

Contents

Design	8
Data structures and sources	8
Authentication	8
Basic authentication	8
OAuth2 authentication	9
Web authentication flow	10
Implementation of the authorization flow	11
Data models and loading	28
Loader	29
Models	31
Base structure	33
 BaseActivity	33
onCreate	35
onDestroy	35
Network cancelling	35
Memory Leak	36
onToolbarBackPressed	36
 NetworkImageView	37
 ViewSafeFragment	42
 FabHideScrollListener	43
 SimpleTextChangeWatcher	44
 KeyBoardVisibilityChecker	44
 Utility methods	46
 User Interface utility methods	46
 Logging	51
 Analytics	54
 Automated build system	55
 Link handling	57
 Possible paths	61
 Showing a chooser	63
 Markdown	64

Spans	64
CharacterStyle	64
ParagraphStyle	65
UpdateAppearance	65
UpdateLayout	65
ReplacementSpan	65
Implementing GitHub Markdown features	66
General strategy	66
Username mentions	68
Issue links	69
Relative links	70
Image links	71
Emoji	72
Loading Emoji	72
Displaying Emoji	75
Text Utilities and Android Regex patterns	76
Linkify	76
Adding links	86
Multiple pattern matching and string escaping	87
Background colour selection	87
Other utility methods	89
Displaying README files	92
JavaScript interface methods	92
Touch interception	94
Vertical scrolling	94
Horizontal scrolling	96
Displaying short Markdown sections	98
CleanURLSpan	98
HorizontalRuleSpan	100
QuoteSpan	102
ListNumberSpan	104
Calculating list number formats	107
RoundedBackgroundEndSpan	108

InlineCodeSpan	110
Dealing with more complex content	112
CodeSpan	112
Drawing	115
Initialisation	116
TableSpan	116
ClickableImageSpan	118
Handling clicks	119
Handling clicks on ReplacementSpans	120
Stopping the onClickHandler	121
MarkdownTextView	123
Handlers	123
Image loading and caching	129
Loading images from Assets and Resources	134
MarkdownTextView	135
MarkdownEditText	141
Tag handling	145
Overriding tags	146
Handlers	147
Tag opening and closing	147
Span opening and closing	150
Span classes	152
Attribute extraction	153
List tags	154
Unordered list opening	155
Ordered list opening	155
List item opening	155
Unordered list closing	155
Ordered list closing	155
List item closing	156
Table tags	156
Table tag opening	157
Table tag closing	157

Font tags	157
Font tag opening	157
Font tag closing	158
Code tags	160
Center tags	161
Strikethrough tags	161
Table row, header, and data	161
Horizontal rule tags	161
Blockquote tags	161
A tags	162
Image tags	162
InlineCode tags	163
Markdown editing	164
Implementing a re-usable editor	164
Utilities	167
KeyBoardDismisingDialogFragment	167
MultiChoiceDialog	168
MarkdownButtonAdapter	171
The EditorActivity	175
Image uploading	180
Image upload process	183
Image source choice	183
Pre-existing image link	183
New image capture	183
Existing image from gallery	184
Image upload	184
Character insertion	184
CharacterAdapter	189
Returning the chosen character	191
Emoji insertion	191
Implementations of EditorActivity	197
CardEditor	197
CommentEditor	205

IssueEditor	209
onCreate	217
Choosing assignees	218
Choosing labels	218
onDone	219
ProjectEditor	219
User Activity	223
UserFragment	231
UserInfoFragment	231
Animation	235
Use of ViewSafeFragment	237
ContributionsView	238
Loading contributions	239
Displaying contributions	243
Contributions statistics	251
Displaying user information	253
Following and unfollowing users	255
UserReposFragment	256
RepositoriesAdapter	258
States	265
insertPinnedRepos	265
Objective 2.b.vi: RepoPinChecker	266
Binding	266
Objective 2.c: UserStarsFragment	267
Objective 2.d: UserGistsFragment	270
Objectives 2.e and 2.f: UserFollowingFragment and UserFollowersFragment	275
Search	280
Objective 3: RepoActivity	286
RepoFragment	290
Objective 3.a: RepoInfoFragment	291
Objective 3.c: RepoReadmeFragment	297
Objective 3.d: RepoCommitsFragment	300
Branch loading	300

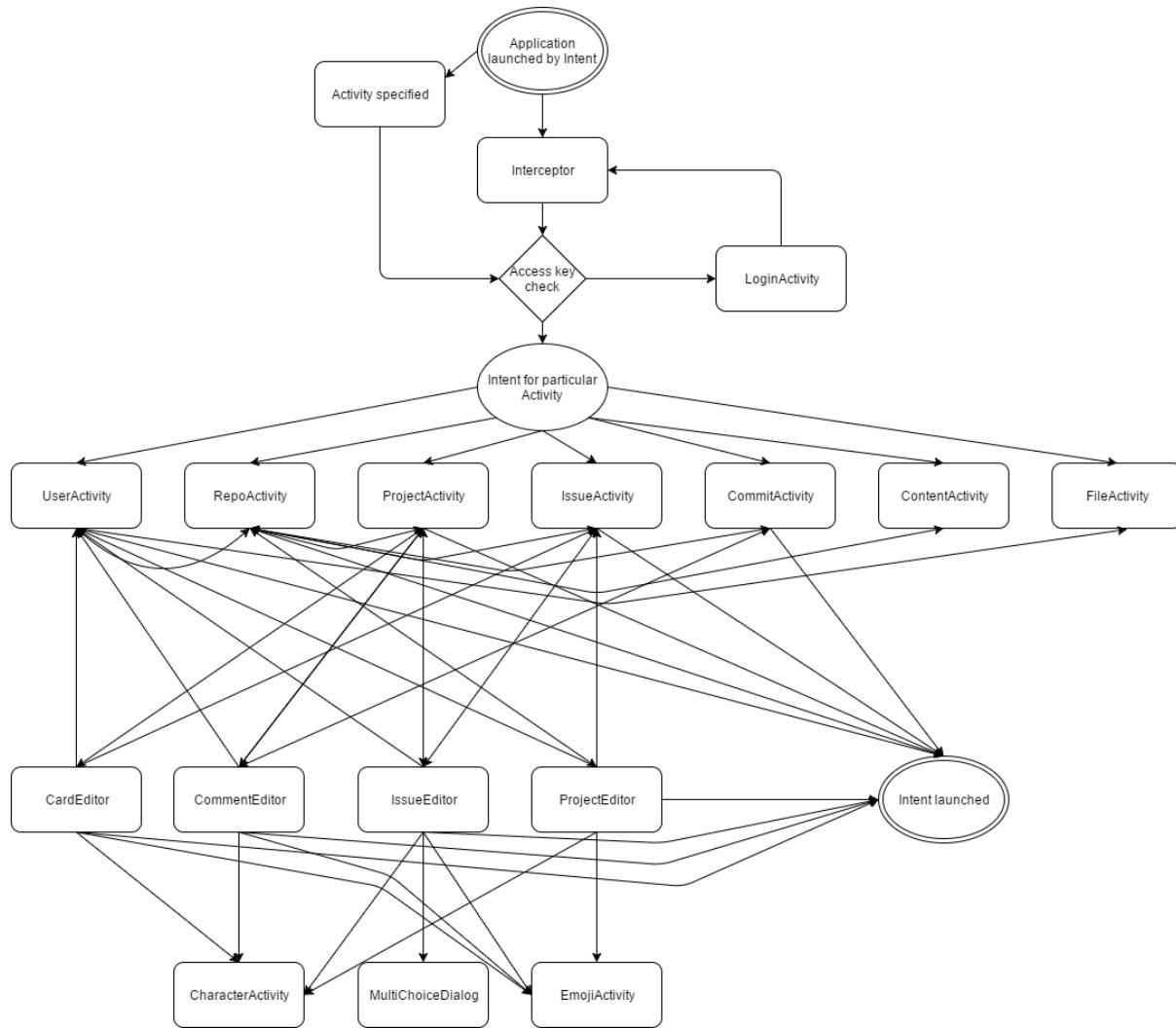
Binding	304
Objective 3.e: RepoIssuesFragment	309
Filtering	318
Editing	319
Creation	320
State changes	320
Adapter	320
Objective 3.f: RepoProjectsFragment	326
Objective 3.b: ContentActivity	333
ContentAdapter	346
Loading files	350
Loading submodule	351
Loading directories	351
Background loading	351
Moving to and from nodes	352
FileActivity	353
Objective 5: CommitActivity	356
Objective 5.a, 5.b, and 5.d: CommitInfoFragment	359
Objective 5.d: Statuses	359
Objective 5.b: CommitDiffAdapter	363
Objective 5.c: CommitCommentsFragment	368
CommitCommentsAdapter	372
Objective 4: IssueActivity	377
Objective 4.a: IssueInfoFragment	382
displayIssue	392
displayAssignees	394
displayMilestone	395
updateIssue	395
toggleIssueState	395
IssueEventsAdapter	395
IssueCommentsFragment	412
IssueCommentsAdapter	416
ProjectActivity	422

Loading the Project	438
 Loading the columns	438
 Adding a new column	438
 Deleting columns	439
 Dragging and dropping Fragments in a ViewPager	439
 NavigationDragListener	440
 ColumnFragment	441
 CardAdapter	455
 Loading issue cards	462
 CardDragListener and the ACTION_DRAG_ENTERED event	462
 ACTION_DRAG_ENTERED	462
 CardDragListener	463
 ACTION_DRAG_ENTERED and ACTION_DRAG_EXITED	465
 ACTION_DROP	466
 ProjectSearchAdapter	467
Notifications	471
 NotificationServiceStartBroadcastReceiver	471
 NotificationEventReceiver	471
 NotificationIntentService	474
 Notification loading and displaying	477
 Dismissing notifications	478

Design

Activity structure

Each rounded rectangular node represents a particular Activity, and each arrow shows a possible entry point for the Activity.



Data structures and sources

Almost all of the data used in the application is acquired through the GitHub API

Authentication

Basic authentication

Requests can be authenticated by sending the user's username and password in the request header, however this is not an acceptable method of authentication for an Android client.

The first problem is that, while HTTPS requests are encrypted, the request itself is likely to be logged by the Android system, and may pass through another service if the user has a VPN active.

The second problem is that the user's account may require more than a password to authenticate.

GitHub supports two factor authentication.

If the user has activated two factor authentication, the two factor pin must be sent with each request.

This is not a usable experience as the pin changes each minute.

OAuth2 authentication

OAuth2 authentication allows applications to request authorization to a user's account without having access to their password.

This method also allows tokens to be limited to specific types of data, and can be revoked by the user.

In order to use the OAuth API, the application must be registered with GitHub.

Register a new OAuth application

Application name

 ✖

Something users will recognize and trust

Homepage URL

The full URL to your application homepage

Application description

Application description is optional

This is displayed to all potential users of your application

Authorization callback URL

Your application's callback URL. Read our [OAuth documentation](#) for more information.

Register application

[Cancel](#)

The application is registered with information recognisable to the user to ensure their trust when authorizing the application.

The call-back URL is the URL which GitHub redirects to once the authorization is complete.

Web authentication flow

1. Redirect users to request access

Display the webpage <https://github.com/login/oauth/authorize>

In order to successfully authenticate we must also pass parameters with the request.

The only required parameter is the client id which was received when the application was registered.

The scope parameter is used to specify what level of access is required to the user's account.

2. Redirect

If the user signs in and accepts the authorization request, GitHub redirects back to your site with a temporary parameter.

The code parameter has a limited time frame to be exchanged for an authorization token.

This is done by posting to https://github.com/login/oauth/access_token.

The post request must have three parameters:

1. The client id, which must match that provided when loading the authorization page
2. The client secret
3. The code received in

If the request is successful the response will be a string containing the access token in the form:

access_token=some_base_64_string&scope=the_scope_requested_in_step_1.

3. Once the access token has been received it can be used for authorization by including the authorization header with each request.

Implementation of the authorization flow

In order to avoid duplication of values used throughout the GitHub API, I have used a single abstract class to contain the headers and path keys used throughout the project

APIHandler.java

```
package com(tpb.github.data;

import android.content.Context;
import android.support.annotation.Nullable;
import android.support.annotation.StringRes;

import com.androidnetworking.error.ANError;
import com(tpb.github.R;
import com(tpb.github.data.auth.GitHubSession;

import java.util.HashMap;

/**
 * Created by theo on 18/12/16.
 */

public abstract class APIHandler {
    static final String TAG = APIHandler.class.getSimpleName();

    protected static final String GIT_BASE = "https://api.github.com";
    private static final String ACCEPT_HEADER_KEY = "Accept";
```

```
    private static final String ACCEPT_HEADER =
"application/vnd.github.v3+json";
    private static final String ORGANIZATIONS_PREVIEW_ACCEPT_HEADER =
"application/vnd.github.korra-preview";
    private static final String PROJECTS_PREVIEW_ACCEPT_HEADER =
"application/vnd.github.inertia-preview+json";
    private static final String REPO_LICENSE_PREVIEW_ACCEPT_HEADER =
"application/vnd.github.drax-preview+json";
    private static final String PAGES_PREVIEW_ACCEPT_HEADER =
"application/vnd.github.mister-fantastic-preview+json";
    private static final String REACTIONS_PREVIEW_ACCEPT_HEADER = "
application/vnd.github.squirrel-girl-preview";
    private static final String AUTHORIZATION_HEADER_KEY = "Authorization";
    private static final String AUTHORIZATION_TOKEN_FORMAT = "token %1$s";
    private static GitHubSession mSession;

    protected static final HashMap<String, String> API_AUTH_HEADERS = new
HashMap<>();
    static final HashMap<String, String> PROJECTS_API_AUTH_HEADERS = new
HashMap<>();
    static final HashMap<String, String> ORGANIZATIONS_API_AUTH_HEADERS = new
HashMap<>();
    static final HashMap<String, String> LICENSES_API_AUTH_HEADERS = new
HashMap<>();
    static final HashMap<String, String> PAGES_API_AUTH_HEADERS = new
HashMap<>();
    static final HashMap<String, String> REACTIONS_API_PREVIEW_AUTH_HEADERS = new
HashMap<>();

    protected static final String SEGMENT_USER = "/user";
    static final String SEGMENT_USERS = "/users";
    static final String SEGMENT_REPOS = "/repos";
    static final String SEGMENT_README = "/readme";
    static final String SEGMENT_COLLABORATORS = "/collaborators";
    static final String SEGMENT_LABELS = "/labels";
    static final String SEGMENT_PROJECTS = "/projects";
    static final String SEGMENT_COLUMNS = "/columns";
    static final String SEGMENT_ISSUES = "/issues";
    static final String SEGMENT_PERMISSION = "/permission";
    static final String SEGMENT_CARDS = "/cards";
    static final String SEGMENT_MOVES = "/moves";
    static final String SEGMENT_COMMENTS = "/comments";
    static final String SEGMENT_EVENTS = "/events";
    static final String SEGMENT_STARRED = "/starred";
    static final String SEGMENT_SUBSCRIPTION = "/subscription";
    static final String SEGMENT_MILESTONES = "/milestones";
    static final String SEGMENT_GISTS = "/gists";
    static final String SEGMENT_FOLLOWING = "/following";
    static final String SEGMENT_FOLLOWERS = "/followers";
    static final String SEGMENT_COMMITS = "/commits";
    static final String SEGMENT_NOTIFICATIONS = "/notifications";

    protected APIHandler(Context context) {
        if(mSession == null) {
            mSession = GitHubSession.getSession(context);
            initHeaders();
        }
    }
```

```

protected final void initHeaders() {
    final String accessToken = mSession.getAccessToken();
    API_AUTH_HEADERS.put(AUTHORIZATION_HEADER_KEY,
        String.format(AUTHORIZATION_TOKEN_FORMAT, accessToken)
    );
    API_AUTH_HEADERS.put(ACCEPT_HEADER_KEY, ACCEPT_HEADER);

    ORGANIZATIONS_API_AUTH_HEADERS.put(AUTHORIZATION_HEADER_KEY,
        String.format(AUTHORIZATION_TOKEN_FORMAT, accessToken)
    );
    ORGANIZATIONS_API_AUTH_HEADERS.put(ACCEPT_HEADER_KEY,
    ORGANIZATIONS_PREVIEW_ACCEPT_HEADER);

    PROJECTS_API_AUTH_HEADERS.put(AUTHORIZATION_HEADER_KEY,
        String.format(AUTHORIZATION_TOKEN_FORMAT, accessToken)
    );
    PROJECTS_API_AUTH_HEADERS.put(ACCEPT_HEADER_KEY,
    PROJECTS_PREVIEW_ACCEPT_HEADER);

    LICENSES_API_AUTH_HEADERS.put(AUTHORIZATION_HEADER_KEY,
        String.format(AUTHORIZATION_TOKEN_FORMAT, accessToken)
    );
    LICENSES_API_AUTH_HEADERS.put(ACCEPT_HEADER_KEY,
    REPO_LICENSE_PREVIEW_ACCEPT_HEADER);

    PAGES_API_AUTH_HEADERS.put(AUTHORIZATION_HEADER_KEY,
        String.format(AUTHORIZATION_TOKEN_FORMAT, accessToken)
    );
    PAGES_API_AUTH_HEADERS.put(ACCEPT_HEADER_KEY,
    PAGES_PREVIEW_ACCEPT_HEADER);

    REACTIONS_API_PREVIEW_AUTH_HEADERS.put(AUTHORIZATION_HEADER_KEY,
        String.format(AUTHORIZATION_TOKEN_FORMAT, accessToken)
    );
    REACTIONS_API_PREVIEW_AUTH_HEADERS.put(ACCEPT_HEADER_KEY,
    REACTIONS_PREVIEW_ACCEPT_HEADER);
}

private static final String CONNECTION_ERROR = "connectionError";

public static final int HTTP_OK_200 = 200; //OK

public static final String HTTP_REDIRECT_NEW_LOCATION = "Location";
public static final int HTTP_301_REDIRECTED = 301; //Should redirect
through the value in location

public static final int HTTP_302_TEMPORARY_REDIRECT = 302; //Redirect for
this request only
public static final int HTTP_307_TEMPORARY_REDIRECT = 307; //Same as above

private static final int HTTP_BAD_REQUEST_400 = 400; //Bad request
problems parsing JSON

public static final String KEY_MESSAGE = "message";
private static final String MESSAGE_BAD_CREDENTIALS = "Bad credentials";
private static final int HTTP_UNAUTHORIZED_401 = 401; //Login required,
account locked, permission error

```

```
private static final String MESSAGE_MAX_LOGIN_ATTEMPTS = "Maximum number of login attempts exceeded.";

public static final String KEY_HEADER_RATE_LIMIT_RESET = "X-RateLimit-Reset";
private static final String MESSAGE_RATE_LIMIT_START = "API rate limit exceeded";
private static final String MESSAGE_ABUSE_LIMIT = "You have triggered an abuse detection mechanism";
private static final int HTTP_FORBIDDEN_403 = 403; //Forbidden server locked or other reasons

private static final int HTTP_NOT_FOUND_404 = 404;

private static final int HTTP_NOT_ALLOWED_405 = 405; //Not allowed (managed server)

private static final int HTTP_409 = 409; //Returned when loading commits for empty repo

private static final int HTTP_419 = 419; //This function can only be executed with an CL-account

public static final String ERROR_MESSAGE_UNPROCESSABLE = "Validation Failed";
public static final String ERROR_MESSAGE_VALIDATION_MISSING = "missing";
public static final String ERROR_MESSAGE_VALIDATION_MISSING_FIELD = "missing_field";
public static final String ERROR_MESSAGE_VALIDATION_INVALID = "invalid";
public static final String ERROR_MESSAGE_VALIDATION_ALREADY_EXISTS = "already_exists";
private static final String ERROR_MESSAGE_EMPTY_REPOSITORY = "Git Repository is empty.";
private static final int HTTP_UNPROCESSABLE_422 = 422; // Validation failed

//600 codes are server codes https://github.com/GleSYS/API/wiki/API-Error-codes#6xx---server

//700 codes are ip errors https://github.com/GleSYS/API/wiki/API-Error-codes#7xxx---ip

//800 codes are archive codes https://github.com/GleSYS/API/wiki/API-Error-codes#8xx---archive

//900 domain https://github.com/GleSYS/API/wiki/API-Error-codes#9xx---domain

//1000 email https://github.com/GleSYS/API/wiki/API-Error-codes#10xx---email

//1100 livechat https://github.com/GleSYS/API/wiki/API-Error-codes#11xx---livechat

//1200 invoice https://github.com/GleSYS/API/wiki/API-Error-codes#11xx---livechat

//1300 glera https://github.com/GleSYS/API/wiki/API-Error-codes#13xx---glera
```

```
//1400 transaction https://github.com/GleSYS/API/wiki/API-Error-
codes#14xx---transaction

//1500 vpn https://github.com/GleSYS/API/wiki/API-Error-codes#15xx---vpn

public static final int GIT_LOGIN_FAILED_1601 = 1601; //Login failed

public static final int GIT_LOGIN_FAILED_1602 = 1602; //Login failed
unknown

public static final int GIT_GOOGLE_AUTHENTICATOR OTP REQUIRED_1603 = 1603;
//Google auth error

public static final int GIT_YUBIKEY_1604 = 1604; //Yubikey OTP required

public static final int GIT_NOT_LOGGED_IN_1605 = 1605; //Not logged in as
user

//1700 invite https://github.com/GleSYS/API/wiki/API-Error-codes#17xx---invite

//1800 test account https://github.com/GleSYS/API/wiki/API-Error-
codes#18xx---test-account

//1900 network https://github.com/GleSYS/API/wiki/API-Error-codes#19xx---network

static APIError parseError(ANError error) {
    APIError apiError;
    if(CONNECTION_ERROR.equals(error.getErrorDetail())) {
        apiError = APIError.NO_CONNECTION;
    } else {
        switch(error.getErrorCode()) {
            case HTTP_BAD_REQUEST_400:
                apiError = APIError.BAD_REQUEST;
                break;
            case HTTP_UNAUTHORIZED_401:
                apiError = APIError.UNAUTHORIZED;
                if(error.getErrorBody() != null) {

if(error.getErrorBody().contains(MESSAGE_BAD_CREDENTIALS)) {
                    apiError = APIError.BAD_CREDENTIALS;
                } else
if(error.getErrorBody().contains(MESSAGE_MAX_LOGIN_ATTEMPTS)) {
                    apiError = APIError.MAX_LOGIN_ATTEMPTS;
                }
            }
            break;
        case HTTP_FORBIDDEN_403:
            apiError = APIError.FORBIDDEN;
            if(error.getErrorBody() != null) {

if(error.getErrorBody().contains(MESSAGE_RATE_LIMIT_START)) {
                    apiError = APIError.RATE_LIMIT;
                } else
if(error.getErrorBody().contains(MESSAGE_ABUSE_LIMIT)) {
                    apiError = APIError.ABUSE_LIMIT;
                }
            }
        }
    }
}
```

```

        }
        break;
    case HTTP_NOT_ALLOWED_405:
        apiError = APIError.NOT_ALLOWED;
        break;
    case HTTP_NOT_FOUND_404:
        apiError = APIError.NOT_FOUND;
        break;
    case HTTP_UNPROCESSABLE_422:
        apiError = APIError.UNPROCESSABLE;
        break;
    case HTTP_409:
        if(error.getErrorBody() != null &&
error.getErrorBody().contains(
            ERROR_MESSAGE_EMPTY_REPOSITORY)) {
            apiError = APIError.EMPTY_REPOSITORY;
            break;
        }
    default:
        apiError = APIError.UNKNOWN;
    }
}
apiError.error = error;
return apiError;
}

public enum APIError {
    NO_CONNECTION(R.string.error_no_connection),
    UNAUTHORIZED(R.string.error_unauthorized),
    FORBIDDEN(R.string.error_forbidden),
    NOT_FOUND(R.string.error_not_found),
    UNKNOWN(R.string.error_unknown),
    RATE_LIMIT(R.string.error_rate_limit),
    ABUSE_LIMIT(R.string.error_abuse_limit),
    MAX_LOGIN_ATTEMPTS(R.string.error_max_login_attempts),
    UNPROCESSABLE(R.string.error_unprocessable),
    BAD_CREDENTIALS(R.string.error_bad_credentials),
    NOT_ALLOWED(R.string.error_not_allowed),
    BAD_REQUEST(R.string.error_bad_request),
    EMPTY_REPOSITORY(R.string.error_empty_repository);

    @StringRes
    public final int resId;

    @Nullable ANError error;

    APIError(@StringRes int resId) {
        this.resId = resId;
    }
}

}

```

The APIHandler class is mostly static constants:

- GIT_BASE - The base URL for all GitHub API requests
- ACCEPT_HEADER_KEY - The key for the content type header

- ACCEPT_HEADER - The default content type, for JSON results
- ORGANIZATIONS_PREVIEW_ACCEPT_HEADER - A content type header required for some features of the API, specifically requesting collaborators on a repository
- PROJECTS_PREVIEW_API_HEADER - A content type header required for requesting information related to projects
- REPO_LICENSE_PREVIEW_API_HEADER - A content type header required to load information about a repositories license when loading the repository
- PAGES_PREVIEW_ACCEPT_HEADER - A content type header required to load information about a repositories pages when loading the repository
- AUTHORIZATION_HEADER_KEY - A header key for providing the authorization token
- AUTHORIZATION_TOKEN_FORMAT - A format string used when inserting the authorization key into a header

Next we have the headers themselves.

As headers are key value pairs, they are represented as string to string maps.

We then have the SEGMENT_ constants. These are segments of the API paths which are used across numerous different API requests.

The APIHandler constructor checks if the single instance of GitHubSession is null, and if so access the singleton session instance before initialising the headers.

Next each header map is initialised with the authorization token header, and their own respective accept headers.

The class which actually stores the authorization information is GitHubSession, which was used once above in APIHandler.

GitHubSession.java

```
package com(tpb.github.data.auth;

import android.content.Context;
import android.content.SharedPreferences;
import android.support.annotation.NonNull;

import com(tpb.github.data.models.User;

import org.json.JSONException;
import org.json.JSONObject;

public class GitHubSession {
    private static final String TAG = GitHubSession.class.getSimpleName();

    private static GitHubSession session;
    private final SharedPreferences prefs;

    private static final String SHARED = "GitHub_Preferences";
    private static final String API_LOGIN = "username";
    private static final String API_ID = "id";
```

```

private static final String API_ACCESS_TOKEN = "access_token";
private static final String INFO_USER = "user_json";

private GitHubSession(Context context) {
    prefs = context.getSharedPreferences(SHARED, Context.MODE_PRIVATE);
}

public static GitHubSession getSession(Context context) {
    if(session == null) session = new GitHubSession(context);
    return session;
}

void storeUser(JSONObject json) {
    final SharedPreferences.Editor editor = prefs.edit();
    editor.putString(INFO_USER, json.toString());
    final User user = new User(json);
    editor.putInt(API_ID, user.getId());
    editor.putString(API_LOGIN, user.getLogin());
    editor.apply();
}

void storeAccessToken(@NonNull String accessToken) {
    final SharedPreferences.Editor editor = prefs.edit();
    editor.putString(API_ACCESS_TOKEN, accessToken);
    editor.apply();
}

public User getUser() {
    try {
        final JSONObject obj = new JSONObject(prefs.getString(INFO_USER,
 ""));
        return new User(obj);
    } catch(JSONException jse) {
        return null;
    }
}

public String getUserLogin() {
    return prefs.getString(API_LOGIN, null);
}

public String getAccessToken() {
    return prefs.getString(API_ACCESS_TOKEN, null);
}

public boolean hasAccessToken() {
    return getAccessToken() != null;
}
}

```

GitHubSession is a singleton class which saves and loads the user credentials and authorization token to and from shared preferences.

The private constructor is used to initialise the SharedPreferences instance, this either opens the pre-existing map or creates a new one if it does not exist.

When the user authorizes the app the access token is stored with `storeAccessToken(@NonNull String accessToken)`.

Once we have an authorization token, we can load the user's data and store it for later use.

The LoginActivity consists of two layouts, only one of which is visible at a time. The first layout is a WebView which is used to display the user authentication page, and the second is a layout to display the user's information once they have signed in.

LoginActivity.java

```
package com(tpb.projects.login;

import android.annotation.SuppressLint;
import android.content.Intent;
import android.graphics.Bitmap;
import android.os.Bundle;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.CardView;
import android.view.View;
import android.view.ViewGroup;
import android.webkit.CookieManager;
import android.webkit.CookieSyncManager;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import android.widget.FrameLayout;
import android.widget.LinearLayout;
import android.widget.ProgressBar;

import com.google.firebaseio.analytics.FirebaseAnalytics;
import com(tpb.github.data.auth.OAuthHandler;
import com(tpb.github.data.models.User;
import com(tpb.projects.BuildConfig;
import com(tpb.projects.R;
import com(tpb.projects.markdown.Formatter;
import com(tpb.projects.user.UserActivity;
import com(tpb.projects.util.Analytics;
import com(tpb.projects.util.UI;

import butterknife.BindView;
import butterknife.ButterKnife;

import static com(tpb.projects.flow.ProjectsApplication.mAnalytics;

/**
 * A login screen that offers login via email/password.
 */
public class LoginActivity extends AppCompatActivity implements
OAuthHandler.OAuthAuthenticationListener {
    private static final String TAG = LoginActivity.class.getSimpleName();
    private boolean mLoginShown = false;

    @BindView(R.id.login_webview) WebView mWebView;
    @BindView(R.id.login_form) CardView mLogin;
    @BindView(R.id.progress_spinner) ProgressBar mSpinner;
    @BindView(R.id.user_details) LinearLayout mUserDetails;

    private Intent mLaunchIntent;
```

```
    private static final FrameLayout.LayoutParams FILL = new
FrameLayout.LayoutParams(
    ViewGroup.LayoutParams.FILL_PARENT,
    ViewGroup.LayoutParams.FILL_PARENT
);

@SuppressWarnings("SetJavaScriptEnabled")
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login);
    ButterKnife.bind(this);

    final OAuthHandler OAuthHandler = new OAuthHandler(this,
        BuildConfig.GITHUB_CLIENT_ID,
        BuildConfig.GITHUB_CLIENT_SECRET,
        BuildConfig.GITHUB_REDIRECT_URL,
        this
    );

    if(getIntent().hasExtra(Intent.EXTRA_INTENT)) {
        mLaunchIntent =
getIntent().getParcelableExtra(Intent.EXTRA_INTENT);
    } else {
        mLaunchIntent = new Intent(LoginActivity.this,
UserActivity.class);
    }

    mWebView.setVerticalScrollBarEnabled(false);
    mWebView.setHorizontalScrollBarEnabled(false);
    mWebView.setWebViewClient(new OAuthWebViewClient(OAuthHandler));
    mWebView.getSettings().setJavaScriptEnabled(true);
    mWebView.loadUrl(OAuthHandler.getAuthUrl());
    mWebView.setLayoutParams(FILL);

    mUserDetails.setVisibility(View.GONE);
    UI.expand(mLogin);
}

@Override
public void onSuccess() {
    mWebView.setVisibility(View.GONE);
    mSpinner.setVisibility(View.VISIBLE);
}

@Override
public void onFailure(String error) {
}

@Override
public void userLoaded(User user) {
    mSpinner.setVisibility(View.GONE);
    Formatter.displayUser(mUserDetails, user);
    final Bundle bundle = new Bundle();
    bundle.putString(Analytics.TAG_LOGIN, Analytics.VALUE_SUCCESS);
    mAnalytics.logEvent(FirebaseAnalytics.Event.LOGIN, bundle);
    new Handler().postDelayed(() -> {
```

```

        CookieSyncManager.createInstance(this);
        final CookieManager cookieManager = CookieManager.getInstance();
        cookieManager.removeAllCookie();
        startActivity(mLaunchIntent);
        overridePendingTransition(R.anim.slide_up, R.anim.none);
        finish();
    }, 1500);
}

private void ensureWebViewVisible() {
    if(!mLoginShown) {
        new Handler().postDelayed(() -> {
            mWebView.setVisibility(View.VISIBLE);
            mSpinner.setVisibility(View.GONE);
            mLoginShown = true;
        }, 150);
    }
}

private class OAuthWebViewClient extends WebViewClient {
    private final OAuthHandler mAuthHandler;

    OAuthWebViewClient(OAuthHandler handler) {
        mAuthHandler = handler;
    }

    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
        return
    !(url.startsWith("https://github.com/login/oauth/authorize") ||
        url.startsWith("https://github.com/login?") ||
        url.startsWith("https://github.com/session") ||
        url.startsWith(BuildConfig.GITHUB_REDIRECT_URL));
    }

    @Override
    public void onPageStarted(WebView view, String url, Bitmap favicon) {
        if(url.contains("?code=")) {
            final String[] parts = url.split "=";
            mAuthHandler.getAccessToken(parts[1]);
        }
        super.onPageStarted(view, url, favicon);
    }

    @Override
    public void onPageFinished(WebView view, String url) {
        ensureWebViewVisible();
        super.onPageFinished(view, url);
    }
}
}

```

The LoginActivity binds four views.

1. The `WebView` which displays the login webpage
2. The `CardView` which holds the `WebView` and user layout
3. The spinning `ProgressBar` which is displayed to indicate progress while the `WebView` is loading or the user's information is being loaded.
4. The `LinearLayout` which will be filled with views showing the user's information

In the `onCreate` method the `WebView` is set up not to allow scrolling, to enable JavaScript, and to use a client implementation to override page loading.

```
mWebView.setVerticalScrollBarEnabled(false);
mWebView.setHorizontalScrollBarEnabled(false);
mWebView.setWebViewClient(new OAuthWebViewClient(OAuthHandler.getListener()));
mWebView.getSettings().setJavaScriptEnabled(true);
mWebView.loadUrl(OAuthHandler.getAuthUrl());
mWebView.setLayoutParams(FILL);
```

The `OAuthWebViewClient` extends `WebViewClient` and is used to capture the code once the user has logged in, as well as ensuring that the user only navigates through the pages required to log in.

The method `onPageStarted(WebView view, String url, Bitmap favicon)` is called whenever a page load begins.

The client checks if the URL contains `?code=`, and if so, passes the segment after that point to the `OAuthHandler` which then requests the authorization token.

This completes objective 1.a

OAuthHandler.java

```
package com(tpb.github.data.auth;

/**
 * Created by theo on 15/12/16.
 */

import android.content.Context;
import android.util.Log;

import com.androidnetworking.AndroidNetworking;
import com.androidnetworking.error.ANError;
import com.androidnetworking.interfaces.JSONObjectRequestListener;
import com.androidnetworking.interfaces.StringRequestListener;
import com(tpb.github.data.APIHandler;
import com(tpb.github.data.models.User;

import org.json.JSONObject;

public class OAuthHandler extends APIHandler {
    private static final String TAG = OAuthHandler.class.getSimpleName();

    private final GitHubSession mSession;
    private final OAuthAuthenticationListener mListener;
    private final String mAuthUrl;
    private final String mTokenUrl;
    private String mAccessToken;
```

```

    private static final String AUTH_URL =
"https://github.com/login/oauth/authorize?";
    private static final String TOKEN_URL =
"https://github.com/login/oauth/access_token?";
    private static final String SCOPE = "user public_repo repo gist";

    private static final String TOKEN_URL_FORMAT = TOKEN_URL +
"client_id=%1$s&client_secret=%2$s&redirect_uri=%3$s";
    private static final String AUTH_URL_FORMAT = AUTH_URL +
"client_id=%1$s&scope=%2$s&redirect_uri=%3$s";

    public OAuthHandler(Context context, String clientId, String clientSecret,
                        String callbackUrl,
                        OAuthAuthenticationListener listener) {
        super(context);
        mSession = GitHubSession.getSession(context);
        mTokenUrl = String.format(TOKEN_URL_FORMAT, clientId, clientSecret,
callbackUrl);
        mAuthUrl = String.format(AUTH_URL_FORMAT, clientId, SCOPE,
callbackUrl);
        mListener = listener;
    }

    public void getAccessToken(final String code) {
        AndroidNetworking.get(mTokenUrl + "&code=" + code)
            .build()
            .getAsString(new StringRequestListener() {
                @Override
                public void onResponse(String response) {
                    mAccessToken = response.substring(
                        response.indexOf("access_token=") +
13,
                        response.indexOf("&scope"))
                );
                mSession.storeAccessToken(mAccessToken);
                initHeaders();
                mListener.onSuccess();
                fetchUser();
            }
            @Override
            public void onError(ANError anError) {
                mListener.onFail(anError.getErrorDetail());
            }
        });
    }

    private void fetchUser() {
        AndroidNetworking.get(GIT_BASE + SEGMENT_USER)
            .addHeaders(API_AUTH_HEADERS)
            .build()
            .getAsJSONObject(new JSONObjectRequestListener() {
                @Override
                public void onResponse(JSONObject response) {
                    mSession.storeUser(response);
                    mListener.userLoaded(mSession.getUser());
                }
            }
            @Override

```

```

        public void onError(ANError anError) {
            mListener.onFail(anError.getErrorDetail());
            Log.e(TAG, "onError: " +
anError.getErrorDetail());
        }
    });

    public String getAuthUrl() {
        return mAuthUrl;
    }

    public interface OAuthAuthenticationListener {
        void onSuccess();

        void onFail(String error);

        void userLoaded(User user);
    }
}

```

The OAuthHandler is used to load the authenticated user for the first time. `getAccessToken(final String code)` performs a get request to the formatted token URL, and parses the response as a string.

The access token is extracted from the string between "access_token=" and "&scope".

Once the access token has been extracted:

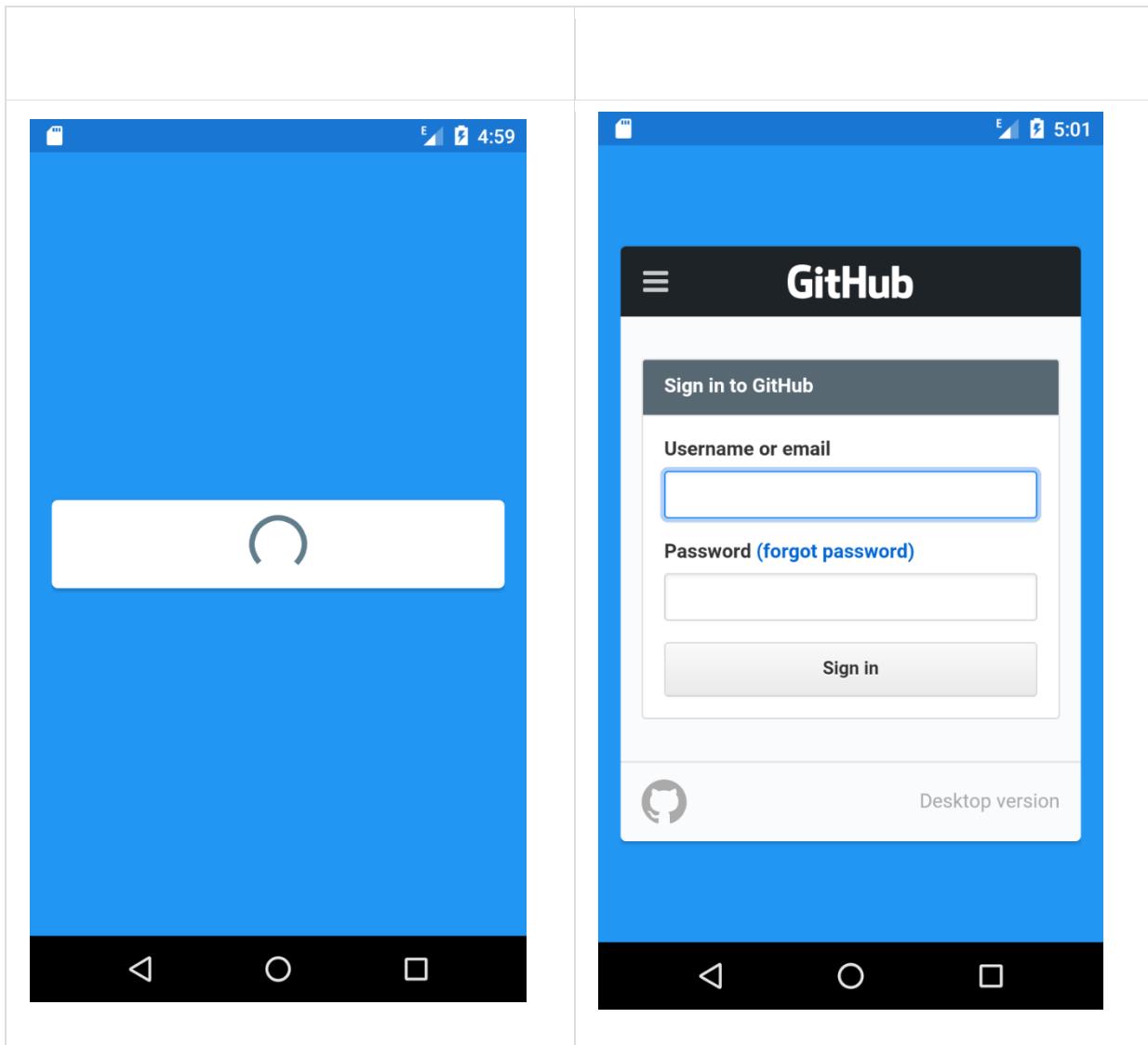
- The token is stored with GitHubSession
- The headers are initialised with the token
- The authentication listener (LoginActivity) is notified, allowing it to update the ProgressBar
- `fetchUser()` is called to load the authenticated user model

`fetchUser()` performs another request, this to time /user, which loads the authenticated user if provided with an authorization token.

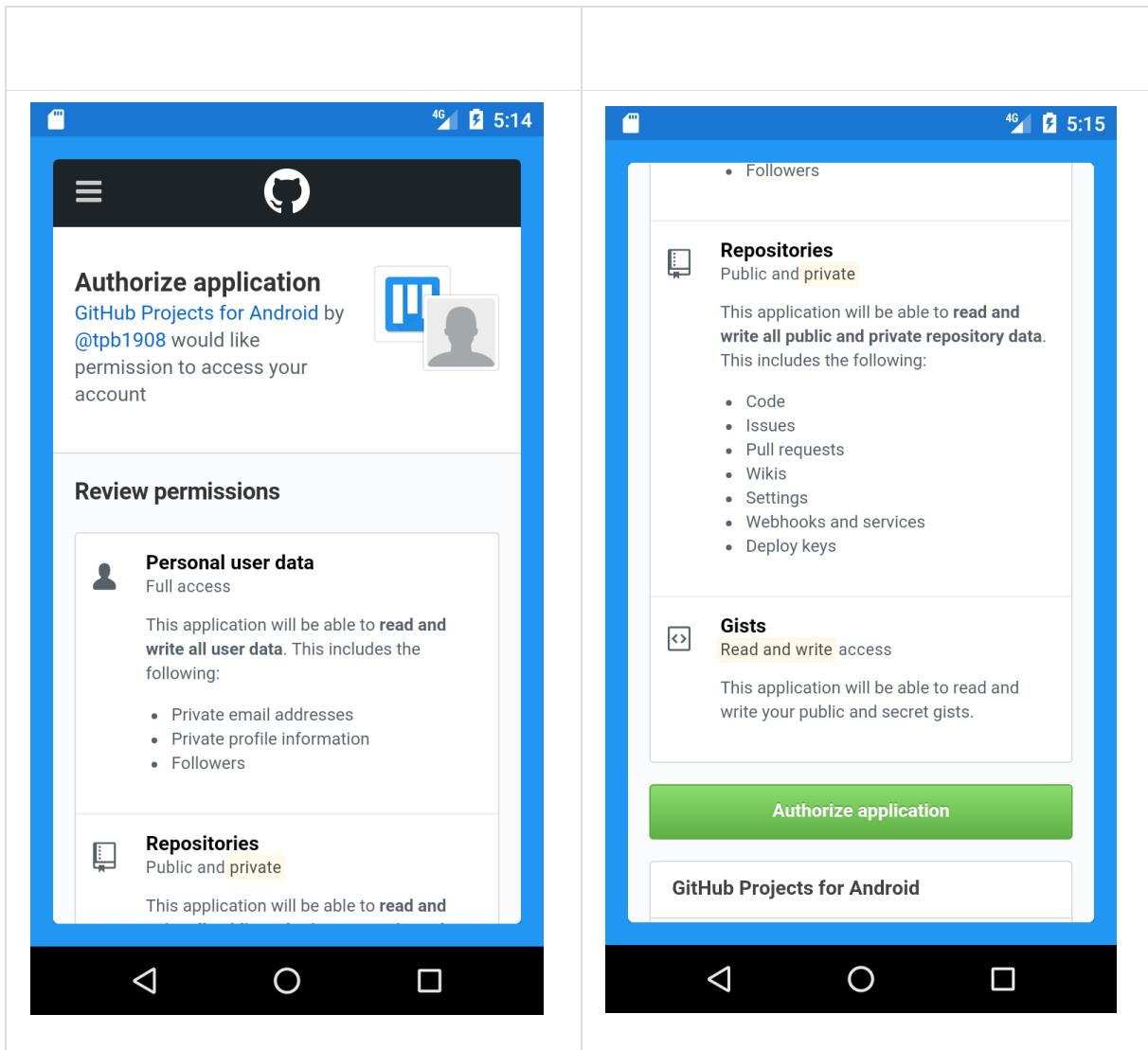
The response is returned as a `JSONObject`, Java's built in JSON model, and is passed to GitHubSession where it is stored as a string for later use, and parsed into a `User` object.

The authentication listener is then called again, with the `User` object.

Prior to the `WebView` loading the login page, the `LoginActivity` shows the view with a spinning `ProgressBar`, and once it has finished the login page is displayed



Once the user has logged in, the GitHub authentication page will show the access which the app is asking for, and ask the user to grant access:



If the user presses the authorize button, the page will be redirected through URL containing the path parameter "code".

In on the overridden `onPageStarted` method of the `OAuthWebViewClient` this results in the `code` parameter being passed to the `OAuthHandler` through `fetchAccessToken`.

LoginActivity.java

```
public void onPageStarted(WebView view, String url, Bitmap favicon) {
    if(url.contains("?code=")) {
        final String[] parts = url.split "=";
        mAuthHandler.fetchAccessToken(parts[1]);
    }
    super.onPageStarted(view, url, favicon);
}
```

OAuthHandler.java

```
public void getAccessToken(final String code) {
    AndroidNetworking.get(mTokenUrl + "&code=" + code)
        .build()
        .getAsString(new StringRequestListener() {
            @Override
```

```
    public void onResponse(String response) {
        mAccessToken = response.substring(
            response.indexOf("access_token=") +
13,
            response.indexOf("&scope")
        );
        mSession.storeAccessToken(mAccessToken);
        initHeaders();
        mListener.onSuccess();
        fetchUser();
    }

    @Override
    public void onError(ANError anError) {
        mListener.onFail(anError.getErrorDetail());
    }
};
```

The `fetchAccessToken` method performs a get request to the token URL created with the apps client id and secret, adding the code as a path parameter.

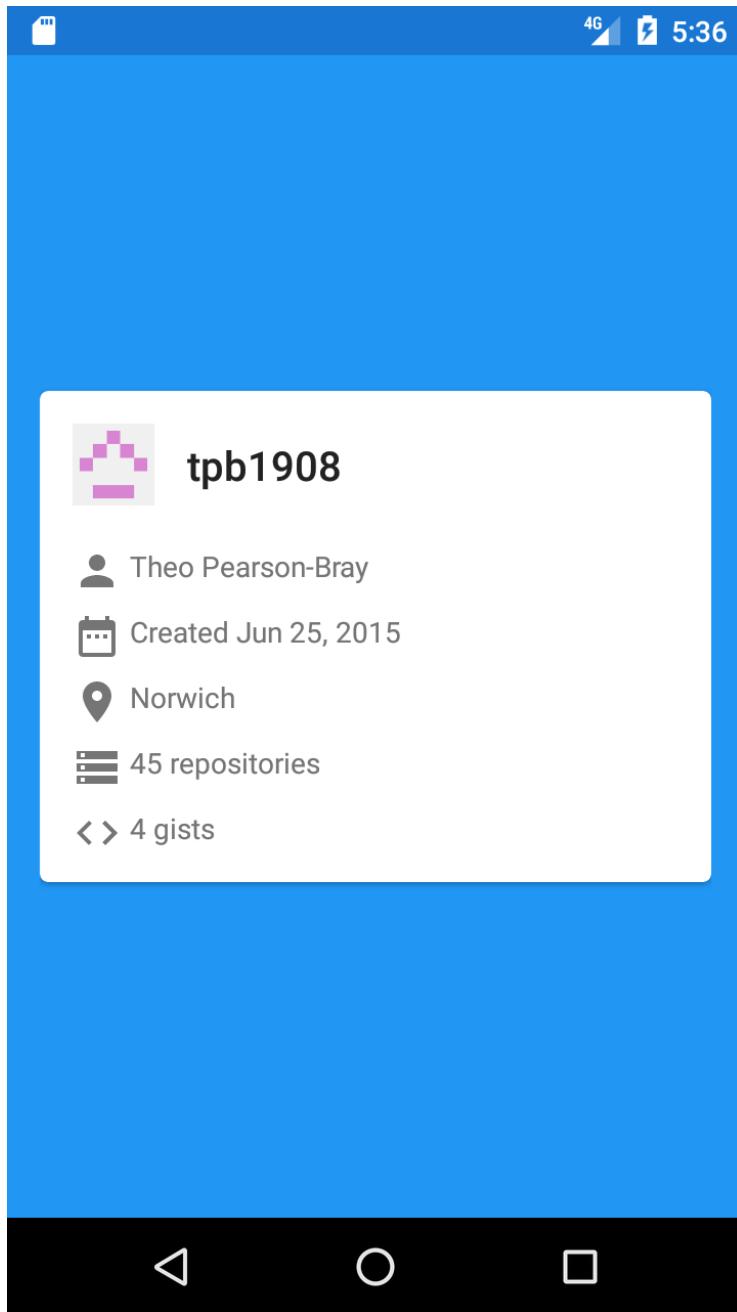
On a successful response the access token is split from the returned value and stored through `GitHubSession`.

This completes objective 1.b

The authorization headers are initialised with a call to `initHeaders` and the `OAuthAuthenticationListener` (`LoginActivity`) is notified of the success.

Finally, `fetchUser` is called.

This method performs a second get request to the Git API, this time to load the `User` model and store the JSON data, as well as notifying the `OAuthAuthenticationListener` that the user has been loaded, allowing the user information to be displayed.



Data models and loading

All of the models from the GitHub API are returned as JSON unless another content type is specified.

Each endpoint returns either a single model, or a single dimensional array of models.

All models used extend the abstract class `DataModel` which contains some of the commonly used keys, as well as the object creation date which is used across all models.

DataModel.java

```

package com(tpb.github.data.models;

/**
 * Created by theo on 15/12/16.
 */

public abstract class DataModel {

    static final String ID = "id";
    static final String NAME = "name";
    static final String CREATED_AT = "created_at";
    static final String UPDATED_AT = "updated_at";
    static final String URL = "url";
    public static final String JSON_NULL = "null";

    long createdAt;

    public abstract long getCreatedAt();

}

}

```

Networking is split into two classes extending APIHandler is split across four classes.

Loader

The Loader class is responsible for almost all get requests sent to the GitHub API. Each method is responsible for load a single model type, and takes the path or filter parameters required to load the model(s) as well as an implementation of a generic loader.

The first interface ItemLoader is used when loading a single model or value.

```

public interface ItemLoader<T> {

    void loadComplete(T data);

    void loadError(APIError error);

}

```

Any class implementing ItemLoader must implement loadComplete(T data) as well as loadError(APIError error).

Most uses of ItemLoader load an instance of DataModel.

An example is loadIssue(@NonNull final ItemLoader<Issue> loader, String repoFullName, int issueNumber, boolean highPriority)

Loader.java

```

public Loader loadIssue(@NonNull final ItemLoader<Issue> loader, String
repoFullName, int issueNumber, boolean highPriority) {
    get(GIT_BASE + SEGMENT_REPO + "/" + repoFullName + SEGMENT_ISSUES +
"/" + issueNumber)
        .addHeaders(REACTIONS_API_PREVIEW_AUTH_HEADERS)
        .setPriority(highPriority ? Priority.HIGH : Priority.MEDIUM)
        .setTag(loader)
}

```

```

    .build()
    .getAsJSONObject(new JSONObjectRequestListener() {
        @Override
        public void onResponse(JSONObject response) {
            loader.loadComplete(new Issue(response));
        }

        @Override
        public void onError(ANError anError) {
            loader.loadError(parseError(anError));
        }
    });
    return this;
}

```

This method is used to load a single `Issue` model given a full repository name (user login and repository name) and the issue number.

Some single methods also have prefetching when a null `ItemLoader` is passed to them:

Loader.java

```

public Loader loadProject(@Nullable final ItemLoader<Project> loader, int id)
{
    final ANRequest req = get(GIT_BASE + SEGMENT_PROJECTS + "/" + id)
        .addHeaders(PROJECTS_API_AUTH_HEADERS)
        .setTag(loader)
        .build();
    if(loader == null) {
        req.prefetch();
    } else {
        req.getAsJSONObject(new JSONObjectRequestListener() {
            @Override
            public void onResponse(JSONObject response) {
                loader.loadComplete(new Project(response));
            }

            @Override
            public void onError(ANError anError) {
                loader.loadError(parseError(anError));
            }
        });
    }
    return this;
}

```

In this case the `ANRequest` instance is built and only requested as a `JSONObject` when there is an `ItemLoader` to deal with the model.

This allows the response to be pre-loaded before an Activity is started, and only parsed to a `DataModel` once a user interface is present to use it.

The `Loader` class also contains the `ListLoader` interface

```
public interface ListLoader<T> {  
    void listLoadComplete(List<T> data);  
    void listLoadError(APIError error);  
}
```

which is used to return objects parsed from a JSONArray as a list of DataModels.
The Loader is a singleton accessed by Loader.getLoader

Models

Numerous different models are required to fulfil different objectives

1. User model:
 - o 1. Sign in
 - o 2. Users
 - o 3. Repositories
 - o 4. Issues
 - o 5. Commits
 - o 6. Projects
2. Repository model:
 - o 2.b User repositories
 - o 2.c User starred repositories
 - o 3. Repositories
3. Node model
 - o 3.b Repository files
4. Issue and label models:
 - o 3.e Repository issues
 - o 4. Issues
 - o 6.d.ii Project issue cards
 - o 6.g Editing project issue cards
 - o 6.h.ii Creation of new issue cards
 - o 6.h.iii Creation of new issue cards from existing issues
5. IssueEvent and MergedModel models:
 - o 4.a.x Issue events
6. Gist model:
 - o 2.d User gists
7. Project model:
 - o 3.f Repository projects
 - o 6. Projects

8. Column model:
 - 6. Projects
9. Card model:
 - 6. Projects
10. Commit model:
 - 3.d Repository commits
 - 5. Commits
11. DiffFile model:
 - 5.b Commit diffs
12. Status and CompleteStatus models
 - 5.d commit statuses
13. State model:
 - Issue model
 - Project model
 - Milestone model
14. Comment model:
 - 4.b Issue comments
 - 5.c Commit comments
15. Notification model:
 - 8. Notifications

Base structure

In order to maintain a cleaner application structure, it is normal to separate repeated logic into a base class or a set of utility classes.

BaseActivity

In this case BaseActivity is used to check that there is an authentication token stored, to provide an onClick method for the toolbar back button, cancel network requests, and to fix a memory leak caused by Android system transitions keeping a reference to the DecorView which in turn references the Activity.

BaseActivity.java

```
package com(tpb.projects.common;

import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.transition.Transition;
import android.support.transition.TransitionManager;
import android.support.v4.app.Fragment;
import android.support.v7.app.AppCompatActivity;
import android.util.ArrayMap;
import android.view.View;
import android.view.ViewGroup;

import com.androidnetworking.AndroidNetworking;
import com(tpb.github.data.auth.GitHubSession;
import com(tpb.projects.login.LoginActivity;
import com(tpb.projects.notifications.receivers.NotificationEventReceiver;

import java.lang.ref.WeakReference;
import java.lang.reflect.Field;
import java.util.ArrayList;
import java.util.List;

/**
 * Created by theo on 10/03/17.
 */

public abstract class BaseActivity extends AppCompatActivity {

    public boolean mHasAccess = true;
    private final List<WeakReference<Fragment>> mWeakFragments = new
ArrayList<>();

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        if(GitHubSession.getSession(this).hasAccessToken()) {
            mHasAccess = true;
            NotificationEventReceiver.setupAlarm(getApplicationContext());
        } else {
```

```

        mHasAccess = false;
        final Intent intent = new Intent(BaseActivity.this,
LoginActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NEW_TASK);
        if(getIntent() != null &&
!getIntent().getAction().equals(Intent.ACTION_MAIN)) {
            intent.putExtra(Intent.EXTRA_INTENT, getIntent());
        }
        startActivity(intent);
        finish();
    }

}

public void onToolbarBackPressed(View view) {
    onBackPressed();
}

@Override
protected void onDestroy() {
    super.onDestroy();
    removeActivityFromTransitionManager();
    cancelNetworkRequests();
}

private void removeActivityFromTransitionManager() {
    final Class transitionManagerClass = TransitionManager.class;
    try {
        final Field runningTransitionsField = transitionManagerClass
            .getDeclaredField("sRunningTransitions");
        runningTransitionsField.setAccessible(true);
        //noinspection unchecked
        final ThreadLocal<WeakReference<ArrayMap<ViewGroup,
ArrayList<Transition>>> runningTransitions
            = (ThreadLocal<WeakReference<ArrayMap<ViewGroup,
ArrayList<Transition>>>)
                runningTransitionsField.get(transitionManagerClass);
        if(runningTransitions.get() == null ||
runningTransitions.get().get() == null) {
            return;
        }
        ArrayMap map = runningTransitions.get().get();
        View decorView = getWindow().getDecorView();
        if(map.containsKey(decorView)) {
            map.remove(decorView);
        }
    } catch(Exception ignored) {
    }
}

@Override
public void onAttachFragment (Fragment fragment) {
    mWeakFragments.add(new WeakReference<>(fragment));
}

private void cancelNetworkRequests() {
    AndroidNetworking.cancel(this);
    for(WeakReference<Fragment> ref : mWeakFragments) {

```

```

        if(ref.get() != null) {
            AndroidNetworking.cancel(ref.get());
        }
    }
}

```

onCreate

The `BaseActivity` `onCreate` method checks that `GitHubSession` has an access token, setting the `mHasAccess` flag accordingly.

If there is an access token, the notification service (Objective 8) is started.

If there is not an access token the `mHasAccess` flag is set to false, allowing any extending `Activities` to return from their `onCreate` methods without performing any unnecessary initialisation, such as `view inflation`.

An `Intent` is then created to launch `LoginActivity`, setting the flags `Intent.FLAG_ACTIVITY_CLEAR_TASK` and `Intent.FLAG_ACTIVITY_NEW_TASK` which result in the current task, a group of `Activities` being destroyed and a new task being created for the `LoginActivity`.

Next, the method checks if the `Intent` which started `BaseActivity` is not from the home screen (`ACTION_MAIN`).

If the `Intent` is not from the homescreen, the `Activity` was launched from a URL which the user likely wants to view once they have signed in. This can be achieved by passing the launch `Intent` forward to the `LoginActivity`.

onDestroy

Network cancelling

Each network request made uses the calling object, e.g. an implementation of `ItemLoader` as a tag.

The `BaseActivity` retains a `WeakReferences` to each of the `Fragments` attached to it, and uses these to cancel network requests as the `Activity` is destroyed.

BaseActivity.java

```

public void onAttachFragment (Fragment fragment) {
    mWeakFragments.add(new WeakReference<>(fragment));
}

```

`onAttachFragment` adds a `WeakReference` to the `Fragment` to `mWeakFragments`, and `cancelNetworkRequests` uses these to cancel network requests started by each `Fragment`.

BaseActivity.java

```

private void cancelNetworkRequests() {
    AndroidNetworking.cancel(this);
}

```

```

    for(WeakReference<Fragment> ref : mWeakFragments) {
        if(ref.get() != null) {
            AndroidNetworking.cancel(ref.get());
        }
    }
}

```

Memory Leak

A bug introduced in Android 5.0, launched in November 2014, which results in a reference to the Activity being kept in the TransitionManager.

The memory leak can be solved by using reflection to remove the Activity from the TransitionManager map.

```

final ThreadLocal<WeakReference<ArrayMap<ViewGroup, ArrayList<Transition>>>>
runningTransitions = (ThreadLocal<WeakReference<ArrayMap<ViewGroup,
ArrayList<Transition>>>> runningTransitionsField.get(transitionManagerClass));

```

The runningTransitionsField refers to a ThreadLocal WeakReference to a map of ViewGroups to ArrayLists of Transitions.

ThreadLocal is a reference to a field, such that each Thread which access a thread-local variable via the ThreadLocals get and set methods have their own copy of the variable.

A WeakReference is a reference which is not strong enough to prevent garbage collection.

Finally we have the map which may contain the reference to our Activity's DecorView.

removeActivityFromTransitionManager checks that both the ThreadLocal's WeakReference is not null, and that the WeakReference's ArrayMap is not null before checking the ArrayMap for the DecorView and removing it if is present.

onToolbarBackPressed

onToolbarBackPressed is used solely to reduce clutter in other Activity classes.

When a view is declared in XML, its OnClickListener can be set with the onClick attribute.

Rather than adding this method in all of the Activities, it can be added to BaseActivity.

```

<ImageButton
    android:id="@+id/back_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@android:color/transparent"
    android:src="@drawable/ic_arrow_back"
    android:onClick="onToolbarBackPressed"/>

```

The back button shown in each Toolbar then references this method, which calls the Activity method onBackPressed to perform the same behaviour as pressing the navigation back key.

NetworkImageView

The NetworkImageView is a subclass of ImageView used to asynchronously load images from a given URL and display them once they have finished downloading.

NetworkImageView.java

```
package com(tpb.projects.common;

import android.content.Context;
import android.content.res.TypedArray;
import android.support.annotation.IdRes;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v7.widget.AppCompatImageView;
import android.text.TextUtils;
import android.util.AttributeSet;
import android.view.ViewGroup;

import com.androidnetworking.error.ANError;
import com.androidnetworking.internal.ANImageLoader;
import com(tpb.projects.R;

/**
 * Created by theo on 01/04/17.
 */

public class NetworkImageView extends AppCompatImageView {

    private String mUrl;

    @IdRes private int mDefaultImageResId;
    @IdRes private int mErrorImageResId;

    private ANImageLoader.ImageContainer mImageContainer;

    public NetworkImageView(Context context) {
        super(context);
    }

    public NetworkImageView(Context context, @Nullable AttributeSet attrs) {
        super(context, attrs);
        if(attrs != null) init(attrs, 0);
    }

    public NetworkImageView(Context context, @Nullable AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        if(attrs != null) init(attrs, defStyleAttr);
    }

    private void init(AttributeSet attrs, int defStyleAttr) {
        final TypedArray array = getContext()
            .obtainStyledAttributes(attrs, R.styleable.NetworkImageView,
        defStyleAttr, 0);
        mDefaultImageResId = array
            .getResourceId(R.styleable.NetworkImageView_default_image_resource,
```

```
        R.drawable.ic_avatar_default
    );
    mErrorImageResId = array

.getResourceId(R.styleable.NetworkImageView_error_image_resource, 0);
    array.recycle();
}

public void setImageUrl(@NonNull String url) {
    mUrl = url;
    loadImage(false);
}

public void setDefaultImageResId(@IdRes int defaultImage) {
    mDefaultImageResId = defaultImage;
}

public void setErrorImageResId(@IdRes int errorImage) {
    mErrorImageResId = errorImage;
}

private void loadImage(final boolean isInLayoutPass) {
    final int width = getWidth();
    final int height = getHeight();

    boolean wrapWidth = false, wrapHeight = false;
    if(getLayoutParams() != null) {
        wrapWidth = getLayoutParams().width ==
ViewGroup.LayoutParams.WRAP_CONTENT;
        wrapHeight = getLayoutParams().height ==
ViewGroup.LayoutParams.WRAP_CONTENT;
    }

    if(width == 0 && height == 0 && !(wrapWidth && wrapHeight)) {
        //Can't display the image as not size
        return;
    }

    if(TextUtils.isEmpty(mUrl)) {
        if(mImageContainer != null) {
            mImageContainer.cancelRequest();
            mImageContainer = null;
        }
        displayDefaultImage();
        return;
    }

    if(mImageContainer != null && mImageContainer.getRequestUrl() != null)
{
        if(mImageContainer.getRequestUrl().equals(mUrl)) {
            return;
        } else {
            mImageContainer.cancelRequest();
        }
    }

    final int maxWidth = wrapWidth ? 0 : width;
    final int maxHeight = wrapHeight ? 0 : height;
```

```
final ScaleType scaleType = getScaleType();
mImageContainer = ANImageLoader.getInstance().get(mUrl,
    new ANImageLoader.ImageListener() {
        @Override
        public void onResponse(final ANImageLoader.ImageContainer
response,
                           boolean isImmediate) {
            if(isImmediate && isInLayoutPass) {
                post(() -> onResponse(response, false));
                return;
            }

            if(response.getBitmap() != null) {
                setImageBitmap(response.getBitmap());
            } else if(mDefaultImageResId != 0) {
                displayDefaultImage();
            }
        }

        @Override
        public void onError(ANError error) {
            if(mErrorImageResId != 0) {
                setImageResource(mErrorImageResId);
            }
        }
    }, maxWidth, maxHeight, scaleType
);

}

public void resetImage() {
    setImageDrawable(null);
    displayDefaultImage();
}

private void displayDefaultImage() {
    if(getDrawable() != null) return; //Drawable has been set manually
    if(mDefaultImageResId != 0) {
        setImageResource(mDefaultImageResId);
    } else {
        setImageBitmap(null);
    }
}

@Override
protected void onLayout(boolean changed, int left, int top, int right, int
bottom) {
    super.onLayout(changed, left, top, right, bottom);
    loadImage(true);
}

@Override
protected void drawableStateChanged() {
    super.drawableStateChanged();
    invalidate();
}
}
```

The `NetworkImageView` has three instance variables; the URL to be loaded as well as two resource identifiers for the loading and error states.

When the `NetworkImageView` is instantiated it checks the `AttributeSet` for resource identifiers set in XML.

NetworkImageView.java

```
private void init(AttributeSet attrs, int defStyleAttr) {
    final TypedArray array = getContext()
        .obtainStyledAttributes(attrs, R.styleable.NetworkImageView,
defStyleAttr, 0);
    mDefaultImageResId = array

.getResourceId(R.styleable.NetworkImageView_default_image_resource,
        R.drawable.ic_avatar_default
    );
    mErrorImageResId = array

.getResourceId(R.styleable.NetworkImageView_error_image_resource, 0);
    array.recycle();
}
```

The `loadImage` method is responsible for loading and displaying the image.

NetworkImageView.java

```
private void loadImage(final boolean isInLayoutPass) {
    final int width = getWidth();
    final int height = getHeight();

    boolean wrapWidth = false, wrapHeight = false;
    if(getLayoutParams() != null) {
        wrapWidth = getLayoutParams().width ==
ViewGroup.LayoutParams.WRAP_CONTENT;
        wrapHeight = getLayoutParams().height ==
ViewGroup.LayoutParams.WRAP_CONTENT;
    }

    if(width == 0 && height == 0 && !(wrapWidth && wrapHeight)) {
        //Can't display the image as not size
        return;
    }

    if(TextUtils.isEmpty(mUrl)) {
        if(mImageContainer != null) {
            mImageContainer.cancelRequest();
            mImageContainer = null;
        }
        displayDefaultImage();
        return;
    }

    if(mImageContainer != null && mImageContainer.getRequestUrl() != null)
{
        if(mImageContainer.getRequestUrl().equals(mUrl)) {
            return;
        } else {
            mImageContainer.cancelRequest();
        }
    }
}
```

```

    }

    final int maxWidth = wrapWidth ? 0 : width;
    final int maxHeight = wrapHeight ? 0 : height;

    final ScaleType scaleType = getScaleType();
    mImageContainer = ANImageLoader.getInstance().get(mUrl,
        new ANImageLoader.ImageListener() {
            @Override
            public void onResponse(final ANImageLoader.ImageContainer
response,
                                boolean isImmediate) {
                if(isImmediate && isInLayoutPass) {
                    post(() -> onResponse(response, false));
                    return;
                }

                if(response.getBitmap() != null) {
                    setImageBitmap(response.getBitmap());
                } else if(mDefaultImageResId != 0) {
                    displayDefaultImage();
                }
            }

            @Override
            public void onError(ANError error) {
                if(mErrorImageResId != 0) {
                    setImageResource(mErrorImageResId);
                }
            }
        },
        maxWidth, maxHeight, scaleType
    );
}

```

The first check performed in `loadImage` is whether the image can actually be drawn.

The `View` width and height are collected, before the `LayoutParams` are checked to determine whether the `NetworkImageView` is of a fixed size, or should expand to the size of its image.

If the `NetworkImageView` has 0 width and height, and doesn't wrap in either direction, then `loadImage` returns as it cannot display the image.

The second check is whether the URL is empty.

If a null or empty URL is passed, any current request is cancelled, and the default image is set.

The third check is whether the URL which is currently being loaded, or has been loaded is the same as the URL passed to the `NetworkImageView`.

If the URL is already being loaded, `loadImage` returns, otherwise it cancels the current request in preparation for loading a new image.

Finally, the maximum width and height of the `NetworkImageView` are determined and the image is loaded.

The `onResponse` callback has two arguments, the response, and the `isImmediate` flag which indicates whether the response was returned from cache.

If both `isImmediate` and `isInLayoutPass` are true, the `NetworkImageView` has not yet been properly drawn and the image cannot yet be displayed.

In this case, a `post` call is made, which will add a runnable to the UI thread `MessageQueue` for execution once all other work on the UI thread is complete.

When the `NetworkImageView` is able to set the image, it checks whether the `Bitmap` is non null, and sets either the `Bitmap` or the default resource accordingly.

The `NetworkImageView` is used to display user avatars, in objectives 2.a.ii, 2.e, 2.f, 3.d.iii, 3.e.ii.3, 3.e.ii.6, 4.a.v, 4.a.vii, 4.a.xi, 4.b.i.1, 5.a.iii, 5.c.i.1, and 6.d.ii.6.

ViewSafeFragment

The lifecycle of a Fragment is different than that of an Activity.

The first method called is `onAttach` when the Fragment is attached to an Activity.

Next, `onCreate` is called, which might be used to initialise non view-dependent logic.

Most importantly, `onCreateView` is called.

`onCreateView` returns a `View` object and must create the layout for the Fragment.

As the Activity is created prior to the Fragment Views being created, the Fragment may have data to bind before it has to Views to bind them to.

In order to ensure that the Fragment doesn't attempt to bind data to a null view, the Fragment has a flag set when its Views are created, and set back when its Views are destroyed.

ViewSafeFragment.java

```
package com(tpb.projects.common;

import android.support.v4.app.Fragment;

/**
 * Created by theo on 11/04/17.
 */

public class ViewSafeFragment extends Fragment {

    protected boolean mAreViewsValid;

    protected boolean areViewsValid() {
        return mAreViewsValid && getActivity() != null;
    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();
        mAreViewsValid = false;
    }

}
```

In a concrete instance of `ViewSafeFragment` the `mAreViewsValid` flag should be set after inflation in `onCreateView` and used to check view validity before performing any binding.

FabHideScrollListener

A `FloatingActionButton` is a button which, as its name suggests, floats over other `Views`. It is often positioned in the bottom right of a screen to provide a button for the primary action.

The floating nature of the button can cause problems when it is displayed over a `RecyclerView` as it obscures the bottom most item.

To solve this the `FloatingActionButton` should hide when the `RecyclerView` scrolls down, and show again when it scrolls back up.

FabHideScrollListener.java

```
package com(tpb.projects.common.fab;

import android.support.v7.widget.RecyclerView;

/**
 * Created by theo on 25/03/17.
 */

public class FabHideScrollListener extends RecyclerView.OnScrollListener {

    private FloatingActionButton mFab;

    public FabHideScrollListener(FloatingActionButton fab) {
        mFab = fab;
    }

    public void setFab(FloatingActionButton fab) {
        mFab = fab;
    }

    @Override
    public void onScrolled(RecyclerView recyclerView, int dx, int dy) {
        super.onScrolled(recyclerView, dx, dy);
        if(mFab == null) return;
        if(dy > 10) {
            mFab.hide(true);
        } else if(dy < -10) {
            mFab.show(true);
        }
    }
}
```

The `FabHideScrollListener` extends `RecyclerView.OnScrollListener` and overrides `onScrolled`, checking if the change in the `y` value is sufficient to indicate scrolling.

SimpleTextChangeWatcher

The `SimpleTextChangeWatcher` is a simplified abstract implementation of `TextWatcher` which forwards the `onTextChanged` call to `textChanged` without any parameters.

Rather than requiring the implementation of `beforeTextChanged`, `onTextChanged`, and `afterTextChanged` when the logic only needs to know that the text has changed.

SimpleTextChangeWatcher.java

```
package com(tpb.projects.util.input;

import android.text.Editable;
import android.text.TextWatcher;

/**
 * Created by theo on 24/02/17.
 * Simplified TextWatcher which updates onTextChanged
 */

public abstract class SimpleTextChangeWatcher implements TextWatcher {

    @Override
    public final void beforeTextChanged(CharSequence s, int start, int count,
    int after) {
    }

    @Override
    public final void afterTextChanged(Editable s) {
    }

    @Override
    public final void onTextChanged(CharSequence s, int start, int before, int
    count) {
        textChanged();
    }

    /**
     * Called onTextChanged
     */
    public abstract void textChanged();
}
```

KeyBoardVisibilityChecker

A long running gripe with Android's text input system is that there is no standard way to detect whether the keyboard is currently visible, or listen for when its visibility changes.

This functionality is achieved by listening for changes on the `viewTreeObserver` of the root `view` of an `Activity` and comparing it to the display frame size.

KeyBoardVisibilityChecker.java

```
package com(tpb.projects.util.input;

import android.graphics.Rect;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.view.View;

/**
 * Created by theo on 21/02/17.
 * Utility for listening for keyboard state
 */

public class KeyBoardVisibilityChecker {

    private boolean mIsKeyboardOpen = false;

    public KeyBoardVisibilityChecker(@NonNull View content) {
        this(content, null);
    }

    public KeyBoardVisibilityChecker(@NonNull View content, @Nullable
KeyBoardVisibilityListener listener) {
        content.getViewTreeObserver().addOnGlobalLayoutListener(() -> {
            final Rect r = new Rect();
            content.getWindowVisibleDisplayFrame(r);
            final int screenHeight = content.getRootView().getHeight();

            // r.bottom is the position above soft keypad or device button.
            // if keypad is shown, the r.bottom is smaller than that before.
            final int kbHeight = screenHeight - r.bottom;

            if(kbHeight > screenHeight * 0.15) { // 0.15 ratio is perhaps
enough to determine keypad height.
                mIsKeyboardOpen = true;
                if(listener != null) listener.keyboardShown();
            } else {
                mIsKeyboardOpen = false;
                if(listener != null) listener.keyboardHidden();
            }
        });
    }

    public boolean isKeyboardOpen() {
        return mIsKeyboardOpen;
    }

    /**
     * Interface for listening to {@link KeyBoardVisibilityChecker}
     */
    public interface KeyBoardVisibilityListener {

        void keyboardShown();

        void keyboardHidden();

    }
}
```

```
}
```

The KeyBoardVisibilityChecker adds an `onGlobalLayoutListener` which checks the size of the window with `getWindowVisibleDisplayFrame`.

This method applies the dimensions of “the overall visible display size in which the window this view is attached to has been positioned in” to a given `Rect`. When the keyboard is shown, the root content layout is pushed upward and resized. The bottom of the content layout is therefore at the same position as the top of the keyboard.

Next the screen height is found from the height of the root `View` returned by the content layout.

If the calculated height is greater than 15% of the screen, it can be assumed that the keyboard is showing.

The `KeyBoardVisibilityListener` is an interface which can be used to listen for changes in keyboard visibility.

Utility methods

The `Util` class contains numerous utility methods for formatting and finding array indices.

The `indexof` methods are used to find the index of a value within an array of integers, strings, pairs, or a generic type.

Each method performs a linear search for the key item, returning -1 if it does not exist.

The `formatBytes` and `formatKB` methods are used to format a number of bytes or kilobytes into a 2 decimal place string representation of the highest unit suffix. The next method, `formatDateLocally` is used to format a date in the expected manner for the device locale.

`isNotNullOrEmpty` is used to check that a string is not null, not empty, and not a “null” string returned from a `JSONObject`.

Finally, the `insertString` methods are used to insert a string at the currently selected position in an `EditText`, before moving the cursor to the end of the inserted string, or to a provided offset.

User Interface utility methods

The `UI` class contains numerous helper methods for performing unit conversions as well as helping with view animations.

UI.java

```
package com(tpb.projects.util;

import android.animation.ArgbEvaluator;
```

```
import android.animation.ObjectAnimator;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.res.Resources;
import android.graphics.drawable.BitmapDrawable;
import android.support.annotation.ColorInt;
import android.support.annotation.Dimension;
import android.support.annotation.NonNull;
import android.support.annotation.Px;
import android.support.v4.util.Pair;
import android.util.TypedValue;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;
import android.view.WindowManager;
import android.view.animation.Transformation;
import android.widget.ImageView;

import com(tpb.projects.R;
import com(tpb.projects.common.CircularRevealActivity);

/**
 * Created by theo on 16/12/16.
 */

public class UI {

    private UI() {}

    /**
     * Sets the expansion point for a {@link CircularRevealActivity} Intent
     * to the midpoint of a View
     *
     * @param i      The Intent to launch a {@link CircularRevealActivity}
     */
    public static void setViewPositionForIntent(Intent i, View view) {
        final int[] pos = new int[2];
        view.getLocationOnScreen(pos);
        pos[0] += view.getWidth() / 2;
        pos[1] += view.getHeight() / 2;
        i.putExtra(view.getContext().getString(R.string.intent_position_x),
pos[0]);
        i.putExtra(view.getContext().getString(R.string.intent_position_y),
pos[1]);
    }

    /**
     * @param context Required to get string resource values
     * @param i      The Intent to launch a {@link CircularRevealActivity}
     */
    public static void setClickPositionForIntent(Context context, Intent i,
float[] pos) {
        i.putExtra(context.getString(R.string.intent_position_x), (int)
pos[0]);
    }
}
```

```
i.putExtra(context.getString(R.string.intent_position_y), (int)
pos[1]);
}

public static void expand(final View v) {
    v.measure(ViewGroup.LayoutParams.MATCH_PARENT,
ViewGroup.LayoutParams.WRAP_CONTENT);
    final int targetHeight = v.getMeasuredHeight();

    // Older versions of android (pre API 21) cancel animations for views
with a height of 0.
    v.setLayoutParams().height = 1;
    v.setVisibility(View.VISIBLE);
    final android.view.animation.Animation a = new
android.view.animation.Animation() {
        @Override
        protected void applyTransformation(float interpolatedTime,
Transformation t) {
            v.setLayoutParams().height = interpolatedTime == 1
                ? ViewGroup.LayoutParams.WRAP_CONTENT
                : (int) (targetHeight * interpolatedTime);
            v.requestLayout();
        }

        @Override
        public boolean willChangeBounds() {
            return true;
        }
    };
}

// 1dp/ms
a.setDuration(
    (int) (targetHeight /
v.getContext().getResources().getDisplayMetrics().density));
    v.startAnimation(a);
}

public static void collapse(final View v) {
    final int initialHeight = v.getMeasuredHeight();

    android.view.animation.Animation a = new
android.view.animation.Animation() {
        @Override
        protected void applyTransformation(float interpolatedTime,
Transformation t) {
            if(interpolatedTime == 1) {
                v.setVisibility(View.GONE);
            } else {
                v.setLayoutParams().height = initialHeight - (int)
(initialHeight * interpolatedTime);
                v.requestLayout();
            }
        }

        @Override
        public boolean willChangeBounds() {
            return true;
        }
    };
}
```

```
// 1dp/ms
a.setDuration(
    (int) (initialHeight /
v.getContext().getResources().getDisplayMetrics().density));
    v.startAnimation(a);
}

/**
 * Fades the background color of a View from original to flash and back
 *
 * @param view      The view to flash
 * @param original The current background color
 * @param flash     The color to fade to
 */
public static void flashViewBackground(View view, @ColorInt int original,
@ColorInt int flash) {
    final ObjectAnimator colorFade = ObjectAnimator.ofObject(
        view,
        "backgroundColor",
        new ArgbEvaluator(),
        original,
        flash
    );
    colorFade.setDuration(300);
    colorFade.setRepeatMode(ObjectAnimator.REVERSE);
    colorFade.setRepeatCount(1);
    colorFade.start();
}

@Dimension
public static float dpFromPx(final float px) {
    return px / Resources.getSystem().getDisplayMetrics().density;
}

@Px
public static int pxFromDp(final float dp) {
    return Math.round(dp *
Resources.getSystem().getDisplayMetrics().density);
}

@Px
public static int pxFromDp(final int dp) {
    return (int) (dp * Resources.getSystem().getDisplayMetrics().density);
}

@Px
public static int pxFromSp(final float sp) {
    return (int) TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_SP, sp,
        Resources.getSystem().getDisplayMetrics()
    );
}

public static void setStatusBarColor(Window window, @ColorInt int color) {
    // clear FLAG_TRANSLUCENT_STATUS flag:
    window.clearFlags(WindowManager.LayoutParams.FLAG_TRANSLUCENT_STATUS);
    // add FLAG_DRAW_SYSTEM_BAR_BACKGROUNDS flag to the window
}
```

```

window.addFlags(WindowManager.LayoutParams.FLAG_DRAWABLE_SYSTEM_BAR_BACKGROUNDS);
    // finally change the color
    window.setStatusBarColor(color);
}

/**
 * Checks whether the device has a navigation bar, and if so returns a
pair
 * for {@see ActivityOptionsCompat#makeSceneTransition}
 *
 * @param activity Any activity in which to find the navigation bar
 * @return The navigation bar view view pair, or an empty view pair
 */
public static Pair<View, String>
getSafeNavigationBarTransitionPair(@NonNull Activity activity) {
    final View nav =
activity.findViewById(android.R.id.navigationBarBackground);
    return nav == null ?
        Pair.create(new View(activity), "not_for_transition") :
        Pair.create(nav,
Window.NAVIGATION_BAR_BACKGROUND_TRANSITION_NAME);
}

public static void setDrawableForIntent(@NonNull ImageView iv, @NonNull
Intent i) {
    if(iv.getDrawable() instanceof BitmapDrawable) {
        i.putExtra(iv.getResources().getString(R.string.intent_drawable),
((BitmapDrawable) iv.getDrawable()).getBitmap()
    );
    }
}
}

```

The first two methods, `setViewPositionForIntent` and `setClickPositionForIntent` are used when passing the position of a view or the position of a click to a new Activity, allowing it to launch from a point.

The `expand` and `collapse` methods are used to animate views. `expand` animates a view from no height to its measured height, and `collapse` shrinks a view from its measured height to no height before hiding it.

`flashViewBackground` is a method used to fade the background colour of a view from its original colour to a highlight colour, and back again. This method can be used to highlight an important view, such as when jumping to a search result, such as in objective 6.i.iii.

The next four methods are used for unit conversion, converting pixels to density independent pixels, as well as converting density independent pixels or scale independent pixels to pixels.

As its name suggest, `setStatusBarColor` is used to set the status bar colour for a Window, which is required if the Activity uses a theme with transparency. Finally, `getSafeNavigationBarTransitionPair` is a utility for shared element transitions.

When a view in a RecyclerView is used in a shared element transition between two Activities, the View is drawn through the `viewOverlay` layer, which draws above the software navigation bar.

If a View is partially below the navigation bar it will jump in elevation to display above the navigation bar, and jump back under it on the return transition.

In order to prevent this jumpy transition, the navigation bar can be added to the transition, resulting in it being drawn in the `viewOverlay` above the transitioning View.

The navigation bar can be found by searching an Activity for the id `android.R.id.navigationBarBackground`, however it may not exist.

Many devices, notably Samsung phones, do not use software navigation keys, and the null view returned from `findViewById` would result in a crash.

In order to prevent this, a Pair containing an empty view instance with an unused key is returned if the navigation bar does not exist.

Logging

As explained in the analysis, logs are printed with the `Log` class and are a useful method of debugging. However, log messages should only be shown in the debug variant of the application.

The `Logger` class wraps the methods in `Log` with checks for the `BuildConfig` flag `IS_IN_DEBUG`.

Logger.java

```
package com(tpb.projects.util;

import android.annotation.SuppressLint;
import android.util.Log;

import java.io.IOException;

import okhttp3.Interceptor;
import okhttp3.Request;
import okhttp3.Response;

/**
 * Created by theo on 11/03/17.
 */

public class Logger {
    private static final boolean DEBUG =
com(tpb.projects.BuildConfig.IS_IN_DEBUG;

    public static void logLong(String TAG, String s) {
```

```
    if(s.length() > 4000) {
        Log.d(TAG, s.substring(0, 4000));
        logLong(TAG, s.substring(4000));
    } else
        Log.d(TAG, s);
}

public static void v(String tag, String msg) {
    if(DEBUG) Log.v(tag, msg);
}

public static void v(String tag, String msg, Throwable tr) {
    if(DEBUG) Log.v(tag, msg, tr);
}

public static void d(String tag, String msg) {
    if(DEBUG) Log.d(tag, msg);
}

public static void d(String tag, String msg, Throwable tr) {
    if(DEBUG) Log.d(tag, msg, tr);
}

public static void w(String tag, String msg) {
    if(DEBUG) Log.w(tag, msg);
}

public static void w(String tag, String msg, Throwable tr) {
    if(DEBUG) Log.w(tag, msg, tr);
}

public static void w(String tag, Throwable tr) {
    if(DEBUG) Log.w(tag, tr);
}

public static void i(String tag, String msg) {
    if(DEBUG) Log.i(tag, msg);
}

public static void i(String tag, String msg, Throwable tr) {
    if(DEBUG) Log.i(tag, msg, tr);
}

public static void e(String tag, String msg) {
    if(DEBUG) Log.e(tag, msg);
}

public static void e(String tag, String msg, Throwable tr) {
    if(DEBUG) Log.e(tag, msg, tr);
}

public static class LoggingInterceptor implements Interceptor {

    private static final String TAG =
LoggingInterceptor.class.getSimpleName();

    @SuppressLint("DefaultLocale")
    @Override
    public Response intercept(Chain chain) throws IOException {
```

```

        final Request request = chain.request();
        final long ts = System.nanoTime();
        Logger.i(TAG, String.format("Sending request %s on %s%n%s",
                request.url(), chain.connection(), request.headers()
        ));

        final Response response = chain.proceed(request);

        Logger.i(TAG, String.format("Received response for %s in
%.1fms%n%s",
                response.request().url(), (System.nanoTime() - ts) / 1e6d,
        response.headers()
        ));

        return response;
    }
}

}

```

The Logger class also contains the LoggingInterceptor class which is a network interceptor used to log all network request made throughout the app.

The LoggingInterceptor is added in the ProjectsApplication class.

It produces two log messages for each call, the first details the request being sent, for example a request to the notifications API:

```

Sending request https://api.github.com/notifications on
Connection{api.github.com:443, proxy=DIRECT@hostAddress=api.github.com/192.30.253.116:443
cipherSuite=TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 protocol=http/1.1}
Accept: application/vnd.github.v3+json
Authorization: token
an_authorization_token
Cache-Control: no-store
Host: api.github.com
Connection: Keep-Alive
Accept-Encoding: gzip
User-Agent: okhttp/3.6.0

```

and the received response:

```

Received response for https://api.github.com/notifications in 419.7ms
Server: GitHub.com
Date: Tue, 11 Apr 2017 23:36:57 GMT
Content-Type: application/json;
charset=utf-8
Content-Length: 2
Status: 200 OK
X-RateLimit-Limit: 5000
X-RateLimit-Remaining: 4959
X-RateLimit-Reset: 1491955996
Cache-Control: private, max-age=60, s-
maxage=60
Vary: Accept, Authorization, Cookie, X-
GitHub-OTP
ETag:
"3ef243829743ac436b782dbd8981e769"
X-Poll-Interval: 60
X-OAuth-Scopes: gist, repo, user

```

```

repo X-Accepted-OAuth-Scopes: notifications,
the_client_id_of_the_app X-OAuth-Client-Id:
format=json X-GitHub-Media-Type: github.v3;
               Access-Control-Expose-Headers: ETag,
Link, X-GitHub-OTP, X-RateLimit-Limit, X-RateLimit-Remaining, X-RateLimit-
Reset, X-OAuth-Scopes, X-Accepted-OAuth-Scopes, X-Poll-Interval
               Access-Control-Allow-Origin: *
               Content-Security-Policy: default-src
'none' Strict-Transport-Security: max-
age=31536000; includeSubdomains; preload
               X-Content-Type-Options: nosniff
               X-Frame-Options: deny
               X-XSS-Protection: 1; mode=block
               Vary: Accept-Encoding
               X-Served-By:
07ff1c8a09e44b62e277fae50a1b1dc4 X-GitHub-Request-Id:
A8DC:2A61:2B933E:372939:58ED6898

```

Analytics

In order to help debug crashes which happen when not connected to a computer, I have chosen to incorporate Google's FireBase analytics service to log crashes and other events.

If a crash occurs, the information about the crash is exported and can then be used to debug the issue later.

Each issue shows the version codes for which it occurred, as well as more detailed information about the circumstances of the crash.

Instances	Users	Versions	Issue	Stack trace
2	1	1123	java.lang.NullPointerException ProjectActivity.java - Line 171 Fatal	com(tpb.projects.project.ProjectActivity.lambda\$onCreate. com(tpb.projects.project.ProjectActivity\$\$Lambda\$3.can.

The crash information contains the full stack trace as well as information about the device on which the crash occurred.

Data			
User	Performance	Device	Error
Country Code: –	VM free: 3.01MB	Manufacturer: HTC	Date: 22 Mar 2017, 14:58:25
Carrier Code: –	VM total: 10.92MB	Model: One	App Version: 1123 (1.1.0)
Operator: T-Mobile	VM max.: 192MB	Board: M7	
Locale: en-GB	Battery level: 42%	Android API: 25	
	Charging state: Unplugged	Android OS: 7.1.1	
	Connection State: Wwan	Brand: Htc	
		RAM: 1.78GB	
		Orientation: Portrait	
		Proximity to user: –	

Automated build system

As explained in the analysis section, continuous integration tools are often integrated with GitHub to build projects as they are committed and add statuses to each commit.

I have used the Travis build system to build the project on each commit and pull request, adding a status to each commit allowing me to see immediately if a build failed.

The system is set up with a configuration file names ".travis.yml".

```
language: android
jdk: oraclejdk8

sudo: true

# Handle git submodules yourself
git:
  submodules: false
# Use sed to replace the SSH URL with the public URL, then initialize
submodules
before_install:
  - sed -i 's/git@github.com:/https:\/\/github.com\/\//' .gitmodules
  - git submodule update --init --recursive

android:
  components:
    - platform-tools
    - tools

  # The BuildTools version
  - build-tools-25.0.2

  # The SDK version
  - android-25

  # Additional components
  - extra-google-m2repository
  - extra-android-m2repository
  - addon-google_apis-google-25
```

```
- sys-img-armeabi-v7a-android-23

licenses:
    - 'android-sdk-preview-license-.+'
    - 'android-sdk-license-.+'
    - 'google-gdk-license-.+'
    - ".+"

script: ./gradlew build

cache:
  directories:
    - $HOME/.gradle/caches/
    - $HOME/.gradle/wrapper/
    - $HOME/.android/build-cache
```

This config file updates any included Git submodules, installs the correct build tools and SDK version, and then runs a gradle build before caching the gradle cache.

Link handling

In order to receive `Intents` when a user attempts to open a link to GitHub, the application must register an intent filter in its manifest.

The intent filter system allows specifying a host and a scheme to capture. It also allows specifying a path pattern to match the path against.

Unfortunately the pattern matching system is very limited.

An asterisk, `"*"`, matches a sequence of 0 or more occurrences of the character immediately preceding it.

A period, `". "`, followed by an asterisk, `"*"`, matches any sequence of 0 or more characters.

This is of no use when matching GitHub URLs.

Instead, the application must match all GitHub URLs and reject those which it cannot handle by allowing the user to choose another application.

The manifest entry for the `Interceptor` Activity is therefore

```
<activity
    android:name=".flow.Interceptor"
    android:theme="@android:style/Theme.NoDisplay">
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>

        <data
            android:host="github.com"
            android:scheme="http"/>
        <data
            android:host="github.com"
            android:scheme="https"/>

        <category android:name="android.intent.category.DEFAULT"/>
        <category android:name="android.intent.category.BROWSABLE"/>
    </intent-filter>
</activity>
```

The `NoDisplay` theme is specified, as the `Activity` should not display any content.

The `Interceptor` Activity should also ensure that it calls `finish` before it exits the `onCreate` method.

The intent filter specifies both schema for github.com, and adds the `DEFAULT` category, allowing the application to be chosen as the default for a particular URL, and the `BROWSABLE`

category which allows the application to be started by a web-browser.

Interceptor.java

```

package com(tpb.projects.flow;

import android.app.Activity;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.content.pm.ResolveInfo;
import android.os.Bundle;
import android.os.Parcelable;

import com(tpb.github.data.models.Notification;
import com(tpb.projects.R;
import com(tpb.projects.commits.CommitActivity;
import com(tpb.projects.issues.IssueActivity;
import com(tpb.projects.milestones.MilestonesActivity;
import com(tpb.projects.notifications.NotificationIntentService;
import com(tpb.projects.project.ProjectActivity;
import com(tpb.projects.repo.RepoActivity;
import com(tpb.projects.repo.content.ContentActivity;
import com(tpb.projects.repo.content.FileActivity;
import com(tpb.projects.user.UserActivity;

import java.util.ArrayList;
import java.util.List;

/**
 * Created by theo on 01/01/17.
 */

public class Interceptor extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        final Intent l = getIntent();
        if(l.getAction().equals(Intent.ACTION_VIEW) &&
           l.getData() != null &&
           "github.com".equals(l.getData().getHost())) {
            if(l.getStringExtra("notif")) {
                final Notification notif = l.getParcelableExtra("notif");

                startService(NotificationIntentService.generateBroadcastDismissIntent(this,
notif));
            }
            final List<String> segments = l.getData().getPathSegments();
            if(segments.size() == 0) {
                fail();
            } else if(segments.size() == 1) {
                final Intent u = new Intent(Interceptor.this,
UserActivity.class);
                u.putExtra(getString(R.string.intent_username),
segments.get(0));
                startActivity(u);
                overridePendingTransition(R.anim.slide_up, R.anim.none);
                finish();
            } else {
                final Intent i = new Intent();
                i.putExtra(getString(R.string.intent_repo), segments.get(0) +
"/" + segments.get(1));
                switch(segments.size()) {

```

```

        case 2: //Repo
            i.setClass(Interceptor.this, RepoActivity.class);
            break;
        case 3:
            if("projects".equals(segments.get(2))) {
                i.setClass(Interceptor.this, RepoActivity.class);
                i.putExtra(getString(R.string.intent_pager_page),
                           RepoActivity.PAGE_PROJECTS
                );
            } else if("issues".equals(segments.get(2))) {
                i.setClass(Interceptor.this, RepoActivity.class);
                i.putExtra(getString(R.string.intent_pager_page),
                           RepoActivity.PAGE_ISSUES
                );
            } else if("milestones".equals(segments.get(2))) {
                i.setClass(Interceptor.this,
MilestonesActivity.class);
            } else if("commits".equals(segments.get(2))) {
                i.setClass(Interceptor.this, RepoActivity.class);
                i.putExtra(getString(R.string.intent_pager_page),
                           RepoActivity.PAGE_COMMITS
                );
            }
            break;
        case 4:
            if("projects".equals(segments.get(2))) {
                i.setClass(Interceptor.this,
ProjectActivity.class);
            }

            i.putExtra(getString(R.string.intent_project_number),
                       safelyExtractInt(segments.get(3))
            );
            final String path = l.getDataString();
            final StringBuilder id = new StringBuilder();
            for(int j = path
                .indexOf('#',
path.indexOf(segments.get(3)) + 6; j < path
                .length(); j++) {
                if(path.charAt(j) >= '0' && path.charAt(j) <=
'9') {
                    id.append(path.charAt(j));
                }
            }
            final int cardId =
safelyExtractInt(id.toString());
            if(cardId != -1)
i.putExtra(getString(R.string.intent_card_id), cardId);
            } else if("issues".equals(segments.get(2))) {
                i.setClass(Interceptor.this, IssueActivity.class);
            }

            i.putExtra(getString(R.string.intent_issue_number),
                       safelyExtractInt(segments.get(3))
            );
        } else if("milestone".equals(segments.get(2))) {
            i.setClass(Interceptor.this,
MilestonesActivity.class);
        }

        i.putExtra(getString(R.string.intent_milestone_number),
                   safelyExtractInt(segments.get(3))

```

```

        );
    } else if("commit".equals(segments.get(2)) ||
"commits".equals(segments.get(2))) {
        i.setClass(Interceptor.this,
CommitActivity.class);
        i.putExtra(getString(R.string.intent_commit_sha),
segments.get(3));
    }
    break;
default:
    if("tree".equals(segments.get(2))) {
        i.setClass(Interceptor.this,
ContentActivity.class);
        final StringBuilder path = new StringBuilder();
        for(int j = 3; j < segments.size(); j++) {
            path.append(segments.get(j));
            path.append('/');
        }
        i.putExtra(getString(R.string.intent_path),
path.toString());
    } else if("blob".equals(segments.get(2))) {
        i.setClass(Interceptor.this, FileActivity.class);
        final StringBuilder path = new StringBuilder();
        for(int j = 2; j < segments.size(); j++) {
            path.append('/');
            path.append(segments.get(j));
        }
        i.putExtra(getString(R.string.intent_blob_path),
path.toString());
    }
}
if(i.getComponent() != null && i.getComponent().getClassName() != null) {
    startActivity(i);
    overridePendingTransition(R.anim.slide_up, R.anim.none);
    finish();
} else {
    fail();
}
}
} else {
    fail();
}
}

private static int safelyExtractInt(String possibleInt) {
try {
    return Integer.parseInt(possibleInt.replace("\\s+", ""));
} catch(NumberFormatException nfe) {
    return -1;
}
}

private void fail() {
try {
    startActivity(generateFailIntentWithoutApp());
} catch(Exception e) {
    e.printStackTrace();
} finally {

```

```

        finish();
    }

private Intent generateFailIntentWithoutApp() {
    try {
        final Intent intent = new Intent(getIntent().getAction());
        intent.setData(getIntent().getData());
        intent.addCategory(Intent.CATEGORY_BROWSABLE);
        intent.addCategory(Intent.CATEGORY_DEFAULT);

        final List<ResolveInfo> resolvedInfo = getPackageManager()
            .queryIntentActivities(intent,
PackageManager.MATCH_DEFAULT_ONLY);

        if(!resolvedInfo.isEmpty()) {
            final List<Intent> targetedShareIntents = new ArrayList<>();
            for(ResolveInfo resolveInfo : resolvedInfo) {
                final String packageName =
resolveInfo.activityInfo.packageName;
                if(!packageName.equals(getPackageName())) {
                    final Intent targetedShareIntent = new
Intent(getIntent().getAction());
                    targetedShareIntent.setData(getIntent().getData());
                    targetedShareIntent.setPackage(packageName);
                    targetedShareIntents.add(targetedShareIntent);
                }
            }
            final Intent chooserIntent =
Intent.createChooser(targetedShareIntents.remove(0),
                    getString(R.string.text_interceptor_open_with)
);
            if(targetedShareIntents.size() > 0) {
                chooserIntent.putExtra(Intent.EXTRA_INITIAL_INTENTS,
targetedShareIntents
                    .toArray(new
Parcelable[targetedShareIntents.size()]));
            }
            return chooserIntent;
        }
    } catch(Exception ignored) {
    }
    return null;
}
}

```

Possible paths

When onCreate is called, the first check is that the Intent action is ACTION_VIEW, the Intent has data, and the data host is github.com, if it is not fail is called. If the URL is from GitHub, the List of path segments are extracted. If the segments size is 0, the URL is github.com, and the Interceptor fails because it has not information to parse.

If the segments size is 1, the only possibility is a user account, so an Intent for the UserActivity is created and the first segment is added as an extra under the "username" key.

The Intent is then started and Interceptor finishes.

If there is more than one segment, there are many more possibilities.

Each of these possibilities contains a path to a repository. The URL is of the form "[github.com/user/repository/...](https://github.com/user/repository/)"

A new Intent is created, and the repository path is added to the Intent extras under the "repo" key.

A switch statement over the size of the segments List is then used to determine the Intent class.

2- The class is set to the repository Activity

3- The URL could be to a repository's projects, issues, milestones, or commits.

The third item in the List is checked, the class is set, and a page number is added to the Intent extras under the "page" key, allowing the Activity to scroll to a particular page on launch.

4- The URL could be a particular project, issue, milestone, or commit

- Projects

A projects URL should contain the integer value of the project's id as the fourth item in the List.

It may also contain a URL parameter for the id of the card reference in the project, this is an integer value after the string "#card-".

Both of these values are added to the Intent as extras and the class is set to the Project Activity.

- Issues

A path with a third segment of "issues" and a length of 4 refers to a particular issue.

The id of the issue is extracted from the fourth segment and added to the Intent, the class is set to the issue Activity.

- Milestone

The milestone id is extracted in the same way as an issue id, and the Activity class is set accordingly.

- Commit

The commit path does not contain a numeric id, but instead contains a SHA hash which is added as an extra to the Intent after the commit Activity is set.

Other-

Any values greater than 5 items refers to a file or directory in the repository files.

If the third path segment is "tree", the URL refers to a directory within a project.

In this case the tree path is built and added to the Intent as an extra.

If the third path segment is "blob", the URL refers to an individual file within the project. The blob path is built and added to the Intent as an extra.

Outside of the switch, the method checks that the component class has been set on the Intent.

If it has, the Intent is launched and Interceptor finishes.

Otherwise the fail method is called.

Showing a chooser

In the event that the Interceptor fails to determine an Activity to handle a particular URL, it should suggest other applications which might be able to handle the URL, objective 7.b.

In order to open a link in another app a chooser dialog should be shown. This is done by building an implicit Intent, not declaring the class to handle the Intent but allowing the user to choose from the available applications. This is handled in generateFailIntentWithoutApp.

The method creates a new Intent, with the same action as the Intent that launched Interceptor.

It then continues to add the data that was launched with Interceptor to the new Intent, and add the BROWSABLE and DEFAULT categories.

Next a List of ResolveInfo objects is collected, each of which contains information about an application which could handle the new Intent.

If the List is not empty, a new List of Intents is created, and a new Intent is created for each of the applications, using their package names, if the package name is not the name of this app.

Finally, the chooser Intent is created using the first Intent in the targetedShareIntents List, and the rest of the Intents are added to the chooser.

Markdown

As GitHub uses Markdown throughout its content, a method for displaying Markdown must be implemented before the creation of the rest of the user interface.

GitHub flavoured Markdown follows the CommonMark specification while extending it with extra features, however these features are not present in the Markdown returned from the GitHub API.

The simplest way to display Markdown would be to display the Markdown as HTML in a `WebView`, however the performance of a `WebView` is not acceptable when displaying multiple different sections of text.

Instead, the Markdown must be displayed in a `TextView`. The `TextView` has no native support for Markdown formatting, nor does it have direct support for HTML.

In order to display styled text without applying the styling to the entire text body, spans are used.

Spans

The `Spanned` interface is “the interface for text that has markup objects attached to ranges of it.”

There are three key interfaces and an abstract class for different types of span used in the `TextView`.

CharacterStyle

The abstract class `CharacterStyle` is used for spans which affect character level text formatting.

`CharacterStyle` has three methods.

The first, `updateDrawState` takes the `TextPaint` used to draw the `TextView` in order to allow changes to be made to the text styling.

The second and third methods are used when a single `CharacterStyle` needs to be applied to multiple different regions of a `Spanned`.

The `wrap` method takes a `CharacterStyle` which will actually perform the text manipulation. When a `CharacterStyle` is used, the `getUnderlying` method is called, which

should return `this` for instances which actually perform manipulation, or return

the wrapping `CharacterStyle` for spans which are only placeholders for actual implementations.

Implementations

of `CharacterStyle` include `BackgroundColorSpan`, `ForegroundColorSpan`, and `UnderLineSpan` which each change a single parameter on `TextPaint`.

ParagraphStyle

As its name suggest, `ParagraphStyle` affects paragraph level text formatting. The interface has no methods, and is instead used only as a marker to indicate that the span or span interface affects text at a wider level.

Direct span implementations of `ParagraphStyle` include `BulletSpan`, `QuoteSpan`, and `DrawableMarginSpan`, while numerous interfaces such as `AlignmentSpan` and `LeadingMarginSpan` also extend from it.

UpdateAppearance

The `UpdateAppearance` interface is for spans which affect character-level text "in a way that modifies their appearance when one is added or removed".

Implementations include `AbsoluteSizeSpan` which is used to scale text using either density independent pixels, or an absolute pixel size.

UpdateLayout

The `UpdateLayout` interface is an extension of `UpdateApperance` with the difference being that the modifications made by an implementation of `UpdateLayout` are such that a

text layout update is triggered when one is added or removed.

Implementations include `SuperscriptSpan`, `SubscriptSpan`, and `ImageSpan`.

ReplacementSpan

An important implementation of `CharacterStyle` is `ReplacementSpan`.

Rather than simply allowing updates to `TextPaint`, the `ReplacementSpan` is able to draw directly to the canvas.

The two abstract methods defined in `ReplacementSpan` are `getSize` and `draw`.

`getSize` takes a `Paint` object, the `CharSequence` displayed in the `TextView`, the start and end

positions of the span within the `CharSequence`, and the `FontMetrics` being used to draw the text. Is expected to return the width of the span.

`draw` takes the canvas, the `CharSequence` displayed in the `TextView`, the start and end positions of the span within the `CharSequence`, a `Paint` object, and the x, y, top, and bottom positions of the span on the canvas.

Implementing GitHub Markdown features

General strategy

None of the markings used are more than three characters in length, meaning that at any one time only the previous two characters need be retained.

The formatted markdown is to be appended to a `StringBuilder` as the array of characters in the markdown is formatted.

Each format method is to take the character array, current position, and the `StringBuilder` and attempt to append the formatted markdown to the `StringBuilder` before returning the new position to continue from in the character array.

Markdown.java

```
public static String formatMD(@NonNull String s, @Nullable String fullRepoPath, boolean linkUsernames) {
    final StringBuilder builder = new StringBuilder();
    char p = ' ';
    char pp = ' ';
    final char[] chars = s.toCharArray();
    for(int i = 0; i < chars.length; i++) {
        if(linkUsernames && chars[i] == '@' && isWhiteSpace(p)) {
            i = parseUsername(builder, chars, i);
        } else if(chars[i] == '#' && isWhiteSpace(p) && fullRepoPath != null) {
            i = parseIssue(builder, chars, i, fullRepoPath);
        } else if(chars[i] == ']' && p == '[' && pp == '!') {
            builder.setLength(builder.length() - 2);
            builder.append("!["No description"]");
        } else if(pp == '[' && (p == 'x' || p == 'X') && chars[i] == ']' && !isEscaped(chars, i-2)) {
            builder.setLength(builder.length() - 2);
            builder.append("\u2611"); // ballot box with check
        } else if(p == '[' && chars[i] == ']' && !isEscaped(chars, i-1)) {
//Closed box
            builder.setLength(builder.length() - 1);
            builder.append("\u2610"); // ballot box
        } else if(pp == '[' && p == ' ' && chars[i] == ']' && !isEscaped(chars, i-2)) { //Open box
            builder.setLength(builder.length() - 2);
            builder.append("\u2610");
        } else if(chars[i] == '(' && fullRepoPath != null) {
            builder.append("(");
            i = parseImageLink(builder, chars, i, fullRepoPath);
        } else if(chars[i] == ':' && !isEscaped(chars, i)) {
            i = parseEmoji(builder, chars, i);
        } else if(pp == '`' && p == '`' && chars[i] == '`') {
//We jump over the code block
            pp = ' ';
            p = ' ';
            int j = i;
            for(; j < chars.length; j++) {
                builder.append(chars[j]);
            }
        }
    }
    return builder.toString();
}
```

```

        if(pp == ' ' && p == ' ' && chars[j] == ' ') {
            i = j;
            p = ' ';
            break;
        } else {
            pp = p;
            p = chars[j];
        }
    }

} else {
    builder.append(chars[i]);
}
pp = p;
p = chars[i];
}
return builder.toString();
}

```

Markdown.java

```

private static boolean isWhiteSpace(char c) {
    //Space tab, newline, line tabulation, carriage return, form feed
    return c == ' ' || c == '\t' || c == '\n' || c == '\u000B' || c ==
'\r' || c == '\u000C';
}

```

Markdown.java

```

private static boolean isLineEnding(char[] cs, int i) {
    return i == cs.length - 1 || cs[i] == '\n' || cs[i] == '\r';
}

```

`isWhiteSpace` and `isLineEnding` are both utility methods. `isWhiteSpace` checks if the character is a space, a tab, a newline, a line tabulation character, a carriage return, or a form feed, while `isLineEnding` checks whether the character is a new line or carriage return or the position is the last in the character array.

The `formatMD` method takes the `Markdown` string, an optional repository path, and a flag for linking usernames.

Within each iteration, the first check is for usernames.

If the current character is the "@" key used for usernames, and the previous character is whitespace the position is jumped with a call to `parseUsername`.

If the current character is the "#" key used for issues, and the previous character is whitespace, and the repository name is non-null the position is jumped with a call to `parseIssue`.

The next three checks deal with formatting GitHub's checkbox lists to use ballot characters rather than non-formatted [] and [x] sequences.

The first check is for an upper or lowercase "x" contained between two square braces. The last two characters are removed from the builder and the unicode "ballot box with check" is added.

The second two checks are both for ballot boxes without checks, either written as "[]" or "[]".

This fulfills objective 9.b.vii.

The next check is for image links, which need to be parsed both in order to deal with links relative to the repository and to add spacing around them as they will be displayed as images.

The next check is for emojis, which are contained between two colons.

If there is a sequence of three backticks, every character from the backticks onward is appended without formatting until the next set of backticks is found.

Finally, if none of the above conditions apply, the character is appended to the builder.

At the end of each iteration the previous and previous previous characters are updated.

Username mentions

GitHub usernames are strings of text up to 39 characters in length, containing only alphanumeric characters and hyphens.

Markdown.java

```
private static int parseUsername(StringBuilder builder, char[] cs, int pos) {
    final StringBuilder nameBuilder = new StringBuilder();
    char p = ' ';
    for(int i = pos + 1; i < cs.length; i++) {
        if(((cs[i] >= 'A' && cs[i] <= 'Z') ||
           (cs[i] >= 'a' && cs[i] <= 'z') ||
           (cs[i] >= '0' && cs[i] <= '9') ||
           (cs[i] == '-' && p != '-')) &&
           i - pos <= 39 &&
           i != cs.length - 1) {
            nameBuilder.append(cs[i]);
            p = cs[i];
            //nameBuilder.length() > 0 stop us linking a single @
        } else if((isWhiteSpace(cs[i]) || i == cs.length - 1) && i - pos >
1) {
            if(i == cs.length - 1) {
                nameBuilder.append(cs[i]); //Otherwise we would miss the
last char of the name
            }
            builder.append("[@");
            builder.append(nameBuilder.toString());
            builder.append("]("https://github.com/");
            builder.append(nameBuilder.toString());
            builder.append(')');
            if(i != cs.length - 1) {
                builder.append(cs[i]); // We still need to append the
space or newline
            }
        return i;
    } else {
        break;
    }
}
```

```

        }
    }
    builder.append("@");
    return pos;
}

```

`parseUsername` iterates through the character array from the position after the `"@"`.

If the character is alphanumeric or the character is `"-"` and the previous character is not, and the name limit has not been exceeded, the character is appended to the `nameBuilder` and

the previous character is set.

If the character is not a valid part of the name there are two possibilities. Either the name should be matched, or the name is not valid.

If the character is whitespace, or we are at the end of the character array the name is valid if it is also of non-zero length.

If the name is valid the name link is appended to the `StringBuilder` used for the formatted markdown.

Aside from appending the link, there are two other cases which must be dealt with.

If the break point for the name is the end of the character array, then the last character in the array must be added.

Second, if the break point is not the end of the character array, then the whitespace character must be added.

Once the name has been added, the counter position is returned.

If the name is not valid, the loop breaks, `"@"` character is appended, and the original position is returned.

This method completes objective 9.v.iv.

Issue links

GitHub issue links are hashes, `"#"`, followed by integer strings.

Markdown.java

```

private static int parseIssue(StringBuilder builder, char[] cs, int pos,
String fullRepoPath) {
    final StringBuilder numBuilder = new StringBuilder();
    for(int i = pos + 1; i < cs.length; i++) {
        if(cs[i] >= '0' && cs[i] <= '9' && i != cs.length - 1) {
            numBuilder.append(cs[i]);
        } else if((isWhiteSpace(cs[i]) || isLineEnding(cs, i)) && (i > pos
+ 1 || i == cs.length - 1)) {
            if(i == cs.length - 1) {
                if(cs[i] >= '0' && cs[i] <= '9') {
                    numBuilder.append(cs[i]);
                } else if(!isWhiteSpace(cs[i])){
                    break;
                }
            }
        }
    }
    builder.append(numBuilder);
    return pos;
}

```

```

        }
    }
    builder.append("[#");
    builder.append(numBuilder.toString());
    builder.append("](https://github.com/");
    builder.append(fullRepoPath);
    builder.append("/issues/");
    builder.append(numBuilder.toString());
    builder.append("#");
    if(i != cs.length - 1) {
        builder.append(cs[i]); // We still need to append the
whitespace
    }
    return i;
} else {
    break;
}
}
builder.append("#");
return pos;
}

```

`parseIssue` checks if each character is a numeric value, adding it to `numBuilder` if so.

If the character is instead whitespace or a line ending the issue link may be valid. If we are at the end of the character array the final character must be checked for validity, and added to `numBuilder`, otherwise the loop breaks.

The link is built, and if the counter is not at the end of the array the original whitespace is appended.

If the character was not a valid issue link, the hash, "#", is appended and the original index is returned.

This method completes objective 9.b.v.

Relative links

When Markdown is rendered in a GitHub repository, links can be relative to the repository.

In order to load content from these links they need to be changed to a full link including the repository path.

Markdown.java

```

private static String concatenateRawContentUrl(String url, String
fullRepoName) {
    if(url.startsWith("http://") || url.startsWith("https://")) return url;
    int offset = 0;
    if(url.startsWith("./")) offset = 2;
    else if(url.startsWith("/")) offset = 1;
    return "https://raw.githubusercontent.com/" + fullRepoName +
"/master/" + url
                           .substring(offset);
}

```

A relative URL can be only a file name or it can start with either "/" or "./" specifying a path in the repository.

If the URL begins with "http://" or "https://" it is assumed to be valid and returned.

Otherwise, the offset is calculated and the URL is added as the file path in a link to githubusercontent.

The concatenateRawContentUrl function is used when parsing image links, as well as when checking links in a repository README.

Image links

Image links are checked both to ensure that they are not relative, and to add spacing around each image so that it does not interfere with the text line spacing.

Markdown.java

```
private static int parseImageLink(StringBuilder builder, char[] cs, int pos,
String fullRepoPath) {
    for(int i = pos + 1; i < cs.length; i++) {
        if(cs[i] == ')') {
            final String link = new String(Arrays.copyOfRange(cs, pos + 1,
i));
            final String extension = link.substring(link.lastIndexOf('.') +
1);
            if("png".equalsIgnoreCase(extension) ||
"jpg".equalsIgnoreCase(extension) ||
"gif".equalsIgnoreCase(extension) ||
"bmp".equalsIgnoreCase(extension) ||
"webp".equalsIgnoreCase(extension)) {
                if(TextUtils.isValidURL(link)) {
                    builder.append(link);
                } else {
                    builder.append(concatenateRawContentUrl(link,
fullRepoPath));
                }
                builder.append(") <br><br>");
            } else {
                builder.append(link);
                builder.append(")");
            }
            return i;
        } else if(isWhiteSpace(cs[i])) {
            break;
        }
    }
    return pos;
}
```

Recalling that a markdown image link is formatted as `![Description](link)`, `parseImageLink` must check the text between the opening bracket, “(”, and the closing bracket “)”.

If whitespace is found, the function can break early.

When the closing bracket is found, the extension can be checked for any of the image extensions supported by Android.

If the URL is already valid, it is added to the builder. Otherwise the concatenated URL is added.

Finally, the closing bracket and breaks are added.

If the URL does not end with an image extension, it is just added to the builder.

This method completes objective 9.b.iii.1.

Emoji

Emoji are added to GitHub Markdown by specifying their alias between two colons. For example “:smiley:” should be rendered as 😊.

Loading Emoji

In order to parse each alias to a unicode string, and later allow searching, a table of emojis is required.

I used the emoji json file used in GitHub’s gemoji, a Ruby gem for “character information about native emoji”.

After stripping the unicode version, ios version, and fitzpatrick information from the file, and minifying it I reduced it from 298kb to 139kb.

Each Emoji contains its description, aliases, tags, and unicode string.

Emoji.java

```
package com(tpb.mdtext.emoji;

import java.util.Collections;
import java.util.List;

/**
 * Created by theo on 16/04/17.
 */

public class Emoji {

    private final String description;
    private final List<String> aliases;
    private final List<String> tags;
    private final String unicode;

    Emoji(String unicode, String description, List<String> aliases,
        List<String> tags) {
```

```

        this.unicode = unicode;
        this.description = description;
        this.aliases = Collections.unmodifiableList(aliases);
        this.tags = Collections.unmodifiableList(tags);
    }

    public String getDescription() {
        return description;
    }

    public List<String> getAliases() {
        return aliases;
    }

    public List<String> getTags() {
        return tags;
    }

    public String getUnicode() {
        return unicode;
    }

    @Override
    public boolean equals(Object o) {
        return o instanceof Emoji && unicode.equals(((Emoji) o).unicode);
    }

    @Override
    public int hashCode() {
        return unicode.hashCode();
    }

    @Override
    public String toString() {
        return "Emoji{" +
            "description='" + description + '\'' +
            ", aliases=" + aliases +
            ", tags=" + tags +
            ", unicode='" + unicode + '\'' +
            '}';
    }
}

```

The emoji are loaded from the resource directory and added to a master list of emojis as well as maps for aliases and tags.

EmojiLoader.java

```

package com(tpb.mdtext.emoji;

import android.content.res.AssetManager;
import android.support.annotation.NonNull;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.BufferedReader;

```

```

import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;

/**
 * Created by theo on 16/04/17.
 */

public class EmojiLoader {

    private static final Map<String, Emoji> ALIAS_MAP = new HashMap<>();
    private static final Map<String, Set<Emoji>> TAG_MAP = new HashMap<>();
    private static final List<Emoji> EMOJIS = new ArrayList<>();

    public static void loadEmojis(AssetManager assets) {
        if(EMOJIS.size() > 0) return;
        try {
            final JSONArray JSON = new
JSONArray(inputStreamToString(assets.open("json/emojis.json")));
            for(int i = 0; i < JSON.length(); i++) {
                final Emoji emoji = buildEmojiFromJSON(JSON.getJSONObject(i));
                if(emoji != null) {
                    EMOJIS.add(emoji);
                    for(String tag : emoji.getTags()) {
                        if(TAG_MAP.get(tag) == null) TAG_MAP.put(tag, new
HashSet<Emoji>());
                        TAG_MAP.get(tag).add(emoji);
                    }
                    for(String alias : emoji.getAliases()) {
                        ALIAS_MAP.put(alias, emoji);
                    }
                }
            }
        } catch(Exception ignored) {}
    }

    private static String inputStreamToString(InputStream is) throws
IOException {
        final BufferedReader reader = new BufferedReader( new
InputStreamReader(is));
        String line;
        final StringBuilder builder = new StringBuilder();
        while((line = reader.readLine()) != null){
            builder.append(line);
        }
        is.close();
        return builder.toString();
    }

    private static Emoji buildEmojiFromJSON(JSONObject json) throws
JSONException {
        if (!json.has("emoji")) {
            return null;
        }
    }
}

```

```

    }
    String emoji = json.getString("emoji");
    String description = null;
    if (json.has("description")) {
        description = json.getString("description");
    }

    List<String> aliases =
JSONArrayToStringList(json.getJSONArray("aliases"));
    List<String> tags = JSONArrayToStringList(json.getJSONArray("tags"));
    return new Emoji(emoji, description, aliases, tags);
}

private static List<String> JSONArrayToStringList(JSONArray array) throws
JSONException {
    final List<String> strings = new ArrayList<>(array.length());
    for (int i = 0; i < array.length(); i++) {
        strings.add(array.getString(i));
    }
    return strings;
}

public static List<Emoji> getAllEmoji() {
    return EMOJIS;
}

public static Set<Emoji> getEmojiForTag(@NonNull String tag) {
    return TAG_MAP.get(tag);
}

public static Emoji getEmojiForAlias(@NonNull String alias) {
    if(alias.startsWith(":")) alias = alias.substring(1, alias.length());
    if(alias.endsWith(":")) alias = alias.substring(0, alias.length() -
1);
    return ALIAS_MAP.get(alias);
}
}

```

The JSON file is opened as an `InputStream` which is then converted to a `String` which can be read as a `JSONArray` for conversion to `Emoji` objects. For each `Emoji` successfully created the object is added to `EMOJIS`, added to `ALIAS_MAP`, and added to a `HashSet` of `Emojis` in `TAG_MAP`. The two `HashMaps` can later be used to retrieve `Emojis` by their tags or aliases. `getEmojiForAlias` also checks whether the alias is in the “`:alias:`” format and strips the colons before searching.

Displaying Emoji

Markdown.java

```

private static int parseEmoji(StringBuilder builder, char[] cs, int pos) {
    final StringBuilder emojiBuilder = new StringBuilder();

    for(int i = pos + 1; i < cs.length; i++) {

```

```

        if((cs[i] >= 'A' && cs[i] <= 'Z') ||
           (cs[i] >= '0' && cs[i] <= '9') ||
           (cs[i] >= 'a' && cs[i] <= 'z') ||
           cs[i] == '_') {
            emojiBuilder.append(cs[i]);
        } else if(cs[i] == ':') {
            final Emoji eww =
EmojiLoader.getEmojiForAlias(emojiBuilder.toString());
            if(eww == null) break;
            builder.append(eww.getUnicode());
            return i;
        } else {
            break;
        }
    }
    builder.append(":");
    return pos;
}

```

`parseEmoji` iterates through the character array after a colon, adding each alphanumeric or underscore character to a `StringBuilder`. If another colon is reached, the `Emoji` is loaded from `EmojiLoader` and if it is non null, the emoji unicode string is added to the parsed text. Otherwise the colon is appended and the original position is returned. This method completes objective [9.b.vi](#).

Text Utilities and Android Regex patterns

Linkify

`Linkify` is Android's utility for adding clickable links to text. It can match URLs, email addresses, phone numbers, and map addresses.

There is no need to match phone numbers and map addresses in GitHub Markdown, however URLs and email addresses should be linked.

The Android regex for URLs is quite large.

It contains the IANA (Internet Assigned Numbers Authority) top level domain names, as well as the characters in the RFC 3987 specification which extends upon the Uniform Resource Identifier scheme.

The `Pattern` class also contains the `WORD_BOUNDARY` regex which is “`(?:\b|$|^)`”. The regex begins with “`?:`”. This begins a non-capturing group, which matches a pattern, but does not include it in the result.

The “`\b`” character is an anchor which matches any word boundary position. A word boundary is any of the three following positions

- Before the first character in the string, if the first character is a word character.
- After the last character in the string, if the last character is a word character.

- Between two characters in the string, where one is a word character and the other is not a word character.

The dollar, "\$", is used to match the position before the first newline in a string.

Finally, the hat, "^", is used to match the beginning of a string.

This pattern is used to ensure that invalid domain names such as ".coma" do not match as valid domain names such as ".com"

The Pattern used for autolinking web URLs is ""(" + WEB_URL_WITH_PROTOCOL + "|" + WEB_URL_WITHOUT_PROTOCOL + ")" which matches any URL with or without a protocol.

This Pattern is likely to cause problems with code in the Markdown.

For example, "System.out.println()" would be matched as a URL.

This problem can be fixed by modifying the regex to required that the character after the URL is whitespace or a line ending, making the regex

"(" + WEB_URL_WITH_PROTOCOL + "|" + WEB_URL_WITHOUT_PROTOCOL + ")(\$|\s)")"

The regex patterns, for autolinking URLs and email addresses can then be combined to a single pattern "AUTOLINK_WEB_URL + "|" + AUTOLINK_EMAIL_ADDRESS"

These patterns are included in the MDPattern class as they are private in the Pattern source class.

MDPattern.java

```
package com(tpb.mdtext;

import java.util.regex.Pattern;

/**
 * Created by theo on 05/03/17.
 */

public class MDPattern {

    /**
     * Regular expression to match all IANA top-level domains.
     * <p>
     * List accurate as of 2015/11/24. List taken from:
     * http://data.iana.org/TLD/tlds-alpha-by-domain.txt
     * This pattern is auto-generated by frameworks/ex/common/tools/make-iana-
     * tld-pattern.py
     *
     * @hide
     */
    private static final String IANA_TOP_LEVEL_DOMAINS =
        "(?:"
```

```
+  
"(?:aaa|aarp|abb|abbott|abogado|academy|accenture|accountant|accountants|aco|active"  
+  
"|actor|ads|adult|aeg|aero|afl|agency|aig|airforce|airtel|allfinanz|alsace|ami  
ca|amsterdam"  
+  
"|android|apartments|app|apple|aquarelle|aramco|archi|army|arpa|arte|asia|asso  
ciates"  
+  
"|attorney|auction|audio|auto|autos|axa|azure|a[cdefgilmoqrstuwxz])"  
+  
"|(?:band|bank|bar|barcelona|barclaycard|barclays|bargains|bauhaus|bayern|bbc|  
bbva"  
+  
"|bcn|beats|beer|bentley|berlin|best|bet|bharti|bible|bid|bike|bing|bingo|bio  
biz|black"  
+  
"|blackfriday|bloomberg|blue|bms|bmw|bnl|bnpparibas|boats|bom|bond|boo|boots|b  
outique"  
+  
"|bradesco|bridgestone|broadway|broker|brother|brussels|budapest|build|builder  
s|business"  
+ "|buzz|bzh|b[abdefghijmnorstvwyz])"  
+  
"|(?:cab|cafe|cal|camera|camp|cancerresearch|canon|capetown|capital|car|carava  
n|cards"  
+  
"|care|career|careers|cars|cartier|casa|cash|casino|cat|catering|cba|cbn|ceb|c  
enter|ceo"  
+  
"|cern|cfa|cfd|chanel|channel|chat|cheap|chloe|christmas|chrome|church|ciprian  
i|cisco"  
+  
"|citic|city|cityeats|claims|cleaning|click|clinic|clothing|cloud|club|clubmed  
|coach"  
+  
"|codes|coffee|college|cologne|com|commbank|community|company|computer|comsec  
|condos"  
+  
"|construction|consulting|contractors|cooking|cool|coop|corsica|country|coupon  
s|courses"  
+  
"|credit|creditcard|creditunion|cricket|crown|crs|cruises|csc|cuisinella|cymru  
|cyou|c[acdfghiklmnoruvwxyz])"  
+  
"|(?:dabur|dad|dance|date|dating|datsun|day|dclk|deals|degree|delivery|dell|de  
lta"  
+  
"|democrat|dental|dentist|desi|design|dev|diamonds|diet|digital|direct|direc  
tory|discount"  
+  
"|dnp|docs|dog|doha|domains|doosan|download|drive|durban|dvag|d[ejkmoz])"  
+  
"|(?:earth|eat|edu|education|email|emerck|energy|engineer|engineering|enterpri  
ses"  
+  
"|epson|equipment|erni|esq|estate|eurovision|eus|events|everbank|exchange|expe  
rt|exposed"
```

```

        + " |express|e[cegrstu])"
        +
"|(?:fage|fail|fairwinds|faith|family|fan|fans|farm|fashion|feedback|ferrero|film"
        +
" |final|finance|financial|firmdale|fish|fishing|fit|fitness|flights|florist|flowers|flsmidth"
        +
" |fly|foo|football|forex|forsale|forum|foundation|frl|frogans|fund|furniture|football|fyi"
        + " |f[ijkmor])"
        +
"|(?:gal|gallery|game|garden|gbiz|gdn|gea|gent|genting|ggee|gift|gifts|gives|giving"
        +
" |glass|gle|global|globlo|gmail|gmo|gmx|gold|goldpoint|golf|goo|goog|google|gov|gov|grainger"
        +
" |graphics|gratis|green|gripe|group|gucci|guge|guide|guitars|guru|g[abdefghilmnpqrstuwy])"
        +
"|(?:hamburg|hangout|haus|healthcare|help|here|hermes|hiphop|hitachi|hiv|hockey|holdings"
        +
" |holiday|homedepot|homes|honda|horse|host|hosting|hoteles|hotmail|house|how|h
sbc|hyundai"
        + " |h[kmnrtu])"
        +
"|(?:ibm|icbc|ice|icu|ifm|iinet|immo|immobilien|industries|infiniti|info|ing|ink|institute"
        +
" |insure|int|international|investments|ipiranga|irish|ist|istanbul|itau|iwc|i[delmnoqrst])"
        +
"|(?:jaguar|java|jcb|jetzt|jewelry|jlc|jll|jobs|joburg|jprs|juegos|j[emop])"
        +
"|(?:kaufen|kddi|kia|kim|kinder|kitchen|kiwi|koeln|komatsu|krd|kred|kyoto|k[eg
himnprwyz])"
        +
"|(?:lacaixa|lancaster|land|landrover|lasalle|lat|latrobe|law|lawyer|lds|lease
|leclerc"
        +
" |legal|lexus|lgbt|liaison|lidl|life|lifestyle|lighting|limited|limo|linde|lin
k|live"
        +
" |lixil|loan|loans|lol|london|lotte|lotto|love|ltd|ltda|lupin|luxe|luxury|l[ab
cikrstuvy])"
        +
"|(?:madrid|maif|maison|man|management|mango|market|marketing|markets|marriott
|mba)"
        +
" |media|meet|melbourne|meme|memorial|men|menu|meo|miami|microsoft|mil|mini|mma
|mobi|moda"
        +
" |moe|moi|mom|monash|money|montblanc|mormon|mortgage|moscow|motorcycles|mov|m
ovie|movistar"
        +
" |mtn|mtpc|mtr|museum|mutuelle|m[acdeghklmnopqrstuvwxyz])"

```

```

+
" |(?:nadex|nagoya|name|navy|nec|net|netbank|network|neustar|new|news|nexus|ngo
|nhk"
+
" |nico|ninja|nissan|nokia|nra|nrw|ntt|nyc|n[acefgilopruz])"
+
" |(?:obi|office|okinawa|omega|one|ong|onl|online|ooo|oracle|orange|org|organic
|osaka"
+ " |otsuka|ovh|om"
+
" |(?:page|panerai|paris|partners|parts|party|pet|pharmacy|philips|photo|photog
raphy"
+
" |photos|physio|piaget|pics|pictet|pictures|ping|pink|pizza|place|play|playsta
tion|plumbing"
+
" |plus|pohl|poker|porn|post|praxi|press|pro|prod|productions|prof|properties|p
roperty"
+ " |protection|pub|p[aefghijklmnrstwy])"
+ " |(?:qpon|quebec|qa)"
+
" |(?:racing|realtor|realty|recipes|red|redstone|rehab|reise|reisen|reit|ren|re
nt|rentals"
+
" |repair|report|republican|rest|restaurant|review|reviews|rich|ricoh|rio|rip|r
ocher|rocks"
+ " |rodeo|rsvp|ruhr|run|rwe|ryukyu|r[eosuw])"
+
" |(?:saarland|sakura|sale|samsung|sandvik|sandvikcoromant|sanofi|sap|sapo|sar
l|saxo"
+
" |sbs|sca|scb|schmidt|scholarships|school|schule|schwarz|science|scor|scot|se
at|security"
+
" |seek|sener|services|seven|sew|sex|sexy|shiksha|shoes|show|shriram|singles|si
te|ski"
+
" |sky|skype|snfc|soccer|social|software|sohu|solar|solutions|sony|soy|space|sp
iegel|spreadbetting"
+
" |srl|stada|starhub|statoil|stc|stcgroup|stockholm|studio|study|style|sucks|su
pplies"
+
" |supply|support|surf|surgery|suzuki|swatch|swiss|sydney|systems|s[abcdefghijkl
mnortuvwxyz])"
+
" |(?:tab|taipei|tatamotors|tatar|tattoo|tax|taxi|team|tech|technology|tel|tele
fonica"
+
" |temasek|tennis|thd|theater|theatre|tickets|tienda|tips|tires|tirol|today|tok
yo|tools"
+
" |top|toray|toshiba|tours|town|toyota|toys|trade|trading|training|travel|trust
|tui|t[cdfghijklmnortvwz])"
+ " |(?:ubs|university|uno|uo[u[agksyz])"
+
" |(?:vacations|vana|vegas|ventures|versicherung|vet|viajes|video|villas|vin|vi
rgin"

```

```

+
" | vision | vista | vistaprint | viva | vlaanderen | vodka | vote | voting | voto | voyage | v[aceg
inu])"
+
" | (? : wales | walter | wang | watch | webcam | website | wed | wedding | weir | whoswho | wien | wiki
| williamhill"
    + " | win | windows | wine | wme | work | works | world | wtc | wtf | w[fs])"
    +
" | (? : \u03b5\u03bb | \u0431\u0435\u043b | \u0434\u0435\u0442\u0438 | \u043a\u043e\u04
3c | \u043c\u043a\u0434"
    +
" | \u043c\u043e\u043d | \u043c\u043e\u0441\u043a\u0432\u0430 | \u043e\u043d\u043b\u0430\u0439\u043d"
    +
" | \u043e\u0440\u0433 | \u0440\u0443\u0441 | \u0440\u0444 | \u0441\u0430\u0439\u0442 |
\u0441\u0440\u0431"
    +
" | \u0443\u043a\u0440 | \u049b\u0430\u0437 | \u0570\u0561\u0575 | \u05e7\u05d5\u05dd |
\u0627\u0631\u0627\u0645\u0643\u0648"
    +
" | \u0627\u0644\u0627\u0631\u062f\u0646 | \u0627\u0644\u062c\u0632\u0627\u0626\u0
631 | \u0627\u0644\u0633\u0639\u0648\u062f\u064a\u0629"
    +
" | \u0627\u0644\u0645\u063a\u0631\u0628 | \u0627\u0645\u0627\u0631\u0627\u062a | \u0
627\u06cc\u0631\u0627\u0646"
    +
" | \u0628\u0627\u0632\u0627\u0631 | \u0628\u06be\u0627\u0631\u062a | \u062a\u0648\u0
646\u0633"
    +
" | \u0633\u0648\u062f\u0627\u0646 | \u0633\u0648\u0631\u064a\u0629 | \u0634\u0628\u0
643\u0629"
    +
" | \u0639\u0631\u0627\u0642 | \u0639\u0645\u0627\u0646 | \u0641\u0644\u0633\u0637\u0
64a\u0646"
    +
" | \u0642\u0637\u0631 | \u0643\u0648\u0645 | \u0645\u0635\u0631 | \u0645\u0644\u064a\u0
633\u064a\u0627"
    +
" | \u0645\u0648\u0642\u0639 | \u0915\u0949\u092e | \u0928\u0947\u091f | \u092d\u093e\u09
30\u0924"
    +
" | \u0938\u0902\u0917\u0920\u0928 | \u09ad\u09be\u09b0\u09a4 | \u0a2d\u0a3e\u0a30\u0
a24 | \u0aad\u0abe\u0ab0\u0aa4"
    +
" | \u0b87\u0ba8\u0bcd\u0ba4\u0bbf\u0baf\u0bbe | \u0b87\u0bb2\u0b99\u0bcd\u0b95\u0
bc8 | \u0b9a\u0bbf\u0b99\u0bcd\u0b95\u0baa\u0bcd\u0baa\u0bc2\u0bb0\u0bcd"
    +
" | \u0c2d\u0c3e\u0c30\u0c24\u0c4d | \u0dbd\u0d82\u0d9a\u0dcf | \u0e04\u0e2d\u0e21 |
\u0e44\u0e17\u0e22"
    +
" | \u10d2\u10d4 | \u307f\u3093\u306a | \u30b0\u30fc\u30b0\u30eb | \u30b3\u30e0 | \u4e16
\u754c"
    +
" | \u4e2d\u4fe1 | \u4e2d\u56fd | \u4e2d\u570b | \u4e2d\u6587\u7f51 | \u4f01\u4e1a | \u4f5
b\u5c71"
    +
" | \u4fe1\u606f | \u5065\u5eb7 | \u516b\u5366 | \u516c\u53f8 | \u516c\u76ca | \u53f0\u6e7
e | \u53f0\u7063"

```

```

+
"|\u5546\u57ce|\u5546\u5e97|\u5546\u6807|\u5728\u7ebf|\u5927\u62ff|\u5a31\u4e50|\u5de5\u884c"
+
"|\u5e7f\u4e1c|\u6148\u5584|\u6211\u7231\u4f60|\u624b\u673a|\u653f\u52a1|\u653f\u5e9c"
+
"|\u65b0\u52a0\u5761|\u65b0\u95fb|\u65f6\u5c1a|\u673a\u6784|\u6de1\u9a6c\u9521|\u6e38\u620f"
+
"|\u70b9\u770b|\u79fb\u52a8|\u7ec4\u7ec7\u673a\u6784|\u7f51\u5740|\u7f51\u5e97|\u7f51\u7edc"
+
"|\u8c37\u6b4c|\u96c6\u56e2|\u98de\u5229\u6d66|\u9910\u5385|\u9999\u6e2f|\ub2f7\ub137"
    +
    + "|\ub2f7\ucef4|\uc0bc\uc131|\ud55c\ud6d|xbox"
    + "|xerox|xin|xn\\-\\-11b4c3d|xn\\-\\-1qqw23a|xn\\-\\-30rr7y|xn\\-\\-3bst00m|xn\\-\\-3ds443g"
        +
        + "|xn\\-\\-3e0b707e|xn\\-\\-3pxu8k|xn\\-\\-42c2d9a|xn\\-\\-45brj9c|xn\\-\\-45q11c|xn\\-\\-4gbrim"
        +
        + "|xn\\-\\-55qw42g|xn\\-\\-55qx5d|xn\\-\\-6frz82g|xn\\-\\-6qq986b3x1|xn\\-\\-80adxhks"
        +
        + "|xn\\-\\-80ao21a|xn\\-\\-80asehdb|xn\\-\\-80aswg|xn\\-\\-90a3ac|xn\\-\\-90ais|xn\\-\\-9dbq2a"
        +
        + "|xn\\-\\-9et52u|xn\\-\\-b4w605ferd|xn\\-\\-c1avg|xn\\-\\-c2br7g|xn\\-\\-cg4bki|xn\\-\\-clchc0ea0b2g2a9gcd"
        +
        + "|xn\\-\\-czr694b|xn\\-\\-czrs0t|xn\\-\\-czru2d|xn\\-\\-d1acj3b|xn\\-\\-d1alf|xn\\-\\-efvy88h"
        +
        + "|xn\\-\\-estv75g|xn\\-\\-fhbei|xn\\-\\-fiq228c5hs|xn\\-\\-fiq64b|xn\\-\\-fiqs8s|xn\\-\\-fiqz9s"
        +
        + "|xn\\-\\-fjq720a|xn\\-\\-flw351e|xn\\-\\-fpqrj9c3d|xn\\-\\-fzc2c9e2c|xn\\-\\-gecrj9c"
        +
        + "|xn\\-\\-h2brj9c|xn\\-\\-hxt814e|xn\\-\\-i1b6b1a6a2e|xn\\-\\-imr513n|xn\\-\\-io0a7i"
        +
        + "|xn\\-\\-j1aef|xn\\-\\-j1amh|xn\\-\\-j6w193g|xn\\-\\-kcrx77d1x4a|xn\\-\\-kprw13d|xn\\-\\-kqry57d"
        +
        + "|xn\\-\\-kput3i|xn\\-\\-l1acc|xn\\-\\-lgbbat1ad8j|xn\\-\\-mgb9awbf|xn\\-\\-mgbba3a3ejt"
        +
        + "|xn\\-\\-mgbba3a4f16a|xn\\-\\-mgbbaam7a8h|xn\\-\\-mgbab2bd|xn\\-\\-mgbayh7gpa|xn\\-\\-mgbbh1a71e"
        +
        + "|xn\\-\\-mgbc0a9azcg|xn\\-\\-mgberp4a5d4ar|xn\\-\\-mgbpl2fh|xn\\-\\-mgbtx2b|xn\\-\\-mgbx4cd0ab"
        +
        + "|xn\\-\\-mk1bu44c|xn\\-\\-mxtq1m|xn\\-\\-ngbc5azd|xn\\-\\-node|xn\\-\\-nqv7f|xn\\-\\-nqv7fs00ema"
        +
        + "|xn\\-\\-nyqy26a|xn\\-\\-o3cw4h|xn\\-\\-ogbpf8f1|xn\\-\\-p1acf|xn\\-\\-p1ai|xn\\-\\-pgbs0dh"
        +
        + "|xn\\-\\-pssy2u|xn\\-\\-q9jyb4c|xn\\-\\-qcka1pmc|xn\\-\\-qxam|xn\\-\\-rhqv96g|xn\\-\\-s9brj9c"
        +
        + "|xn\\-\\-ses554g|xn\\-\\-t60b56a|xn\\-\\-tckwe|xn\\-\\-unup4y|xn\\-\\-vermgensberater\\-ctb"
        +
        + "|xn\\-\\-vermgensberatung\\-pwb|xn\\-\\-vhquv|xn\\-\\-vuq861b|xn\\-\\-wgbh1c|xn\\-\\-wgb16a"
        +
        + "|xn\\-\\-xhq521b|xn\\-\\-xkc2a13hye2a|xn\\-\\-xkc2d13a5ee0h|xn\\-\\-y9a3aq|xn\\-\\-yfro4i67o"
        +
        + "|xn\\-\\-ygb12ammx|xn\\-\\-zfr164b|xperia|xxx|xyz)"
        +
    +
"|(?:yachts|yamaxun|yandex|yodobashi|yoga|yokohama|youtube|y[et])"
    +
    + "|(?:zara|zip|zone|zuerich|z[amw]))";

```

```

public static final Pattern IP_ADDRESS
    = Pattern.compile(
        "((25[0-5]|2[0-4][0-9]|0-1)[0-9]{2}|1-9)[0-9]|1-9))\\.(25[0-
5]|2[0-4]" +
        + "[0-9]|0-1)[0-9]{2}|1-9)[0-9]|1-9)|0)\\.(25[0-5]|2[0-
4][0-9]|0-1]" +
        + "[0-9]{2}|1-9)[0-9]|1-9)|0)\\.(25[0-5]|2[0-4][0-9]|0-
1)[0-9]{2}" +
        + "|1-9)[0-9]|0-9]))");
}

/**
 * Valid UCS characters defined in RFC 3987. Excludes space characters.
 */
private static final String UCS_CHAR = "[" +
    "\u00A0-\uD7FF" +
    "\uF900-\uFDCE" +
    "\uFDF0-\uFFEF" +
    "\uD800\uDC00-\uD83F\uDFFF" +
    "\uD840\uDC00-\uD87F\uDFFF" +
    "\uD880\uDC00-\uD8BF\uDFFF" +
    "\uD8C0\uDC00-\uD8FF\uDFFF" +
    "\uD900\uDC00-\uD93F\uDFFF" +
    "\uD940\uDC00-\uD97F\uDFFF" +
    "\uD980\uDC00-\uD9BF\uDFFF" +
    "\uD9C0\uDC00-\uD9FF\uDFFF" +
    "\uDA00\uDC00-\uDA3F\uDFFF" +
    "\uDA40\uDC00-\uDA7F\uDFFF" +
    "\uDA80\uDC00-\uDABF\uDFFF" +
    "\uDAC0\uDC00-\uDAFF\uDFFF" +
    "\uDB00\uDC00-\uDB3F\uDFFF" +
    "\uDB44\uDC00-\uDB7F\uDFFF" +
    "&&[\^u00A0[\u2000-\u200A]\u2028\u2029\u202F\u3000]]";
}

/**
 * Valid characters for IRI label defined in RFC 3987.
 */
private static final String LABEL_CHAR = "a-zA-Z0-9" + UCS_CHAR;

/**
 * Valid characters for IRI TLD defined in RFC 3987.
 */
private static final String TLD_CHAR = "a-zA-Z" + UCS_CHAR;

/**
 * RFC 1035 Section 2.3.4 limits the labels to a maximum 63 octets.
 */
private static final String IRI_LABEL =
    "[" + LABEL_CHAR + "](:[" + LABEL_CHAR + "\\-]{0,61}[" +
    LABEL_CHAR + "]){{0,1}}";

/**
 * RFC 3492 references RFC 1034 and limits Punycode algorithm output to 63
characters.
 */
private static final String PUNYCODE_TLD = "xn\\-\\-[\\w\\-]{0,58}\\w";

private static final String TLD = "(" + PUNYCODE_TLD + "|" + "[" +
TLD_CHAR + "]{2,63}" + ")";

```

```

private static final String HOST_NAME = "(" + IRI_LABEL + "\\.)+" + TLD;
private static final String PROTOCOL = "(?i:http|https|rtsp):\/\/\/";
/* A word boundary or end of input. This is to stop foo.sure from
matching as foo.su */
private static final String WORD_BOUNDARY = "(?:\\b|$|^)";

private static final String USER_INFO = "(?:[a-zA-Z0-9\\$\\\\-
\\_\\.\\.+\\\\!\\\\*\\\\'(\\\\')"
+ "\\\",\\\";\\\"?\\\"&\\\"=]\\|(?:\\\\%[a-fA-F0-9]{2}){1,64}(?:\\:\\:(?:[a-zA-Z0-
9\\$\\\\-\\\\_"
+ "\\\".\\\"+\\\\!\\\\*\\\\'\\\\'(\\\\')\\\",\\\";\\\"?\\\"&\\\"=]\\|(?:\\\\%[a-fA-F0-
9]{2}){1,25})?\\@";

private static final String PORT_NUMBER = "\\:\\\\d{1,5}";

private static final String PATH_AND_QUERY = "\\//(?:(:[" + LABEL_CHAR
+ "\\;\\\\/\\\\?\\\\:@\\\\&\\#=\\\\#\\\\~" // plus optional query params
+ "\\-\\\\.\\\"+\\\\!\\\\*\\\\'\\\\'(\\\\')\\\",\\\"_])|(?:\\\\%[a-fA-F0-9]{2}))*";

/**
 * Regular expression that matches known TLDs and punycode TLDs
 */
private static final String STRICT_TLD = "(?:" +
IANA_TOP_LEVEL_DOMAINS + "|" + PUNYCODE_TLD + ")";

/**
 * Regular expression that matches host names using {@link #STRICT_TLD}
 */
private static final String STRICT_HOST_NAME = "(?:(?:" + IRI_LABEL +
"\\.)+"
+ STRICT_TLD + ")";

/**
 * Regular expression that matches domain names using either {@link
#STRICT_HOST_NAME} or
 * {@link #IP_ADDRESS}
 */
private static final Pattern STRICT_DOMAIN_NAME
= Pattern.compile("(?:" + STRICT_HOST_NAME + "|\\" + IP_ADDRESS +
")");

/**
 * Regular expression that matches domain names without a TLD
 */
private static final String RELAXED_DOMAIN_NAME =
"(?:" + "(?:" + IRI_LABEL + "(?:\\.(?=\\$))" + "?)+" + "|" +
IP_ADDRESS + ")";

/**
 * Regular expression to match strings that do not start with a supported
protocol. The TLDs
 * are expected to be one of the known TLDs.
 */
private static final String WEB_URL_WITHOUT_PROTOCOL = "("
+ WORD_BOUNDARY
+ "(?<!:\\\\/\\\\/)"

```

```

+ "("
+ "(?:" + STRICT_DOMAIN_NAME + ")"
+ "(?:" + PORT_NUMBER + ")?""
+ ")"
+ "(?:" + PATH_AND_QUERY + ")?""
+ WORD_BOUNDARY
+ ")";

/**
 * Regular expression to match strings that start with a supported
protocol. Rules for domain
 * names and TLDs are more relaxed. TLDs are optional.
*/
private static final String WEB_URL_WITH_PROTOCOL = "("
    + WORD_BOUNDARY
    + "(?:"
    + "(?:" + PROTOCOL + "(?:" + USER_INFO + ")?" + ")"
    + "(?:" + RELAXED_DOMAIN_NAME + ")?""
    + "(?:" + PORT_NUMBER + ")?""
    + ")"
    + "(?:" + PATH_AND_QUERY + ")?""
    + WORD_BOUNDARY
    + ")";

/**
 * Regular expression pattern to match IRIIs. If a string starts with
http(s):// the expression
 * tries to match the URL structure with a relaxed rule for TLDs. If the
string does not start
 * with http(s):// the TLDs are expected to be one of the known TLDs.
*
* @hide
*/
static final Pattern AUTOLINK_WEB_URL = Pattern.compile(
    "(^|\s)((" + WEB_URL_WITH_PROTOCOL + "| " +
WEB_URL_WITHOUT_PROTOCOL + ")($|\s))");

/**
 * Regular expression for valid email characters. Does not include some of
the valid characters
 * defined in RFC5321: #&~!^`{}=/=$*?|
*/
private static final String EMAIL_CHAR = LABEL_CHAR + "\\\\+\\\\-_%'";

/**
 * Regular expression for local part of an email address. RFC5321 section
4.5.3.1.1 limits
 * the local part to be at most 64 octets.
*/
private static final String EMAIL_ADDRESS_LOCAL_PART =
    "[" + EMAIL_CHAR + "]" + "(?:"[ " + EMAIL_CHAR + "\\.]\\.{1,62}[" +
EMAIL_CHAR + "])?";

/**
 * Regular expression for the domain part of an email address. RFC5321
section 4.5.3.1.2 limits
 * the domain to be at most 255 octets.
*/
private static final String EMAIL_ADDRESS_DOMAIN =

```

```

        "(?=.{1,255}(?:\\s|$|^))" + HOST_NAME;

    /**
     * Regular expression pattern to match email addresses. It excludes double
     quoted local parts
     * and the special characters #&~!^`{}=/=$*?| that are included in RFC5321.
     *
     * @hide
     */
    public static final Pattern AUTOLINK_EMAIL_ADDRESS = Pattern.compile("(
WORD_BOUNDARY +
        "(?:" + EMAIL_ADDRESS_LOCAL_PART + "@" + EMAIL_ADDRESS_DOMAIN +
")" +
        WORD_BOUNDARY + ")"
);

public static final Pattern SPACED_MATCH_PATTERN = Pattern
    .compile(AUTOLINK_WEB_URL + "|" + AUTOLINK_EMAIL_ADDRESS);

private MDPattern() {
}

}

```

Adding links

Having defined the regex for matching URLs and emails, the `addLinks` method can be explained.

TextUtils.java

```

public static boolean addLinks(@NonNull Spannable spannable) {
    boolean hasMatches = false;
    final Matcher m = MDPattern.SPACED_MATCH_PATTERN.matcher(spannable);
    while(m.find()) {
        spannable.setSpan(
            new CleanURLESpan(m.group(0)),
            m.start(),
            m.end(),
            Spanned.SPAN_EXCLUSIVE_EXCLUSIVE
        );
        hasMatches = true;
    }

    return hasMatches;
}

```

The method uses a matcher from the `SPACED_MATCH_PATTERN` and sets a `CleanURLESpan`, a subclass of `URLESpan` to be explained later, across the indices of the match.

The `Spanned.SPAN_EXCLUSIVE_EXCLUSIVE` flag means that if a character or another span is inserted at either end of the `CleanURLESpan` it will not be viewed as part of the `CleanURLESpan`.

This method completes objective 9.c.

Multiple pattern matching and string escaping

In order not to attempt to display HTML tags in titles, and to replace HTML tags in order to stop Android capturing them, multiple string replace calls must be used.

Rather than calling the `replace` method multiple times, each incurring a full traversal of the string, multiple matches can be compiled into a single pattern.

TextUtils.java

```
static Pattern generatePattern(@NonNull Set<String> keys) {
    final StringBuilder b = new StringBuilder();
    int i = 0;
    for(String s : keys) {
        b.append(REGEX_ESCAPE_CHARS.matcher(s).replaceAll("\\\\\$0"));
        if(++i != keys.size()) b.append('|');
    }
    return Pattern.compile(b.toString());
}
```

`generateKeys` takes a set of strings and builds an or separated pattern from the strings.

In order to ensure that the strings themselves are only matched as text, they must be escaped.

Each match key is escaped with the “[\\<\\\\\\\\{\\\\\\\\\\\\^\\\\\\\\-\\\\\\\\=\\\\\$\\\\\\\\\\\\\\\\\\\\\\\\?*\\\\\\\\+\\\\\\\\.\\\\\\\\>]” pattern, which matches regex control characters and allows them to be replaced with their escaped form.

Once a valid pattern has been generated, a single Matcher can be used to replace a set of key value pairs from a Map.

TextUtils.java

```
static String replace(@Nullable String s, Map<String, String> replacements,
Pattern pattern) {
    if(s == null) return null;
    final StringBuffer buffer = new StringBuffer();
    final Matcher matcher = pattern.matcher(s);
    while(matcher.find()) {
        matcher.appendReplacement(buffer,
replacements.get(matcher.group())));
    }
    matcher.appendTail(buffer);
    return buffer.toString();
}
```

In each iteration of the while loop `appendReplacement` appends all of the text between the previous match and the current one.

Finally, the `appendTail` call appends any text after the final match.

Background colour selection

When displaying text on a coloured background, as will happen when displaying labels, it is important to ensure that the foreground text is a suitable colour to ensure legibility.

In order to determine the background colour, the relative luminance is used.

For linear RGB values, the relative luminance is given by $Y = 0.2126R + 0.7152G + 0.0722B$. This formula shows that green contributes most to the human perception of luminosity, and blue the least.

In order to use this formula, an inverse gamma function must be applied to the RGB values to account for the non-linear relationship between the intensity of the primary colours and the actual values stored.

The numeric approximation to the function is given below

$$\begin{cases} \frac{C_{sRGB}}{12.92} & C_{sRGB} \leq 0.04045 \\ \left(\frac{C_{sRGB} + 0.055}{1.055} \right)^{2.4} & C_{sRGB} \geq 0.04045 \end{cases}$$

Each RGB value is then used in the relative luminance formula to determine whether to use a light or dark text colour.

TextUtils.java

```
public static int getTextColorForBackground(int bg) {
    double r = Color.red(bg) / 255d;
    if(r <= 0.04045) {
        r = r / 12.92;
    } else {
        r = Math.pow((r + 0.055) / 1.055, 2.4);
    }
    double g = Color.green(bg) / 255d;
    if(g <= 0.04045) {
        g = g / 12.92;
    } else {
        g = Math.pow((g + 0.055) / 1.055, 2.4);
    }
    double b = Color.blue(bg) / 255d;
    if(b <= 0.04045) {
        b = b / 12.92;
    } else {
        b = Math.pow((b + 0.055) / 1.055, 2.4);
    }
    return (0.2126 * r + 0.7152 * g + 0.0722 * b) > 0.35 ? Color.BLACK :
Color.WHITE;
}
```

This method completes objective 9.b.viii.

Other utility methods

TextUtils.java

```
package com(tpb.mdtext;

import android.graphics.Color;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.text.Spannable;
import android.text.Spanned;

import com(tpb.mdtext.views.spans.CleanURLSpan;

import java.util.Map;
import java.util.Set;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * Created by theo on 21/03/17.
 */

public class TextUtils {

    private TextUtils() {}

    private static final Pattern REGEX_ESCAPE_CHARS =
        Pattern.compile("[\\<\\\\{\\\\}^\\\\-\\\\=\\\\$\\\\!\\\\|\\\\]\\\\{\\\\}\\\\?\\\\*\\\\+\\\\.\\\\>]");

    static String replace(@Nullable String s, Map<String, String>
replacements) {
        return replace(s, replacements,
generatePattern(replacements.keySet()));
    }

    static String replace(@Nullable String s, Map<String, String>
replacements, Pattern pattern) {
        if(s == null) return null;
        final StringBuffer buffer = new StringBuffer();
        final Matcher matcher = pattern.matcher(s);
        while(matcher.find()) {
            matcher.appendReplacement(buffer,
replacements.get(matcher.group()));
        }
        matcher.appendTail(buffer);
        return buffer.toString();
    }

    static Pattern generatePattern(@NonNull Set<String> keys) {
        final StringBuilder b = new StringBuilder();
        int i = 0;
        for(String s : keys) {
            b.append(REGEX_ESCAPE_CHARS.matcher(s).replaceAll("\\\\\\\\$0"));
            if(++i != keys.size()) b.append('|');
        }
        return Pattern.compile(b.toString());
    }
}
```

```

}

public static boolean addLinks(@NonNull Spannable spannable) {
    boolean hasMatches = false;
    final Matcher m = MDPattern.SPACED_MATCH_PATTERN.matcher(spannable);
    while(m.find()) {
        spannable.setSpan(
            new CleanURLSpan(m.group(0)),
            m.start(),
            m.end(),
            Spanned.SPAN_EXCLUSIVE_EXCLUSIVE
        );
        hasMatches = true;
    }

    return hasMatches;
}

public static int getTextColorForBackground(int bg) {
    double r = Color.red(bg) / 255d;
    if(r <= 0.04045) {
        r = r / 12.92;
    } else {
        r = Math.pow((r + 0.055) / 1.055, 2.4);
    }
    double g = Color.green(bg) / 255d;
    if(g <= 0.04045) {
        g = g / 12.92;
    } else {
        g = Math.pow((g + 0.055) / 1.055, 2.4);
    }
    double b = Color.blue(bg) / 255d;
    if(b <= 0.04045) {
        b = b / 12.92;
    } else {
        b = Math.pow((b + 0.055) / 1.055, 2.4);
    }
    return (0.2126 * r + 0.7152 * g + 0.0722 * b) > 0.35 ? Color.BLACK :
Color.WHITE;
}

public static boolean isValidURL(String possible) {
    return MDPattern.AUTOLINK_WEB_URL.matcher(possible).matches();
}

public static String capitaliseFirst(String s) {
    if(s == null || s.length() == 0) return s;
    return s.substring(0, 1).toUpperCase() + s.substring(1).toLowerCase();
}

/**
 * Counts the instances of a string within another string
 *
 * @param s The string to search
 * @param sub The string to count instances of
 * @return The number of instances of s2 in s1
 */
public static int instancesOf(@NonNull String s, @NonNull String sub) {

```

```

        if(s.length() == 0 || sub.length() == 0 || sub.length() > s.length())
    return 0;
    int last = 0;
    int count = 0;
    while(last != -1) {
        last = s.indexOf(sub, last);
        if(last != -1) {
            count++;
            last++;
        }
    }
    return count;
}

public static boolean isInteger(String s) {
    return isInteger(s, 10);
}

public static boolean isInteger(String s, int radix) {
    if(s.isEmpty()) return false;
    for(int i = 0; i < s.length(); i++) {
        if(i == 0 && s.charAt(i) == '-') {
            if(s.length() == 1) return false;
            else continue;
        }
        if(Character.digit(s.charAt(i), radix) < 0) return false;
    }
    return true;
}
}

```

The four other methods in `TextUtils` are `isValidUrl`, `capitaliseFirst`, `instancesOf`, and `isInteger`.

`isValidURL` matches a string against the AUTOLINK_WEB_URL pattern.

`capitaliseFirst` capitalises the first character of a string.

`instancesOf` determines the number of instances of a substring in another.

It first performs a check for whether either of the strings are empty, or if the substring is larger than the string to be searched.

Otherwise, `instancesOf` searches the string for the substring, each time searching from the index after the previous position found.

`isInteger` is used to determine whether a `String` is a an integer in a given base.

It first checks if the string is empty.

If the string is not empty the method iterates through each character in the string.

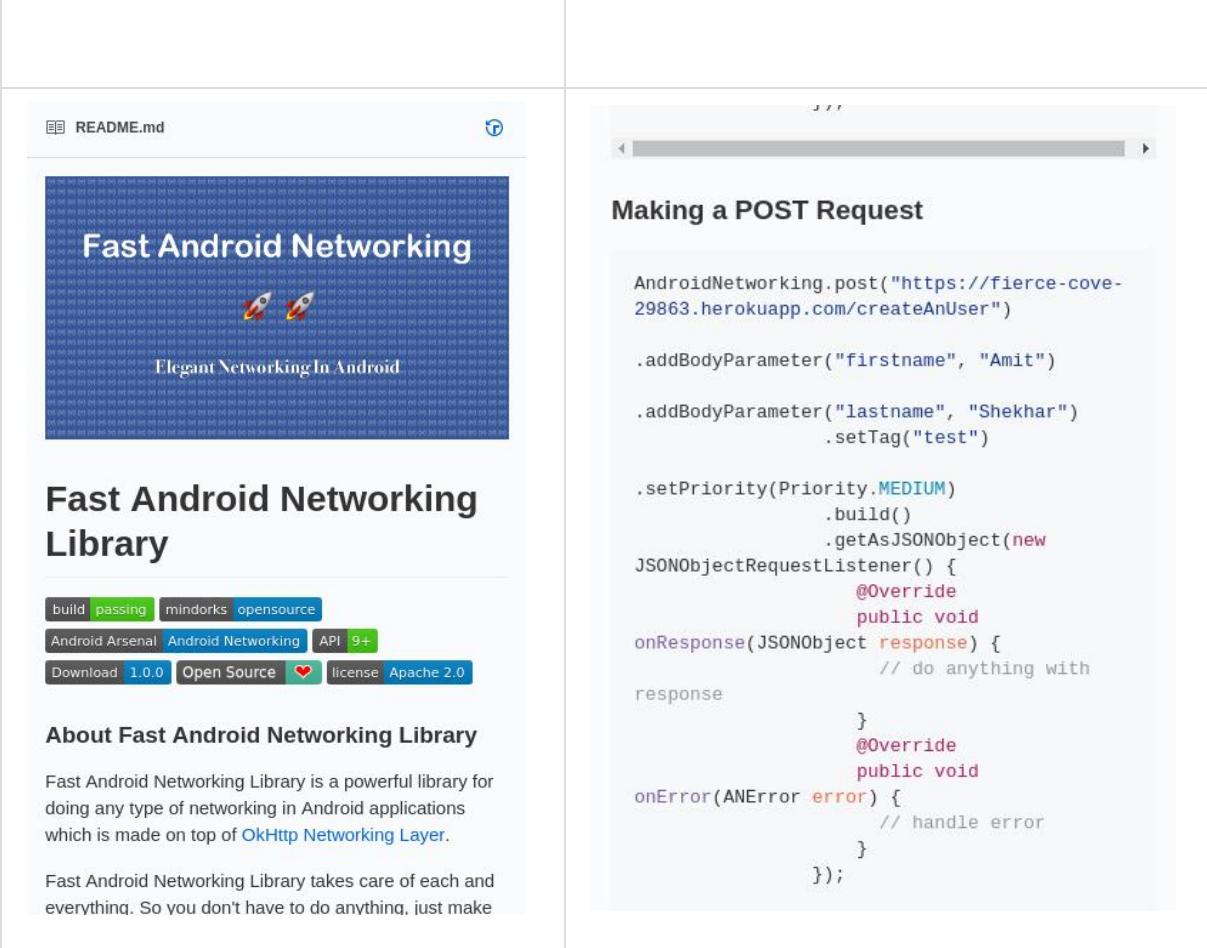
If the first character of the string is a minus, “-”, and the string is longer than one character, the string may still be valid.

Otherwise, each iteration checks the character for its numeric value in the given base with the `Character` class, which will return -1 if the character has no numeric value in the given base.

Displaying README files

While most Markdown strings will be relatively short and simple, many README files are quite complex as they are used to explain the repository purpose and how one might go about using it.

The GitHub mobile website displays READMEs inline with code made horizontally scrollable.



Making a POST Request

```
AndroidNetworking.post("https://fierce-cove-29863.herokuapp.com/createAnUser")
    .addBodyParameter("firstname", "Amit")
    .addBodyParameter("lastname", "Shekhar")
    .setTag("test")
    .setPriority(Priority.MEDIUM)
    .build()
    .getAsJSONObject(new JSONObjectRequestListener() {
        @Override
        public void onResponse(JSONObject response) {
            // do anything with
            response
        }
        @Override
        public void onError(ANError error) {
            // handle error
        }
    });
}
```

In order to implement exactly the same styling, the same CSS as used in the GitHub website can be used. A dark theme is also used, with the colour scheme changed.

In order to ensure that the Markdown is rendered in exactly the same manner, the GitHub Markdown API is used.

This API takes a Markdown string and renders it to HTML. It also takes an optional context argument, specifying the repository which the Markdown is being rendered for.

JavaScript interface methods

In order to interface between Java code used in the application and the JavaScript used in a WebView JavaScriptInterface methods are used.

These methods are annotated with the @JavascriptInterface annotations, which makes them accessible from JavaScript run in the WebView through an interface set on the WebView.

In order to call JavaScript functions from the Java code the evaluateJavascript function is called, which evaluates a string of JavaScript.

MarkdownWebView.java

```
private void init() {
    setWebViewClient(new WebViewClient() {
        public void onPageFinished(WebView view, String url) {
            evaluateJavascript(previewText, null);

        }
    });
    addJavascriptInterface(this, "TouchIntercept");
    if(darkTheme) {
        loadUrl("file:///android_asset/html/md_preview_dark.html");
    } else {
        loadUrl("file:///android_asset/html/md_preview.html");
    }

    getSettings().setJavaScriptEnabled(true);
    getSettings().setAllowUniversalAccessFromFileURLs(true);

    if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {

getSettings().setMixedContentMode(WebSettings.MIXED_CONTENT_ALWAYS_ALLOW);
    }
}
```

The init method sets a custom WebView client which evaluates the Javascript string used to load the rendered Markdown into the WebView once the page has finished loading.

It also adds a Javascript interface called "TouchIntercept" to the WebView.

The two files which can be loaded into the WebView are md_preview and md_preview_dark.

```
<!doctype html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <script src="js/highlight.pack.js"></script>
    <script src="js/md_preview.js"></script>

    <link rel="stylesheet" href="css/github.css"/>

    <style>
        img { max-width: 100%; }
    </style>

</head>
<body>
    <div class="container" id="preview"></div>
</body>
```

```
</html>
```

Each specifies the stylesheet to use, and loads both the Javascript used to display the Markdown and the HighlightJS library used by GitHub to highlight code.

The body contains a single div element which is used to display the Markdown once it has been loaded.

Touch interception

There are two problems with the use of the `WebView` to display the README, both relating to its use within a `SwipeRefreshLayout` which is itself within a `ViewPager` with an `AppBar` layout behaviour.

Vertical scrolling

Each of the `Fragments` used throughout the app uses a `SwipeRefreshLayout` in order to allow the user to drag down from the top of a scrolling layout in order to refresh its contents.

Each of the scrolling children inside the `SwipeRefreshLayout` has a layout behaviour specified, which should result in the `ToolBar`, and `TabLayout` if applicable, being hidden as the user scrolls down.

This scrolling behaviour only occurs when the scrolling child implements `NestedScrollingChild` which the `WebView` does not.

Most of the methods in `NestedScrollingChild` are forward through to a `NestedScrollingChildHelper` which deals with the interpretation of the touch events.

`MarkdownWebView.java`

```
@Override
public boolean onTouchEvent(MotionEvent ev) {
    if(mInterceptTouchEvent && getParent() != null) {
        getParent().requestDisallowInterceptTouchEvent(mInterceptTouchEvent);
        return super.onTouchEvent(ev);
    }

    boolean rv = false;

    MotionEvent event = MotionEvent.obtain(ev);
    final int action = MotionEventCompat.getActionMasked(event);
    if(action == MotionEvent.ACTION_DOWN) {
        mNestedOffsetY = 0;
    }
    int eventY = (int) event.getY();
    event.offsetLocation(0, mNestedOffsetY);
    switch(action) {
        case MotionEvent.ACTION_MOVE:
            int deltaY = mLastY - eventY;
            // NestedPreScroll
    }
}
```

```
        if(dispatchNestedPreScroll(0, deltaY, mScrollConsumed,
mScrollOffset)) {
            deltaY -= mScrollConsumed[1];
            mLastY = eventY - mScrollOffset[1];
            event.offsetLocation(0, -mScrollOffset[1]);
            mNestedOffsetY += mScrollOffset[1];
        }
        rv = super.onTouchEvent(event);

        // NestedScroll
        if(dispatchNestedScroll(0, mScrollOffset[1], 0, deltaY,
mScrollOffset)) {
            event.offsetLocation(0, mScrollOffset[1]);
            mNestedOffsetY += mScrollOffset[1];
            mLastY -= mScrollOffset[1];
        }
        break;
    case MotionEvent.ACTION_DOWN:
        rv = super.onTouchEvent(event);
        mLastY = eventY;
        // start NestedScroll
        startNestedScroll(ViewCompat.SCROLL_AXIS_VERTICAL);
        break;
    case MotionEvent.ACTION_UP:
    case MotionEvent.ACTION_CANCEL:
        rv = super.onTouchEvent(event);
        // end NestedScroll
        stopNestedScroll();
        break;
    }
    return rv;
}

// Nested Scroll implements
@Override
public void setNestedScrollingEnabled(boolean enabled) {
    mChildHelper.setNestedScrollingEnabled(enabled);
}

@Override
public boolean isNestedScrollingEnabled() {
    return mChildHelper.isNestedScrollingEnabled();
}

@Override
public boolean startNestedScroll(int axes) {
    return mChildHelper.startNestedScroll(axes);
}

@Override
public void stopNestedScroll() {
    mChildHelper.stopNestedScroll();
}

@Override
public boolean hasNestedScrollingParent() {
    return mChildHelper.hasNestedScrollingParent();
}
```

```

@Override
public boolean dispatchNestedScroll(int dxConsumed, int dyConsumed, int
dxUnconsumed, int dyUnconsumed,
                                    int[] offsetInWindow) {
    return mChildHelper.dispatchNestedScroll(dxConsumed, dyConsumed,
dxUnconsumed, dyUnconsumed,
                                           offsetInWindow
);
}

@Override
public boolean dispatchNestedPreScroll(int dx, int dy, int[] consumed, int[]
offsetInWindow) {
    return mChildHelper.dispatchNestedPreScroll(dx, dy, consumed,
offsetInWindow);
}

@Override
public boolean dispatchNestedFling(float velocityX, float velocityY, boolean
consumed) {
    return mChildHelper.dispatchNestedFling(velocityX, velocityY, consumed);
}

@Override
public boolean dispatchNestedPreFling(float velocityX, float velocityY) {
    return mChildHelper.dispatchNestedPreFling(velocityX, velocityY);
}

```

The `onTouchEvent` forwards each `MotionEvent` through the helper in order to receive the scrolling offsets required to scroll multiple views in sync.

Horizontal scrolling

The second problem occurs when displaying code blocks.

When a code block overflows horizontally, as happens often on vertical mobile screens, it is expected to scroll horizontally.

This causes a problem when the `WebView` is displayed in a `ViewPager` because the horizontal touch movements are intercepted by the `ViewPager` and result in the entire `Fragment` being scrolled horizontally.

These events can be overridden by calling `requestDisallowInterceptTouchEvent` on the `WebView`'s parent, however we must only call this method for touch events which are on the code blocks, otherwise all events will be intercepted by the `webView` and the user will not be able to exit the `Fragment`.

This is achieved by adding touch listeners in Javascript and then notifying the `WebView` through Javascript interface methods

md_preview.js

```

function touchStart(event) {
    TouchIntercept.beginTouchIntercept();
}

```

```

function touchEnd(event) {
    TouchIntercept.endTouchIntercept();
}

function preview(md_html) {
    if(md_html == "") {
        return false;
    }
    document.getElementById("preview").innerHTML = md_html.replace(/\n/g,
"\n")
    var codes = document.getElementsByClassName('code');
    for(var i = 0; i < codes.length; i++) {
        codes[i].style.display = 'block';
        codes[i].style.wordWrap = 'normal';
        codes[i].style.overflowX = 'scroll';
        if(!codes[i].innerHTML.includes("license")) {
            hljs.highlightBlock(codes[i]);
        }
        codes[i].addEventListener("touchstart", touchStart, false);
        codes[i].addEventListener("touchend", touchEnd, false)
    }

    var pres = document.getElementsByTagName('pre');
    for(var i = 0; i < pres.length; i++) {
        pres[i].addEventListener("touchstart", touchStart, false);
        pres[i].addEventListener("touchend", touchEnd, false)
    }
    var tables = document.getElementsByTagName('table');
    for(var i = 0; i < tables.length; i++) {
        tables[i].addEventListener("touchstart", touchStart, false);
        tables[i].addEventListener("touchend", touchEnd, false)
    }
}

```

The Javascript above first sets the inner HTML of the preview div mentioned earlier, and then searches for each of the elements which scroll.

The style on each code element is set such that it scrolls on any overflow in x, and if it is not displaying a license it is highlighted.

All code, pre, and table elements are assigned event listeners for the "touchstart" and "touchend" events which call the Java methods in the WebView.

The two Java methods are beginTouchIntercept and endTouchIntercept which set the mInterceptTouchEvent flag.

MarkdownWebView.java

```

@JavascriptInterface
public void beginTouchIntercept() {
    mInterceptTouchEvent = true;
}

@JavascriptInterface
public void endTouchIntercept() {
    mInterceptTouchEvent = false;
}

```

In the `onTouchEvent` method shown in the section above, the flag is checked:

```
if(mInterceptTouchEvent && getParent() != null) {
    getParent().requestDisallowInterceptTouchEvent(mInterceptTouchEvent);
    return super.onTouchEvent(ev);
}
```

and the event is intercepted if the user is touching a code, pre, or table element.

Displaying short Markdown sections

In order to display GitHub Markdown formatted text in the Android `TextView`, custom spans are required.

CleanURLSpan

This span type was reference earlier.

It is used to display links to web addresses and email addresses.

CleanURLSpan.java

```
package com(tpb.mdtext.views.spans;

import android.content.Intent;
import android.graphics.Typeface;
import android.net.Uri;
import android.os.Parcel;
import android.support.annotation.Nullable;
import android.text.TextPaint;
import android.text.style.URLSpan;
import android.view.View;

import com(tpb.mdtext.MDPattern;
import com(tpb.mdtext.handlers.LinkClickHandler;

/**
 * Created by theo on 27/02/17.
 */

public class CleanURLSpan extends URLSpan {
    private LinkClickHandler mHandler;

    public CleanURLSpan(String url) {
        super(ensureValidURL(url));
    }

    public CleanURLSpan(String url, LinkClickHandler handler) {
        super(ensureValidURL(url));
        mHandler = handler;
    }

    @Override
    public void onClick(View widget) {
        if(mHandler == null) {
            final Intent i = new Intent(Intent.ACTION_VIEW);
            i.setData(Uri.parse(getURL()));
        }
    }
}
```

```

        widget.getContext().startActivity(i);
    } else {
        mHandler.onClick(getURL());
    }
}

private static String ensureValidURL(@Nullable String url) {
    if(url == null) return null;

    if(MDPattern.AUTOLINK_EMAIL_ADDRESS.matcher(url).matches()) {
        return "mailto:" + url;
    } else if(!MDPattern.IP_ADDRESS.matcher(url).matches() &&
!url.startsWith("https://") && !url.startsWith("http://")) {
        return "http://" + url;
    }
    return url;
}

@Override
public void updateDrawState(TextPaint ds) {
    super.updateDrawState(ds);
    // Links are bold without underline
    ds.setUnderlineText(false);
    ds.setTypeface(Typeface.DEFAULT_BOLD);
}

@Override
public int describeContents() {
    return 0;
}

@Override
public void writeToParcel(Parcel dest, int flags) {
}

CleanURLSpan(Parcel in) {
    super(in);
}

public static final Creator<CleanURLSpan> CREATOR = new
Creator<CleanURLSpan>() {
    @Override
    public CleanURLSpan createFromParcel(Parcel source) {
        return new CleanURLSpan(source);
    }

    @Override
    public CleanURLSpan[] newArray(int size) {
        return new CleanURLSpan[size];
    }
};

}
}

```

The CleanURLSpan uses the LinkClickHandler interface, which provides an onClick method for a URL

LinkClickHandler.java

```
package com(tpb.mdtext.handlers;
```

```
/*
 * Created by theo on 27/02/17.
 */

public interface LinkClickHandler {
    void onClick(String url);
}
```

This interface is used to allow capturing all link clicks which occur in a `TextView`. The `ensureValidURL` method checks if the link is an email address, formatting it accordingly. If the link is a web address, the correct protocol is prefixed. The `CleanURLESpan` overrides `updateDrawState` to remove the underline usually displayed on links, and to use a bold typeface.

HorizontalRuleSpan

The `HorizontalRuleSpan` solves the problem of drawing a line across the `TextView`. As trivial as this problem sounds, it cannot be achieved with any string of text. While a line could be drawn more easily with a box drawing character, specifically U+2500 which draws lines with no gap —— or the thicker U+2501 ———, these characters would not span the full width of the `TextView` without guesswork, approximations, and some luck. Once a layout pass has been completed, the number of characters per line of a `TextView` can be calculated by repeatedly measuring a string with the `TextView`'s `Paint`.

```
private static boolean isTextTooLong(TextView tv, String text) {
    final float textWidth = tv.getPaint().measureText(text);
    return (textWidth >= tv.getMeasuredWidth ());
}

private static boolean findCharactersPerLine(TextView tv) {
    String s = "";
    while(!isTextTooLong(tv, s)) {
        s += " ";
    }
    return s.length()
}
```

This method has numerous problems:

First, it relies on the `TextView` using a monospace font. Second, it requires a layout pass to have been completed. This means that in order to display the horizontal rule, the `TextView` would have to:

1. Check for horizontal spans when its text is set
2. Add a listener for its layout call

3. Within this listener, calculate the maximum number of characters which can be displayed per line
4. Replace each horizontal rule placeholder with a new string of the correct length
5. Redraw the entire `TextView`, and ensure that there isn't an infinite loop of redrawing the `TextView`

Clearly this is not a reasonable way to display horizontal rules in a `TextView`. Instead, a `ReplacementSpan` can be used to draw a line across the canvas.

HorizontalRuleSpan.java

```
package com(tpb.mdttext.views.spans;

import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.RectF;
import android.support.annotation.NonNull;
import android.text.style.ReplacementSpan;

/**
 * Created by theo on 02/03/17.
 */

public class HorizontalRuleSpan extends ReplacementSpan {

    private RectF mRectF;

    public HorizontalRuleSpan() {
        mRectF = new RectF();
    }

    @Override
    public int getSize(@NonNull Paint paint, CharSequence text, int start, int end, Paint.FontMetricsInt fm) {
        return 0;
    }

    @Override
    public void draw(@NonNull Canvas canvas, CharSequence text, int start, int end, float x, int top, int y, int bottom, @NonNull Paint paint) {
        final int mid = (top + bottom) / 2;
        final int quarter = (bottom - top) / 4;
        paint.setColor(Color.GRAY);
        mRectF.left = x;
        mRectF.top = mid - quarter;
        mRectF.right = x + canvas.getWidth();
        mRectF.bottom = mid + quarter;
        canvas.drawRect(mRectF, paint);
        final int eighth = quarter / 2;
        paint.setColor(Color.LTGRAY);
        mRectF.left += eighth;
        mRectF.right -= eighth;
        mRectF.top += eighth;
        mRectF.bottom -= eighth;
    }
}
```

```

        canvas.drawRect(mRectF, paint);
    }
}

```

The draw method in `HorizontalRuleSpan` draws a bordered rectangle by drawing two rectangles.

First, it calculates the mid-point of the space available to it (a single line).

Second, it calculates one quarter of the height of the space available to it.

Third, it assigns the given x position, and the calculated mid-point and quarter height to a `RectF` object in order to draw a rectangle across the canvas between the upper and lower

quartiles of the line. This makes the total area covered half of the available line. The second rectangle to be drawn fills half of the vertical space within the first rectangle.

One eighth of the height is calculated as half of the quarter, and the bounds of the rectangle are changed to give the new rectangle a border of this size.

The `Paint` colour is then changed to light grey and the new rectangle is drawn.

A horizontal rule



Separating some text

The only caveat to this method is that if the span it must be ensured that there is an empty line for the `HorizontalRuleSpan` to fill.

This span completes objective 9.b.ii

QuoteSpan

Android already includes a span for quotes, however it only draws a line to the start of the text and is the colour is not configurable, instead using blue (0, 255, 0).

QuoteSpan.java

```

package com(tpb.mdtext.views.spans;

import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.text.Layout;
import android.text.TextPaint;
import android.text.style.CharacterStyle;
import android.text.style.LeadingMarginSpan;

```

```

/**
 * Created by theo on 20/03/17.
 */

public class QuoteSpan extends CharacterStyle implements LeadingMarginSpan {
    private static final int STRIPE_WIDTH = 4;
    private static final int GAP_WIDTH = 8;
    private final int mColor;

    public QuoteSpan() {
        super();
        mColor = Color.WHITE;
    }

    public QuoteSpan(int color) {
        super();
        mColor = color;
    }

    public int describeContents() {
        return 0;
    }

    @Override
    public void updateDrawState(TextPaint tp) {
        tp.setAlpha((tp.getColor() & 0x00FFFFFF) > 0x800000 ? 179 : 138);
    }

    @Override
    public int getLeadingMargin(boolean first) {
        return STRIPE_WIDTH + GAP_WIDTH;
    }

    @Override
    public void drawLeadingMargin(Canvas c, Paint p, int x, int dir,
                                 int top, int baseline, int bottom,
                                 CharSequence text, int start, int end,
                                 boolean first, Layout layout) {
        final Paint.Style style = p.getStyle();
        final int color = p.getColor();
        p.setStyle(Paint.Style.FILL);
        p.setColor(mColor);
        c.drawRect(x, top, x + dir * STRIPE_WIDTH, bottom, p);
        p.setStyle(style);
        p.setColor(color);
    }
}

```

QuoteSpan extends CharacterStyle, allowing it to set modify the TextPaint, as well as implementing LeadingMarginSpan in order to draw the quote line.

The QuoteSpan follows the material guidelines for secondary text, which specify using an opacity of 54% for secondary text using a dark text on light backgrounds, and using 70%

opacity for secondary text using a white text on light backgrounds.

Each Android colour is stored as an integer with alpha, red, green, and blue occupying each byte.

The alpha value is stripped from the colour by and-ing it with 00FFFFFF, and it is

then compared to the middle colour value to approximate whether it is a light or dark colour.

In `drawLeadingMarginSpan` the original style and colour are saved, and a rectangle is drawn across the whole vertical space of the line, and across 4 pixels horizontally.

The original style and colour are then saved.

ListNumberSpan

`ListNumberSpan` implements `LeadingMarginSpan` and is used to draw the keys in an ordered list.

HTML ordered lists can specify four types of keys.

- Numbers, indexed from 1
- Letters, indexed from a
- Capital letters, indexed from A
- Roman numerals, indexed from i
- Capital Roman numerals, indexed from I

The `ListNumberSpan` needs to specify the margin for for list indentation, as well as drawing the list item number.

ListNumberSpan.java

```
package com(tpb.mdtext.views.spans;

import android.graphics.Canvas;
import android.graphics.Paint;
import android.support.annotation.NonNull;
import android.text.Layout;
import android.text.Spanned;
import android.text.TextPaint;
import android.text.style.LeadingMarginSpan;

import com(tpb.mdtext.TextUtils;

import java.util.TreeMap;

/**
 * Created by theo on 02/03/17.
 */

public class ListNumberSpan implements LeadingMarginSpan {
    private final String mNumber;
    private final int mTextWidth;

    public ListNumberSpan(TextPaint textPaint, int number, ListType type) {
        mNumber = ListType.getFormattedNumber(number + type.start,
        type).concat(".");
        mTextWidth = (int) textPaint.measureText(mNumber);
    }
}
```

```
@Override
public int getLeadingMargin(boolean first) {
    return mTextWidth;
}

@Override
public void drawLeadingMargin(Canvas c, Paint p, int x, int dir, int top,
int baseline,
                                int bottom, CharSequence text, int start,
int end,
                                boolean first, Layout l) {
    //Check if we are at the correct depth to draw text rather than just
spacing
    if(text instanceof Spanned) {
        if(((Spanned) text).getSpanStart(this) == start) {
            c.drawText(mNumber, x, baseline, p);
        }
    }
}

public enum ListType {

    NUMBER,
    LETTER,
    LETTER_CAP,
    ROMAN,
    ROMAN_CAP;

    int start = 0;

    public static ListType fromString(@NonNull String val) {
        if(val.isEmpty()) return NUMBER;
        if(TextUtils.isInteger(val)) {
            final ListType num = NUMBER;
            num.start = Integer.parseInt(val) - 1;
            return num;
        } else {
            switch(val.charAt(0)) {
                case 'a': return LETTER;
                case 'A': return LETTER_CAP;
                case 'i': return ROMAN;
                case 'I': return ROMAN_CAP;
                default: return NUMBER;
            }
        }
    }
}

public static String getFormattedNumber(int num, ListType type) {
    switch(type) {
        case LETTER:
            return getLetter(num);
        case LETTER_CAP:
            return getLetter(num).toUpperCase();
        case ROMAN:
            return getRoman(num);
        case ROMAN_CAP:
            return getRoman(num).toUpperCase();
        default:
```

```

        return Integer.toString(num);
    }

private static String getLetter(int num) {
    final StringBuilder builder = new StringBuilder();
    while(num-- > 0) { //1 = a, not 0 = a
        final int rmdr = num % 26;
        builder.append((char) (rmdr + 'a'));
        num = (num - rmdr) / 26;
    }
    return builder.reverse().toString();
}

private static TreeMap<Integer, String> map = new TreeMap<>();

static {
    map.put(1000, "m");
    map.put(900, "cm");
    map.put(500, "d");
    map.put(400, "cd");
    map.put(100, "c");
    map.put(90, "xc");
    map.put(50, "l");
    map.put(40, "xl");
    map.put(10, "x");
    map.put(9, "xi");
    map.put(5, "v");
    map.put(4, "iv");
    map.put(1, "i");
}
}

private static String getRoman(int num) {
    final int l = map.floorKey(num);
    if(l == num) {
        return map.get(num);
    }
    return map.get(l) + getRoman(num - l);
}
}

}

```

The `ListNumberSpan` constructor takes a `TextPaint`, which is used to calculate the width of the list item text, the number to display, and the `ListType` enum.

The string `mNumber` is calculated with `getFormattedNumber` and concatenated with `". "`.

The width of `mNumber` is then calculated.

In `getLeadingMargin` the text width calculated in the constructor is returned. `drawLeadingMargin` has to check if the start position is the start position of the span, in order to ensure that only the span at the deepest indentation level draws its text.

First, it is checked that the text is an instance of `Spanned`, as the position cannot be found otherwise.

If the text is an instance, the span start position returned from the cast Spanned is checked against the start position passed to `drawLeadingMargin`. If these values are the same, the number text is drawn with the parameters passed to `drawLeadingMargin`.

Calculating list number formats

The `fromString` method in `ListType` is used to convert the type parameter in an ordered list tag to a `ListType` enum.

If the string is empty, `NUMBER` is returned as the default type.

If the string is an integer, the integer value is parsed to become the starting index for the list, and `NUMBER` is returned.

Otherwise, the method switches on the first character in the string:

- If it is 'a', `LETTER` is returned
- If it is 'A' `LETTER_CAP` is returned
- If it is 'i' `ROMAN` is returned
- If it is 'I' `ROMAN_CAP` is returned
- If it is none of the above, `NUMBER` is returned

Once the `ListType` has been calculated, it needs to be formatted into a string.

This is done in `getFormattedNumber` which takes an integer and a `ListType`.

If the type is `LETTER`, `getLetter` is returned.

```
private static String getLetter(int num) {
    final StringBuilder builder = new StringBuilder();
    while(num-- > 0) { //1 = a, not 0 = a
        final int rmdr = num % 26;
        builder.append((char) (rmdr + 'a'));
        num = (num - rmdr) / 26;
    }
    return builder.reverse().toString();
}
```

This method converts an integer value to a base 26 string.

It does this by computing the remainder of dividing the number by 26, and offsetting this by the numeric value of the character 'a' (97).

The remainder is then subtracted from the number, and it is divided by 26.

This process repeats until the reversed form of the string has been generated.

The `StringBuilder` is then reversed and its value is returned.

If the type is `LETTER_CAP`, `getLetter` is called, and its return value shifted to uppercase.

If the type is `ROMAN`, `getRoman` is called.

`getRoman` uses a `TreeMap` to store the unique roman numeral values of different integers.

The `TreeMap` maintains the order of the mapped pairs.

The greatest key less than or equal to the given key is found with `floorKey`. If this value is the same as the number, there is a direct match and the string is returned.

Otherwise, the string at the lowest key is concatenated with a recursive call for the value of the number minus the lowest key value.

```
private static TreeMap<Integer, String> map = new TreeMap<>();

static {
    map.put(1000, "m");
    map.put(900, "cm");
    map.put(500, "d");
    map.put(400, "cd");
    map.put(100, "c");
    map.put(90, "xc");
    map.put(50, "l");
    map.put(40, "xl");
    map.put(10, "x");
    map.put(9, "xi");
    map.put(5, "v");
    map.put(4, "iv");
    map.put(1, "i");
}

private static String getRoman(int num) {
    final int l = map.floorKey(num);
    if(l == num) {
        return map.get(num);
    }
    return map.get(l) + getRoman(num - l);
}
```

Example:

Converting 46 to roman numerals.

The first call calls `map.floorKey(46)`, which returns 40.

As $40 \neq 46$, the function returns `map.get(40) + getRoman(46 - 40)`.

The recursive call `getRoman(6)` calls `map.floorKey(6)`, which returns 5.

As $5 \neq 6$, the function returns `map.get(5) + getRoman(6 - 5)`.

The recursive call `getRoman(1)` calls `map.floorKey(1)` which returns 1.

As $1 == 1$, the function returns `map.get(1)`.

As there are no further recursive calls, the function returns `map.get(40) + map.get(5) + map.get(1)`.

This evaluates to "xl" + "v" + "i", which is correct.

This span completes objective 9.iv.b

RoundedBackgroundEndSpan

GitHub labels are shown with coloured backgrounds which have rounded corners.

bug**duplicate****enhancement****question**

The background colour can be achieved using a `BackgroundColorSpan`, which sets the background colour of the `TextPaint`.

However, this draws a rectangular block of colour behind the segment of text, and doesn't allow drawing any other shape.

The problem can be solved by drawing the spans separately, adding a `ReplacementSpan` at each end of the `BackgroundColorSpan` which draws the rounded corners.

RoundedBackgroundEndSpan.java

```
package com(tpb.mdtext.views.spans;

import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.RectF;
import android.support.annotation.NonNull;
import android.text.style.ReplacementSpan;

/**
 * Created by theo on 28/03/17.
 */

public class RoundedBackgroundEndSpan extends ReplacementSpan {

    private int mCharacterWidth;
    private final int mBgColor;
    private final boolean mIsEndSpan;

    public RoundedBackgroundEndSpan(int bgColor, boolean isEndSpan) {
        mBgColor = bgColor;
        mIsEndSpan = isEndSpan;
    }

    @Override
    public int getSize(@NonNull Paint paint, CharSequence text, int start, int end, Paint.FontMetricsInt fm) {
        mCharacterWidth = (int) paint.measureText("t");
        return mCharacterWidth;
    }

    @Override
    public void draw(@NonNull Canvas canvas, CharSequence text, int start, int end, float x, int top, int y, int bottom, @NonNull Paint paint) {
        RectF rect = new RectF(x, top, x + mCharacterWidth, bottom);
        paint.setColor(mBgColor);
        canvas.drawRoundRect(rect, paint.getTextSize() / 6,
            paint.getTextSize() / 6, paint);
        if(mIsEndSpan) {
            rect = new RectF(x, top, (x + x + mCharacterWidth) / 2, bottom);
        } else {
            rect = new RectF((x + x + mCharacterWidth) / 2, top, x +
                mCharacterWidth, bottom);
        }
    }
}
```

```

        canvas.drawRect(rect, paint);
    }
}

```

The RoundedBackgroundEndSpan will either be drawn at the start or the end of a BackgroundColorSpan. The direction in which it draws is controlled by mIsEndSpan, which is passed to the constructor along with the background colour to use.

The RoundedBackgroundEndSpan uses one character of space, which is measured in getSize.

The drawing takes place with two calls.

The CSS on the GitHub website uses a corner size of one sixth the text size, which can be duplicated when painting the rounded rectangle.

The rounded rectangle is drawn across the measured character width.

At this point, the rounded section of the end span has been drawn, but it is rounded on both sides which would leave a gap between the rounded rectangle and the text.

The next draw call is dependent on whether the span is at the start or end of the BackgroundColorSpan.

If the span is at the start of the BackgroundColorSpan, it needs to fill in the space after itself and before the BackgroundColorSpan whereas if it is after the BackgroundColorSpan it needs to fill in the space after the BackgroundColorSpan and before itself.

If the span is at the end of the BackgroundColorSpan the rectangle is created from the given x position (which is the end of the BackgroundColorSpan), to the x position

halfway between the start and end of the span.

If the span is at the start of the BackgroundColorSpan the rectangle is created from the halfway position to the end of the span.

The rectangle is then drawn.

InlineCodeSpan

The InlineCodeSpan is used to draw short sections of code within the TextView.

InlineCodeSpan.java

```

package com(tpb.mdtext.views.spans;

import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Typeface;
import android.support.annotation.IntRange;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.text.TextPaint;
import android.text.style.ReplacementSpan;

```

```

/*
 * Created by theo on 22/03/17.
 */

public class InLineCodeSpan extends ReplacementSpan {
    private final float mTextSize;

    private float mPadding;

    public InLineCodeSpan(float textSize) {
        mTextSize = textSize;
    }

    @Override
    public void updateDrawState(TextPaint tp) {
        tp.setTextSize(mTextSize);
        tp.setTypeface(Typeface.MONOSPACE);
    }

    @Override
    public int getSize(@NonNull Paint paint,
                       CharSequence text,
                       @IntRange(from = 0) int start,
                       @IntRange(from = 0) int end,
                       @Nullable Paint.FontMetricsInt fm) {
        mPadding = paint.measureText("c");
        return (int) (paint.measureText(text, start, end) + mPadding * 2);
    }

    @Override
    public void draw(@NonNull Canvas canvas,
                    CharSequence text,
                    @IntRange(from = 0) int start,
                    @IntRange(from = 0) int end,
                    float x,
                    int top,
                    int y,
                    int bottom,
                    @NonNull Paint paint) {
        canvas.drawText(text, start, end, x + mPadding, y, paint);
        paint.setColor(Color.GRAY);
        paint.setAlpha(50);
        final int leading = paint.getFontMetricsInt().leading;
        canvas.drawRect((int) x, top - leading, (int) x + canvas.getWidth(),
                        bottom + leading, paint);
    }
}

```

InLineCodeSpan extends ReplacementSpan and is used to draw short segments of code.

The InLineCodeSpan sets the typeface to monospace in updateDrawState as well as setting the font size to a value provided in the constructor.

In getSize, the padding size is computed as the width of a character before the width is returned as the measured width of the text plus two times the padding.

Finally, in `draw` the `InlineCodeSpan` first draws the text, offset by the padding computed earlier, and then draws the code background.

The background is an opaque grey rectangle which fills the full width of the `TextView`.

```
System.out.println("Hello");
```

Standard text

```
if(Math.random() > 0.5) {  
    System.out.println("Do something");  
}
```

This span completes objective 9.b.i.1

Dealing with more complex content

As explained earlier, some content, notably large code segments and tables, which is usually displayed on the desktop is not well suited for small vertical displays.

As such, it would not be sensible to display this content directly in the `TextView`. Instead, placeholders in the `TextView` can be used to link to a more suitable method for displaying the content.

CodeSpan

The first problem to be dealt with is larger blocks of code.

As was written in the markdown section of the background information, code blocks are written

``` Language

Some code

...

where the “Language” string is optional.

A `CodeSpan` needs to be a large enough item in the `TextView` that it can be easily clicked.

It must also be obvious to the user that the span should be clicked.

As such, the span is styled like a button.

## CodeSpan.java

```
package com(tpb.mdtext.views.spans;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.PorterDuff;
import android.graphics.PorterDuffColorFilter;
import android.graphics.RectF;
import android.graphics.drawable.Drawable;
import android.support.annotation.IntRange;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.text.style.ReplacementSpan;
import android.util.Base64;

import com(tpb.mdtext.TextUtils;
import com(tpb.mdtext.handlers.CodeClickHandler;

import org.sufficientlysecure.htmltextview.R;

import java.lang.ref.WeakReference;

/**
 * Created by theo on 06/03/17.
 */

public class CodeSpan extends ReplacementSpan implements
WrappingClickableSpan.WrappedClickableSpan {
 private WeakReference<CodeClickHandler> mHandler;
 private String mCode;
 private String mLanguge;
 private static String mLangugeFormatString = "%1$s code";
 private static String mNoLanguageString = "Code";
 private static Bitmap mCodeBM;
 private PorterDuffColorFilter mBMPFilter;
 private int mBaseOffset = 5;

 public CodeSpan(String code, CodeClickHandler handler) {
 setCode(code);
 mHandler = new WeakReference<>(handler);
 }

 public void setCode(String code) {
 final int ls = code.indexOf('[');
 final int le = code.indexOf(']');
 if(ls != -1 && le != -1 && le - ls > 0 && le < code.indexOf("\n")) {
 mLanguge = TextUtils.capitalizeFirst(code.substring(ls + 1, le));
 mCode = code.substring(le + 1);
 } else {
 mCode = code;
 }
 mCode = new String(Base64.decode(mCode, Base64.DEFAULT));
 }

 @Override
```

```

 public int getSize(@NonNull Paint paint, CharSequence text, @IntRange(from
= 0) int start, @IntRange(from = 0) int end, @Nullable Paint.FontMetricsInt
fm) {
 mBaseOffset = (int) paint.measureText("c");
 return 0;
}

@Override
public void draw(@NonNull Canvas canvas, CharSequence text, @IntRange(from
= 0) int start, @IntRange(from = 0) int end, float x, int top, int y, int
bottom, @NonNull Paint paint) {
 paint.setTextSize(paint.getTextSize() - 1);
 final int textHeight = paint.getFontMetricsInt().descent -
paint.getFontMetricsInt().ascent;

 int offset = mBaseOffset;
 if(mCodeBM != null) offset += mCodeBM.getWidth();

 final int textStart = top + textHeight / 4;

 if(mLanguage != null && !mLanguage.isEmpty()) {
 canvas.drawText(String.format(mLanguageFormatString, mLanguage), x
+ mBaseOffset + offset, textStart,
 paint
);
 } else {
 canvas.drawText(mNoLanguageString, x + mBaseOffset + offset,
textStart, paint);
 }

 paint.setStyle(Paint.Style.STROKE);
 paint.setStrokeWidth(mBaseOffset / 4);
 canvas.drawRoundRect(new RectF(x, top + top - bottom, x +
canvas.getWidth(), bottom), 7, 7,
 paint
);

 if(mCodeBM != null) {
 if(mBMFilter == null) mBMFilter = new
PorterDuffColorFilter(paint.getColor(), PorterDuff.Mode.SRC_IN);
 paint.setColorFilter(mBMFilter);
 canvas.drawBitmap(mCodeBM, x + mBaseOffset, textStart -
textHeight, paint);
 }
}

public void onClick() {

 if(mHandler.get() != null) mHandler.get().codeClicked(mCode,
mLanguage);
}

public static void initialise(Context context) {
 final Drawable drawable =
context.getResources().getDrawable(R.drawable.ic_code);
 final Bitmap bitmap =
Bitmap.createBitmap(drawable.getIntrinsicWidth(),
 drawable.getIntrinsicHeight(), Bitmap.Config.ARGB_8888
);
}

```

```

final Canvas canvas = new Canvas(bitmap);
drawable.setBounds(0, 0, canvas.getWidth(), canvas.getHeight());
drawable.draw(canvas);
mCodeBM = bitmap;
mLanguageFormatString =
context.getString(R.string.code_span_language_format);
mNoLanguageString = context.getString(R.string.code_span_no_language);
}

public static boolean isInitialised() {
 return mCodeBM != null;
}

}

```

The CodeSpan is more complicated than the other spans because it also draws a BitMap image and deals with being clicked.

mHandler is a WeakReference to a CodeClickHandler, which is called when the span is clicked.

The code string and language are stored when the span is created, and the format strings are used to format the displayed text dependent on whether or not the language has been set.

setCode is used to check if a language has been embedded in the code string. While it has not been explained yet, when the code is parsed it will begin with two square brackets followed by a newline, with the language between the two brackets.

If the brackets exist, and are before the first index of a newline, the language is extracted.

getSize measures the size of a single character, and then returns 0, as the span is not drawing any text in the normal manner.

## Drawing

draw is used to draw a button shape, the “button” text, and the BitMap.

First, the text size is reduced, as the text is to be drawn between the bounds of the “button”.

Next, the text height is computed from the Paint font metrics.

The offset is used when drawing text to ensure that it is correctly aligned in the horizontal.

If the BitMap is non-null, its width is added to the offset.

The vertical start position of the text is computed as the top position, plus a quarter of the text height.

If the language is non-null and non-empty, the formatted string containing the language is drawn.

Otherwise, the no-language string is drawn.

In order to draw the “button” shape, the `Paint` style is changed to `stroke`, and its width set to one quarter of the base offset, which is one character width.

A round rectangle is then drawn across two lines of vertical space, and the entire width of the canvas.

Finally, if the `Bitmap` has been initialised, a colour filter is set to ensure that it is the same colour as the text, and it is drawn at the start of the span, at a vertical position

the same as the start of the text.

### Initialisation

As the `CodeSpan` has no access to a `Context` instance, it cannot load items from resources.

This is done with a static initialiser which is called in the custom `TextView` written for displaying markdown.

This allows the `Bitmap` to be loaded and the correct strings to be loaded for a particular language.

### Named code block

{ } Python code

### Un-named block

{ } Code

This span completes objective 9.b.i.2

### TableSpan

The `TableSpan` is structured in a very similar manner to `CodeSpan` as it also draws a “button” style span across two lines, and draws a bitmap if it has been initialised.

### TableSpan.java

```
package com(tpb.mdtext.views.spans;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.PorterDuff;
import android.graphics.PorterDuffColorFilter;
import android.graphics.RectF;
import android.graphics.drawable.Drawable;
import android.support.annotation.IntRange;
```

```
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.text.style.ReplacementSpan;

import com(tpb).mdtext.handlers.TableClickHandler;

import org.sufficientlysecure.htmltextview.R;

import java.lang.ref.WeakReference;

/**
 * Created by theo on 08/04/17.
 */

public class TableSpan extends ReplacementSpan implements
WrappingClickableSpan.WrappedClickableSpan {

 private WeakReference<TableClickHandler> mHandler;
 private static String mTableString = "Table";
 private static Bitmap mTableBM;
 private PorterDuffColorFilter mBMPFilter;
 private String mHtml;
 private int mBaseOffset = 7;

 public TableSpan(String html, TableClickHandler handler) {
 mHtml = html;
 mHandler = new WeakReference<>(handler);
 }

 @Override
 public int getSize(@NonNull Paint paint, CharSequence text, @IntRange(from =
0) int start, @IntRange(from = 0) int end, @Nullable Paint.FontMetricsInt
fm) {
 mBaseOffset = (int) paint.measureText("c");
 return 0;
 }

 @Override
 public void draw(@NonNull Canvas canvas, CharSequence text, @IntRange(from =
0) int start, @IntRange(from = 0) int end, float x, int top, int y, int
bottom, @NonNull Paint paint) {
 paint.setTextSize(paint.getTextSize() - 1);
 final int textHeight = paint.getFontMetricsInt().descent -
paint.getFontMetricsInt().ascent;

 int offset = mBaseOffset;
 if(mTableBM != null) offset += mTableBM.getWidth();

 final int textStart = top + textHeight / 4;

 canvas.drawText(mTableString, x + mBaseOffset + offset, textStart,
paint);

 paint.setStyle(Paint.Style.STROKE);
 paint.setStrokeWidth(mBaseOffset / 4);
 canvas.drawRoundRect(new RectF(x, top + top - bottom, x +
canvas.getWidth(), bottom), 7, 7,
 paint
);
}
```

```

 if(mTableBM != null) {
 if(mBMFilter == null)
 mBMFilter = new PorterDuffColorFilter(paint.getColor(),
PorterDuff.Mode.SRC_IN);
 paint.setColorFilter(mBMFilter);
 canvas.drawBitmap(mTableBM, x + mBaseOffset, textStart -
textHeight, paint);
 }
 }

 public void onClick() {
 if(mHandler.get() != null) mHandler.get().onClick(mHtml);
 }

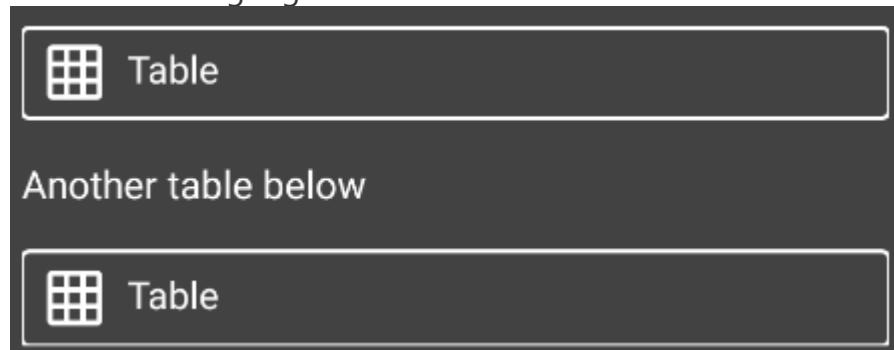
 public static void initialise(Context context) {
 final Drawable drawable =
context.getResources().getDrawable(R.drawable.ic_table);
 final Bitmap bitmap =
Bitmap.createBitmap(drawable.getIntrinsicWidth(),
 drawable.getIntrinsicHeight(), Bitmap.Config.ARGB_8888
);
 final Canvas canvas = new Canvas(bitmap);
 drawable.setBounds(0, 0, canvas.getWidth(), canvas.getHeight());
 drawable.draw(canvas);
 mTableBM = bitmap;
 mTableString = context.getString(R.string.table_span);
 }

 public static boolean isInitialised() {
 return mTableBM != null;
 }

}

```

The key differences are that `TableSpan` only ever draws a constant string, and that it takes a `TableClickHandler` which only takes one parameter, the table HTML, rather than code and a language.



This span completes objective 9.b.xi

### **ClickableImageSpan**

`ClickableImageSpan` extends `ImageSpan` and is used to implement click listening for the `ImageClickHandler`.

It also ensures that the actual drawable is returned from a `URLDrawable`, which will be explained in the "Image loading and caching" section.

### **ClickableImageSpan.java**

```
package com(tpb.mdtext.views.spans;

import android.graphics.drawable.Drawable;
import android.text.style.ImageSpan;

import com(tpb.mdtext.handlers.ImageClickHandler;
import com(tpb.mdtext.imagegetter.HttpImageGetter;

import java.lang.ref.WeakReference;

/**
 * Created by theo on 22/04/17.
 */

public class ClickableImageSpan extends ImageSpan implements
WrappingClickableSpan.WrappedClickableSpan {

 private WeakReference<ImageClickHandler> mImageClickHandler;

 public ClickableImageSpan(Drawable d, ImageClickHandler handler) {
 super(d);
 mImageClickHandler = new WeakReference<>(handler);
 }

 @Override
 public Drawable getDrawable() {
 if(super.getDrawable() instanceof HttpImageGetter.URLDrawable &&
((HttpImageGetter.URLDrawable) super.getDrawable()).getDrawable() != null) {
 return ((HttpImageGetter.URLDrawable)
super.getDrawable()).getDrawable();
 }
 return super.getDrawable();
 }

 @Override
 public void onClick() {
 if(mImageClickHandler.get() != null) {
 mImageClickHandler.get().imageClicked(getDrawable());
 }
 }
}
```

This span completes objective 9.b.iii.3, as well as 9.b.iii.5 once click handling is implemented below.

### **Handling clicks**

ReplacementSpans have not click listeners.

Clicks on individual spans in a TextView are handled by ClickableSpans and the MovementMethod.

ClickableSpan is an abstract class extending CharacterStyle and implementing UpdateAppearance. It sets the TextPaint colour to the link colour, and underlines the clickable content. ClickableSpan has the abstract method onClick which takes the View which was clicked.

The calling of onClick is handled by the MovementMethod.

A MovementMethod "provides cursor positioning, scrolling and text selection functionality in a TextView".

The TextView delegates handling of key events and touches to the movement method for content navigation.

### Handling clicks on ReplacementSpans

As ClickableSpan is an abstract class, ReplacementSpans cannot directly implement the onClickmethod.

Instead, their click handling must be handled by a ClickableSpan which is set across the same subsection of text as the ReplacementSpan.

### WrappingClickableSpan.java

```
package com(tpb.mdtext.views.spans;

import android.support.annotation.NonNull;
import android.text.style.ClickableSpan;
import android.view.View;

/**
 * Created by theo on 11/04/17.
 */

public class WrappingClickableSpan extends ClickableSpan {

 private final WrappedClickableSpan mWrappedClickableSpan;

 public WrappingClickableSpan(@NonNull WrappedClickableSpan child) {
 mWrappedClickableSpan = child;
 }

 @Override
 public void onClick(View widget) {
 mWrappedClickableSpan.onClick();
 }

 public interface WrappedClickableSpan {

 void onClick();

 }
}
```

As was shown above, `CodeSpan`, `TableSpan` and `ImageSpan` implement `WrappedClickableSpan` which allows touch events on a parent `ClickableSpan` to be forwarded to the `ReplacementSpan`.

### Stopping the `onClickHandler`

The problem with having clickable elements in the `TextView` is that it interferes with any click listeners set on the `TextView` itself.

This problem is solved with a custom `MovementMethod` and `TextView`.

### **ClickableMovementMethod.java**

```
package com(tpb.mdtext;

import android.text.Layout;
import android.text.Selection;
import android.text.Spannable;
import android.text.method.LinkMovementMethod;
import android.text.method.Touch;
import android.text.style.ClickableSpan;
import android.view.MotionEvent;
import android.widget.TextView;

import com(tpb.mdtext.views.MarkdownTextView;

/**
 * Copied from http://stackoverflow.com/questions/8558732
 */
public class ClickableMovementMethod extends LinkMovementMethod {

 private static ClickableMovementMethod instance;

 private ClickableMovementMethod() {}

 public static ClickableMovementMethod getInstance() {
 if(instance == null) instance = new ClickableMovementMethod();
 return instance;
 }

 @Override
 public boolean onTouchEvent(final TextView widget, final Spannable buffer,
MotionEvent event) {
 final int action = event.getAction();

 if(action == MotionEvent.ACTION_UP || action ==
MotionEvent.ACTION_DOWN) {

 int x = (int) event.getX();
 int y = (int) event.getY();

 x -= widget.getTotalPaddingLeft();
 y -= widget.getTotalPaddingTop();

 x += widget.getScrollX();
```

```

 y += widget.getScrollY();

 final Layout layout = widget.getLayout();
 final int line = layout.getLineForVertical(y);
 final int off = layout.getOffsetForHorizontal(line, x);

 final ClickableSpan[] clickable = buffer.getSpans(off, off,
ClickableSpan.class);

 if(clickable.length != 0) {
 if(action == MotionEvent.ACTION_UP) {
 clickable[0].onClick(widget);
 triggerSpanHit(widget);
 }
 return true;
 } else {
 Selection.removeSelection(buffer);
 Touch.onTouchEvent(widget, buffer, event);
 return false;
 }
 }

 return Touch.onTouchEvent(widget, buffer, event);
}

private void triggerSpanHit(TextView widget) {
 if(widget instanceof MarkdownTextView) {
 ((MarkdownTextView) widget).setSpanHit();
 }
}
}

```

The `ClickableMovementMethod` extends `LinkMovementMethod` and overrides `onTouchEvent` to deal with all clickable spans, as well as notifying the `TextView`.

If the event action is an up or down movement, the event is captured.

The x and y positions are collected from the event, and then offset by both the padding and scroll position of the `TextView`.

The `TextView` layout is then used to calculate the line of text, and the offset within the line for the click position.

Using this offset, the array of `ClickableSpans` present at this position is found from the buffer.

If the array is not empty, and the event type is up (The end of a click) the `ClickableSpan onClick` method is called, and the span hit is triggered on the `TextView`. Returning

true results in further events being passed to the `MovementMethod`.

If the array is empty, the selection is removed, the touch event is triggered, and false is returned so that no further events in this chain are passed to the `MovementMethod`.

Within the `TextView`, `setSpanHit` is used to set a flag for triggering click events.

Usually, to handle click events for a `View`, one would call `setOnClickListener` which would then be called when the `TextView` is clicked.

The problem with this is that the `OnClickListener` would receive span click events. To solve this problem, the `TextView` itself implements `OnClickListener`.

The `TextView` also overrides `setOnClickListener`. In this method it stores the `onClickListener`.

In `onClick`, it checks if the span hit flag is false, and the listener is non null, and if both of these are true it forwards the click to the listener.

It then sets the span hit flag back to false.

## MarkdownTextView

`MarkdownTextView` is the `TextView` descendent used for displaying markdown. It handles click handling for links, images, tables, and code, as well as dealing with background parsing of content and caching of parsed content.

## Handlers

There are four handler interfaces used for click events on different items in the `MarkdownTextView`.

There are also three default implementations of these interfaces.

### CodeClickHandler.java

```
package com(tpb.mdtext.handlers;

import android.support.annotation.Nullable;

/**
 * Created by theo on 27/02/17.
 */

public interface CodeClickHandler {
 void codeClicked(String code, @Nullable String language);
}
```

### ImageClickHandler.java

```
package com(tpb.mdtext.handlers;

import android.graphics.drawable.Drawable;

/**
 * Created by theo on 27/02/17.
 */

public interface ImageClickHandler {
 void imageClicked(Drawable drawable);
}
```

```
}
```

### LinkClickHandler.java

```
package com(tpb.mdtext.handlers;

/**
 * Created by theo on 27/02/17.
 */

public interface LinkClickHandler {
 void onClick(String url);
}
```

### TableClickHandler.java

```
package com(tpb.mdtext.handlers;

/**
 * Created by theo on 08/04/17.
 */

public interface TableClickHandler {
 public void onClick(String html);
}
```

The default implementations  
of CodeClickHandler, ImageClickHandler and TableClickHandler are all dialogs used  
to show the content over a larger area.

They can be replaced with any other implementation of their respective  
interfaces.

### CodeDialog.java

```
package com(tpb.mdtext.dialogs;

import android.app.AlertDialog;
import android.app.Dialog;
import android.content.Context;
import android.support.annotation.Nullable;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;

import com.pddstudio.highlightjs.HighlightJsView;
import com.pddstudio.highlightjs.models.Language;
import com.pddstudio.highlightjs.models.Theme;
import com(tpb.mdtext.handlers.CodeClickHandler;

import org.sufficientlysecure.htmltextview.R;

/***
```

```

 * Created by theo on 27/02/17.
 */

public class CodeDialog implements CodeClickHandler {

 private Context mContext;

 public CodeDialog(Context context) {
 mContext = context;
 }

 @Override
 public void codeClicked(String code, @Nullable String language) {
 final AlertDialog.Builder builder = new AlertDialog.Builder(mContext);
 final LayoutInflater inflater = LayoutInflater.from(mContext);

 final View view = inflater.inflate(R.layout.dialog_code, null);

 builder.setView(view);

 final HighlightJsView wv = (HighlightJsView)
view.findViewById(R.id.dialog_highlight_view);
 wv.setTheme(Theme.ANDROID_STUDIO);

 if(language != null) wv.setHighlightLanguage(getLanguage(language));
 wv.setSource(code);
 final Dialog dialog = builder.create();

 dialog.getWindow()
 .setLayout(ViewGroup.LayoutParams.MATCH_PARENT,
ViewGroup.LayoutParams.MATCH_PARENT);
 dialog.getWindow().requestFeature(Window.FEATURE_NO_TITLE);
 dialog.show();
 }

 private static Language getLanguage(String lang) {
 for(Language l : Language.values()) {
 if(l.toString().equalsIgnoreCase(lang)) return l;
 }
 return Language.AUTO_DETECT;
 }
}

```

The CodeDialog creates a dialog to display a `HighlightJsView`, which is a `WebView` with the `highlightjs` library embedded.

It also attempts to find the correct language for highlighting the code. This completes objective 9.b.i.2

## ImageDialog.java

```

package com(tpb.mdtext.dialogs;

import android.app.AlertDialog;
import android.app.Dialog;
import android.content.Context;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;

```

```
import android.graphics.drawable.Drawable;
import android.view.LayoutInflater;
import android.view.View;
import android.view.WindowManager;

import com(tpb).mdtext.handlers.ImageClickHandler;

import org.sufficientlysecure.htmltextview.R;

/**
 * Created by theo on 27/02/17.
 */

public class ImageDialog implements ImageClickHandler {

 private final Context mContext;

 public ImageDialog(Context context) {
 mContext = context;
 }

 @Override
 public void imageClicked(Drawable drawable) {
 final AlertDialog.Builder builder = new AlertDialog.Builder(mContext);
 final LayoutInflater inflater = LayoutInflater.from(mContext);

 final View view = inflater.inflate(R.layout.dialog_image, null);

 builder.setView(view);

 final FillingImageView fiv = (FillingImageView)
 view.findViewById(R.id.dialog_imageview);
 fiv.setImageDrawable(drawable);

 final Dialog dialog = builder.create();
 dialog.getWindow().setBackgroundDrawable(new
 ColorDrawable(Color.TRANSPARENT));

 dialog.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
 WindowManager.LayoutParams.FLAG_FULLSCREEN
);
 dialog.getWindow().setLayout(WindowManager.LayoutParams.MATCH_PARENT,
 WindowManager.LayoutParams.MATCH_PARENT
);

 fiv.setOnClickListener(new View.OnClickListener() {
 @Override
 public void onClick(View view) {
 dialog.dismiss();

 }
 });

 dialog.show();
 }
}
```

The `ImageDialog` is used to show an image across the entire screen, while maintaining its aspect ratio.

It uses a `FillingImageView` which overrides the `onMeasure` method of `ImageView` to ensure that the image aspect ratio is maintained.

This completes objective 9.b.iii.5

### FillingImageView.java

```
package com(tpb.mdtext.dialogs;

import android.content.Context;
import android.graphics.drawable.Drawable;
import android.support.annotation.Nullable;
import android.support.v7.widget.AppCompatImageView;
import android.util.AttributeSet;

/**
 * Created by theo on 31/01/17.
 */

public class FillingImageView extends AppCompatImageView {

 public FillingImageView(Context context) {
 super(context);
 }

 public FillingImageView(Context context, @Nullable AttributeSet attrs) {
 super(context, attrs);
 }

 public FillingImageView(Context context, @Nullable AttributeSet attrs, int defStyleAttr) {
 super(context, attrs, defStyleAttr);
 }

 @Override
 protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
 try {
 final Drawable drawable = getDrawable();

 if(drawable == null) {
 setMeasuredDimension(0, 0);
 } else {
 final float imageSideRatio = (float)
 drawable.getIntrinsicWidth() / (float) drawable
 .getIntrinsicHeight(); //Image aspect ratio
 final float viewSideRatio = (float) MeasureSpec
 .getSize(widthMeasureSpec) / (float) MeasureSpec
 .getSize(heightMeasureSpec); //Aspect ratio of parent
 if(imageSideRatio >= viewSideRatio) {
 // Image is wider than the display (ratio)
 final int width = MeasureSpec.getSize(widthMeasureSpec);
 final int height = (int) (width / imageSideRatio);
 setMeasuredDimension(width, height);
 } else {
 // Image is taller than the display (ratio)
 }
 }
 } catch (Exception e) {
 e.printStackTrace();
 }
 }
}
```

```

 final int height = MeasureSpec.getSize(heightMeasureSpec);
 final int width = (int) (height * imageSideRatio);
 setMeasuredDimension(width, height);
 }
}
} catch(Exception e) {
 super.onMeasure(widthMeasureSpec, heightMeasureSpec);
}
}

}

```

The method calculates the aspect ratio of the image, and the aspect ratio of the parent view.

If the image has a wider ratio than the display ratio, then the width is calculated as the available width of the ImageView, and the height is calculated as the ratio of this width

matching the ratio calculated earlier.

If the image has a taller ratio than the display ratio, then the height is calculated as the available height of the ImageView, and the width is calculated as the ratio of this

height matching the ratio calculated earlier.

### TableDialog.java

```

package com(tpb.mdtext.dialogs;

import android.app.AlertDialog;
import android.app.Dialog;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;

import com(tpb.mdtext.handlers.TableClickHandler;
import com(tpb.mdtext.webview.MarkdownWebView;

import org.sufficientlysecure.htmltextview.R;

/**
 * Created by theo on 08/04/17.
 */

public class TableDialog implements TableClickHandler {

 private Context mContext;

 public TableDialog(Context context) {
 mContext = context;
 }

 @Override
 public void onClick(String html) {
 final AlertDialog.Builder builder = new AlertDialog.Builder(mContext);
 final LayoutInflater inflater = LayoutInflater.from(mContext);

```

```

 final View view = inflater.inflate(R.layout.dialog_table, null);

 builder.setView(view);

 final MarkdownWebView wv = (MarkdownWebView)
view.findViewById(R.id.dialog_web_view);
 wv.enableDarkTheme();
 wv.setMarkdown(html);
 final Dialog dialog = builder.create();

 dialog.getWindow()
 .setLayout(ViewGroup.LayoutParams.MATCH_PARENT,
ViewGroup.LayoutParams.MATCH_PARENT);
 dialog.getWindow().requestFeature(Window.FEATURE_NO_TITLE);
 dialog.show();
 }

}

```

The TableDialog uses the MarkdownWebView created earlier to display the HTML of the table in a WebView using the GitHub style CSS.  
This dialog completes objective 9.b.xi.

## Image loading and caching

The `HttpImageGetter` implements `Html.ImageGetter` which is used when retrieving images for img tags.

As well as loading images, the `HttpImageGetter` also caches them, which is especially useful when editing markdown segments containing images or in a comment feed where multiple users may use the same image, as it stops the image being reloaded each time the editor is toggled from raw to formatted markdown.

### `HttpImageGetter.java`

```

package com(tpb.mdtext.imagegetter;

import android.content.res.Resources;
import android.graphics.Canvas;
import android.graphics.drawable.BitmapDrawable;
import android.graphics.drawable.Drawable;
import android.os.AsyncTask;
import android.support.annotation.Nullable;
import android.support.v4.util.Pair;
import android.text.Html.ImageGetter;
import android.view.View;
import android.widget.TextView;

import java.io.IOException;
import java.io.InputStream;
import java.lang.ref.WeakReference;
import java.net.URI;
import java.net.URL;
import java.util.HashMap;

```

```
import java.util.Iterator;
import java.util.Map;

public class HttpImageGetter implements ImageGetter {

 private static final HashMap<String, Pair<Drawable, Long>> cache = new
HashMap<>();

 private final TextView mContainer;

 public HttpImageGetter(TextView container) {
 this.mContainer = container;
 }

 public Drawable getDrawable(String source) {
 final URLDrawable ud = new URLDrawable();
 new ImageGetterAsyncTask(ud, this, mContainer).execute(source);
 return ud;
 }

 private static class ImageGetterAsyncTask extends AsyncTask<String, Void,
Drawable> {
 private final WeakReference<URLDrawable> mDrawableReference;
 private final WeakReference<HttpImageGetter> mGetterReference;
 private final WeakReference<View> mContainerReference;
 private final WeakReference<Resources> mResources;
 private String mSource;
 private float mScale;

 ImageGetterAsyncTask(URLDrawable d, HttpImageGetter imageGetter, View
container) {
 this.mDrawableReference = new WeakReference<>(d);
 this.mGetterReference = new WeakReference<>(imageGetter);
 this.mContainerReference = new WeakReference<>(container);
 this.mResources = new WeakReference<>(container.getResources());
 }

 @Override
 protected Drawable doInBackground(String... params) {
 mSource = params[0];
 synchronized(cache) {
 Map.Entry<String, Pair<Drawable, Long>> entry;
 final Iterator<Map.Entry<String, Pair<Drawable, Long>>> it =
cache.entrySet().iterator();
 for(; it.hasNext();) {
 entry = it.next();
 if(System.currentTimeMillis() > entry.getValue().second +
60000) {
 it.remove();
 }
 }
 if(cache.containsKey(mSource)) {
 if(System.currentTimeMillis() > cache.get(mSource).second +
45000) {
 // The drawable is still being accessed, so we update
it
 fetchDrawable(mResources.get(), mSource);
 }
 }
 }
 }
 }
}
```

```

 }
 return
cache.get(mSource).first.getConstantState().newDrawable();
 } else if(mResources.get() != null) {
 return fetchDrawable(mResources.get(), mSource);
 }
}

return null;
}

@Override
protected void onPostExecute(Drawable result) {
 final URLDrawable urlDrawable = mDrawableReference.get();
 final HttpImageGetter imageGetter = mGetterReference.get();
 // We exist outside of the lifespan of the view
 if(result == null || urlDrawable == null || imageGetter == null)
return;

// Scale is set here as drawable may be cached and view may have
changed
setDrawableScale(result);

// Set the correct bound according to the result from HTTP call
urlDrawable.setBounds(0, 0, (int) (result.getIntrinsicWidth() *
mScale),
 (int) (result.getIntrinsicHeight() * mScale)
);

// Change the reference of the current urlDrawable to the result
from the HTTP call
urlDrawable.mDrawable = result;

// redraw the image by invalidating the container
imageGetter.mContainer.invalidate();
// re-set text to fix images overlapping text
imageGetter.mContainer.setText(imageGetter.mContainer.getText());
}

@Nullable
private Drawable fetchDrawable(Resources res, String urlString) {
 try {
 final InputStream is = fetch(urlString);
 final Drawable drawable = new BitmapDrawable(res, is);
 synchronized(cache) {
 cache.put(mSource, Pair.create(drawable,
System.currentTimeMillis()));
 }
 return drawable;
 } catch(Exception e) {
 return null;
 }
}

private void setDrawableScale(Drawable drawable) {
 mScale = getScale(drawable);
}

```

```

 drawable.setBounds(0, 0, (int) (drawable.getIntrinsicWidth() * mScale),
 (int) (drawable.getIntrinsicHeight() * mScale)
);
 }

 private float getScale(Drawable drawable) {
 final View container = mContainerReference.get();
 if(container == null) {
 return 1f;
 }
 float maxWidth = container.getWidth();
 float originalDrawableWidth = drawable.getIntrinsicWidth();
 return maxWidth / originalDrawableWidth;
 }

 @Nullable
 private InputStream fetch(String urlString) throws IOException {
 URL url;
 final HttpImageGetter imageGetter = mGetterReference.get();
 if(imageGetter == null) {
 return null;
 }
 url = URI.create(urlString).toURL();

 return (InputStream) url.getContent();
 }
}

@SuppressWarnings("deprecation")
public class URLDrawable extends BitmapDrawable {
 Drawable mDrawable;

 @Override
 public void draw(Canvas canvas) {
 if(mDrawable != null) {
 mDrawable.draw(canvas);
 }
 }

 public Drawable getDrawable() {
 return mDrawable;
 }

}
}

```

The `HttpImageGetter` is constructed with a reference to the `TextView` for which it is loading images.

The `URLDrawable` is used in order to only draw the `Drawable` to the `canvas` once it has been loaded.

When `getDrawable` is called, `HttpImageGetter` creates a new `URLDrawable`, then creates a new `ImageGetterAsyncTask` and then returns the `URLDrawable`.

The `ImageGetterAsyncTask` then begins loading the image in the background.

The `ImageGetterAsyncTask` takes a `URLDrawable`, the `HTTPImageGetter` and the

containing `View`, and stores `WeakReferences` to each of them as well as a `WeakReference` to the `Resources` from the container `View`.

When an `AsyncTask` is executed, first two methods are executed on the main thread, and the third is executed in the background.

The first on the main thread is `onPreExecute`, which is not needed here as there is no setup process.

Next, `doInBackground` is called, which performs work on another thread.

Finally, `onPostExecute` is called on the main thread, with the result from `doInBackground`.

In `doInBackground` the source is first extracted from the first parameter.

Next the cache is checked for the source. This must be synchronised as the cache may be accessed from multiple threads at once.

The cache is a `Map` and must be accessed with an `Iterator` to allow removal at the same time.

The `Map` contains pairs of strings, the source URLs, and pairs of the drawable and the last time that it was updated.

If the drawable was last updated over a minute ago, it is removed from the cache. Once the cache has been cleaned, the cache is checked for the current key, which is reloaded if it was last accessed more than 45 seconds ago.

The `Drawable` is returned with `getConstantState().newDrawable()`. This ensures that each `Drawable` displayed has the correct aspect ratio for the `TextView` in which it is displayed, otherwise only the last mutation would take effect.

Otherwise, the `Resources` reference is checked, and if it is non-null, the `Drawable` is loaded.

`fetchDrawable` creates an `InputStream` by calling `fetch`.

`fetch` checks if the `HttpImageGetter` still exists, and if so creates a `URL` from which to load the content.

`fetchDrawable` then creates a `BitmapDrawable` from this `InputStream` and the `Resources` object which is used to determine specifics about how the `Drawable` should be displayed.

The `Drawable` is then added to the cache and returned.

In `onPostExecute` the validity of the `HttpImageGetter`, `result` and `URLDrawable` are checked, and if they are valid the `Drawable` is scaled.

`setDrawableScale` calls `getScale` which returns the scale of the view maximum width to the `Drawable` width.

`setDrawableScale` then sets the bounds of the `Drawable` to the scaled values of its original width and height, maintaining its aspect ratio while filling the full width of the containing `View`.

Once the scale of the `result` `Drawable` has been set, the scale of the `URLDrawable` is also set, and the `URLDrawable` drawable is changed for the `result` `Drawable`.

If the `DrawableCatcher` is non-null, the `drawable` and `URL` are passed to it.

Finally, in order to draw the `URLDrawable` the container is invalidated and its text is set to its current text to force a layout refresh.

This completes objective 9.b.iii.1, 9.b.iii.2, and 9.b.iii.4.

### Loading images from Assets and Resources

While they are not currently used in this project, some may want to load images from the device itself. Either those included in the assets directory, or in resources.

This can be handled with the `AssetsImageGetter` and `ResImageGetter`.

#### **AssetsImageGetter.java**

```
package com(tpb.mdtextr.imagegetter;

import android.content.Context;
import android.graphics.drawable.Drawable;
import android.text.Html;
import android.widget.TextView;

import java.io.IOException;
import java.io.InputStream;

/**
 * Assets Image Getter
 * <p>
 * Load image from assets folder
 *
 * @author Daniel Passos
 */
class AssetsImageGetter implements Html.ImageGetter {

 private final Context context;

 public AssetsImageGetter(Context context) {
 this.context = context;
 }

 public AssetsImageGetter(TextView textView) {
 this.context = textView.getContext();
 }

 @Override
 public Drawable getDrawable(String source) {

 try {
 final InputStream inputStream = context.getAssets().open(source);
 final Drawable d = Drawable.createFromStream(inputStream, null);
 d.setBounds(0, 0, d.getIntrinsicWidth(), d.getIntrinsicHeight());
 return d;
 } catch(IOException e) {
 return null;
 }
 }
}
```

```
}
```

The `AssetsImageGetter` creates an `InputStream` from an assets path and loads the drawable from it.

### **ResImageGetter.java**

```
package com(tpb.mdtext.imagegetter;

import android.content.Context;
import android.graphics.drawable.Drawable;
import android.text.Html;
import android.widget.TextView;

/**
 * Copied from http://stackoverflow.com/a/22298833
 */
class ResImageGetter implements Html.ImageGetter {
 private final TextView container;

 public ResImageGetter(TextView textView) {
 this.container = textView;
 }

 public Drawable getDrawable(String source) {
 final Context context = container.getContext();
 int id = context.getResources().getIdentifier(source, "drawable",
context.getPackageName());

 if(id == 0) {
 //Drawable not in this package, might be somewhere else
 id = context.getResources().getIdentifier(source, "drawable",
"android");
 }

 if(id == 0) {
 return null;
 } else {
 final Drawable d = context.getResources().getDrawable(id);
 d.setBounds(0, 0, d.getIntrinsicWidth(), d.getIntrinsicHeight());
 return d;
 }
 }
}
```

The `ResImageGetter` attempts to load a drawable from a resource name, checking the current package as well as the built in drawables.

### **MarkdownTextView**

`MarkdownTextView` is used to implement each of the individual objectives described above. It handles background parsing of the markdown as well as dealing with click handling and the caching of content.

The MarkdownTextView contains  
a LinkClickHandler, ImageClickHandler, TableClickHandler, and CodeClickHandler.  
**MarkdownTextView.java**

```
package com(tpb.mdtext.views;

import android.content.Context;
import android.graphics.Color;
import android.os.Build;
import android.os.Handler;
import android.os.Looper;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.annotation.RawRes;
import android.support.v7.widget.AppCompatTextView;
import android.text.Html;
import android.text.SpannableString;
import android.text.Spanned;
import android.util.AttributeSet;
import android.util.Log;
import android.view MotionEvent;
import android.view.View;

import com(tpb.mdtext.ClickableMovementMethod;
import com(tpb.mdtext.HtmlTagHandler;
import com(tpb.mdtext.Markdown;
import com(tpb.mdtext.SpanCache;
import com(tpb.mdtext.TextUtils;
import com(tpb.mdtext.dialogs.CodeDialog;
import com(tpb.mdtext.dialogs.ImageDialog;
import com(tpb.mdtext.dialogs.TableDialog;
import com(tpb.mdtext.handlers.CodeClickHandler;
import com(tpb.mdtext.handlers.ImageClickHandler;
import com(tpb.mdtext.handlers.LinkClickHandler;
import com(tpb.mdtext.handlers.TableClickHandler;
import com(tpb.mdtext.views.spans.CodeSpan;
import com(tpb.mdtext.views.spans.TableSpan;

import java.io.InputStream;
import java.lang.ref.WeakReference;
import java.util.Scanner;

public class MarkdownTextView extends AppCompatTextView implements
View.OnClickListener {

 public static final String TAG = MarkdownTextView.class.getSimpleName();

 @Nullable private LinkClickHandler mLinkHandler;
 @Nullable private ImageClickHandler mImageClickHandler;
 @Nullable private TableClickHandler mTableHandler;
 @Nullable private CodeClickHandler mCodeHandler;
 @Nullable private Handler mParseHandler;

 private boolean mSpanHit = false;
 private OnClickListener mOnClickListener;
 private float[] mLastClickPosition = new float[] {-1, -1};
```

```
private WeakReference<SpanCache> mSpanCache;

public MarkdownTextView(Context context, AttributeSet attrs, int defStyle)
{
 super(context, attrs, defStyle);
 init(context);
}

public MarkdownTextView(Context context, AttributeSet attrs) {
 super(context, attrs);
 init(context);
}

public MarkdownTextView(Context context) {
 super(context);
 init(context);
}

private void init(Context context) {
 if(!CodeSpan.isInitialised()) CodeSpan.initialise(context);
 if(!TableSpan.isInitialised()) TableSpan.initialise(context);
 setDefaultHandlers(context);
 setTextIsSelectable(true);
 setCursorVisible(false);
 setClickable(true);
 setTextColor(Color.WHITE);
}

public void setParseHandler(@Nullable Handler parseHandler) {
 mParseHandler = parseHandler;
}

public void setLinkClickHandler(LinkClickHandler handler) {
 mLinkHandler = handler;
}

public void setImageHandler(ImageClickHandler imageHandler) {
 mImageClickHandler = imageHandler;
}

public void setCodeClickHandler(CodeClickHandler handler) {
 mCodeHandler = handler;
}

public void setTableClickHandler(TableClickHandler handler) {
 mTableHandler = handler;
}

public void setDefaultHandlers(Context context) {
 setCodeClickHandler(new CodeDialog(context));
 setImageHandler(new ImageDialog(context));
 setTableClickHandler(new TableDialog(context));
}

public void setMarkdown(@NonNull String markdown) {
 setMarkdown(markdown, null, null);
}

public void setMarkdown(@RawRes int resId) {
```

```

 setMarkdown(resId, null);
 }

 private void setMarkdown(@RawRes int resourceId, @Nullable Html.ImageGetter
imageGetter) {
 final InputStream inputStreamText =
getContext().getResources().openRawResource(resourceId);
 setMarkdown(convertStreamToString(inputStreamText), imageGetter,
null);
}

public void setMarkdown(@NonNull final String markdown, @Nullable final
Html.ImageGetter imageGetter, @Nullable SpanCache cache) {
 mSpanCache = new WeakReference<>(cache);
 //If we have a handler use it
 if(mParseHandler != null) {
 mParseHandler.post(new Runnable() {
 @Override
 public void run() {
 parseAndSetMd(markdown, imageGetter);
 }
 });
 } else {
 parseAndSetMd(markdown, imageGetter);
 }
}

private void parseAndSetMd(@NonNull String markdown, @Nullable final
Html.ImageGetter imageGetter) {
 // Override tags to stop Html.fromHtml destroying some of them
 markdown = HtmlTagHandler.overrideTags(Markdown.parseMD(markdown));
 final HtmlTagHandler htmlTagHandler = new HtmlTagHandler(this,
 imageGetter, mLinkHandler, mImageClickHandler, mCodeHandler,
mTableHandler
);
 try {
 final Spanned text;
 if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
 text = removeHtmlBottomPadding(
 Html.fromHtml(markdown, Html.FROM_HTML_MODE_LEGACY,
imageGetter,
 htmlTagHandler
));
 } else {
 text = removeHtmlBottomPadding(
 Html.fromHtml(markdown, imageGetter, htmlTagHandler));
 }
 // Convert to a buffer to allow editing
 final SpannableString buffer = new SpannableString(text);

 //Add links for emails and web-urls
 TextUtils.addLinks(buffer);
 if(Looper.myLooper() == Looper.getMainLooper()) {
 setMarkdownText(buffer);
 } else {
 //Post back on UI thread
 MarkdownTextView.this.postDelayed(new Runnable() {
 @Override

```

```
 public void run() {
 setMarkdownText(buffer);
 }
 }, 17);
}
} catch(Exception e) {
 Log.e("TextView", "WTF", e);
 markdown = "Error parsing markdown\n\n\n" +
 Html.fromHtml(Html.escapeHtml(markdown));
 setText(markdown);
}
}

private void setMarkdownText(SpannableString buffer) {
 setText(buffer);
 if(!(getMovementMethod() instanceof ClickableMovementMethod)) {
 setMovementMethod(ClickableMovementMethod.getInstance());
 }
 if(mSpanCache != null && mSpanCache.get() != null) {
 mSpanCache.get().cache(buffer);
 mSpanCache = null;
 }
}

@NonNull
private static String convertStreamToString(@NonNull InputStream is) {
 final Scanner s = new Scanner(is).useDelimiter("\n");
 return s.hasNext() ? s.next() : "";
}

private static Spanned removeHtmlBottomPadding(Spanned text) {
 while(text.length() > 0 && text.charAt(text.length() - 1) == '\n') {
 text = (Spanned) text.subSequence(0, text.length() - 1);
 }
 return text;
}

@Override
public boolean onTouchEvent(MotionEvent event) {
 if(event.getAction() == MotionEvent.ACTION_DOWN) {
 mLastClickPosition[0] = event.getRawX();
 mLastClickPosition[1] = event.getRawY();
 if(hasSelection()) clearFocus();
 } else if(event.getAction() == MotionEvent.ACTION_UP) {
 mSpanHit = false;
 }
 return super.onTouchEvent(event);
}

public float[] getLastClickPosition() {
 if(mLastClickPosition[0] == -1) {
 // If we haven't been clicked yet, get the centre of the view
 final int[] pos = new int[2];
 getLocationOnScreen(pos);
 mLastClickPosition[0] = pos[0] + getWidth() / 2;
 mLastClickPosition[1] = pos[1] + getHeight() / 2;
 }
 return mLastClickPosition;
}
```

```

public void setSpanHit() {
 mSpanHit = true;
}

@Override
public void onClick(View v) {
 if(!mSpanHit && mOnClickListener != null) mOnClickListener.onClick(v);
}

@Override
public void setOnClickListener(@Nullable OnClickListener l) {
 mOnClickListener = l;
 super.setOnClickListener(this);
}

@Override
protected boolean getDefaultEditable() {
 return Build.VERSION.SDK_INT >= Build.VERSION_CODES.N ||
super.getDefaultEditable();
}

@Override
public boolean isSuggestionsEnabled() {
 return false;
}
}

```

Each of the constructors calls `init` which checks for the initialisation of `CodeSpan` and `TableSpan`, creates the default handlers, enables text selection and clicking, and sets the text colour.

After this are setters for each of the handlers.

The raw markdown text can be set from a string or a resource id.

Each of these methods can also take an `ImageGeter`.

The `setMarkdown` method which takes an a `SpanCacher` creates a `WeakReference` to the `SpanCacher` and then checks if there is a `Handler` to perform parsing on another thread.

If the handler exists, the call to `parseAndSetMd` is run on the `Handler`, otherwise it is called on the UI thread.

In `parseAndSetMd` the `HtmlTagHandler` is created to parse HTML to spans.

The span text is then overriden to stop the built in `Html.fromHtml` capturing but ignoring numerous tags.

If the Android version is API 25 or greater, the `LEGACY` flag is used to ensure the styling is the same as on previous versions.

In either case, `removeHtmlBottomPadding` is called, as `Html.fromHtml` adds two newlines to the end of the parsed HTML.

In order to add the URL and email address links, the `Spanned` object must be converted to an `Editable`.

The links are then added, and the text can be set.

If the current Looper is the main Looper, then the code is running on the UI thread and the text can be set immediately.

Otherwise, it must be posted on the TextView itself so that it runs on the UI thread.

setMarkdownText sets the text, and ensures that the movement method is a ClickableMovementMethod.

Finally, if a SpanCacher exists, it is passed the SpannableString.

The remaining methods are used to handle clicking on the TextView.

onTouchEvent captures all touch events.

If the event action is down the position is stored for use in animations, and any selection is cleared.

If the event is up, the span hit flag is reset.

setSpanHit is used to set the span hit flag, as described in ClickableMovementMethod.

onClick is used to forward the click events to the actual OnClickListener if it exists and a span has not been hit in the same click event.

setOnClickListener is overriden to store the OnClickListener and ensure that click events are passed to the MarkdownTextView itself.

getDefaultValue checks the Android version, returning true if the SDK version is 25 or above, otherwise returning the default value.

This fixes a problem introduced in SDK 25 where span priority is not respected, which can cause problems when displaying indented content, such as lists.

isSuggestionEnabled is overriden to ensure that suggestions are never shown on the MarkdownTextView, even though it may appear to be editable.

## MarkdownEditText

The MarkdownEditText is very similar to MarkdownTextView except that it extends EditText, a subclass of TextView, does not include support for processing on another thread, and adds support for toggling between and editable and non-editable state.

### MarkdownEditText.java

```
package com(tpb.mdtext.views;

import android.content.Context;
import android.os.Build;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.annotation.RawRes;
import android.support.v7.widget.AppCompatEditText;
import android.text.Editable;
import android.text.Html;
import android.text.SpannableString;
import android.text.SpannableStringBuilder;
import android.text.Spanned;
```

```
import android.util.AttributeSet;
import android.util.Log;

import com(tpb.mdtext.ClickableMovementMethod;
import com(tpb.mdtext.HtmlTagHandler;
import com(tpb.mdtext.Markdown;
import com(tpb.mdtext.TextUtils;
import com(tpb.mdtext.dialogs.CodeDialog;
import com(tpb.mdtext.dialogs.ImageDialog;
import com(tpb.mdtext.dialogs.TableDialog;
import com(tpb.mdtext.handlers.CodeClickHandler;
import com(tpb.mdtext.handlers.ImageClickHandler;
import com(tpb.mdtext.handlers.LinkClickHandler;
import com(tpb.mdtext.handlers.TableClickHandler;
import com(tpb.mdtext.views.spans.Span;
import com(tpb.mdtext.views.spans.TableSpan;

import java.io.InputStream;
import java.util.Scanner;

/**
 * Created by theo on 27/02/17.
 */

public class MarkdownEditText extends AppCompatEditText {

 public static final String TAG = MarkdownEditText.class.getSimpleName();

 private boolean mIsEditing = true;
 private Editable mSavedText = new SpannableStringBuilder();
 @Nullable private LinkClickHandler mLinkHandler;
 @Nullable private ImageClickHandler mImageClickHandler;
 @Nullable private TableClickHandler mTableHandler;
 @Nullable private CodeClickHandler mCodeHandler;

 public MarkdownEditText(Context context, AttributeSet attrs, int defStyle)
 {
 super(context, attrs, defStyle);
 init(context);
 }

 public MarkdownEditText(Context context, AttributeSet attrs) {
 super(context, attrs);
 init(context);
 }

 public MarkdownEditText(Context context) {
 super(context);
 init(context);
 }

 private void init(Context context) {
 if(!CodeSpan.isInitialised()) CodeSpan.initialise(context);
 if(!TableSpan.isInitialised()) TableSpan.initialise(context);
 setDefaultHandlers(context);
 setPadding(0, getPaddingTop(), 0, getPaddingBottom());
 }
}
```

```
public void setLinkClickHandler(LinkClickHandler handler) {
 mLinkHandler = handler;
}

public void setImageHandler(ImageClickHandler imageHandler) {
 mImageClickHandler = imageHandler;
}

public void setTableClickHandler(TableClickHandler handler) {
 mTableHandler = handler;
}

public void setCodeClickHandler(CodeClickHandler handler) {
 mCodeHandler = handler;
}

public void setDefaultHandlers(Context context) {
 setCodeClickHandler(new CodeDialog(context));
 setImageHandler(new ImageDialog(context));
 setTableClickHandler(new TableDialog(context));
}

public void setMarkdown(@NonNull String html) {
 setMarkdown(html, null);
}

public void setMarkdown(@RawRes int resId) {
 setMarkdown(resId, null);
}

private void setMarkdown(@RawRes int resId, @Nullable Html.ImageGetter
imageGetter) {
 InputStream inputStreamText =
getContext().getResources().openRawResource(resId);
 setMarkdown(convertStreamToString(inputStreamText), imageGetter);
}

public void setMarkdown(@NonNull String markdown, @Nullable final
Html.ImageGetter imageGetter) {
 // Override tags to stop Html.fromHtml destroying some of them
 final String overridden =
HtmlTagHandler.overrideTags(Markdown.parseMD(markdown));
 final HtmlTagHandler htmlTagHandler = new HtmlTagHandler(this,
 imageGetter, mLinkHandler, mImageClickHandler, mCodeHandler,
mTableHandler
);
 try {
 final Spanned text;
 if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
 text = removeHtmlBottomPadding(
 Html.fromHtml(overridden, Html.FROM_HTML_MODE_LEGACY,
imageGetter,
 htmlTagHandler
));
 } else {
 text = removeHtmlBottomPadding(
 Html.fromHtml(overridden, imageGetter,
htmlTagHandler));
 }
 } catch (Exception e) {
 Log.e("Markdown", "Error setting markdown", e);
 }
}
```

```
 }

 // Convert to a buffer to allow editing
 final SpannableString buffer = new SpannableString(text);

 //Add links for emails and web-urls
 TextUtils.addLinks(buffer);
 setText(buffer);
 if(!(getMovementMethod() instanceof ClickableMovementMethod)) {
 setMovementMethod(ClickableMovementMethod.getInstance());
 }
 } catch(Exception e) {
 Log.e("TextView", "WTF", e);
 markdown = "Error parsing markdown\n\n\n" +
 Html.fromHtml(Html.escapeHtml(markdown));
 setText(markdown);
 }
}

@NonNull
private static String convertStreamToString(@NonNull InputStream is) {
 final Scanner s = new Scanner(is).useDelimiter("\n");
 return s.hasNext() ? s.next() : "";
}

@Nullable
private static Spanned removeHtmlBottomPadding(@Nullable Spanned text) {
 if(text == null) {
 return null;
 }

 while(text.length() > 0 && text.charAt(text.length() - 1) == '\n') {
 text = (Spanned) text.subSequence(0, text.length() - 1);
 }
 return text;
}

public boolean isEditing() {
 return mIsEditing;
}

public void enableEditing() {
 if(mIsEditing) return;
 setFocusable(true);
 setFocusableInTouchMode(true);
 setCursorVisible(true);
 setEnabled(true);
 mIsEditing = true;
}

public void disableEditing() {
 if(!mIsEditing) return;
 setFocusable(false);
 setCursorVisible(false);
 mIsEditing = false;
}

public void saveText() {
 mSavedText = getText();
```

```

 }

 public void restoreText() {
 setText(mSavedText);
 }

 public Editable getInputText() {
 return mIsEditing ? getText() : mSavedText;
 }

 @Override
 public boolean isSuggestionsEnabled() {
 return mIsEditing;
 }
}

```

As before, the `CodeSpan` and `TableSpan` are initialised, and the default handlers are set.

The `MarkdownEditText` then removes padding on either side of it, as the spans should be shown across the entire canvas and have no access to the padding information.

As before, markdown can be set from a string or a resource id, with or without an `ImageGetter`.

As text is expected to change, the `MarkdownEditText` does not support caching the parsed spans.

In order to support two text states, the `MarkdownTextView` contains an editing flag, and an `Editable` containing any saved text.

`saveText` saves the current text to this `Editable`, and `restoreText` restores it.

The `getInputText` method is used to always return the text being edited, rather than the parsed form.

`enableEditing` and `disableEditing` do as their names suggest, enabling or disable the cursor and focusing, and setting the `mIsEditing` flag.

## Tag handling

Android's `Html.fromHtml` method parses simple Html to a `Spanned` object.

The method supports the following tags:

- bold
- big
- blockquote
- break
- cite
- define
- emphasis
- font color and face

- headers
- italics
- paragraphs
- small
- strong
- subscript
- superscript
- underline
- href

Most of these spans only modify the `TextPaint` to draw text with a certain size, colour or styling.

In order to implement the spans required for GitHub flavoured markdown, some of these tags must be overridden, and others implemented.

Overridden tags-

- unordered list
- ordered list
- list item
- blockquote
- href
- font
- image

Implemented tags

- table
- table row
- table header
- table data
- code
- strikethrough
- inlinecode

### **Overriding tags**

Only tags which are not recognised are delegated to the `TagHandler`.

In order to capture these tags they must be overridden prior to parsing.

This is done with the `TextUtils.replace` method implemented earlier.

```
private static final String UNORDERED_LIST_TAG = "ESCAPED_UL_TAG";
```

```

private static final String ORDERED_LIST_TAG = "ESCAPED_OL_TAG";
private static final String LIST_ITEM_TAG = "ESCAPED_LI_TAG";
private static final String BLOCKQUOTE_TAG = "ESCAPED_BLOCKQUOTE_TAG";
private static final String A_TAG = "ESCAPED_A_TAG";
private static final String FONT_TAG = "ESCAPED_FONT_TAG";
private static final String IMAGE_TAG = "ESCAPED_IMG_TAG";

private static final Map<String, String> ESCAPE_MAP = new HashMap<>();

static {
 ESCAPE_MAP.put("<ul", "<" + UNORDERED_LIST_TAG);
 ESCAPE_MAP.put("", "</>" + UNORDERED_LIST_TAG + ">");
 ESCAPE_MAP.put("<ol", "<" + ORDERED_LIST_TAG);
 ESCAPE_MAP.put("", "</>" + ORDERED_LIST_TAG + ">");
 ESCAPE_MAP.put("<li", "<" + LIST_ITEM_TAG);
 ESCAPE_MAP.put("", "</>" + LIST_ITEM_TAG + ">");
 ESCAPE_MAP.put("<blockquote>", "<" + BLOCKQUOTE_TAG + ">");
 ESCAPE_MAP.put("</blockquote>", "</>" + BLOCKQUOTE_TAG + ">");
 ESCAPE_MAP.put("<a", "<" + A_TAG);
 ESCAPE_MAP.put("", "</>" + A_TAG + ">");
 ESCAPE_MAP.put("<font", "<" + FONT_TAG);
 ESCAPE_MAP.put("", "</>" + FONT_TAG + ">");
 ESCAPE_MAP.put("<img", "<" + IMAGE_TAG);
 ESCAPE_MAP.put("", "</>" + IMAGE_TAG + ">");
}
}

private static final Pattern ESCAPE_PATTERN =
TextUtils.generatePattern(ESCAPE_MAP.keySet());

public static String overrideTags(@Nullable String html) {
 return TextUtils.replace(html, ESCAPE_MAP, ESCAPE_PATTERN);
}

```

## Handlers

The `HtmlTagHandler` is passed the handlers which are required for each of the span types.

It is also passed the `TextView` itself, which is required for measuring indentations.

### `HtmlTagHandler.java`

```

public HtmlTagHandler(TextView tv, Html.ImageGetter imageGetter,
 @Nullable LinkClickHandler linkHandler,
 @Nullable ImageClickHandler imageClickHandler,
 @Nullable CodeClickHandler codeHandler,
 @Nullable TableClickHandler tableHandler) {
 mTextPaint = tv.getTextPaint();
 mSingleIndent = (int) mTextPaint.measureText("t");
 mImageGetter = imageGetter;
 mImageClickHandler = imageClickHandler;
 mLinkHandler = linkHandler;
 mCodeHandler = codeHandler;
 mTableHandler = tableHandler;
}

```

## Tag opening and closing

Each tag is delegated to the `HtmlTagHandler` through the `handleTag` method, which has the parameters `boolean opening`, `String tag`, `Editable output`, `XMLReader xmlReader`.

If `opening` is true, the tag, output and `xmlReader` are passed to `handleOpeningTag`. Otherwise the tag and output are passed to `handleClosingTag`.

### **HtmlTagHandler.java**

```
public void handleTag(final boolean opening, final String tag, Editable output, final XMLReader xmlReader) {
 if(opening) {
 handleOpeningTag(tag, output, xmlReader);
 } else {
 handleClosingTag(tag, output);
 }
 storeTableTags(opening, tag);
}
```

`handleOpeningTag` switches the tag to begin the correct span type in the `Editable`

### **HtmlTagHandler.java**

```
private void handleOpeningTag(final String tag, Editable output, final XMLReader xmlReader) {
 switch(tag.toUpperCase()) {
 case UNORDERED_LIST_TAG:
 mLists.push(
 Triple.create(
 tag,
 safelyParseBoolean(
 getAttribute("bulleted", xmlReader,
 "true"), true
),
 ListNumberSpan.ListType.NUMBER
)
);
 break;
 case ORDERED_LIST_TAG:
 final ListNumberSpan.ListType type = ListNumberSpan.ListType
 .fromString(getAttribute("type", xmlReader, ""));
 mLists.push(
 Triple.create(
 tag,
 safelyParseBoolean(
 getAttribute("numbered", xmlReader,
 "true"), true
),
 type
)
);
 m0lIndices.push(Pair.create(1, type));
 break;
 case LIST_ITEM_TAG:
 if(output.length() > 0 && output.charAt(output.length() - 1)
!= '\n') {
 output.append("\n");
 }
 if(!mLists.isEmpty()) {
 final String parentList = mLists.peek().first;
```

```
 if(parentList.equalsIgnoreCase(ORDERED_LIST_TAG)) {
 start(output, new O1());
 mO1Indices.push(Pair.create(mO1Indices.pop().first +
1, mLists.peek().third));
 } else if(parentList.equalsIgnoreCase(UNORDERED_LIST_TAG))
{
 start(output, new U1());
 }
 } else {
 start(output, new O1());
 mO1Indices.push(Pair.create(1,
ListNumberSpan.ListType.NUMBER));
 }
 break;
case "TABLE":
 start(output, new Table());
 if(mTableLevel == 0) {
 mTableHtmlBuilder = new StringBuilder();
 }
 mTableLevel++;
 break;
case FONT_TAG:
 final String font = getAttribute("face", xmlReader, "");
 final String fgColor = getAttribute("color", xmlReader, "");
 final String bgColor = getAttribute("background-color",
xmlReader, "");
 final boolean rounded =
safelyParseBoolean(getAttribute("rounded", xmlReader, ""),
false
);
 if(font != null && !font.isEmpty()) {
 start(output, new Font(font));
 }
 if(fgColor != null && !fgColor.isEmpty()) {
 start(output, new ForegroundColor(fgColor));
 }
 if(bgColor != null && !bgColor.isEmpty()) {
 start(output, new BackgroundColor(bgColor, rounded));
 }
 break;
case "CODE":
 start(output, new Code());
 break;
case "CENTER":
 start(output, new Center());
 break;
case "S":
case "STRIKE":
 start(output, new StrikeThrough());
 break;
case "TR":
 start(output, new Tr());
 break;
case "TH":
 start(output, new Th());
 break;
case "TD":
 start(output, new Td());
 break;
```

```

 case "HR":
 start(output, new HorizontalRule());
 break;
 case BLOCKQUOTE_TAG:
 start(output, new BlockQuote());
 break;
 case A_TAG:
 start(output, new A(getAttribute("href", xmlReader,
"invalid_url")));
 break;
 case IMAGE_TAG:
 handleImageTag(output, getAttribute("src", xmlReader, ""));
 break;
 case "INLINECODE":
 start(output, new InlineCode());
 break;

 }
}

```

After the tag has been handled, any table tags are stored with `storeTableTags`. This checks if the current table depth is greater than 0, or the tag is "table". If so, the opening bracket is added to the `mTableHtmlBuilder` `StringBuilder`, along with the closing forward slash if the tag is being closed. The tag is then appended and closed.

### **HtmlTagHandler.java**

```

private void storeTableTags(boolean opening, String tag) {
 if(mTableLevel > 0 || tag.equalsIgnoreCase("table")) {
 mTableHtmlBuilder.append("<");
 if(!opening) {
 mTableHtmlBuilder.append("/");
 }
 mTableHtmlBuilder
 .append(tag.toLowerCase())
 .append(">");
 }
}

```

This builds the HTML for a table so that it can be displayed later.

### **Span opening and closing**

The `TagHandler` works by placing MARK spans in the `Editable`. A MARK span is a span which must be removed later and acts as a placeholder for a new span.

### **HtmlTagHandler.java**

```

private void start(Editable output, Object mark) {
 final int point = output.length();
 output.setSpan(mark, point, point, Spannable.SPAN_MARK_MARK);
}

```

The MARK span has 0 length and is placed at the end of the `Editable`.

### **HtmlTagHandler.java**

```

private void end(Editable output, Class kind, boolean paragraphStyle,
Object... replaces) {
 final Object obj = getLast(output, kind);
 final int start = output.getSpanStart(obj);
 final int end = output.length();

 // If we're in a table, then we need to store the raw HTML for later
 if(mTableLevel > 0) {
 mTableHtmlBuilder.append(extractSpanText(output, kind));
 }

 output.removeSpan(obj);
 if(start != end) {
 int len = end;
 // paragraph styles like AlignmentSpan need to end with a new
line!
 if(paragraphStyle) {
 output.append("\n");
 len++;
 }
 for(Object replace : replaces) {
 if(output.length() > 0 && (mTableLevel == 0 || end <
output.length())) {
 output.setSpan(replace, start, len,
Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);
 }
 }
 }
}

```

When a tag is ended it must be replaced by one or more spans, or removed.

The span is extracted with `getLast`

### **HtmlTagHandler.java**

```

private static <T> T getLast(Editable text, Class<T> kind) {
 final T[] objs = text.getSpans(0, text.length(), kind);
 if(objs.length == 0) {
 return null;
 } else {
 //In reverse as items are returned in order they were inserted
 for(int i = objs.length; i > 0; i--) {
 if(text.getSpanFlags(objs[i - 1]) == Spannable.SPAN_MARK_MARK)
{
 return objs[i - 1];
 }
 }
 return null;
 }
}

```

This finds all of the objects of the given class and iterates backwards through them searching for a span with the MARK flags.

If the tag is within a table, the text contained within it is extracted and added to the table `StringBuilder`.

The span start and end positions are then stored and it is removed from the `Editable`.

If the span is of non-zero length, it is replaced.

If the span is an instance of `ParagraphStyle`, a newline must be added after it.

Once this check has ben completed, each of the spans in the replaces array are place over the position which was previously occupied by the MARK span.

## Span classes

Each span has its own marker class, which is used to extract it from the `Editable`

```
private static class Ul {
}

private static class Ol {
}

private static class Code {
}

private static class Center {
}

private static class StrikeThrough {
}

private static class Table {
}

private static class Tr {
}

private static class Th {
}

private static class Td {
}

private static class HorizontalRule {
}

private static class BlockQuote {
}

private static class InLineCode {
}

private static class A {
 String href;

 A(String href) {
 this.href = href;
 }
}

private static class ForegroundColor {
```

```

 String color;

 ForegroundColor(String color) {
 this.color = color;
 }

 }

private static class BackgroundColor {
 String color;
 boolean rounded;

 BackgroundColor(String color, boolean rounded) {
 this.color = color;
 this.rounded = rounded;
 }
}

private static class Font {
 String face;

 Font(String face) {
 this.face = face;
 }
}

```

## Attribute extraction

Unfortunately, the TagHandler has no direct access to the attributes of the tags that are passed to it.

However, it is able to access them when the tag is being opened.

This is done with `getAttribute`

### HtmlTagHandler.java

```

private static String getAttribute(@NotNull String attr, @NotNull XMLReader
reader, String defaultAttr) {
 try {
 final Field fElement =
reader.getClass().getDeclaredField("theNewElement");
 fElement.setAccessible(true);
 final Object element = fElement.get(reader);
 final Field fAtts =
element.getClass().getDeclaredField("theAtts");
 fAtts.setAccessible(true);
 final Object attrs = fAtts.get(element);
 final Field fData = attrs.getClass().getDeclaredField("data");
 fData.setAccessible(true);
 final String[] data = (String[]) fData.get(attrs);
 final Field fLength = attrs.getClass().getDeclaredField("length");
 fLength.setAccessible(true);
 final int len = (Integer) fLength.get(attrs);
 for(int i = 0; i < len; i++) {
 if(attr.equals(data[i * 5 + 1])) {
 return data[i * 5 + 4];
 }
 }
 }
}

```

```

 } catch(Exception e) {
 Log.e(TAG, "handleTag: ", e);
 }
 return defaultAttr;
}

```

This method uses reflection to attempt to extract the attribute from the XMLReader.

First the element field is collected, made accessible and accessed from the XMLReader.

Next the attributes field is collected, made accessible and accessed from the element.

Next the data field is collected, made accessible and accessed from the attributes.

The data field is a string array containing the attribute names, types, and values. A href to a user might appear as [, href, href, CDATA, <https://github.com/user>, null, null]

while a font tag with a colour and background colour may appear as [, background-color, background-color, CDATA, navy, , color, color, CDATA, red, null, null]

Once the attribute is first found in the array, the next item will be the attribute again, the item after will be the data type, and the item after that will be the actual data.

The length value extracted from the attrs object is the actual number of attributes present.

The loop iterates through every 5th item, beginning at index 1, as index 0 is always empty.

If the index contains the tag, the value 4 positions after is returned.

## List tags

List tags and numberings are stored in two stacks.

`mLists` is a stack of triples containing the tag, whether the list is bulleted, and the list type.

`mOlIndices` is a stack of pairs of integers and `ListTypes` which is used when exiting a nested span to continue the previous lists' numbering.

Note:

The `Triple` is a generic class like `Pair` which holds three generic types.

```
private static class Triple<T, U, V> {
```

```

 T first;
 U second;
```

```

 V third;

 Triple(T t, U u, V v) {
 first = t;
 second = u;
 third = v;
 }

 static <T, U, V> Triple<T, U, V> create(T t, U u, V v) {
 return new Triple<>(t, u, v);
 }
}

```

### **Unordered list opening**

If the tag is an unordered list tag, a new `Triple` is pushed to the stack, containing the tag, the “bulleted” attribute used to specify whether bullets should be shown, and the

`NUMBER ListType`, although it will not be used.

### **Ordered list opening**

If the tag is an ordered list tag, a new `Triple` is push to the stack, containing the tag, the numbered attribute, and the `ListType` found from the string value of the “type” attribute.

An index of 1 and the `ListType` are then push to the `mOIIIndices` stack.

### **List item opening**

If the tag is a list tag, the last character of the output is checked to ensure that it is a newline, otherwise the line will not wrap.

If `mLists` is not empty, the parent list tag is checked.

If it is an ordered list, the `o1` span is started, and a `Pair` containing the top index in the stack plus one, and the parent `ListType` is pushed to the stack.

If it is an unordered list, the `u1` span is started

If `mLists` is empty, this means that there is list item without a parent.

A new `o1` span is started, and the new index is pushed to `mOIIIndices`.

### **Unordered list closing**

`mLists` is popped, removing the `u1` tag.

### **Ordered list closing**

`mLists` and `mOIIIndices` are popped, removing the `o1` tag and its index.

## List item closing

If mLists is non-empty the parent tag is checked.

If it is an unordered list tag, the Editable is passed to handleULTag.

### HtmlTagHandler.java

```
private void handleULTag(Editable output) {
 if(output.length() > 0 && output.charAt(output.length() - 1) != '\n')
 {
 output.append("\n");
 }

 if(mLists.peek().second) {
 //Check for checkboxes
 if(output.length() > 2 &&
 ((output.charAt(0) >= '\u2610' && output.charAt(0) <=
'\u2612') ||
 (output.charAt(1) >= '\u2610' && output
 .charAt(1) <= '\u2612'))
) {
 end(output, Ul.class, false,
 new LeadingMarginSpan.Standard(
 mListIndent * (mLists.size() - 1)))
);
 } else {
 end(output, Ul.class, false,
 new LeadingMarginSpan.Standard(
 mListIndent * (mLists.size() - 1)),
 new BulletSpan(mSingleIndent)
);
 }
} else {
 end(output, Ul.class, false,
 new LeadingMarginSpan.Standard(mListIndent *
(mLists.size() - 1))
);
}
}
```

This method checks the last character, ensuring that the tag ends in a newline.

It then checks if the list should have bullets, if not the tag is ended with a new `LeadingMarginSpan` proportional to the list size.

If the unordered list should have bullets, the bullets can either be bullet points or checkboxes.

If the `Editable` is sufficiently long for the character to be there, the output is checked to see whether the first or second character is within the range of the checkbox characters.

If it is, a `LeadingMarginSpan` is applied.

Otherwise, a `LeadingMarginSpan` and a `BulletSpan` are applied.

## Table tags

## Table tag opening

When a table tag is opened, the Table span is started.  
 If the table depth is 0, the StringBuilder is recreated.  
 The table level is then incremented.

## Table tag closing

A closing table tag is passed to handleTableTag

### HtmlTagHandler.java

```
private void handleTableTag(Editable output) {
 mTableLevel--;
 // When we're back at the root-level table
 if(mTableLevel == 0) {
 final Table obj = getLast(output, Table.class);
 final int start = output.getSpanStart(obj);
 output.removeSpan(obj); //Remove the old span
 output.insert(start, "\n \n");
 final TableSpan table = new
TableSpan(mTableHtmlBuilder.toString(), mTableHandler);
 output.setSpan(table, start, start + 2,
Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);
 output.setSpan(new WrappingClickableSpan(table), start, start + 3,
 Spanned.SPAN_EXCLUSIVE_EXCLUSIVE
);
}

} else {
 end(output, Table.class, false);
}
}
```

The table level is decremented, as we have just closed a table tag.

If the table level is 0, the last Table is found, its start position stored, and the span removed.

Two newlines separated by a space are then inserted, making space for the TableSpan.

The TableSpan is created with the value of the table StringBuilder, and the TableClickHandler.

The span is inserted, and then a WrappingClickableSpan is inserted around the TableSpan.

If the table level is not 0, the Table span is closed.

## Font tags

### Font tag opening

When a font tag is opened, four attributes are extracted.

These are the font, the text colour, the background colour, and whether the background colour should be rounded.

If the font is non-null and non-empty, a new `Font` span is started with the `Font`.  
 If the foreground colour is non-null and non-empty, a new `ForegroundColor` span  
 is started with the font colour.

If the background colour is non-null and non-empty, a new `BackgroundColor` span  
 is started with the background colour, and whether it should be rounded.

### Font tag closing

A closing font tag is passed to `handleFontTag`

#### `HtmlTagHandler.java`

```
private void handleFontTag(Editable output) {
 final ForegroundColor fgc = getLast(output, ForegroundColor.class);
 final BackgroundColor bgc = getLast(output, BackgroundColor.class);
 final Font f = getLast(output, Font.class);
 if(fgc != null) {
 final int start = output.getSpanStart(fgc);
 final int end = output.length();
 output.removeSpan(fgc);
 output.setSpan(new
ForegroundColorSpan(safelyParseColor(fgc.color)), start, end,
 Spannable.SPAN_EXCLUSIVE_EXCLUSIVE
);
 }
 if(bgc != null) {
 final int start = output.getSpanStart(bgc);
 final int end = output.length();
 output.removeSpan(bgc);

 final int color = safelyParseColor(bgc.color);
 if(bgc.rounded) {
 output.insert(end, "\u00A0");
 output.insert(start, "\u00A0");
 output.setSpan(new RoundedBackgroundEndSpan(color, false),
start, start + 1,
 Spanned.SPAN_INCLUSIVE_EXCLUSIVE
);
 output.setSpan(new RoundedBackgroundEndSpan(color, true), end,
end + 1,
 Spanned.SPAN_EXCLUSIVE_INCLUSIVE
);
 output.setSpan(new BackgroundColorSpan(color), start + 1, end,
 Spannable.SPAN_INCLUSIVE_INCLUSIVE
);
 } else {
 output.setSpan(new BackgroundColorSpan(color), start, end,
 Spannable.SPAN_EXCLUSIVE_EXCLUSIVE
);
 }
 if(fgc == null) {
 output.setSpan(new
ForegroundColorSpan(TextUtils.getTextColorForBackground(color)), start, end,
 Spannable.SPAN_EXCLUSIVE_EXCLUSIVE
);
 }
 }
}
```

```

 }
 if(f != null) {
 final int start = output.getSpanStart(f);
 final int end = output.length();
 output.removeSpan(f);
 output.setSpan(new TypefaceSpan(f.face), start, end,
 Spannable.SPAN_EXCLUSIVE_EXCLUSIVE
);
 }
}

```

This extracts the spans for each of the three possible attributes.

If the `ForegroundColor` is non null, the start and end are stored, the span is removed, and a `ForegroundColorSpan` is set with the parsed color of the the `ForegroundColor` span.

The colour is parsed using `safelyParseColor`.

### **HtmlTagHandler.java**

```

private int safelyParseColor(String color) {
 try {
 return Color.parseColor(color);
 } catch(Exception e) {
 switch(color) {
 case "black":
 return Color.BLACK;
 case "white":
 return Color.WHITE;
 case "red":
 return Color.RED;
 case "blue":
 return Color.BLUE;
 case "green":
 return Color.GREEN;
 case "grey":
 return Color.GRAY;
 case "yellow":
 return Color.YELLOW;
 case "aqua":
 return 0xff00ffff;
 case "fuchsia":
 return 0xffff00ff;
 case "lime":
 return 0xff00ff00;
 case "maroon":
 return 0xff800000;
 case "navy":
 return 0xffff00ff;
 case "olive":
 return 0xff808000;
 case "purple":
 return 0xff800080;
 case "silver":
 return 0xffc0c0c0;
 case "teal":
 return 0xff008080;
 default:

```

```
 return mTextPaint.getColor();
 }
}
```

The method attempts to parse the color with `Color.parseColor`. If this fails, it switches the string across the different HTML colour values, before returning the `TextPaint` colour if a colour is not matched.

If the `BackgroundColor` is non-null, its positions are saved and it is removed.

The background colour is then parsed, and if the span should be rounded, no break space characters are inserted around the span, before TWO RoundedBackgroundEndSpans are inserted around a BackgroundColorSpan.

Otherwise, only the `BackgroundColorSpan` is inserted.

If the `Font` span is non-null, its positions are saved and it is removed.

If the font span is non null, its positions are saved and it A TypefaceSpan is then inserted across its previous range

## Code tags

A code tag is opened by starting a new code span

A code tag is closed with handleCodeTag

A code tag is closed with  
**HtmlTagHandler.java**

```
private void handleCodeTag(Editable output) {
 final Object obj = getLast(output, Code.class);
 final int start = output.getSpanStart(obj);
 final int end = output.length();
 if(end > start + 1) {
 final CharSequence chars = extractSpanText(output, Code.class);
 output.removeSpan(obj);
 output.insert(start, "\n \n"); // Another line for our CodeSpan to
cover
 final CodeSpan code = new CodeSpan(chars.toString(),
mCodeHandler);
 output.setSpan(code, start, start + 2,
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
 output.setSpan(new WrappingClickableSpan(code), start, start + 3,
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE
);
}
}
```

The tag is extracted and its position saved.

If the span has a length greater than 1, the CodeSpan is to be inserted.

First the text is extracted with `extractSpanText`.

## HtmlTagHandler.java

```
private CharSequence extractSpanText(Editable output, Class kind) {
 final Object obj = getLast(output, kind);
 final int start = output.getSpanStart(obj);
 final int end = output.length();

 final CharSequence extractedSpanText = output.subSequence(start, end);
 output.delete(start, end);
```

```

 return extractedSpanText;
}

```

This method finds the span, captures the subsequence from the `Editable`, removes it, and then returns the extracted `CharSequence`. Once the `CharSequence` has been extracted, spacing for the `CodeSpan` is inserted, the `CodeSpan` itself is inserted, and a `WrappingClickableSpan` is inserted around it.

## Center tags

Center tags are opened by starting a `Center` span . They are closed by ending the `center` span with an `AlignmentSpan` with `Layout.Alignment.ALIGN_CENTER`.

## Strikethrough tags

Strikethrough tags are opened by starting a `StrikeThrough` span. They are closed by ending the `strikeThroughSpan` with a `StrikeThroughSpan`.

## Table row, header, and data

Table row, header, and data tags are started with `Tr`, `Th`, and `Td` spans respectively, and ended with these spans.

## Horizontal rule tags

Horizontal rule tags are opened by starting a new `HorizontalRule` span.

They are ended with `handleHorizontalRuleTag`

### `HtmlTagHandler.java`

```

private void handleHorizontalRuleTag(Editable output) {
 final Object obj = getLast(output, HorizontalRule.class);
 final int start = output.getSpanStart(obj);
 output.removeSpan(obj); //Remove the old span
 output.replace(start, output.length(), " "); //We need a non-empty
span
 output.setSpan(new HorizontalRuleSpan(), start, start + 1, 0);
//Insert the bar span
}

```

This finds the `HorizontalRule` span, stores its length, removes the span, inserts a space for the `HorizontalRuleSpan` to occupy, and then inserts the `HorizontalRuleSpan`.

## Blockquote tags

Blockquote tags are opened by starting a new `BlockQuote` span. They are ended with `handleBlockQuoteTag`

This finds the BlockQuote span, stores its start and end positions, removes the BlockQuote span, and inserts a QuoteSpan across these indices.

### HtmlTagHandler.java

```
private void handleBlockQuoteTag(Editable output) {
 final Object obj = getLast(output, BlockQuote.class);
 final int start = output.getSpanStart(obj);
 final int end = output.length();
 output.removeSpan(obj);
 output.setSpan(new QuoteSpan(), start, end,
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
}
```

### A tags

A tags are opened by starting a new A tag with the extracted "href" attribute. They are ended with handleATag

### HtmlTagHandler.java

```
private void handleATag(Editable output) {
 final A obj = getLast(output, A.class);
 final int start = output.getSpanStart(obj);
 final int end = output.length();
 output.removeSpan(obj);
 if(isValidURL(obj.href)) {
 output.setSpan(new CleanURLESpan(obj.href, mLinkHandler), start,
end,
 Spannable.SPAN_INCLUSIVE_EXCLUSIVE
);
} else {
 output.insert(start, obj.href.concat(" "));
}
}
```

This finds and removes the span.

It then checks if the href is valid.

If it is valid, a CleanURLESpan is inserted.

If it is not valid, the href is inserted before the a tag text.

### Image tags

Image tags are handled immediately with handleImageTag.

The Drawable is initially constructed as a transparent ColorDrawable.

If the ImageGetter is non-null, the drawable is loaded from it.

The original length is stored, before an object replacement character, `OBJ`, is inserted.

The ClickableImageSpan is then created, inserted, and wrapped with a WrappingClickableSpan.

As the spans are inserted over the object replacement character, they will draw over it once they have been loaded.

As no MARK span is inserted, it does not need to be (and obviously cannot) be removed.

### InlineCode tags

Inline code tags are started with `InlineCode` spans.

They are closed with `handleInlineCodeTag`

#### HtmlTagHandler.java

```
private void handleInlineCodeTag(Editable output) {
 final Object obj = getLast(output, InlineCode.class);
 final int start = output.getSpanStart(obj);
 final int end = output.length();
 output.removeSpan(obj);
 output.setSpan(new InlineCodeSpan(mTextPaint.getTextSize()), start,
end,
 Spannable.SPAN_INCLUSIVE_EXCLUSIVE
);
}
```

This replaces the `InlineCode` span with an `InlineCodeSpan`, passing the correct text size.

## Markdown editing

Now that markdown parsing has been implemented, markdown editing can be implemented.

The first order objectives under objective 10 were to, "Implement toggling of a text editor between raw markdown and formatted markdown", "Add utility buttons for markdown features", and "Add utility buttons for markdown features".

As each of these objectives are closely linked, being part of the same UI element, it makes sense to implement them together.

### Implementing a re-usable editor

The markdown editor will be used in numerous sections of the app. In some cases such as editing project cards it needs only allow the input and preview of a single block of markdown content, whereas in others such as editing an Issue it must also be able to facilitate editing aspects of the model that it is working with, such as assignees and tags.

This can be achieved by using a single primary layout, containing the control elements for the editor, and a stub to contain the elements related to the particular model being edited.

#### activity\_markdown\_editor.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:orientation="vertical"
 android:layout_width="match_parent"
 android:layout_height="match_parent">

 <LinearLayout
 android:id="@+id/markdown_activity_buttons"
 android:layout_width="match_parent"
 android:layout_height="?android:attr actionBarSize"
 android:layout_alignParentTop="true"
 android:orientation="horizontal"
 android:background="@color/colorPrimary"
 android:elevation="2dp"
 android:baselineAligned="false">

 <FrameLayout
 android:layout_width="0dp"
 android:layout_weight="0.5"
 android:layout_height="wrap_content"
 android:layout_gravity="center_vertical">
```

```
<Button
 android:id="@+id/markdown_editor_discard"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_gravity="center_horizontal"
 android:text="@string/action_discard"
 android:drawableStart="@drawable/ic_cancel"
 android:background="?android:attr/selectableItemBackground"
 style="@style/Widget.AppCompat.Button.Borderless"/>

</FrameLayout>

<FrameLayout
 android:layout_width="0dp"
 android:layout_weight="0.5"
 android:layout_height="wrap_content"
 android:layout_gravity="center_vertical">

 <Button
 android:id="@+id/markdown_editor_done"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_gravity="center_horizontal"
 android:text="@string/action_done"
 android:drawableStart="@drawable/ic_done"
 android:background="?android:attr/selectableItemBackground"
 style="@style/Widget.AppCompat.Button.Borderless"/>

 </FrameLayout>

</LinearLayout>

<ScrollView
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:layout_below="@+id/markdown_activity_buttons"
 android:layout_above="@+id/markdown_edit_scrollview"
 android:fillViewport="true">

 <ViewStub
 android:id="@+id/editor_stub"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"/>

</ScrollView>

<HorizontalScrollView
 android:id="@+id/markdown_edit_scrollview"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_alignParentBottom="true"
 android:scrollbars="none"
 android:layout_marginTop="8dp"
 android:minHeight="48dp"
 android:background="@color/cardview_dark_background">

 <LinearLayout
 android:id="@+id/markdown_edit_buttons"
 android:layout_width="wrap_content"
```

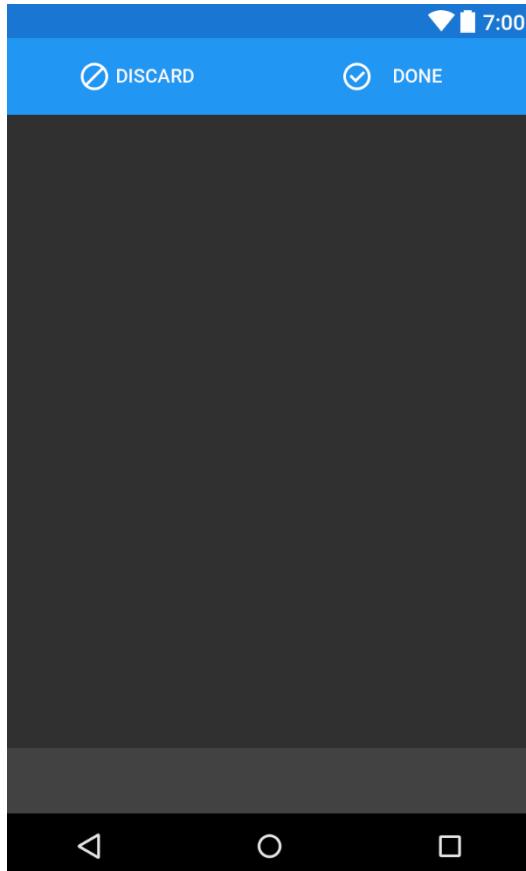
```
 android:layout_height="wrap_content"
 android:orientation="horizontal">

</LinearLayout>

</HorizontalScrollView>

</RelativeLayout>
```

The markdown\_editor layout contains three children.



In vertical order, from top to bottom:

The first child contains the navigation buttons for the entire editor, allowing the edit action to be completed or discarded.

The second child is the content layout. The ScrollView wraps a ViewStub which will be inflated to display the editor layout.

The third and final child is a HorizontalScrollView containing a LinearLayout which will be used to display the list of editor action buttons.

## Utilities

### KeyBoardDismisingDialogFragment

One of the longest running irritations with Android is its handling of the keyboard.

There is no consistent method for changing the keyboard visibility, and the keyboard does not dismiss in scenarios when it would be expected to such as when a dialog is dismissed.

### KeyboardDismissingDialogFragment.java

```
package com(tpb.projects.editors;

import android.content.Context;
import android.os.Bundle;
import android.support.v4.app.DialogFragment;
import android.view.inputmethod.InputMethodManager;

import com(tpb.projects.R;

/**
 * Created by theo on 31/12/16.
 */

public abstract class KeyboardDismissingDialogFragment extends DialogFragment {
 @Override
 public void dismiss() {
 //Check if a text input is focused
 if(getDialog().getCurrentFocus() != null) {
 final InputMethodManager imm = (InputMethodManager) getContext()
 .getSystemService(Context.INPUT_METHOD_SERVICE);
 //Close the keyboard with the window token of the focused text
 input
 imm.hideSoftInputFromWindow(getDialog().getCurrentFocus().getWindowToken(),
 InputMethodManager.HIDE_NOT_ALWAYS
);
 }
 super.dismiss();
 }

 //Does the same as dismiss
 @Override
 public void dismissAllowingStateLoss() {
 if(getDialog().getCurrentFocus() != null) {
 final InputMethodManager imm = (InputMethodManager) getContext()
 .getSystemService(Context.INPUT_METHOD_SERVICE);

 imm.hideSoftInputFromWindow(getDialog().getCurrentFocus().getWindowToken(),
 InputMethodManager.HIDE_NOT_ALWAYS
);
 }
 }
}
```

```

 super.dismissAllowingStateLoss();
 }

 @Override
 public void onActivityCreated(Bundle savedInstanceState) {
 super.onActivityCreated(savedInstanceState);
 //Enable the slide in animation
 getWindow().getAttributes().windowAnimations =
R.style.DialogAnimation;
 }

}

```

The KeyboardDismissingDialogFragment is a subclass of DialogFragment which ensures that the keyboard is closed when the dialog is dismissed. This is done by checking if a View is focused, and if so hiding the keyboard from the focused View.

## MultiChoiceDialog

The MultiChoiceDialog extends KeyboardDismissingDialogFragment and builds a multi choice dialog using the AlertDialog builder, allowing colouring the views.

### MultiChoiceDialog.java

```

package com(tpb.projects.editors;

import android.app.AlertDialog;
import android.app.Dialog;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v7.widget.AppCompatCheckedTextView;
import android.text.SpannableStringBuilder;
import android.text.Spanned;
import android.text.style.BackgroundColorSpan;
import android.text.style.ForegroundColorSpan;
import android.widget.AbsListView;
import android.widget.ListView;

import com(tpb.mdtext.TextUtils;
import com(tpb.projects.R;

/**
 * Created by theo on 29/12/16.
 */

public class MultiChoiceDialog extends KeyboardDismissingDialogFragment {

 private MultiChoiceDialogListener listener;
 private String[] choices;
 private boolean[] checked;
 private ListView listView;
 private int[] colors;

 @NonNull
 @Override
 public Dialog onCreateDialog(Bundle savedInstanceState) {

```

```

 final AlertDialog.Builder builder = new
AlertDialog.Builder(getActivity());

 final Bundle arguments = getArguments();
 final int titleRes =
arguments.getInt(getContext().getString(R.string.intent_title_res));
 builder.setTitle(titleRes);
 builder.setMultiChoiceItems(choices, checked, (dialogInterface, i, b)
-> checked[i] = b);
 builder.setPositiveButton(R.string.action_ok, (dialogInterface, i) ->
{
 if(listener != null) listener.choicesComplete(choices, checked);
 });
 builder.setNegativeButton(R.string.action_cancel, (dialogInterface, i) -> {
 if(listener != null) listener.choicesCancelled();
 });
 final AlertDialog dialog = builder.create();
listView = dialog.getListView();

 dialog.setOnShowListener(dialogInterface -> {
 if(colors != null) addBackgroundSetterListener();
 });

 return dialog;
 }

 private void addBackgroundSetterListener() {
 final SpannableStringBuilder[] cache = new
SpannableStringBuilder[choices.length];
 listView.setOnScrollListener(new AbsListView.OnScrollListener() {
 @Override
 public void onScrollStateChanged(AbsListView absListView, int
scrollState) {
 }

 /*
 We have to get the TextViews after they are bound, so we
wait for a scroll
 and then iterate through the TextView on screen
 */
 }

 @Override
 public void onScroll(AbsListView absListView, int firstVisible,
int visibleCount, int totalCount) {
 for(int i = firstVisible; i < firstVisible + visibleCount;
i++) {
 try {
 if(cache[i] == null) {
 final SpannableStringBuilder builder = new
SpannableStringBuilder();
 builder.append(choices[i]);
 builder.setSpan(
 new BackgroundColorSpan(colors[i]),
 0,
 builder.length(),
 Spanned.SPAN_EXCLUSIVE_EXCLUSIVE
);
 builder.setSpan(

```

```

 new ForegroundColorSpan(
 TextUtils
 .getTextColorForBackground(
 colors[i])
),
 0,
 builder.length(),
 Spanned.SPAN_EXCLUSIVE_EXCLUSIVE
);
 cache[i] = builder;
 }

 ((AppCompatCheckedTextView) listView.getChildAt(i))
 .setText(cache[i]);
 } catch(ClassCastException ignored) {
}

}
}
}

public void setBackgroundColors(int[] colors) {
 this.colors = colors;
}

public void setChoices(String[] choices, boolean[] checked) {
 this.choices = choices;
 this.checked = checked;
}

public void setListener(MultiChoiceDialogListener listener) {
 this.listener = listener;
}

public interface MultiChoiceDialogListener {
 void choicesComplete(String[] choices, boolean[] checked);
 void choicesCancelled();
}
}

```

The `MultiChoiceDialogListener` interface is used for returning the chosen items, or notifying that the choice selection has been cancelled.

When a `MultiChoiceDialog` is created, the choices must be set, and optionally the colours for each choice can be set.

When `onCreateDialog` is called, a new `AlertDialog.Builder` is created, and the title is set from the resource id passed to the dialog.

The multi choice items are then set, and listeners are added for the positive and negative buttons which call `choicesComplete` and `choicesCancelled` respectively.

The listener set with the multi choice items sets the value in the checked array at the toggled position.

The AlertDialog is then built, inflating the layout.

The ListView can then be extracted from the AlertDialog, and if there is a colors array, a listener is added to colour each of the ListView items, as this is not a built in feature.

addBackgroundSetterListener creates an array of SpannableStringBuilder to cache the coloured spans.

It then adds an OnScrollListener, and in onScroll it sets the text of each of the visible TextViews.

If no SpannableStringBuilder has been built, it is constructed with a BackgroundColorSpan and

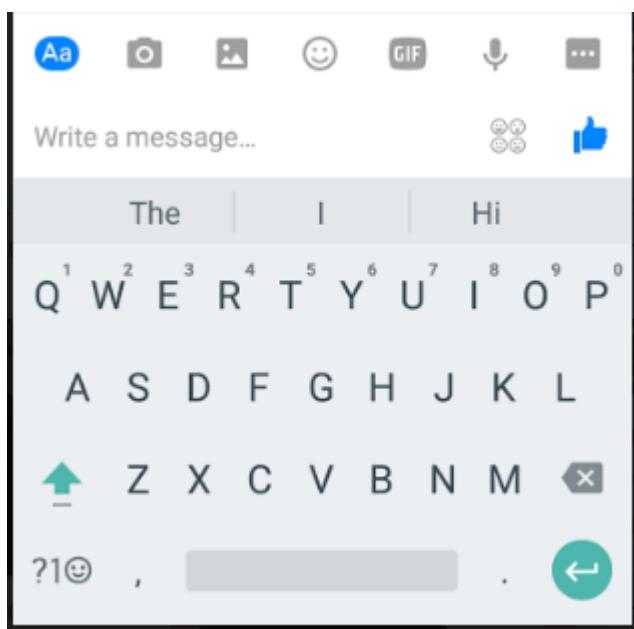
a ForegroundColorSpan using TextUtils.getTextColorForBackground and then stored in the cache array.

The TextView text is then set to the SpannableStringBuilder.

### MarkdownButtonAdapter

Objectives 10.ii and 10.iii are to implement buttons for inserting markdown control sequences into text.

Many apps, such as Facebook messenger below, augment the keyboard by showing an extra row of buttons above it, specifically for the content type being input.



As explained above, the HorizontalScrollView in activity\_markdown\_editor will be used to display these buttons.

As the buttons are the same throughout each of the editors, a general adapter is used with an interface for inserting text or showing the format preview, which allows individual implementations of EditorActivity to deal with the views that they have inflated.

## MarkdownButtonAdapter.java

```
package com(tpb.projects.editors;

import android.content.Intent;
import android.support.annotation.DrawableRes;
import android.support.annotation.NonNull;
import android.support.v7.app.AlertDialog;
import android.view.LayoutInflater;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.LinearLayout;

import com(tpb.projects.R;
import com(tpb.projects.util.UI;

/**
 * Created by theo on 10/02/17.
 */

class MarkdownButtonAdapter {

 private final EditorActivity mParent;
 private final LinearLayout mScrollView;
 private final MarkdownButtonListener mListener;

 MarkdownButtonAdapter(EditorActivity parent, @NonNull LinearLayout scrollView, @NonNull MarkdownButtonListener listener) {
 mParent = parent;
 mScrollView = scrollView;
 mListener = listener;
 initViews();
 }

 private void initViews() {
 ImageButton preview = createImageButton(R.drawable.ic_preview);
 preview.setOnClickListener((v) -> mListener.previewCalled());

 preview = createImageButton(R.drawable.ic_insert_link);
 preview.setOnClickListener((v) -> showInsertLinkDialog());

 preview = createImageButton(R.drawable.ic_photo);
 preview.setOnClickListener((v) -> mParent.showImageUploadDialog());

 preview = createImageButton(R.drawable.ic_format_bold);
 preview.setOnClickListener((v) -> mListener.snippetEntered("****", 2));

 preview = createImageButton(R.drawable.ic_format_italic);
 preview.setOnClickListener((v) -> mListener.snippetEntered("**", 1));

 preview = createImageButton(R.drawable.ic_format_strikethrough);
 preview.setOnClickListener((v) -> mListener.snippetEntered("~~~", 2));

 preview = createImageButton(R.drawable.ic_check_box_checked);
 preview.setOnClickListener((v) -> mListener.snippetEntered(" [x] ", 5));
 }
}
```

```
 preview = createImageButton(R.drawable.ic_check_box_empty);
 preview.setOnClickListener((v) -> mListener.snippetEntered(" [] ",
4));

 preview = createImageButton(R.drawable.ic_horizontal_rule);
 preview.setOnClickListener((v) -> mListener.snippetEntered("\n---\n",
5));

 preview = createImageButton(R.drawable.ic_format_list_bulleted);
 preview.setOnClickListener((v) -> mListener.snippetEntered(" * ", 3));

 preview = createImageButton(R.drawable.ic_format_list_numbered);
 preview.setOnClickListener((v) -> mListener.snippetEntered(" 1. ", 3));

 preview = createImageButton(R.drawable.ic_format_quote);
 preview.setOnClickListener((v) -> mListener.snippetEntered("> ", 2));

 preview = createImageButton(R.drawable.ic_format_code);
 preview.setOnClickListener((v) ->
mListener.snippetEntered("``` \n\n``` ", 4));

 preview = createImageButton(R.drawable.ic_emoticon);
 preview.setOnClickListener((v) -> showInsertEmoticonActivity());

 preview = createImageButton(R.drawable.ic_character);
 preview.setOnClickListener((v) -> showInsertCharacterActivity());
 }

 private ImageButton createImageButton(@DrawableRes int resId) {
 final ImageButton ib = (ImageButton) LayoutInflater
 .from(mParent)
 .inflate(
 R.layout.shard_markdown_button,
 mScrollView,
 false
);
 ib.setImageResource(resId);
 mScrollView.addView(ib);
 return ib;
 }

 private void showInsertLinkDialog() {
 final LinearLayout wrapper = new LinearLayout(mParent);
 wrapper.setOrientation(LinearLayout.VERTICAL);
 wrapper.setPaddingRelative(UI.pxFromDp(16), 0, UI.pxFromDp(16), 0);

 final EditText text = new EditText(mParent);
 text.setHint(R.string.hint_url_description);
 wrapper.addView(text);

 final EditText url = new EditText(mParent);
 url.setHint(R.string.hint_url_url);
 wrapper.addView(url);

 final AlertDialog.Builder builder = new AlertDialog.Builder(mParent);
 builder.setTitle(R.string.title_insert_link);
 builder.setView(wrapper);
```

```

 builder.setPositiveButton(R.string.action_insert, (v, di) -> {
 mListener.snippetEntered(
 String.format(
 mParent.getString(R.string.text_md_link),
 text.getText().toString(),
 url.getText().toString()
),
 0
);
 });
 builder.setNegativeButton(R.string.action_cancel, null);
 }

 private void showInsertEmoticonActivity() {
 mParent.startActivityForResult(new Intent(mParent,
EmojiActivity.class),
 EmojiActivity.REQUEST_CODE_CHOOSE_EMOJI
);
 }

 private void showInsertCharacterActivity() {
 mParent.startActivityForResult(new Intent(mParent,
CharacterActivity.class),
 CharacterActivity.REQUEST_CODE_INSERT_CHARACTER
);
 }

 interface MarkdownButtonListener {
 void snippetEntered(String snippet, int relativePosition);
 String getText();
 void previewCalled();
 }
}

```

The `MarkdownButtonAdapter` is constructed with an `EditorActivity`, the `LinearLayout` within which the `Views` are to be inserted, and the `MarkdownButtonListener` which will be called when a button is clicked. The class variables are assigned, and `initViews` is called. `initViews` uses `createImageButton` for each new `ImageButton`. This method takes a resource id for the image to show on the button, inflates the button, sets its image resource id, adds the button to the `LinearLayout` and returns the button. 15 `ImageButtons` are created, for 15 different actions.

1. The first button is for displaying the markdown in its formatted state, and calls `previewCalled` on the listener.
2. The second button calls `showInsertLinkDialog`  
This creates a new `LinearLayout` with standard 16dp padding, inflates

two `EditTexts` inside the `LinearLayout`, and then builds an `AlertDialog` with the layout.

When the positive button is clicked, the `snippetEntered` method is called on the listener, inserting the formatted URL.

3. The third button calls `showImageUploadDialog` on the `EditorActivity` parent
4. The fourth button inserts the quadruple asterisks for bold text, and positions the cursor between the two pairs of asterisks.
5. The fifth button inserts the double asterisks for italic text, and positions the cursor between the two.
6. The sixth button inserts the quadruple tildes for strikethrough text, and positions the cursor between the two.
7. The seventh button inserts the ticked checkbox characters, and positions the cursor after them.
8. The eighth button inserts the empty checkbox characters, and positions the cursor after them.
9. The ninth button inserts the triple hypens for a thematic break between two newlines, and positions the cursor after them.
10. The tenth button inserts a spaced asterisk for a bullet point list item, and positions the cursor after it.
11. The eleventh button inserts the first item in a numbered list, and positions the cursor after it.
12. The twelfth button inserts the right chevron and positions the cursor after it.
13. The thirteenth button inserts the two sets of triple backticks for a code block, separated by two newlines, and positions the cursor between the two sets.
14. The fourteenth button launches the `EmojiActivity` from the parent Activity.
15. The fifteenth button launches the `CharacterActivity` from the parent Activity.

## The `EditorActivity`

### `EditorActivity.java`

```
package com(tpb.projects.editors;

import android.app.AlertDialog;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Environment;
```

```
import android.os.Handler;
import android.os.Looper;
import android.os.ParcelFileDescriptor;
import android.provider.MediaStore;
import android.support.v4.content.FileProvider;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.util.Base64;
import android.util.Log;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.Toast;

import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Uploader;
import com(tpb.projects.BuildConfig;
import com(tpb.projects.R;
import com(tpb.projects.common.CircularRevealActivity;
import com(tpb.projects.util.UI;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileDescriptor;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;

/**
 * Created by theo on 16/02/17.
 */

public abstract class EditorActivity extends CircularRevealActivity {
 private static final String TAG = EditorActivity.class.getSimpleName();

 private static final int REQUEST_CAMERA = 9403; //Random request codes
 private static final int SELECT_FILE = 6113;
 private String mCurrentFilePath;
 protected ProgressDialog mUploadDialog;

 @Override
 protected void onActivityResult(int requestCode, int resultCode, Intent data) {
 super.onActivityResult(requestCode, resultCode, data);
 if(resultCode == AppCompatActivity.RESULT_OK) {
 if(requestCode == EmojiActivity.REQUEST_CODE_CHOOSE_EMOJI) {

emojiChosen(data.getStringExtra(getString(R.string.intent_emoji)));
 } else if(requestCode ==
CharacterActivity.REQUEST_CODE_INSERT_CHARACTER) {

insertString(data.getStringExtra(getString(R.string.intent_character)));
 } else {
 final ProgressDialog pd = new ProgressDialog(this);
 pd.setCanceledOnTouchOutside(false);
 pd.setCancelable(false);
 if(requestCode == REQUEST_CAMERA) {

pd.setTitle(R.string.title_image_conversion);
```

```
 pd.show();
 AsyncTask.execute(() -> { // Execute asynchronously
 final Bitmap image =
BitmapFactory.decodeFile(mCurrentFilePath);
 final ByteArrayOutputStream stream = new
ByteArrayOutputStream();
 image.compress(Bitmap.CompressFormat.PNG, 100,
stream);
 pd.cancel();
 });

uploadImage(Base64.encodeToString(stream.toByteArray(), Base64.DEFAULT));
 });

} else if(requestCode == SELECT_FILE) {
 final Uri selectedFile = data.getData();
 pd.setTitle(R.string.title_image_conversion);
 pd.show();
 AsyncTask.execute(() -> {
 try {
 final String image =
attemptLoadImage(selectedFile);
 pd.cancel();
 uploadImage(image);
 } catch(IOException ioe) {
 pd.cancel();
 imageLoadException(ioe);
 }
 });
}
}

abstract void imageLoadComplete(String url);

abstract void imageLoadException(IOException ioe);

void showImageUploadDialog() {
 final CharSequence[] items = {
 getString(R.string.text_take_a_picture),
 getString(R.string.text_choose_from_gallery),
 getString(R.string.text_insert_image_link),
 getString(R.string.action_cancel)
 };
 final AlertDialog.Builder builder = new AlertDialog.Builder(this);
 builder.setTitle(getString(R.string.text_upload_an_image));
 builder.setItems(items, (dialog, which) -> {
 if(which == 3) {
 dialog.dismiss();
 } else if(which == 2) {
 displayImageLinkDialog();
 } else {
 if(mUploadDialog == null) {
 mUploadDialog = new ProgressDialog(EditorActivity.this);
 mUploadDialog.setTitle(R.string.title_image_upload);
 }
 if(which == 0) {

```

```
 attemptTakePicture();
 } else if(which == 1) {
 final Intent intent = new Intent(
 Intent.ACTION_GET_CONTENT,
 MediaStore.Images.Media.EXTERNAL_CONTENT_URI
);
 intent.setType("image/*");
 startActivityForResult(intent, SELECT_FILE);
 }
}
builder.show();
}

private void displayImageLinkDialog() {
 final LinearLayout wrapper = new LinearLayout(this);
 wrapper.setOrientation(LinearLayout.VERTICAL);
 wrapper.setPaddingRelative(UI.pxFromDp(16), 0, UI.pxFromDp(16), 0);

 final EditText desc = new EditText(this);
 desc.setHint(R.string.hint_url_description);
 wrapper.addView(desc);

 final EditText url = new EditText(this);
 url.setHint(R.string.hint_url_url);
 wrapper.addView(url);

 final AlertDialog.Builder builder = new AlertDialog.Builder(this);
 builder.setTitle(R.string.title_insert_image_link);
 builder.setView(wrapper);

 builder.setPositiveButton(R.string.action_insert, (v, di) -> {
 insertString(String.format(getString(R.string.text_image_link_with_desc),
 desc.getText().toString(),
 url.getText().toString()
));
 });
 builder.setNegativeButton(R.string.action_cancel, null);
 builder.create().show();
}

private void attemptTakePicture() {
 final Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
 // Check if there is an activity which can take a picture
 if(intent.resolveActivity(getApplicationContext()) != null) {
 File photoFile = null;
 try {
 //Create the file for the image to be stored in
 photoFile = createImageFile();
 } catch(IOException ioe) {
 Log.e(TAG, "attemptTakePicture: ", ioe);
 imageLoadException(ioe);
 }

 if(photoFile != null) {
 final Uri photouri = FileProvider
```

```

 .getUriForFile(this, "com(tpb.projects.provider",
photoFile);
 intent.putExtra(MediaStore.EXTRA_OUTPUT, photoURI);
 startActivityForResult(intent, REQUEST_CAMERA);
 } else {
 imageLoadException(
 new
IOException(getString(R.string.error_image_file_not_created)));
 }
 } else {
 Toast.makeText(this, R.string.error_no_application_for_picture,
Toast.LENGTH_SHORT)
 .show();
 }
}

private File createImageFile() throws IOException {
 //Create an image file with a formatted name
 final String timeStamp = new
SimpleDateFormat("yyyyMMdd_HHmss").format(new Date());
 final String imageFileName = "JPEG_" + timeStamp + "_";
 final File storageDir =
getExternalFilesDir(Environment.DIRECTORY_PICTURES);
 final File image = File.createTempFile(
 imageFileName, /* prefix */
 ".jpg", /* suffix */
 storageDir /* directory */
);
 mCurrentFilePath = image.getAbsolutePath();
 return image;
}

private String attemptLoadImage(Uri uri) throws IOException {
 // Open FileDescriptor in read mode
 final ParcelFileDescriptor parcelFileDescriptor =
 getContentResolver().openFileDescriptor(uri, "r");
 final FileDescriptor fileDescriptor =
parcelFileDescriptor.getFileDescriptor();
 //Decode to a bitmap, and convert to a byte array
 final Bitmap image =
BitmapFactory.decodeFileDescriptor(fileDescriptor);
 final ByteArrayOutputStream stream = new ByteArrayOutputStream();
 image.compress(Bitmap.CompressFormat.PNG, 100, stream);
 //Return base64 string for Imgur
 return Base64.encodeToString(stream.toByteArray(), Base64.DEFAULT);
}

private void uploadImage(String image64) {
 new Handler(Looper.getMainLooper()).postAtFrontOfQueue(() ->
mUploadDialog.show());
 Uploader.uploadImage(
 new Uploader.ImgurUploadListener() {
 @Override
 public void imageUploaded(String link) {
 mUploadDialog.cancel();
 final String snippet =
String.format(getString(R.string.text_image_link), link);
 imageLoadComplete(snippet);
 }
 }
);
}

```

```

 @Override
 public void uploadError(APIHandler.APIError
error) {
 mUploadDialog.cancel();
 Toast.makeText(EditorActivity.this,
error.resId, Toast.LENGTH_SHORT).show();
 }
 },
 image64,
 (bUP, bTotal) -> mUploadDialog.setProgress(Math.round((100 *
bUP) / bTotal)),
 BuildConfig.IMGUR_CLIENT_ID
);
}

protected abstract void emojiChosen(String emoji);

protected abstract void insertString(String c);

}

```

The `EditorActivity` is an abstract class dealing with the process for creating and uploading images, as well as inserting special characters and emojis.

## Image uploading

Objective 10.iii.a is to implement a feature allowing the user to upload an image of their choosing to a hosting service, retrieve the URL for the image, and insert it into the `EditText`.

I chose to use Imgur to host user images as it is established, free, sufficiently fast, and provides an hourly upload limit of 1250 images for authenticated clients which is more than sufficient.

A client ID and secret can be generated on the Imgur website for an app linked to any Imgur user account.

The image upload endpoint is simple to use, requiring only authentication and the image.

<b>Method</b>	<b>POST</b>
Route	<a href="https://api.imgur.com/3/image">https://api.imgur.com/3/image</a>
Alternative Route	<a href="https://api.imgur.com/3/upload">https://api.imgur.com/3/upload</a>
Response Model	Basic

## Parameters

Key	Required	Description
image	required	A binary file, base64 data, or a URL for an image. (up to 10MB)
album	optional	The id of the album you want to add the image to. For anonymous albums, {album} should be the deletehash that is returned at creation.
type	optional	The type of the file that's being sent; file, base64 or URL
name	optional	The name of the file, this is automatically detected if uploading a file with a POST and multipart / form-data
title	optional	The title of the image.
description	optional	The description of the image.

Image uploading is handled with the Uploader.

### Uploader.java

```
package com(tpb.github.data;

import android.content.Context;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.util.Log;

import com.androidnetworking.AndroidNetworking;
import com.androidnetworking.common.Priority;
import com.androidnetworking.error.ANError;
```

```
import com.androidnetworking.interfaces.JSONObjectRequestListener;
import com.androidnetworking.interfaces.UploadProgressListener;

import org.json.JSONObject;

/**
 * Created by theo on 15/02/17.
 */

public class Uploader extends APIHandler {

 private static final String IMGUR_AUTH_KEY = "Authorization";
 private static final String IMGUR_AUTH_FORMAT = "Client-ID %1$s";

 protected Uploader(Context context) {
 super(context);
 }

 public static void uploadImage(@NonNull final ImgurUploadListener
 listener, String image64, @Nullable UploadProgressListener uploadListener,
 @NonNull String clientId) {
 AndroidNetworking.upload("https://api.imgur.com/3/image")
 .addHeaders(IMGUR_AUTH_KEY,
 String.format(IMGUR_AUTH_FORMAT,
 clientId
))
 .addMultipartParameter("image", image64)
 .setPriority(Priority.HIGH)
 .build()
 .setUploadProgressListener(uploadListener)
 .getAsJSONObject(new JSONObjectRequestListener() {
 @Override
 public void onResponse(JSONObject response) {
 try {
 final String link =
 response.getJSONObject("data").getString("link");
 listener.imageUploaded(link);
 } catch(Exception e) {
 Log.e("Uploader", "onResponse: ", e);
 }
 }

 @Override
 public void onError(ANError anError) {
 listener.uploadError(parseError(anError));
 }
 });
 }

 public interface ImgurUploadListener {
 void imageUploaded(String url);

 void uploadError(APIError error);

 }
}
```

Given an `ImgurUploadListener`, base 64 encoded image, an `UploadProgressListener`, and the client id, `uploadImage` attempts to upload the image.

### Image upload process

When the user requests to insert an image link, there are multiple sources from which they may wish to choose their image.

First, they may wish to take a new picture.

Second, they may wish to choose an image from their gallery.

Third, they may already have a link to an image.

#### Image source choice

When `showUploadDialog` is called, it creates a dialog to display these options.

When the user selects an action, the item selection listener triggers the appropriate action.

If the user clicks cancel, the dialog is dismissed.

#### Pre-existing image link

If the user has selected, "Insert image link", `displayImageLinkDialog` is called.

This shows a dialog containing two `EditTexts`, one for the title and one for the description of the image.

If the user selects the "Insert" button, their input is formatted and passed to `insertString` for the implementation of `EditorActivity` to deal with.

If the user has selected another option, the image will need to be uploaded, so the `ProgressDialog` `mUploadDialog` is created if it has not been already.

If the selected option is to take a picture, `attemptTakePicture` is called.

#### New image capture

This creates a new `Intent` with the `MediaStore.ACTION_IMAGE_CAPTURE` action.

The Activity for the `Intent` is then resolved.

If it is null, there is no camera app, or no camera, and an error message is shown.

If there is a camera, a new file must be created in which to store the image.

`createImageFile` is called, which attempts to create a new image file in the system pictures directory, with a name in the form `JPEG_yyyyMMdd_HHmmss.jpg`, with the date format filled in with the current time. This should guarantee a unique image file.

If the file is created successfully, it is stored, and the `Uri` is passed as the `OUTPUT` extra for the `Intent`. Otherwise `imageLoadException` is called.

The `startActivityForResult` call means that a result will be returned to the calling Activity.

In `onActivityResult` the `requestCode` parameter will be the same as that sent with the `Intent`. If the `resultCode` is `RESULT_OK`, the request was successful.

In the case of an image taken with the camera, a new `ProgressDialog` is created and shown, and an `AsyncTask` is started to convert the `Bitmap` image into a base64 encoded

string in the PNG format.

Once the conversion is complete, the `ProgressDialog` is cancelled, and `uploadImage` is called.

### Existing image from gallery

If the user has selected "Choose from gallery", a new `Intent` with the `Intent.ACTION_GET_CONTENT` action, and

the `MediaStore.Images.Media.EXTERNAL_CONTENT_URI` Uri is created.

The `Intent` type is set to image, and the `Intent` is started for a result with the `SELECT_FILE` request code.

If the user chooses a file, the result code will be `RESULT_OK`.

The Uri of the file is accessed from the data `Intent`, the `ProgressDialog` is shown, and an `AsyncTask` is started to attempt to load the image from the `File`.

`attemptLoadImage` opens a `ParcelFileDescriptor` for the image in read only mode.

The `ParcelFileDescriptor` returns a [Java.IO](#) `FileDescriptor` which handles device specific file access.

In this case the `BitmapFactory` is used to load the file as a `Bitmap` image, which is then read as a `ByteArrayOutputStream`, compressed, and returned as a base64 string.

### Image upload

An image from the camera or the user's files must now be uploaded.

This is done when `uploadImage` is called.

The method first posts to the front of the UI queue with a runnable to show the upload dialog.

It then begins the upload process, passing an `ImgurUploadListener` which cancels the dialog and formats the image link once the image has been uploaded, before calling

`imageLoadComplete`.

The call also passes an `UploadProgressListener` which sets the progress of the `ProgressDialog` to the percentage of bytes uploaded out of the total.

### Character insertion

While most mobile keyboards provide a sufficient set of keys for general usage, most have no way to input less common characters.

Objective 10.iii.d is to implement a method for searching for and inserting unicode characters into the markdown being edited.

This has been achieved in the `CharacterActivity`.

### **CharacterActivity.java**

```
package com(tpb.projects.editors;

import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.util.Pair;
import android.support.v7.widget.GridLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.TextView;

import com(tpb.projects.R;
import com(tpb.projects.common BaseActivity;
import com(tpb.projects.util.SettingsActivity;
import com(tpb.projects.util.input.SimpleTextChangeWatcher;

import java.util.ArrayList;
import java.util.List;

import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by theo on 24/03/17.
 */

public class CharacterActivity extends BaseActivity {

 public static final int REQUEST_CODE_INSERT_CHARACTER = 7438;

 @BindView(R.id.search_title) TextView mTitle;
 @BindView(R.id.search_recycler) RecyclerView mRecycler;
 @BindView(R.id.search_search_box) EditText mSearch;

 @Override
 protected void onCreate(@Nullable Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 final SettingsActivity.Preferences prefs =
SettingsActivity.Preferences
 .getPreferences(this);
 setTheme(prefs.isDarkThemeEnabled() ? R.style.AppTheme_Dark :
R.style.AppTheme);
 setContentView(R.layout.activity_simple_search);
 ButterKnife.bind(this);
 }
}
```

```

 mTitle.setText(R.string.title_activity_characters);
 mSearch.setHint(R.string.hint_search_characters);
 mRecycler.setLayoutManager(new LinearLayoutManager(this, 3));
 final CharacterAdapter adapter = new CharacterAdapter();
 mRecycler.setAdapter(adapter);
 mSearch.addTextChangedListener(new SimpleTextChangeWatcher() {
 @Override
 public void textChanged() {
 adapter.filter(mSearch.getText().toString().toUpperCase());
 }
 });
 AsyncTask.execute(() -> {
 final ArrayList<Pair<String, String>> characters = new
 ArrayList<>();
 final int length = Character.MAX_CODE_POINT -
 Character.MIN_CODE_POINT;
 int lastIndex = 0;
 for(int i = Character.MIN_CODE_POINT; i <
 Character.MAX_CODE_POINT; i++) {
 if(Character.isDefined(i) && !Character.isISOControl(i)) {
 characters.add(Pair.create(String.valueOf((char) i),
 Character.getName(i)));
 }
 // 50 gives ~10 chunks
 if((characters.size() - lastIndex) > length / 250 || i ==
 Character.MAX_CODE_POINT - 1) {
 adapter.addCharacters(characters, lastIndex);
 lastIndex = characters.size();
 }
 }
 });
 }

 class CharacterAdapter extends
 RecyclerView.Adapter<CharacterAdapter.CharacterViewHolder> {

 private final ArrayList<Pair<String, String>> mCharacters = new
 ArrayList<>();
 private ArrayList<Integer> mFilteredPositions = new ArrayList<>();
 private ArrayList<Integer> mWorkingPositions = new ArrayList<>();
 private int mSize = 0;
 private String mLastQuery = "";

 void addCharacters(List<Pair<String, String>> characters, int start) {
 for(int i = start; i < characters.size(); i++)
 mCharacters.add(characters.get(i));

 final int originalSize = mFilteredPositions.size();
 for(int i = originalSize; i < mCharacters.size(); i++) {
 mFilteredPositions.add(i);
 }
 CharacterActivity.this.runOnUiThread(() -> {
 mSize = mFilteredPositions.size();
 notifyItemRangeInserted(originalSize,
 mFilteredPositions.size());
 });
 }

 void filter(String query) {

```

```

 AsyncTask.execute(() -> {
 mWorkingPositions = new ArrayList<>();
 if(query.isEmpty()) {
 for(int i = 0; i < mCharacters.size(); i++) {
 mWorkingPositions.add(i);
 }
 } else if(query.startsWith(mLastQuery)) {
 for(int i = 0; i < mFilteredPositions.size(); i++) {

if(mCharacters.get(mFilteredPositions.get(i)).second.contains(query)) {
 mWorkingPositions.add(mFilteredPositions.get(i));
 }
 } else {
 for(int i = 0; i < mCharacters.size(); i++) {
 if(mCharacters.get(i).second.contains(query)) {
 mWorkingPositions.add(i);
 }
 }
 }
 mLastQuery = query;
 CharacterActivity.this.runOnUiThread(() -> {
 mFilteredPositions = mWorkingPositions;
 mSize = mFilteredPositions.size();
 notifyDataSetChanged();
 });
 });
 });

}

private void choose(int pos) {

CharacterActivity.this.choose(mCharacters.get(mFilteredPositions.get(pos)).first);
}

@Override
public CharacterViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
 return new
CharacterViewHolder(LayoutInflater.from(parent.getContext()))
.inflate(R.layout.viewholder_text, parent,
false
));
}

@Override
public void onBindViewHolder(CharacterViewHolder holder, int position)
{
 holder.mCharacter.setText(mCharacters.get(mFilteredPositions.get(position)).fi
rst);

 holder mName.setText(mCharacters.get(mFilteredPositions.get(position)).second)
;
}

```

```

@Override
public int getItemCount() {
 return mSize;
}

class CharacterViewHolder extends RecyclerView.ViewHolder {

 @BindView(R.id.text_content) TextView mCharacter;
 @BindView(R.id.text_info) TextView mName;

 CharacterViewHolder(View itemView) {
 super(itemView);
 ButterKnife.bind(this, itemView);
 itemView.setOnClickListener(v ->
choose(getAdapterPosition()));
 }
}

protected void choose(String c) {
 final Intent result = new Intent();
 result.putExtra(getString(R.string.intent_character), c);
 setResult(RESULT_OK, result);
 finish();
}
}

```

The layout for this Activity consists of an `EditText` for search input, and a `RecyclerView` in which to display the searchable content.

The viewholder layout used for displaying each character consists of two `TextViews`, to display the character and its name.

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:orientation="vertical"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_margin="8dp"
 android:background="?attr/selectableItemBackgroundBorderless">

 <TextView
 android:id="@+id/text_content"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:gravity="center_horizontal"

 android:textAppearance="@android:style/TextAppearance.Material.Title"/>

 <TextView
 android:id="@+id/text_info"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:gravity="center_horizontal"/>

</LinearLayout>

```

In `onCreate`, the layout is inflated, and the text is set on the title and search views. The RecyclerView is then setup with a `GridLayoutManager` to display three viewholders per row, as each viewholder is quite small.

After the adapter is created, a `SimpleTextChangeWatcher` is applied to the search `EditText` to capture input as the user types, and pass it to the adapter for filtering.

The characters are then loaded for the adapter.

The full range of characters defined in the `Character` class is from 0 to 1114111. It is not reasonable to load all of these characters at once, especially not on the UI thread.

Instead, they must be loaded in chunks from another thread.

Within the `AsyncTask` a new `ArrayList` of pairs of strings is created.

The total length is set as `MAX_CODE_POINT - MIN_CODE_POINT`, and `lastIndex` is set as 0, representing the index of the last block added to the adapter.

For each character in the range, if the character is defined and not a control character, it is added to the `ArrayList` along with its name.

If the current size of the `ArrayList` minus the last added index is greater than one  $250^{\text{th}}$  of the entire length, the characters are added to the adapter, and the `lastIndex`

is reset.

This results in the valid characters being chunked into around 56 blocks.

### **CharacterAdapter**

The `CharacterAdapter` holds three `ArrayLists`. The first is `mCharacters`, which holds the pairs of strings which are to be displayed.

The other two `ArrayLists` are lists of integers which point to the positions in the first list.

`mFilteredPositions` is the `ArrayList` which the adapter uses to determine its length and to find the strings which it should be binding to to the viewholders.

`mWorkingPositions` is a separate `ArrayList` used during the filtering of positions on another thread. `mFilteredPositions` cannot be updated from another thread because the

`RecyclerView` expects the data-set not to change without it being notified, and the `RecyclerView` cannot be notified as each character is filtered.

When the `Activity` starts the `AsyncTask` begins to add blocks of characters to the adapter.

When `addCharacters` is called, each of the characters from the start position onwards are added to `mCharacters`.

Next, the original length of `mFilteredPositions` is saved, and each of the integer values from this point to the size of `mCharacters` is added to `mFilteredPositions`. Finally, a new runnable is posted to the UI thread to change the `mSize` variable and call `notifyItemRangeInserted` to notify the `RecyclerView` that a new range has been

inserted.

In this case there is no need for a working `ArrayList` because items are being added, not removed, and as such `mSize` will always be smaller than the length of `mFilteredPositions`, meaning that the `RecyclerView` will never request a position past this point.

When the user types something into the `EditText`, `filter` is called with the query that they typed.

This executes an `AsyncTask` to search the characters for their query.

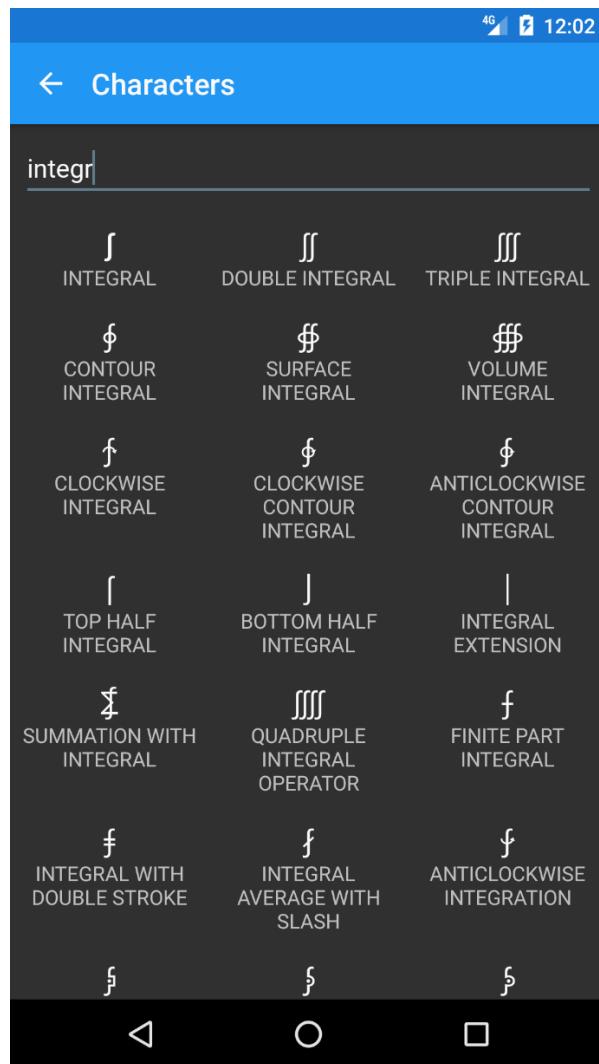
First, `mWorkingPositions` is re-created.

Next, there are three possibilities for the search:

1. The query is empty, in which case all of the positions are added to `mWorkingPositions`
2. The query starts with the last query, meaning that the user has typed another character. In this case the method iterates through the currently filtered positions and only searches the characters at these positions in `mCharacters` for the new query, as if the other characters did not contain the shorter query, they will not contain this one.
3. Otherwise, the entire `mCharacters` list is searched, and the matching positions are added to `mWorkingPositions`.

Once `mWorkingPositions` has been built, the last query is updated, and a new runnable is posted to the UI thread to update the filtered positions.

This swaps `mFilteredPositions`, updates the size, and notifies that the dataset has been changed.



### Returning the chosen character

When the user clicks on a character, `choose` is called in the `CharacterAdapter`, which calls the `choose` method in `CharacterActivity` with the character string. This creates an `Intent` and adds the string as an extra, before setting the result code to `RESULT_OK` and the data to the `Intent`, and finishing the `Activity`. Returning to the `onActivityResult` method of `EditorActivity`, the `resultCode` will be `CharacterActivity.REQUEST_CODE_CHARACTER_INSERTED`. `insertString` is then called with the character string.

### Emoji insertion

The `EmojiActivity` is very similar to the `CharacterActivity` except that it does not have to work on another thread as there are far fewer emojis. The Unicode 9.0 specification includes 1126 emoji characters which are easily searchable without impacting UI performance.

The `EmojiActivity` does however store the last 9 emojis that the user used.

### `EmojiActivity.java`

```
package com(tpb.projects.editors;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v7.widget.GridLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.TextView;

import com(tpb.mdtext.TextUtils;
import com(tpb.mdtext.emoji.Emoji;
import com(tpb.mdtext.emoji.EmojiLoader;
import com(tpb.projects.R;
import com(tpb.projects.common BaseActivity;
import com(tpb.projects.util.SettingsActivity;
import com(tpb.projects.util.input.SimpleTextChangeWatcher;

import java.util.ArrayList;

import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by theo on 23/03/17.
 */

public class EmojiActivity extends BaseActivity {

 public static final int REQUEST_CODE_CHOOSE_EMOJI = 666;
 private static final String PREFS_COMMON_EMOJIS = "COMMON_EMOJIS";

 private SharedPreferences mCommonEmojis;
 @BindView(R.id.search_title) TextView mTitle;
 @BindView(R.id.search_recycler) RecyclerView mRecycler;
 @BindView(R.id.search_search_box) EditText mSearch;

 @Override
 protected void onCreate(@Nullable Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 final SettingsActivity.Preferences prefs =
SettingsActivity.Preferences
 .getPreferences(this);
 setTheme(prefs.isDarkThemeEnabled() ? R.style.AppTheme_Dark :
R.style.AppTheme);
 setContentView(R.layout.activity_simple_search);
 ButterKnife.bind(this);
 mTitle.setText(R.string.title_activity_emoji);
 mSearch.setHint(R.string.hint_search_characters);
 mCommonEmojis = getSharedPreferences(PREFS_COMMON_EMOJIS,
MODE_PRIVATE);
 mRecycler.setLayoutManager(new GridLayoutManager(this, 3));
 final EmojiAdapter adapter = new EmojiAdapter(mCommonEmojis);
 mRecycler.setAdapter(adapter);
 mSearch.addTextChangedListener(new SimpleTextChangeWatcher() {
```

```

 @Override
 public voidTextChanged() {

 adapter.filter(mSearch.getText().toString().toLowerCase().replace(" ", "_"));
 }
}

class EmojiAdapter extends
RecyclerView.Adapter<EmojiAdapter.EmojiViewHolder> {

 private ArrayList<Emoji> mEmojis = new ArrayList<>();
 private ArrayList<Emoji> mFilteredEmojis = new ArrayList<>();

 EmojiAdapter(SharedPreferences prefs) {
 mEmojis.addAll(EmojiLoader.getAllEmoji());
 if(prefs.getString("common", null) != null) {
 final String[] common = prefs.getString("common",
 "").split(",");
 for(String s : common) {
 final Emoji e = EmojiLoader.getEmojiForAlias(s);
 if(e != null) {
 mEmojis.remove(e);
 mEmojis.add(0, e);
 }
 }
 }
 mFilteredEmojis.addAll(mEmojis);
 }

 void filter(String query) {
 mFilteredEmojis.clear();
 if(query.isEmpty()) {
 mFilteredEmojis.addAll(mEmojis);
 } else {
 for(Emoji e : mEmojis) {
 boolean added = false;
 for(String s : e.getAliases()) {
 if(s.contains(query)) {
 mFilteredEmojis.add(e);
 added = true;
 break;
 }
 }
 if(!added) {
 for(String s : e.getTags()) {
 if(s.contains(query)) {
 mFilteredEmojis.add(e);
 break;
 }
 }
 }
 }
 }
 notifyDataSetChanged();
 }

 private void choose(int pos) {
}

```

```
EmojiActivity.this.choose(mFilteredEmojis.get(pos).getAliases().get(0));
 }

 @Override
 public EmojiViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
 return new
EmojiViewHolder(LayoutInflater.from(parent.getContext()))

.inflate(R.layout.viewholder_text, parent,
 false
);
 }

 @Override
 public void onBindViewHolder(EmojiViewHolder holder, int position) {
 holder.mEmoji.setText(mFilteredEmojis.get(position).getUnicode());
 holder mName.setText(
 String.format(":%1$s:",
mFilteredEmojis.get(position).getAliases().get(0)));
 }

 @Override
 public int getItemCount() {
 return mFilteredEmojis.size();
 }

 class EmojiViewHolder extends RecyclerView.ViewHolder {

 @BindView(R.id.text_content) TextView mEmoji;
 @BindView(R.id.text_info) TextView mName;

 EmojiViewHolder(View itemView) {
 super(itemView);
 ButterKnife.bind(this, itemView);
 itemView.setOnClickListener(v ->
choose(getAdapterPosition()));
 }
 }
}

private void choose(String alias) {
 String common = "";
 if(mCommonEmojis.getString("common", null) != null) {
 String current = mCommonEmojis.getString("common", "");
 if(current.contains(alias)) {
 final int index = current.indexOf(alias);
 current = current.substring(0, index) +
current.substring(index + alias.length());
 }
 if(TextUtils.instancesOf(current, ",") > 8) {
 current = current.substring(current.indexOf(',') + 1);
 }
 common = current;
 }
 common += "," + alias;
 mCommonEmojis.edit().putString("common", common).apply();
}
```

```
 final Intent result = new Intent();
 result.putExtra(getString(R.string.intent_emoji), alias);
 setResult(RESULT_OK, result);
 finish();
}

}
```

In `onCreate` the same layout as `CharacterActivity` is inflated, and the title and search hint are set to the appropriate string resources.

The `SharedPreferences` used for storing common emojis is then loaded, and the `RecyclerView` is set up with a `GridLayoutManager`.

The `EmojiAdapter` is created, and set on the `RecyclerView`, and finally a `SimpleTextChangeWatcher` is added to the search `EditText`.

The `SimpleTextChangeWatcher` replaces spaces in the search string with underscores before passing it to the adapter, as multi-word emoji names are separate by underscores rather than spaces.

When the `EmojiAdapter` is created, it adds all of the `Emoji` from `EmojiLoader` to `mEmojis`.

It then checks if anything is stored under the "common" key in the `EmojiActivity` shared preferences.

If the `EmojiActivity` has been used before, the value under the "common" key will be a comma delimited list of emoji aliases, which are split and used to insert each of the common

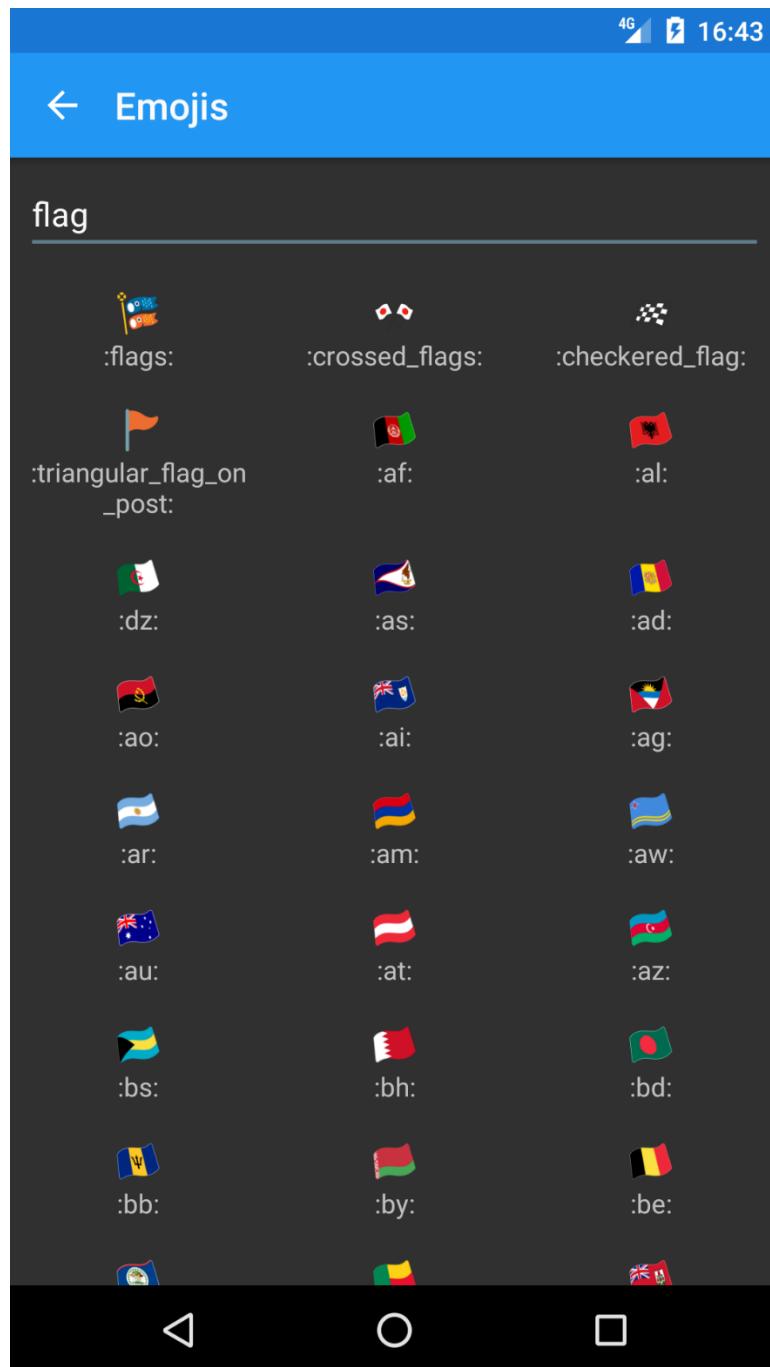
emojis at the start of the array.

When the user enters a query, the currently filtered `Emojis` are cleared.

If the query is empty, all of the `Emojis` are added to `mFilteredEmojis`.

Otherwise, each `Emoji` is checked. If one of the aliases matches, the `Emoji` is added, if not the tags are checked for matches, and if one is found the `Emoji` is added.

`notifyDataSetChanged` is then called.



When the user chooses an Emoji, it is added to the SharedPreferences before being returned.

The common string is declared, and if mCommonEmojis contains the “common” key, the current value is loaded. If the emoji alias is already contained in the string, it is removed. Next, if the string contains more than 8 commas, 9 elements, the first item is removed. Finally, the common string is set to the current string.

Next, the new emoji alias is added to common, and the common string is written to SharedPreferences.

Finally, the result Intent is created, the emoji alias is added as an extra, the result is set, and the EmojiActivity finishes.

## Implementations of EditorActivity

The EditorActivity is used across multiple objectives.

[3.e.vi](#), 3.v.vii, 4.b.ii, 4.b.iii, 4.c, 5.c.ii, 5.c.iii, 6.e, 6.f, 6.g, and 6.h.

The EditorActivity is used for editing cards, comments, issues, and projects.

### CardEditor

The CardEditor deals with the creation and editing of cards, including creating cards from issues.

It inflates the ViewStub with a layout containing a MarkdownEditText, a button used when showing the list of available issues for card creation, and a button for clearing any issue preview information shown.

#### stub\_card\_editor.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 android:orientation="vertical"
 android:layout_width="match_parent"
 android:layout_height="match_parent">

 <android.support.design.widget.TextInputLayout
 android:id="@+id/card_note_wrapper"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_weight="1"
 android:layout_margin="8dp"
 android:hint="@string/hint_card_new"
 app:counterEnabled="true"
 app:counterMaxLength="250">

 <com(tpb).mdtext.views.MarkdownEditText
 android:id="@+id/card_note_edit"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:inputType="textMultiLine|textCapSentences"
 android:maxLength="250"
 android:imeOptions="actionNone"
 android:scrollHorizontally="false"
 android:gravity="top"/>

 </android.support.design.widget.TextInputLayout>

 <Button
 android:id="@+id/card_clear_issue_button"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_gravity="bottom">
```

```

 android:layout_weight="0"
 android:text="@string/text_clear"
 android:background="?android:attr/selectableItemBackground"
 style="@style/Widget.AppCompat.Button.Borderless"
 android:visibility="gone"/>>

 <Button
 android:id="@+id/card_from_issue_button"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_gravity="bottom"
 android:layout_weight="0"
 android:text="@string/hint_from_issue"
 android:background="?android:attr/selectableItemBackground"
 style="@style/Widget.AppCompat.Button.Borderless"
 android:visibility="gone"/>>

</LinearLayout>

```

The MarkdownEditText is wrapped in a TextInputLayout allowing the character limit to be shown above the EditText when the EditText is in use.

## CardEditor.java

```

package com(tpb.projects.editors;

import android.app.Dialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.design.widget.TextInputLayout;
import android.support.v7.app.AlertDialog;
import android.text.InputFilter;
import android.view.View;
import android.view.ViewStub;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.Toast;

import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Loader;
import com(tpb.github.data.models.Card;
import com(tpb.github.data.models.Issue;
import com(tpb.mdtext.Markdown;
import com(tpb.mdtext.imagegetter.HttpImageGetter;
import com(tpb.mdtext.views.MarkdownEditText;
import com(tpb.projects.R;
import com(tpb.projects.markdown.Formatter;
import com(tpb.projects.util.SettingsActivity;
import com(tpb.projects.util.Util;
import com(tpb.projects.util.input.KeyBoardVisibilityChecker;
import com(tpb.projects.util.input.SimpleTextChangeWatcher;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

```

```
import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;

/**
 * Created by theo on 13/02/17.
 */

public class CardEditor extends EditorActivity {
 public static final int REQUEST_CODE_NEW_CARD = 1606;
 public static final int REQUEST_CODE_EDIT_CARD = 7180;

 @BindView(R.id.card_note_edit) MarkdownEditText mEditor;
 @BindView(R.id.card_from_issue_button) Button mIssueButton;
 @BindView(R.id.card_clear_issue_button) Button mClearButton;
 @BindView(R.id.markdown_edit_buttons) LinearLayout mEditButtons;
 @BindView(R.id.markdown_editor_discard) Button mDiscardButton;
 @BindView(R.id.markdown_editor_done) Button mDoneButton;
 @BindView(R.id.card_note_wrapper) TextInputLayout mEditorWrapper;
 private KeyBoardVisibilityChecker mKeyBoardChecker;

 private Card mCard;

 private boolean mHasBeenEdited = false;

 @Override
 protected void onCreate(@Nullable Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 final SettingsActivity.Preferences prefs =
SettingsActivity.Preferences
 .getPreferences(this);
 setTheme(
 prefs.isDarkThemeEnabled() ? R.style.AppTheme_Transparent_Dark
: R.style.AppTheme_Transparent);
 setContentView(R.layout.activity_markdown_editor);

 final ViewStub stub = (ViewStub) findViewById(R.id.editor_stub);

 stub.setLayoutResource(R.layout.stub_card_editor);
 stub.inflate();
 ButterKnife.bind(this);

 final Intent launchIntent = getIntent();

 if(launchIntent.hasExtra(getString(R.string.parcel_card))) { //We are
editing a card
 mCard =
launchIntent.getParcelableExtra(getString(R.string.parcel_card));
 mEditor.setText(mCard.getNote());
 } else {
 mCard = new Card();
 addFromIssueButtonListeners(launchIntent);
 }

 new MarkdownButtonAdapter(this, mEditButtons,
 new MarkdownButtonAdapter.MarkdownButtonListener() {
 @Override
 public void snippetEntered(String snippet, int
relativePosition) {
```

```
 if(mEditor.hasFocus() && mEditor.isEnabled() &&
mEditor.isEditing()) {
 Util.insertString(mEditor, snippet,
relativePosition);
 }
 }

@Override
public String getText() {
 return mEditor.getInputText().toString();
}

@Override
public void previewCalled() {
 if(mEditor.isEditing()) {
 mEditor.saveText();
 mEditor.disableEditing();
 mEditor.setMarkdown(
Markdown.formatMD(mEditor.getInputText().toString()),
new HttpImageGetter(mEditor)
);
 } else {
 mEditor.restoreText();
 mEditor.enableEditing();
 }
}
);

final View content = findViewById(android.R.id.content);
content.setVisibility(View.VISIBLE);

mKeyBoardChecker = new KeyBoardVisibilityChecker(content,
new KeyBoardVisibilityChecker.KeyBoardVisibilityListener() {
 @Override
 public void keyboardShown() {
 mIssueButton.setVisibility(View.GONE);
 }

 @Override
 public void keyboardHidden() {
 if(mIssueButton.hasOnClickListeners()) {
 mIssueButton.postDelayed(() ->
mIssueButton.setVisibility(View.VISIBLE),
100
);
 }
 }
);
};

mEditor.addTextChangedListener(new SimpleTextChangeWatcher() {
 @Override
 public void textChanged() {
 mHasBeenEdited |= mEditor.isEditing();
 }
});
}
```

```
private void bindIssue(Issue issue) {
 mEditor.setMarkdown(Formatter.buildIssueSpan(this, issue, true, true,
true, true, false)
 .toString(),
 new HttpImageGetter(mEditor)
);
}

private void addFromIssueButtonListeners(Intent launchIntent) {
 final String fullRepoName =
launchIntent.getStringExtra(getString(R.string.intent_repo));
 final ArrayList<Integer> invalidIds = launchIntent

.getIntegerArrayListExtra(getString(R.string.intent_int_arraylist));

 mIssueButton.setVisibility(View.VISIBLE);
 mIssueButton.setOnClickListener(v -> {
 final ProgressDialog pd = new ProgressDialog(CardEditor.this);
 pd.setTitle(R.string.text_loading_issues);
 pd.setCancelable(false);
 pd.show();
 Loader.getLoader(CardEditor.this).loadOpenIssues(new
Loader.ListLoader<Issue>() {
 private int selectedIssuePosition = 0;

 @Override
 public void listLoadComplete(List<Issue> loadedIssues) {
 if(isClosing()) return; // There is no window to attach to
 pd.dismiss();

 //We check which Issues are not already attached to a card
 final ArrayList<Issue> validIssues = new ArrayList<>();
 for(Issue i : loadedIssues) {
 if(invalidIds.indexOf(i.getId()) == -1)
validIssues.add(i);
 }
 if(validIssues.isEmpty()) {
 Toast.makeText(CardEditor.this,
R.string.error_no_issues,
 Toast.LENGTH_SHORT
).show();
 return;
 }
 final String[] issues = new String[validIssues.size()];
 for(int i = 0; i < validIssues.size(); i++) {
 issues[i] =
String.format(getString(R.string.text_issue_single_line),
 validIssues.get(i).getNumber(),
validIssues.get(i).getTitle()
);
 }

 final AlertDialog.Builder scBuilder = new
AlertDialog.Builder(CardEditor.this);
 scBuilder.setTitle(R.string.title_choose_issue);
 scBuilder.setSingleChoiceItems(issues, 0,

```

```

 (dialogInterface, i) -> selectedIssuePosition = i
);
 scBuilder.setPositiveButton(R.string.action_ok,
((dialogInterface, i) -> {

mCard.setFromIssue(validIssues.get(selectedIssuePosition));
 mEditor.setFilters(new InputFilter[] {}); //Remove the
length filter
 mEditorWrapper.setCounterEnabled(false); //Remove the
counter
 bindIssue(mCard.getIssue());
 mEditor.setFocusable(false);
 mClearButton.setVisibility(View.VISIBLE); //Enable
clearing
});
 scBuilder.setNegativeButton(R.string.action_cancel,
 (dialogInterface, i) -> dialogInterface.dismiss()
);
 scBuilder.create().show();
}

@Override
public void listLoadError(APIHandler.APIError error) {
 if(isClosing()) return;
 pd.dismiss();
 Toast.makeText(CardEditor.this, error.resId,
Toast.LENGTH_SHORT).show();
}
}, fullRepoName);
});

mClearButton.setOnClickListener((v) -> {
 mEditor.setText(null);
 mEditor.setFilters(
 new InputFilter[] {new InputFilter.LengthFilter(250)});
//Re-enable filter
 mEditorWrapper.setCounterEnabled(true);
 mCard = new Card();
 mClearButton.setVisibility(View.GONE);
});
}

@Override
void imageLoadComplete(String url) {
 Util.insertString(mEditor, url);
}

@Override
void imageLoadException(IOException ioe) {
 //TODO Explain error
}

@OnClick(R.id.markdown_editor_done)
void onDone() {
 final Intent done = new Intent();
 mCard.setNote(mEditor.getInputText().toString());
 done.putExtra(getString(R.string.parcel_card), mCard);
 setResult(RESULT_OK, done);
 mHasBeenEdited = false;
}

```

```

 finish();
 }

 @OnClick(R.id.markdown_editor_discard)
 void onDiscard() {
 onBackPressed();
 }

 @Override
 protected void emojiChosen(String emoji) {
 Util.insertString(mEditor, String.format(":%1$s:", emoji));
 }

 @Override
 protected void insertString(String c) {
 Util.insertString(mEditor, c);
 }

 @Override
 public void finish() {
 if(mHasBeenEdited && !mEditor.getText().toString().isEmpty()) {
 final AlertDialog.Builder builder = new AlertDialog.Builder(this);
 builder.setTitle(R.string.title_discard_changes);
 builder.setPositiveButton(R.string.action_yes, (dialogInterface,
i) -> {
 final InputMethodManager imm = (InputMethodManager)
getSystemService(
 Context.INPUT_METHOD_SERVICE);

imm.hideSoftInputFromWindow(findViewById(android.R.id.content).getWindowToken(),
0);
 mDoneButton.postDelayed(super::finish, 150);
 });
 builder.setNegativeButton(R.string.action_no, null);
 final Dialog deleteDialog = builder.create();
 deleteDialog.getWindow().getAttributes().windowAnimations =
R.style.DialogAnimation;
 deleteDialog.show();
 } else {
 if(mKeyboardChecker.isKeyboardOpen()) {
 final InputMethodManager imm = (InputMethodManager)
getSystemService(
 Context.INPUT_METHOD_SERVICE);

imm.hideSoftInputFromWindow(findViewById(android.R.id.content).getWindowToken(),
0);
 mDoneButton.postDelayed(super::finish, 150);
 } else {
 super.finish();
 }
 }
 }
}

```

When the CardEditor onCreate method is called, the markdown\_editor layout is inflated, the ViewStub is found and inflated with the card\_editor stub layout, and the Views are found.

If the launch Intent contains a Card, a Card is being edited, so the Card is stored, and the EditText text is set to the contents of the Card note.

Otherwise, a new Card is created and addFromIssueButtonListeners is called with the launch Intent.

This collects the full repository name from the Intent, as well as a list of issue ids which have already been used in cards, and therefore cannot be used again.

The issue button is made visible add its OnClickListener is set.

When the button is clicked:

1. A new ProgressDialog is created is created with a title telling the user that it is loading the Issues.
2. When the Issues are loaded, the first check is that the Activity is not closing, in which case the listener returns. Next, the method iterates through all of the returned Issues, adding them to a new list if their id is not in the list of invalid ids.
3. If the list of valid Issues is empty, a toast error message is displayed, telling the user that there are no valid Issues. The method then returns.
4. Otherwise, a string array of the Issue titles is created, and a single choice dialog is created in which to display them.
5. When an Issue is selected
  1. setFromIssue is called on the Card instance, setting its note and Issue.
  2. The InputFilters on the EditText are removed.
  3. The character counter on the EditText is disabled.
  4. The Issue is bound:
    1. The span is built with buildIssueSpan with flags to format the title as a header, insert the numbered link to the Issue, show the assignees, show the time that the issue was closed, and not show the comment count.
  5. The MarkdownEditText is made non-focusable so that it cannot be clicked.
  6. The clear issue button is made visible so that the issue being previewed can be removed.

The OnClickListener is then set on the clear button.

This clears the text in the MarkdownEditText, re-enables the 250 character limit, re-enables the counter, re-creates the Card, and hides the clear issue button.

If the Issues do not load successfully, the dialog is dismissed, and a toast message is shown with the error resource for the APIError.

Returning to the onCreate method, an anonymous MarkdownButtonAdapter is created with the CardEditor Activity, the mEditButtons LinearLayout, and an anonymous MarkdownButtonListener.

The MarkdownButtonListener implements, snippetEntered, which

uses `Util.insertString` to insert the snippet at a relative position.  
`getText` returns the `MarkdownEditText` `Editable` as a string.  
`previewCalled` checks if the `MarkdownEditText` is currently in the editing state. If it is it:

1. Calls `saveText` on the `MarkdownEditText` to save the raw markdown
  2. Calls `disableEditing` on the `MarkdownEditText` to disable any input
  3. Sets the formatted markdown with a new `HttpImageGetter` with the `MarkdownEditText` as its container.
- Otherwise it:
4. Calls `restoreText` to set the text back to the value saved in `saveText`.
  5. Calls `enableEditing` to re-enable input in the `MarkdownEditText`.

An instance of `KeyBoardVisibilityChecker` is created and assigned to `mKeyBoardChecker` with the root content layout, and an anonymous `KeyBoardVisibilityChecker` which hides the issue button when the keyboard is shown, and shows it again after the keyboard is hidden, if it has a listener.

Finally, a `SimpleTextChangeWatcher` is added to the `MarkdownEditText` which ORs `mHasBeenEdited` with the `MarkdownEditText` editing state.

The `CardEditor` implements character and emoji insertion using the `Util.insertString` method.

When `onDone` is called, a new `Intent` is created, the `Card` note is set, and the `Card` is added to the `Intent` which is then set as the result.

The `mHasBeenEdited` flag is set to false, indicating that the content has not been updated since the last time that it was saved, and `finish` is called.

`finish` checks if `mHasBeenEdited` is true, and the `EditText` is non-empty. If so, it displays a dialog asking the user to confirm that they wish to dismiss their changes.

If `mHasBeenEdited` is false, or the user chooses to dismiss their changes, the keyboard is dismissed and `super.finish` is called to close the `Activity`.

## CommentEditor

The `CommentEditor` is very similar to the `CardEditor` and deals with editing comments for issues and commits.

### CommentEditor.java

```
package com(tpb.projects.editors;

import android.app.Dialog;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v7.app.AlertDialog;
```

```
import android.view.ViewStub;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.LinearLayout;

import com(tpb.github.data.models.Comment;
import com(tpb.github.data.models.Issue;
import com(tpb.mdtext.Markdown;
import com(tpb.mdtext.imagegetter.HttpImageGetter;
import com(tpb.mdtext.views.MarkdownEditText;
import com(tpb.projects.R;
import com(tpb.projects.util.SettingsActivity;
import com(tpb.projects.util.Util;
import com(tpb.projects.util.input.KeyBoardVisibilityChecker;
import com(tpb.projects.util.input.SimpleTextChangeWatcher;

import java.io.IOException;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;

/**
 * Created by theo on 14/02/17.
 */

public class CommentEditor extends EditorActivity {

 public static final int REQUEST_CODE_NEW_COMMENT = 1799;
 public static final int REQUEST_CODE_EDIT_COMMENT = 5734;
 public static final int REQUEST_CODE_COMMENT_FOR_STATE = 1400;

 @BindView(R.id.comment_body_edit) MarkdownEditText mEditor;
 @BindView(R.id.markdown_edit_buttons) LinearLayout mEditButtons;
 @BindView(R.id.markdown_editor_discard) Button mDiscardButton;
 @BindView(R.id.markdown_editor_done) Button mDoneButton;
 private KeyBoardVisibilityChecker mKeyBoardChecker;

 private boolean mHasBeenEdited;

 private Comment mComment;
 private Issue mIssue;

 @Override
 protected void onCreate(@Nullable Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 final SettingsActivity.Preferences prefs =
SettingsActivity.Preferences
 .getPreferences(this);
 setTheme(prefs.isDarkThemeEnabled() ? R.style.AppTheme_Dark :
R.style.AppTheme);
 setContentView(R.layout.activity_markdown_editor);

 final ViewStub stub = (ViewStub) findViewById(R.id.editor_stub);
 stub.setLayoutResource(R.layout.stub_comment_editor);
 stub.inflate();

 //Bind after inflating the stub
 }
}
```

```
 ButterKnife.bind(this);

 final Intent launchIntent = getIntent();

 if(launchIntent.getStringExtra(getString(R.string.parcel_comment))) {
 mComment =
 launchIntent.getParcelableExtra(getString(R.string.parcel_comment));
 mEditor.setText(mComment.getBody());
 }
 if(launchIntent.getStringExtra(getString(R.string.parcel_issue))) {
 mIssue =
 launchIntent.getParcelableExtra(getString(R.string.parcel_issue));
 }
 mEditor.addTextChangedListener(new SimpleTextChangeWatcher() {
 @Override
 public voidTextChanged() {
 mHasBeenEdited |= mEditor.isEditing();
 }
 });

 new MarkdownButtonAdapter(this, mEditButtons,
 new MarkdownButtonAdapter.MarkdownButtonListener() {
 @Override
 public voidsnippetEntered(String snippet, int
relativePosition) {
 if(mEditor.hasFocus() && mEditor.isEnabled() &&
mEditor.isEditing()) {
 Util.insertString(mEditor, snippet,
relativePosition);
 }
 }

 @Override
 public StringgetText() {
 return mEditor.getInputText().toString();
 }

 @Override
 public voidpreviewCalled() {
 if(mEditor.isEditing()) {
 mEditor.saveText();
 final String repo = mIssue == null ? null :
mIssue.getRepoFullName();
 mEditor.disableEditing();
 mEditor.setMarkdown(
Markdown.formatMD(mEditor.getInputText().toString(), repo),
 new HttpImageGetter(mEditor)
);
 } else {
 mEditor.restoreText();
 mEditor.enableEditing();
 }
 }
 });
);
 mKeyBoardChecker = new
KeyboardVisibilityChecker(findViewById(android.R.id.content));
}
```

```
}

@Override
protected void emojiChosen(String emoji) {
 Util.insertString(mEditor, String.format(":%1$s:", emoji));
}

@Override
protected void insertString(String c) {
 Util.insertString(mEditor, c);
}

@OnClick(R.id.markdown_editor_done)
void onDone() {
 final Intent done = new Intent();
 if(mComment == null) mComment = new Comment();
 mComment.setBody(mEditor.getInputText().toString());
 done.putExtra(getString(R.string.parcel_comment), mComment);
 if(mIssue != null) done.putExtra(getString(R.string.parcel_issue),
mIssue);
 setResult(RESULT_OK, done);
 mHasBeenEdited = false;
 finish();
}

@OnClick(R.id.markdown_editor_discard)
void onDiscard() {
 onBackPressed();
}

@Override
void imageLoadComplete(String url) {
 Util.insertString(mEditor, url);
}

@Override
void imageLoadException(IOException ioe) {

}

@Override
public void finish() {
 if(mHasBeenEdited && !mEditor.getText().toString().isEmpty()) {
 final AlertDialog.Builder builder = new AlertDialog.Builder(this);
 builder.setTitle(R.string.title_discard_changes);
 builder.setPositiveButton(R.string.action_yes, (dialogInterface,
i) -> {
 final InputMethodManager imm = (InputMethodManager)
getSystemService(
 Context.INPUT_METHOD_SERVICE);

 imm.hideSoftInputFromWindow(findViewById(android.R.id.content).getWindowToken(),
 0);
 mDoneButton.postDelayed(super::finish, 150);
 });
 builder.setNegativeButton(R.string.action_no, null);
 final Dialog deleteDialog = builder.create();
 deleteDialog.getWindow().getAttributes().windowAnimations =
R.style.DialogAnimation;
 }
}
```

```

 deleteDialog.show();
 } else {
 if(mKeyboardChecker.isKeyboardOpen()) {
 final InputMethodManager imm = (InputMethodManager)
getSystemService(
 Context.INPUT_METHOD_SERVICE);

imm.hideSoftInputFromWindow(findViewById(android.R.id.content).getWindowToken(),
), 0);
 mDoneButton.postDelayed(super::finish, 150);
 } else {
 super.finish();
 }
 }
}
}

```

The key differences are in the `onDone` and `previewCalled` methods.  
`onDone` passes the `Issue` with the `Intent` if it is non-null.  
`previewCalled` extracts the repository URL from the `Issue` if it exists, and uses it when formatting the markdown.

## IssueEditor

The `IssueEditor` is more complex as it also deals with setting the labels and collaborators on an issue, as well as creating issues from cards.

### IssueEditor.java

```

package com(tpb.projects.editors;

import android.app.Dialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.util.Pair;
import android.support.v7.app.AlertDialog;
import android.view.View;
import android.view.ViewStub;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.Toast;

import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Loader;
import com(tpb.github.data.models.Card;
import com(tpb.github.data.models.Issue;
import com(tpb.github.data.models.Label;
import com(tpb.github.data.models.User;
import com(tpb.mdtext.Markdown;
import com(tpb.mdtext.imagegetter.HttpImageGetter;
import com(tpb.mdtext.views.MarkdownEditText;

```

```
import com(tpb).mdtext.views.MarkdownTextView;
import com(tpb).projects.R;
import com(tpb).projects.markdown.Formatter;
import com(tpb).projects.util.SettingsActivity;
import com(tpb).projects.util.Util;
import com(tpb).projects.util.input.KeyBoardVisibilityChecker;
import com(tpb).projects.util.input.SimpleTextChangeWatcher;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;

/**
 * Created by theo on 07/02/17.
 */

public class IssueEditor extends EditorActivity {
 private static final String TAG = IssueEditor.class.getSimpleName();

 public static final int REQUEST_CODE_NEW_ISSUE = 3025;
 public static final int REQUEST_CODE_EDIT_ISSUE = 1188;
 public static final int REQUEST_CODE_ISSUE_FROM_CARD = 9836;

 @BindView(R.id.issue_title_edit) EditText mTitleEdit;
 @BindView(R.id.issue_body_edit) MarkdownEditText mBodyEdit;
 @BindView(R.id.markdown_editor_discard) Button mDiscardButton;
 @BindView(R.id.markdown_editor_done) Button mDoneButton;
 @BindView(R.id.issue_labels_text) MarkdownTextView mLabelsText;
 @BindView(R.id.issue_assignees_text) MarkdownTextView mAssigneesText;
 @BindView(R.id.issue_information_layout) View mInfoLayout;
 @BindView(R.id.markdown_edit_buttons) LinearLayout mEditButtons;
 private KeyBoardVisibilityChecker mKeyBoardChecker;

 private final ArrayList<String> mAssignees = new ArrayList<>();
 private final ArrayList<String> mSelectedLabels = new ArrayList<>();

 private Card mLaunchCard;
 private Issue mLaunchIssue;

 private String mRepo;

 private boolean mHasBeenEdited = false;

 @Override
 protected void onCreate(@Nullable Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 final SettingsActivity.Preferences prefs =
SettingsActivity.Preferences
 .getPreferences(this);
 setTheme(prefs.isDarkThemeEnabled() ? R.style.AppTheme_Dark :
R.style.AppTheme);
 setContentView(R.layout.activity_markdown_editor);

 final ViewStub stub = (ViewStub) findViewById(R.id.editor_stub);
```

```
 stub.setLayoutResource(R.layout.stub_issue_editor);
 stub.inflate();
 ButterKnife.bind(this);

 final Intent launchIntent = getIntent();
 mRepo = launchIntent.getStringExtra(getString(R.string.intent_repo));
 if(launchIntent.hasExtra(getString(R.string.parcel_issue))) {
 mLaunchIssue =
 launchIntent.getParcelableExtra(getString(R.string.parcel_issue));
 mLaunchCard =
 launchIntent.getParcelableExtra(getString(R.string.parcel_card));
 mTitleEdit.setText(mLaunchIssue.getTitle());
 mBodyEdit.setText(mLaunchIssue.getBody());

 if(mLaunchIssue.getAssignees() != null) {
 for(User u : mLaunchIssue.getAssignees())
 mAssignees.add(u.getLogin());
 setAssigneesText();
 }

 if(mLaunchIssue.getLabels() != null &&
 mLaunchIssue.getLabels().length > 0) {
 final ArrayList<Pair<String, Integer>> labels = new
 ArrayList<>();
 for(Label l : mLaunchIssue.getLabels()) {
 labels.add(Pair.create(l.getName(), l.getColor()));
 }
 setLabelsText(labels);
 }
 }

 } else if(launchIntent.hasExtra(getString(R.string.parcel_card))) {
 mLaunchCard =
 launchIntent.getParcelableExtra(getString(R.string.parcel_card));

 final String note = mLaunchCard.getNote();
 int index = note.indexOf("\n");
 if(index == -1) index = 137;
 if(index < note.length()) {
 mTitleEdit.setText(note.substring(0, index) + "...");
 mBodyEdit.setText("..." + note.substring(index));
 } else {
 mTitleEdit.setText(note);
 }

 }
 final SimpleTextChangeWatcher editWatcher = new
 SimpleTextChangeWatcher() {
 @Override
 public void textChanged() {

 mHasBeenEdited |= mBodyEdit.isEditing();
 }
 };

 mTitleEdit.addTextChangedListener(editWatcher);
 mBodyEdit.addTextChangedListener(editWatcher);

 final View content = findViewById(android.R.id.content);
```

```
mKeyBoardChecker = new KeyboardVisibilityChecker(content,
 new KeyboardVisibilityChecker.KeyBoardVisibilityListener() {
 @Override
 public void keyboardShown() {
 mInfoLayout.setVisibility(View.GONE);
 }

 @Override
 public void keyboardHidden() {
 if(mBodyEdit.isEditing()) {
 mInfoLayout.postDelayed(() ->
mInfoLayout.setVisibility(View.VISIBLE),
 100
);
 }
 }
);
}

new MarkdownButtonAdapter(this, mEditButtons,
 new MarkdownButtonAdapter.MarkdownButtonListener() {
 @Override
 public void snippetEntered(String snippet, int
relativePosition) {
 mHasBeenEdited = true;
 Util.insertString(mBodyEdit, snippet,
relativePosition);
 }

 @Override
 public String getText() {
 return mBodyEdit.getInputText().toString();
 }

 @Override
 public void previewCalled() {
 if(mBodyEdit.isEditing()) {
 mBodyEdit.saveText();
 String repo = null;
 if(mLaunchIssue != null) repo =
mLaunchIssue.getRepoFullName();
 mBodyEdit.disableEditing();
 mTitleEdit.setEnabled(false);
 mBodyEdit.setMarkdown(
Markdown.formatMD(mBodyEdit.getInputText().toString(), repo),
 new HttpImageGetter(mBodyEdit)
);
 mInfoLayout.setVisibility(View.GONE);
 } else {
 mBodyEdit.restoreText();
 mBodyEdit.enableEditing();
 mTitleEdit.setEnabled(true);
 if(!mKeyBoardChecker.isKeyboardOpen()) {
 mInfoLayout.setVisibility(View.VISIBLE);
 }
 }
 }
 }
}
```

```

);
}

@OnClick(R.id.issue_add_assignees_button)
public void showAssigneesDialog() {
 final ProgressDialog pd = new ProgressDialog(this);
 pd.setTitle(R.string.text_loading_contributors);
 pd.setCancelable(false);
 pd.show();
 Loader.getLoader(this).loadContributors(new Loader.ListLoader<User>()
{
 @Override
 public void listLoadComplete(List<User> contributors) {
 final MultiChoiceDialog mcd = new MultiChoiceDialog();
 final Bundle b = new Bundle();
 b.putInt(getString(R.string.intent_title_res),
R.string.title_choose_assignees);
 mcd.setArguments(b);

 final String[] names = new String[contributors.size()];
 final boolean[] checked = new boolean[names.length];
 for(int i = 0; i < names.length; i++) {
 names[i] = contributors.get(i).getLogin();
 checked[i] = mAssignees.indexOf(names[i]) != -1;
 }
 mcd.setChoices(names, checked);
 mcd.setListener(new
MultiChoiceDialog.MultiChoiceDialogListener() {
 @Override
 public void choicesComplete(String[] choices, boolean[]
checked) {
 mAssignees.clear();
 for(int i = 0; i < choices.length; i++) {
 if(checked[i]) mAssignees.add(choices[i]);
 }
 setAssigneesText();
 mHasBeenEdited = true;
 }
 @Override
 public void choicesCancelled() {
 }
 });
 if(isClosing()) return; //Activity has been closed
 pd.dismiss();
 mcd.show(getSupportFragmentManager(), TAG);
 }

 @Override
 public void listLoadError(APIHandler.APIError error) {
 if(isClosing()) return;
 pd.dismiss();
 Toast.makeText(IssueEditor.this, error.resId,
Toast.LENGTH_SHORT).show();
 }
}, mRepo);
}

```

```
@OnClick(R.id.issue_add_labels_button)
public void showLabelsDialog() {
 final ProgressDialog pd = new ProgressDialog(this);
 pd.setTitle(R.string.text_loading_labels);
 pd.setCancelable(false);
 pd.show();
 Loader.getLoader(this).loadLabels(new Loader.ListLoader<Label>() {
 @Override
 public void listLoadComplete(List<Label> labels) {
 final MultiChoiceDialog mcd = new MultiChoiceDialog();

 final Bundle b = new Bundle();
 b.putInt(getString(R.string.intent_title_res),
R.string.title_choose_labels);
 mcd.setArguments(b);

 final String[] labelTexts = new String[labels.size()];
 final int[] colors = new int[labels.size()];
 final boolean[] choices = new boolean[labels.size()];
 for(int i = 0; i < labels.size(); i++) {
 labelTexts[i] = labels.get(i).getName();
 colors[i] = labels.get(i).getColor();
 choices[i] =
mSelectedLabels.indexOf(labels.get(i).getName()) != -1;
 }

 mcd.setChoices(labelTexts, choices);
 mcd.setBackgroundColors(colors);
 mcd.setListener(new
MultiChoiceDialog.MultiChoiceDialogListener() {
 @Override
 public void choicesComplete(String[] choices, boolean[]
checked) {
 mSelectedLabels.clear();
 final ArrayList<Pair<String, Integer>> labels = new
ArrayList<>();
 for(int i = 0; i < choices.length; i++) {
 if(checked[i]) {
 mSelectedLabels.add(choices[i]);
 labels.add(Pair.create(choices[i],
colors[i]));
 }
 }
 setLabelsText(labels);
 mHasBeenEdited = true;
 }
 @Override
 public void choicesCancelled() {
 }
 });
 if(isClosing()) return;
 pd.dismiss();
 mcd.show(getSupportFragmentManager(), TAG);
 }
 }
}
```

```
 public void listLoadError(APIHandler.APIError error) {
 if(isClosing()) return;
 pd.dismiss();
 Toast.makeText(IssueEditor.this, error.resId,
Toast.LENGTH_SHORT).show();
 }
 }, mRepo);
}

private void setAssigneesText() {
 if(!mAssignees.isEmpty()) {
 final StringBuilder builder = new StringBuilder();
 for(String a : mAssignees) {
 builder.append(
 String.format(getString(R.string.text_href),
"https://github.com/" + a, a));
 builder.append("
");
 }
 mAssigneesText.setVisibility(View.VISIBLE);
 mAssigneesText.setMarkdown(builder.toString());
 } else {
 mAssigneesText.setVisibility(View.GONE);
 }
}

private void setLabelsText(ArrayList<Pair<String, Integer>> labels) {
 mSelectedLabels.clear();

 if(!labels.isEmpty()) {
 final StringBuilder builder = new StringBuilder();
 builder.append("<ul bulleted=\"false\">");
 for(Pair<String, Integer> p : labels) {
 mSelectedLabels.add(p.first);
 builder.append("");
 builder.append(Formatter.getLabelString(p.first, p.second));
 builder.append("");
 }
 builder.append("");
 mLabelsText.setVisibility(View.VISIBLE);
 mLabelsText.setMarkdown(builder.toString());
 } else {
 mLabelsText.setVisibility(View.GONE);
 }
}

@Override
void imageLoadComplete(String url) {
 Util.insertString(mBodyEdit, url);
}

@Override
void imageLoadException(IOException ioe) {

}

@Override
protected void emojiChosen(String emoji) {
 Util.insertString(mBodyEdit, String.format(":%1$s:", emoji));
}
```

```
}

@Override
protected void insertString(String c) {
 Util.insertString(mBodyEdit, c);
}

@OnClick(R.id.markdown_editor_done)
void onDone() {
 final Intent done = new Intent();
 if(mLaunchIssue == null) {
 mLaunchIssue = new Issue();
 }
 mLaunchIssue.setTitle(mTitleEdit.getText().toString());
 mLaunchIssue.setBody(mBodyEdit.getInputText().toString());
 done.putExtra(getString(R.string.parcel_issue), mLaunchIssue);
 if(mLaunchCard != null) done.putExtra(getString(R.string.parcel_card),
mLaunchCard);
 if(!mSelectedLabels.isEmpty()) {
 done.putExtra(getString(R.string.intent_issue_labels),
 mSelectedLabels.toArray(new String[0]))
 };
}
if(!mAssignees.isEmpty()) {
 done.putExtra(getString(R.string.intent_issue_assignees),
 mAssignees.toArray(new String[0]))
};
}

setResult(RESULT_OK, done);
mHasBeenEdited = false;
finish();
}

@OnClick(R.id.markdown_editor_discard)
void onDiscard() {
 onBackPressed();
}

@Override
public void onBackPressed() {
 super.onBackPressed();
}

@Override
public void finish() {
 if(mHasBeenEdited && !mBodyEdit.getText().toString().isEmpty() &&
!mTitleEdit.getText()

.toString()

.isEmpty()) {
 final AlertDialog.Builder builder = new AlertDialog.Builder(this);
 builder.setTitle(R.string.title_discard_changes);
 builder.setPositiveButton(R.string.action_yes, (dialogInterface,
i) -> {
 final InputMethodManager imm = (InputMethodManager)
getSystemService(
 Context.INPUT_METHOD_SERVICE);

```

```
imm.hideSoftInputFromWindow(findViewById(android.R.id.content).getWindowToken(), 0);
 mDoneButton.postDelayed(super::finish, 150);
 });
 builder.setNegativeButton(R.string.action_no, null);
 final Dialog deleteDialog = builder.create();
 deleteDialog.getWindow().getAttributes().windowAnimations =
R.style.DialogAnimation;
 deleteDialog.show();
} else {
 if(mKeyboardChecker.isKeyboardOpen()) {
 final InputMethodManager imm = (InputMethodManager)
getSystemService(
 Context.INPUT_METHOD_SERVICE);

imm.hideSoftInputFromWindow(findViewById(android.R.id.content).getWindowToken(), 0);
 mDoneButton.postDelayed(super::finish, 150);
 } else {
 super.finish();
 }
}
}
```

After inflating the layout and `ViewStub` layout, there are more checks to perform than in the other editors.

## **onCreate**

Every time the `IssueEditor` is launched, a repository path is provided for loading the labels and collaborators.

If the `Intent` contains an `Issue` model:

1. The `Issue` is extracted to `mLaunchIssue`
2. The `Card` is extracted to `mLaunchCard`
3. The title and body `EditTexts` are set with the `Issue` title and body.
4. If the launch `Issue` has a list of assignees:
  1. The assignee logins are added to the `mAssignees ArrayList`.
  2. `setAssigneesText` is called. (To be explained)
5. If the launch `Issue` has a non-null and non-empty list of `Labels`:
  1. An `ArrayList` of `Pairs` of strings and integers is created from the `labels array`.
  2. `setLabelsText` is called (To be explained).

If the `Intent` contains a `card` model:

1. The `Card` is extracted to `mLaunchCard`
2. The `Card note` is stored.
3. The index of the first newline in the note is found.

4. If the index is -1, the index is set to 137
5. If the index is less than the length of the note, the note is split between the title and body and ellipsized.
6. Otherwise the note is set as the title.

A `SimpleTextChangeWatcher` is then added to both the title and body `EditTexts` which ORs `mHasBeenEdited` with the body editing state.

A `KeyBoardVisibilityChecker` is then used to hide the layout containing information about the issues' labels and assignees when the keyboard is shown, and to show it again once the keyboard has been hidden.

Finally, the `MarkdownButtonAdapter` is created, which also deals with the title `EditText` in this case, enabling or disabling it along with the body `MarkdownEditText` as well as hiding the layout for labels and assignees.

### Choosing assignees

When the assignees button is clicked, a new `ProgressDialog` is created and shown while the assignees are loaded.

A call is then made to load the collaborators for the current repository.

Once the collaborators are loaded a new `MultiChoiceDialog` is created, and each of the collaborators names are added to an array, with the respective checked flag set dependent on whether the collaborator name is already in the `mAssignees` list.

The dialog listener is then set to clear the current list of assignees and add the checked assignees before calling `setAssigneesText` to display the updated list. `setAssigneesText` checks if the assignees list is empty, if so it hides the `TextView`, otherwise it builds a list of links to each of the assignees GitHub user pages.

### Choosing labels

When the labels button is clicked, a new `ProgressDialog` is created and shown while the labels are loaded.

Arrays for the label texts, colours, and current choices are then created before being set on the `MultiChoiceDialog`.

The `MultiChoiceDialogListener` is then set to clear the current list of label names, and then add the checked names to the list as well as building an array of pairs of label names and their respective colours.

`setLabelsText` is then called to display the newly selected labels.

If the labels list is empty, the labels `TextView` is hidden, otherwise a `StringBuilder` is used to create a non-bulleted list of labels using `Formatter.getLabelString` to create each font tag with the correct text colour and background colour.

## onDone

The `onDone` method follows the save pattern as the other `onDone` methods, except that it also adds the assignees and labels to the array if they are not empty.

## ProjectEditor

The `ProjectEditor` is relatively simple, as it only deals with the project title and description.

When it is launched, the `Intent` is checked for a `Project` extra which is used to set the name and description, as well as storing the `Project` id for returning to the launching `Activity`.

### ProjectEditor.java

```
package com(tpb.projects.editors;

import android.app.Dialog;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v7.app.AlertDialog;
import android.view.View;
import android.view.ViewStub;
import android.view.inputmethod.InputMethodManager;
import android.widget.EditText;
import android.widget.LinearLayout;

import com(tpb.github.data.models.Project;
import com(tpb.mdtext.Markdown;
import com(tpb.mdtext.imagegetter.HttpImageGetter;
import com(tpb.mdtext.views.MarkdownEditText;
import com(tpb.projects.R;
import com(tpb.projects.util.SettingsActivity;
import com(tpb.projects.util.UI;
import com(tpb.projects.util.Util;
import com(tpb.projects.util.input.SimpleTextChangeWatcher;

import java.io.IOException;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;

/**
 * Created by theo on 25/03/17.
 */

public class ProjectEditor extends EditorActivity {

 public static final int REQUEST_CODE_NEW_PROJECT = 4591;
 public static final int REQUEST_CODE_EDIT_PROJECT = 1932;

 @BindView(R.id.markdown_edit_buttons) LinearLayout mEditButtons;
```

```
 @BindView(R.id.project_description_edit) MarkdownEditText
mDescriptionEditor;
 @BindView(R.id.project_name_edit) EditText mNameEditor;

 private int mProjectNumber = -1;
 private boolean mHasBeenEdited = false;

 @Override
 protected void onCreate(@Nullable Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 final SettingsActivity.Preferences prefs =
SettingsActivity.Preferences
 .getPreferences(this);
 setTheme(
 prefs.isDarkThemeEnabled() ? R.style.AppTheme_Transparent_Dark
: R.style.AppTheme_Transparent);
 setContentView(R.layout.activity_markdown_editor);
 UI.setStatusBarColor(getWindow(),
getResources().getColor(R.color.colorPrimaryDark));
 final ViewStub stub = (ViewStub) findViewById(R.id.editor_stub);

 stub.setLayoutResource(R.layout.stub_project_editor);
 stub.inflate();
 ButterKnife.bind(this);

 new MarkdownButtonAdapter(this, mEditButtons,
 new MarkdownButtonAdapter.MarkdownButtonListener() {
 @Override
 public void snippetEntered(String snippet, int
relativePosition) {
 Util.insertString(mDescriptionEditor, snippet,
relativePosition);
 }

 @Override
 public String getText() {
 return mDescriptionEditor.getInputText().toString();
 }

 @Override
 public void previewCalled() {
 if(mDescriptionEditor.isEditing()) {
 mDescriptionEditor.saveText();
 mDescriptionEditor.setMarkdown(
Markdown.formatMD(mDescriptionEditor.getText().toString(),
null
),
 new HttpImageGetter(mDescriptionEditor)
);
 mDescriptionEditor.disableEditing();
 } else {
 mDescriptionEditor.restoreText();
 mDescriptionEditor.enableEditing();
 }
 }
 });
 };
}
```

```
final View content = findViewById(android.R.id.content);
content.setVisibility(View.VISIBLE);

mNameEditor.addTextChangedListener(new SimpleTextChangeWatcher() {
 @Override
 public void textChanged() {
 mHasBeenEdited |= mDescriptionEditor.isEditing();
 }
});
mDescriptionEditor.addTextChangedListener(new
SimpleTextChangeWatcher() {
 @Override
 public void textChanged() {
 mHasBeenEdited = true;
 }
});

if(getIntent().hasExtra(getString(R.string.parcel_project))) {
 final Project project = getIntent()
 .getParcelableExtra(getString(R.string.parcel_project));
 mProjectNumber = project.getId();
 mNameEditor.setText(project.getName());
 mDescriptionEditor.setText(project.getBody());
}
}

@OnClick(R.id.markdown_editor_done)
void onDone() {
 final Intent data = new Intent();
 if(mProjectNumber != -1) {
 data.putExtra(getString(R.string.intent_project_number),
mProjectNumber);
 }
 data.putExtra(getString(R.string.intent_name),
mNameEditor.getText().toString());
 data.putExtra(getString(R.string.intent_markdown),
mDescriptionEditor.getText().toString());
 setResult(RESULT_OK, data);
 mHasBeenEdited = false;
 finish();
}

@OnClick(R.id.markdown_editor_discard)
void onDiscard() {
 finish();
}

@Override
void imageLoadComplete(String url) {
 Util.insertString(mDescriptionEditor, url);
}

@Override
void imageLoadException(IOException ioe) {

}
```

```
@Override
protected void emojiChosen(String emoji) {
 Util.insertString(mDescriptionEditor, String.format(":%1$s", emoji));
}

@Override
protected void insertString(String c) {
 Util.insertString(mDescriptionEditor, c);
}

@Override
public void finish() {
 if(mHasBeenEdited && !mNameEditor.getText().toString().isEmpty()) {
 final AlertDialog.Builder builder = new AlertDialog.Builder(this);
 builder.setTitle(R.string.title_discard_changes);
 builder.setPositiveButton(R.string.action_yes, (dialogInterface,
i) -> {
 final InputMethodManager imm = (InputMethodManager)
getSystemService(
 Context.INPUT_METHOD_SERVICE);

imm.hideSoftInputFromWindow(findViewById(android.R.id.content).getWindowToken(),
 0);
 super.finish();
 });
 builder.setNegativeButton(R.string.action_no, null);
 final Dialog deleteDialog = builder.create();
 deleteDialog.getWindow().getAttributes().windowAnimations =
R.style.DialogAnimation;
 deleteDialog.show();
 } else {
 super.finish();
 }
}
}
```

## User Activity

Once a user has logged in, their account can be displayed.  
This is objective 2.

The immediate sub-objectives of objective 2 are written to represent the different components shown when displaying a user in a paged Activity as described in the proposed design.

Each section will be shown as a Fragment within a ViewPager.

This makes the UserActivity layout and class simple, as most logic is kept separate, with each Fragment only concerned with its own purpose.

### activity\_user.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 xmlns:app="http://schemas.android.com/apk/res-auto">

 <android.support.design.widget.AppBarLayout
 android:layout_width="match_parent"
 android:layout_height="wrap_content">

 <android.support.v7.widget.Toolbar
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 app:layout_scrollFlags="scroll|enterAlwaysCollapsed">

 <!--suppress AndroidMissingOnButtonClickHandler -->
 <ImageButton
 android:id="@+id/back_button"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:background="@android:color/transparent"
 android:src="@drawable/ic_arrow_back"
 android:onClick="onToolbarBackPressed"
 android:contentDescription="@string/content_description_back"
 android:layout_marginStart="16dp"/>

 <TextView
 android:id="@+id/title_user"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="@string/title_activity_user"
 android:layout_marginStart="16dp"
 android:layout_marginEnd="16dp"

 android:textAppearance="@android:style/TextAppearance.Material.Title"/>

 </android.support.v7.widget.Toolbar>

 <android.support.design.widget.TabLayout
 android:id="@+id/user_fragment_tablayout"
```

```

 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_gravity="center_horizontal"
 app:layout_scrollFlags="scroll|snap|enterAlways"
 app:tabMode="scrollable"
 app:tabGravity="center">

 </android.support.design.widget.TabLayout>

</android.support.design.widget.AppBarLayout>

<android.support.v4.view.ViewPager
 android:id="@+id/user_fragment_viewpager"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 app:layout_behavior="@string/appbar_scrolling_view_behavior">

</android.support.v4.view.ViewPager>

</android.support.design.widget.CoordinatorLayout>

```

The UserActivity layout contains an AppBarLayout allowing the Toolbar to scroll with the contents of any ScrollViews within the Fragments which are contained in the ViewPager.

## UserActivity.java

```

package com(tpb.projects.user;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.design.widget.TabLayout;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;
import android.support.v4.view.ViewPager;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.TextView;

import com.androidnetworking.AndroidNetworking;
import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Loader;
import com(tpb.github.data.auth.GitHubSession;
import com(tpb.github.data.models.User;
import com(tpb.projects.R;
import com(tpb.projects.common BaseActivity;
import com(tpb.projects.common ShortcutDialog;
import com(tpb.projects.user.fragments UserFollowersFragment;
import com(tpb.projects.user.fragments UserFollowingFragment;
import com(tpb.projects.user.fragments UserFragment;
import com(tpb.projects.user.fragments UserGistsFragment;
import com(tpb.projects.user.fragments UserInfoFragment;
import com(tpb.projects.user.fragments UserReposFragment;
import com(tpb.projects.user.fragments UserStarsFragment;
import com(tpb.projects.util.SettingsActivity;

```

```
import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by theo on 10/03/17.
 */

public class UserActivity extends BaseActivity implements
Loader.ItemLoader<User> {
 private static final String TAG = UserActivity.class.getSimpleName();
 private static final String URL =
"https://github.com/tpb1908/AndroidProjectsClient/blob/master/app/src/main/java/com/tpb/projects/user/UserActivity.java";

 @BindView(R.id.title_user) TextView mTitle;
 @BindView(R.id.user_fragment_tablayout) TabLayout mTabs;
 @BindView(R.id.user_fragment_viewpager) ViewPager mPager;

 private UserFragmentAdapter mAdapter;
 private User mUser;

 @Override
 protected void onCreate(@Nullable Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 if(!mHasAccess) return;

 setTheme(SettingsActivity.Preferences.getPreferences(this).isDarkThemeEnabled());
 ? R.style.AppTheme_Dark : R.style.AppTheme);
 setContentView(R.layout.activity_user);
 ButterKnife.bind(this);
 postponeEnterTransition();

 if(mAdapter == null) mAdapter = new
UserFragmentAdapter(getSupportFragmentManager());
 final Loader loader = Loader.getLoader(this);

 if(getIntent() != null &&
getIntent().hasExtra(getString(R.string.intent_username))) {
 final String user =
getIntent().getStringExtra(getString(R.string.intent_username));
 mTitle.setText(user);
 loader.loadUser(this, user);
 } else {
 if(isTaskRoot()) {
 findViewById(R.id.back_button).setVisibility(View.GONE);
 }
 loadComplete(GitHubSession.getSession(this).getUser());
 }
 mTabs.setupWithViewPager(mPager);
 mPager.setAdapter(mAdapter);
 mPager.setOffscreenPageLimit(7);
 }

 @Override
 public void loadComplete(User user) {
 mUser = user;
```

```
mTitle.setText(mUser.getLogin());
mAdapter.notifyUserLoaded();
}

@Override
public void loadError(APIHandler.APIError error) {
}

@Override
public void onAttachFragment(Fragment fragment) {
 super.onAttachFragment(fragment);
 if(mAdapter == null) mAdapter = new
UserFragmentAdapter(getSupportFragmentManager());
 mAdapter.ensureAttached((UserFragment) fragment);
}

private class UserFragmentAdapter extends FragmentPagerAdapter {

 private UserFragment[] mFragments = new UserFragment[6];

 UserFragmentAdapter(FragmentManager fm) {
 super(fm);
 }

 @Override
 public Fragment getItem(int position) {
 switch(position) {
 case 0:
 mFragments[0] = new UserInfoFragment();
 break;
 case 1:
 mFragments[1] = new UserReposFragment();
 break;
 case 2:
 mFragments[2] = new UserStarsFragment();
 break;
 case 3:
 mFragments[3] = new UserGistsFragment();
 break;
 case 4:
 mFragments[4] = new UserFollowingFragment();
 break;
 case 5:
 mFragments[5] = new UserFollowersFragment();
 break;
 }
 if(mUser != null) mFragments[position].userLoaded(mUser);
 return mFragments[position];
 }

 void ensureAttached(UserFragment fragment) {
 if(fragment instanceof UserInfoFragment) mFragments[0] = fragment;
 else if(fragment instanceof UserReposFragment) mFragments[1] =
fragment;
 else if(fragment instanceof UserStarsFragment) mFragments[2] =
fragment;
 else if(fragment instanceof UserGistsFragment) mFragments[3] =
fragment;
 }
}
```

```
 else if(fragment instanceof UserFollowingFragment) mFragments[4] =
fragment;
 else if(fragment instanceof UserFollowersFragment) mFragments[5] =
fragment;
 }

 void notifyUserLoaded() {
 for(UserFragment f : mFragments) {
 if(f != null) f.userLoaded(mUser);
 }
 }

 @Override
 public int getCount() {
 return mFragments.length;
 }

 @Override
 public CharSequence getPageTitle(int position) {
 switch(position) {
 case 0:
 return getString(R.string.title_user_info_fragment);
 case 1:
 return getString(R.string.title_user_repos_fragment);
 case 2:
 return getString(R.string.title_user_stars_fragment);
 case 3:
 return getString(R.string.title_user_gists_fragment);
 case 4:
 return getString(R.string.title_user_following_fragment);
 case 5:
 return getString(R.string.title_user_followers_fragment);
 default:
 return "Error";
 }
 }
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
 //TODO Pass to fragment
 getMenuInflater().inflate(R.menu.menu_activity, menu);
 return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
 switch(item.getItemId()) {
 case R.id.menu_settings:
 startActivity(new Intent(UserActivity.this,
SettingsActivity.class));
 break;
 case R.id.menu_source:
 startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse(URL)));
 break;
 case R.id.menu_share:
 final Intent share = new Intent();
 share.setAction(Intent.ACTION_SEND);
 share.putExtra(Intent.EXTRA_TEXT, mUser.getHtmlUrl());
 }
}
```

```

 share.addCategory(Intent.CATEGORY_BROWSABLE);
 share.setType("text/plain");
 startActivity(share);
 break;
 case R.id.menu_save_to_homescreen:
 final ShortcutDialog dialog = new ShortcutDialog();
 final Bundle args = new Bundle();
 args.putInt(getString(R.string.intent_title_res),
R.string.title_save_user_shortcut);
 args.putString(getString(R.string.intent_name),
mUser.getLogin());

 dialog.setArguments(args);
 dialog.setListener((name, iconFlag) -> {
 final Intent i = new Intent(getApplicationContext(),
UserActivity.class);
 i.putExtra(getString(R.string.intent_username),
mUser.getLogin());

 final Intent add = new Intent();
 add.putExtra(Intent.EXTRA_SHORTCUT_INTENT, i);
 add.putExtra(Intent.EXTRA_SHORTCUT_NAME, name);
 add.putExtra("duplicate", false);
 add.putExtra(Intent.EXTRA_SHORTCUT_ICON_RESOURCE,
Intent.ShortcutIconResource.fromContext(getApplicationContext(),
R.mipmap.ic_launcher));

 add.setAction("com.android.launcher.action.INSTALL_SHORTCUT");
 getApplicationContext().sendBroadcast(add);
 });
 dialog.show(getSupportFragmentManager(), TAG);
 break;
 }
 return true;
}

@Override
protected void onDestroy() {
 super.onDestroy();
 AndroidNetworking.cancelAll();
}
}

```

### **fragment\_user\_info.xml**

```

<?xml version="1.0" encoding="utf-8"?>

<android.support.v4.widget.SwipeRefreshLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 android:id="@+id/user_info_refresher"
 android:orientation="vertical"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 app:layout_behavior="@string/appbar_scrolling_view_behavior">

 <android.support.v4.widget.NestedScrollView
 android:layout_width="match_parent"
 android:layout_height="wrap_content">

```

```
<LinearLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 android:orientation="vertical"
 android:layout_width="match_parent"
 android:layout_height="match_parent">

 <android.support.v7.widget.CardView
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:orientation="vertical"
 android:layout_margin="8dp">

 <include layout="@layout/shard_user_info"/>

 </android.support.v7.widget.CardView>

 <android.support.v7.widget.CardView
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:orientation="vertical"
 android:layout_margin="8dp">

 <LinearLayout
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_marginStart="16dp"
 android:layout_marginEnd="16dp"
 android:layout_marginTop="8dp"
 android:layout_marginBottom="8dp"
 android:orientation="vertical">

 <TextView
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="@string/text_user_activity"

 android:textAppearance="@android:style/TextAppearance.Material.Large"/>

 <com(tpb.contributionsview.ContributionsView
 android:id="@+id/user_contributions"
 android:layout_width="match_parent"
 android:layout_height="80dp"
 app:textSize="12sp"
 app:textColor="?android:textColorPrimary"/>

 <TextView
 android:id="@+id/user_contributions_info"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"/>

 </LinearLayout>

 </android.support.v7.widget.CardView>

</LinearLayout>
```

```
</android.support.v4.widget.SwipeRefreshLayout>
```

When it is created, `UserActivity` calls the `super` method ( `BaseActivity`) which performs the access check, allowing `UserActivity` to return if the app doesn't have an access token.

If this check passes, `onCreate` method continues by checking the theme theme before inflating the `activity_user` layout and bindings its Views.

Once the Views have been inflated, the `UserActivity` can continue by creating the `FragmentPagerAdapter` which is responsible for creating each of the Fragments. The `UserActivity` then determines whether it has been started from a link to a user, in which case the user must be loaded, or if the app is starting from the homescreen to display the authenticated user.

## UserFragment

UserFragment is an abstract class extending ViewSafeFragment used to add a User model instance, the userLoaded method , as well as to save and restore the User instance when the Fragment is added to or removed from the back stack.

### UserFragment.java

```
package com(tpb.projects.user.fragments;

import android.os.Bundle;
import android.support.annotation.Nullable;

import com(tpb.github.data.models.User;
import com(tpb.projects.R;
import com(tpb.projects.common.ViewSafeFragment;
import com(tpb.projects.user.UserActivity;

/**
 * Created by theo on 10/03/17.
 */

public abstract class UserFragment extends ViewSafeFragment {

 protected User mUser;

 public abstract void userLoaded(User user);

 protected UserActivity getParent() {
 return (UserActivity) getActivity();
 }

 @Override
 public void onActivityCreated(@Nullable Bundle savedInstanceState) {
 super.onActivityCreated(savedInstanceState);
 if(savedInstanceState != null && savedInstanceState
 .containsKey(getString(R.string.parcel_user))) {
 mUser =
 savedInstanceState.getParcelable(getString(R.string.parcel_user));
 userLoaded(mUser);
 }
}

 @Override
 public void onSaveInstanceState(Bundle outState) {
 super.onSaveInstanceState(outState);
 outState.putParcelable(getString(R.string.parcel_user), mUser);
 }
}
```

## UserInfoFragment

The first Fragment to be displayed is the UserInfoFragment.

The UserInfoFragment is to fulfill objective 2.a, displaying information about the user.

The `UserInfoFragment` layout has two cards, the first displaying text based information about the user, and the second displaying a graph of the users' contributions.

## fragment\_user\_info.xml

```
<?xml version="1.0" encoding="utf-8"?>

<android.support.v4.widget.SwipeRefreshLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 android:id="@+id/user_info_refresher"
 android:orientation="vertical"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 app:layout_behavior="@string/appbar_scrolling_view_behavior">

 <android.support.v4.widget.NestedScrollView
 android:layout_width="match_parent"
 android:layout_height="wrap_content">

 <LinearLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 android:orientation="vertical"
 android:layout_width="match_parent"
 android:layout_height="match_parent">

 <android.support.v7.widget.CardView
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:orientation="vertical"
 android:layout_margin="8dp">

 <include layout="@layout/shard_user_info"/>

 </android.support.v7.widget.CardView>

 <android.support.v7.widget.CardView
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:orientation="vertical"
 android:layout_margin="8dp">

 <LinearLayout
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_marginStart="16dp"
 android:layout_marginEnd="16dp"
 android:layout_marginTop="8dp"
 android:layout_marginBottom="8dp"
 android:orientation="vertical">

 <TextView
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="User Info Content" />

 </LinearLayout>

 </android.support.v7.widget.CardView>

 </LinearLayout>

 </android.support.v4.widget.NestedScrollView>

</android.support.v4.widget.SwipeRefreshLayout>
```

```

 android:layout_height="wrap_content"
 android:text="@string/text_user_activity"

 android:textAppearance="@android:style/TextAppearance.Material.Large"/>

 <com(tpb.contributionsview.ContributionsView
 android:id="@+id/user_contributions"
 android:layout_width="match_parent"
 android:layout_height="80dp"
 app:textSize="12sp"
 app:textColor="?android:textColorPrimary"/>

 <TextView
 android:id="@+id/user_contributions_info"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"/>

</LinearLayout>

</android.support.v7.widget.CardView>

</LinearLayout>

</android.support.v4.widget.NestedScrollView>

</android.support.v4.widget.SwipeRefreshLayout>

```

The layout included within the first CardView is the same layout used in LoginActivity to display the user's information

### **shard\_user\_info.xml**

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
 android:id="@+id/user_details"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:orientation="vertical"
 android:layout_gravity="top"
 xmlns:android="http://schemas.android.com/apk/res/android">

 <LinearLayout
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:orientation="horizontal">

 <com(tpb.projects.common.NetworkImageView
 android:id="@+id/user_avatar"
 android:layout_width="40dp"
 android:layout_height="40dp"
 android:layout_margin="16dp"
 android:layout_gravity="center_vertical"
 android:transitionName="@string/transition_user_image"/>

 <TextView
 android:id="@+id/user_login"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"

```

```
 android:layout_gravity="center_vertical"
 android:text="@string/text_placeholder"

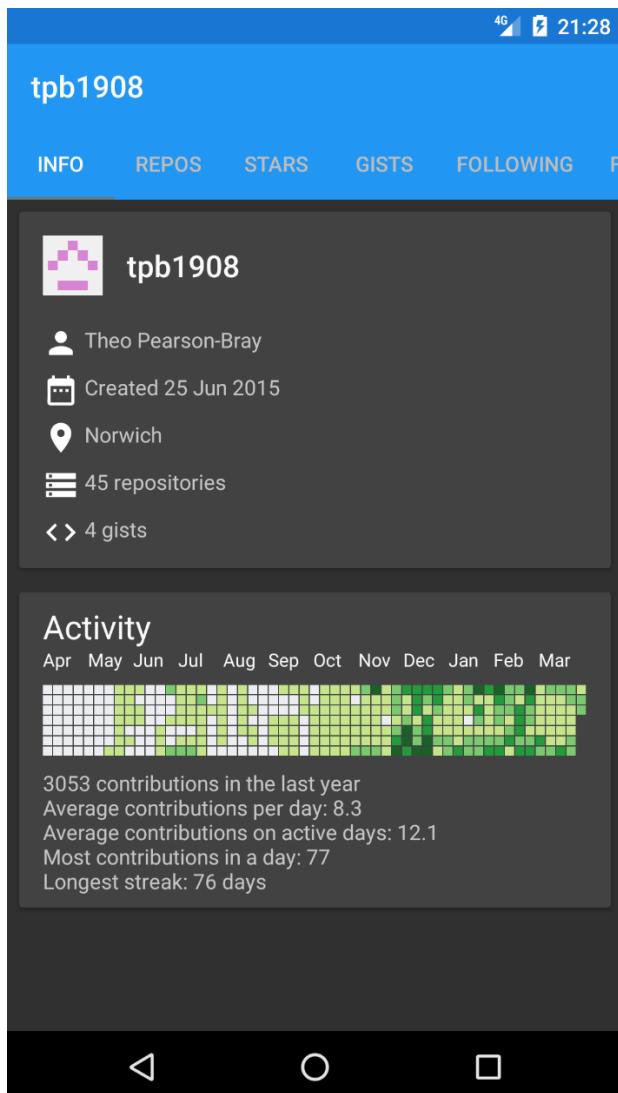
 android:textAppearance="@android:style/TextAppearance.Material.Title"
 android:transitionName="@string/transition_username"/>

</LinearLayout>

<LinearLayout
 android:id="@+id/user_info_layout"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_marginStart="16dp"
 android:layout_marginEnd="16dp"
 android:layout_marginBottom="8dp"
 android:orientation="vertical">

</LinearLayout>
```

and contains a `LinearLayout` to display the user's avatar and username, as well as another `LinearLayout` to display a list of the user's available information. Once inflated and bound with a user's information, the layout will be displayed as below.



## Animation

In Android 5.0, released November 2014, shared element transitions were introduced.

These transitions allow a View to be shared between two Activities via the `ViewOverlay` which is drawn on top of all other Views.

A common use is for an `ImageView` to animate between two Activities. This works well so long as the `ImageView` maintains its aspect ratio.

A common case for the `UserActivity` being opened is when the user clicks on an avatar. The avatar can be animated from its original position to its new position in the `UserActivity`. The user name can also be animated from one `TextView` to another.

If the Activity had a single layout inflated in its `onCreate` method, this animation would be simple to perform, and wouldn't require any Java code only XML attributes.

However, the `NetworkImageView` displaying the user avatar is shown in

the UserInfoFragment which is inflated after the UserActivity. Further, the image stored in

the NetworkImageView in the launching Activity was loaded from a URL, and the new NetworkImageView in the UserInfoFragment would load the image again, defeating

the purpose of the transition.

The first problem is solved by postponing the entry transition.

The line after View binding in UserActivity is the call postponeEnterTransition() which, as its name suggests, postpones the entry transition until the startPostponedEnterTransition is called.

### **UserInfoFragment.java**

```
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
 final View view = inflater.inflate(R.layout.fragment_user_info,
 container, false);
 unbinder = ButterKnife.bind(this, view);
 mRefresher.setRefreshing(true);
 mAvatar.getViewTreeObserver()
 .addOnPreDrawListener(new ViewTreeObserver.OnPreDrawListener()
{
 @Override
 public boolean onPreDraw() {
 if(getActivity().getIntent() != null &&
getActivity().getIntent().hasExtra(
 getString(R.string.intent_drawable))) {
 final Bitmap bm =
getActivity().getIntent().getParcelableExtra(
 getString(R.string.intent_drawable));

mUserLogin.setText(getActivity().getIntent().getStringExtra(
 getString(R.string.intent_username)));
 mAvatar.setImageBitmap(bm);
 }
 }

 mAvatar.getViewTreeObserver().removeOnPreDrawListener(this);
 getActivity().startPostponedEnterTransition();
 return true;
 }
});
 mRefresher.setOnRefreshListener(
 () -> Loader.getLoader(getContext()).loadUser(new
 Loader.ItemLoader<User>() {
 @Override
 public void loadComplete(User user) {
 userLoaded(user);
 }

 @Override
 public void loadError(APIHandler.APIError error) {
 mRefresher.setRefreshing(false);
 }
 }, mUser.getLogin())
);
}
```

```

 mAreViewsValid = true;
 if(mUser != null) userLoaded(mUser);
 return view;
}

```

Once the `view` has been created in `UserInfoFragment` we have the `NetworkImageView` required to perform the transition.

However, the full layout has not been calculated, so the animation cannot yet be performed.

In order to wait until the layout has been completed, an `OnPreDrawListener` is added to the `ViewTreeObserver` of the `NetworkImageView` `mAvatar`.

When `onPreDraw` is called, the listener checks that the the `Intent` contains the key `intent_drawable`, and if it is present it sets the `Bitmap` of `mAvatar` as well as setting the text of the `TextView` `mUserLogin`.

Due to the nature of `view` layouts and drawing, adding a listener to the `ViewTreeObserver` is computationally expensive. It is therefore important to remove the

`onPreDrawListener` from the `NetworkImageView`.

Finally, `startPostponedEnterTransition` can be called on the `UserInfoFragment`'s parent `Activity`.

The second problem, which was assumed to have been solved when setting the drawable above, is to provide the `ImageView` in the new `Activity` with the avatar to be drawn.

This is done in the `UI` utility method `setDrawableForIntent`

### UI.java

```

public static void setDrawableForIntent(@NonNull ImageView iv, @NonNull Intent i) {
 if(iv.getDrawable() instanceof BitmapDrawable) {
 i.putExtra(iv.getResources().getString(R.string.intent_drawable),
 ((BitmapDrawable) iv.getDrawable()).getBitmap());
 }
}

```

The method checks that the `ImageView`'s `Drawable` is an instance of `BitmapDrawable`, which it will be if loaded from a `Bitmap` as `NetworkImageView` does.

The `instanceof` comparator also performs a null check as the language specification defines the result of the operator to be "true if the value of the `RelationalExpression` is not null and the reference could be cast (§15.16) to the `ReferenceType` without raising a `ClassCastException`".

If the `BitmapDrawable` is valid, it is added to the `Intent`.

## Use of `ViewSafeFragment`

When a `UserFragment` is created, there are two possible orders of events.

The first is that `onCreateView` is called, and at some point in the future the network call to load the user returns, `userLoaded` is called and the `Views` are

bound.

The second is that the `userLoaded` is called before `onCreateView`, which would result in a `NullPointerException` if an attempt was made to bind data to null `Views`.

This is solved with two checks.

In `userLoaded` the first line is `mUser = user`, assigning the member variable `mUser`. The second line is `if(!mAreViewsValid) return` which will ensure that no `Views` are bound to if they have not yet been created. This deals with the crash which could happen in

the second case, however we must now ensure that the `Views` are bound once they have been created.

This is achieved with the second check, the last line in `onCreateView` before returning the `view`.

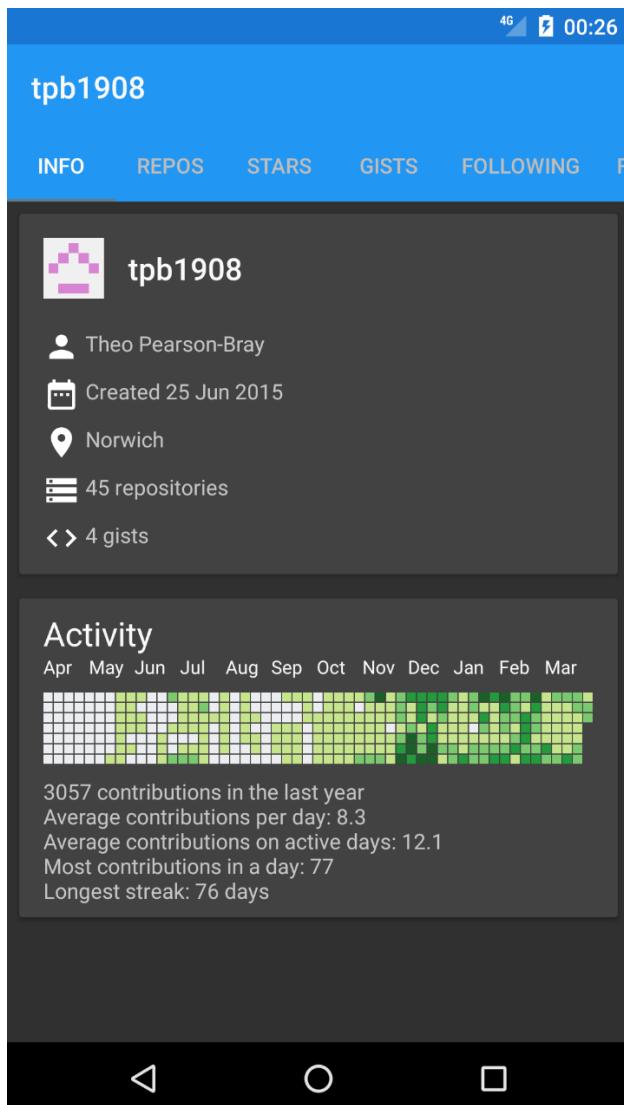
This check is `if(mUser != null) userLoaded(mUser)`. As `mAreViewsValid` has been set to true, this will bind the `Views` if the `User` was initially loaded before the `Views` were created.

This structure is used throughout the different concrete implementations of `UserFragment`.

## **ContributionsView**

Objective 2.a.iv is to display the user's contributions in a graphical form.

As explained in the limitations section, there is no API for loading a user's contributions. Instead this can be achieved by parsing the SVG image of a user's contributions



This image can be loaded  
from [https://github.com/users/user\\_name/contributions](https://github.com/users/user_name/contributions)

### Loading contributions

Each column in the image is made up of a set of rectangles, with three important attributes.

```
<g transform="translate(0, 0)">
 <rect class="day" width="10" height="10" x="13" y="0" fill="#ebedf0"
data-count="0" data-date="2016-04-10"></rect>
 <rect class="day" width="10" height="10" x="13" y="12"
fill="#ebedf0" data-count="0" data-date="2016-04-11"></rect>
 <rect class="day" width="10" height="10" x="13" y="24"
fill="#ebedf0" data-count="0" data-date="2016-04-12"></rect>
 <rect class="day" width="10" height="10" x="13" y="36"
fill="#ebedf0" data-count="0" data-date="2016-04-13"></rect>
 <rect class="day" width="10" height="10" x="13" y="48"
fill="#ebedf0" data-count="0" data-date="2016-04-14"></rect>
```

```

<rect class="day" width="10" height="10" x="13" y="60"
fill="#ebedf0" data-count="0" data-date="2016-04-15"></rect>
<rect class="day" width="10" height="10" x="13" y="72"
fill="#ebedf0" data-count="0" data-date="2016-04-16"></rect>
</g>
```

Firstly, the `fill` attribute provides the hexadecimal colour code used to draw the rectangle, which can be parsed and used to draw the image later.

Secondly, the `data-count` attribute provides the number of contributions made for the particular day.

Finally, the `data-date` attribute is the date represented by the rectangle in the format `yyyy-mm-dd`.

In order to load the SVG, a request can be made to download it as a string.

The `ContributionsLoader` class is used to load a user's contributions.

### **ContributionsLoader.java**

```

package com(tpb.contributionsview;

import android.content.Context;
import android.graphics.Color;
import android.os.Parcel;
import android.os.Parcelable;
import android.support.annotation.ColorInt;
import android.support.annotation.NonNull;

import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;

import java.lang.ref.WeakReference;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;

/**
 * Created by theo on 12/01/17.
 */

public class ContributionsLoader {

 // Format string for svg path
 private static final String IMAGE_BASE =
"https://github.com/users/%s/contributions";

 private final WeakReference<ContributionsRequestListener> mListener;

 ContributionsLoader(@NonNull ContributionsRequestListener listener) {
 mListener = new WeakReference<>(listener);
 }

 void beginRequest(@NonNull Context context, @NonNull String login) {
 final String URL = String.format(IMAGE_BASE, login);
 // Load the svg as a string
 final StringRequest req = new StringRequest(Request.Method.GET, URL,
```

```

 new Response.Listener<String>() {
 @Override
 public void onResponse(String response) {
 parse(response);
 }
 }, new Response.ErrorListener() {
 @Override
 public void onErrorResponse(VolleyError error) {
 if(mListener.get() != null)
 mListener.get().onError(error);
 }
);
 Volley.newRequestQueue(context).add(req);
 }

 private void parse(String response) {
 final ArrayList<ContributionsDay> contributions = new ArrayList<>();
 int first = response.indexOf("<rect");
 int last;
 // Find each rectangle in the image
 while(first != -1) {
 last = response.indexOf(">", first);
 contributions.add(new ContributionsDay(response.substring(first,
last)));
 first = response.indexOf("<rect", last);
 }
 if(mListener.get() != null) mListener.get().onResponse(contributions);
 }

 public static class ContributionsDay implements Parcelable {
 private static final SimpleDateFormat sdf = new
SimpleDateFormat("yyyy-MM-dd");
 @ColorInt public final int color;
 public long date;
 public final int contributions;

 ContributionsDay(String rect) {
 //rect is <rect class="day" width="10" height="10" x="" y=""
fill="#FFFFF" data-count="n" data-date="yyyy-mm-dd"/>
 final int colorIndex = rect.indexOf("fill="") + 6;
 color = Color.parseColor(rect.substring(colorIndex, colorIndex +
7));

 final int countIndex = rect.indexOf("data-count="") + 12;
 final int countEndIndex = rect.indexOf("", countIndex);
 contributions = Integer.parseInt(rect.substring(countIndex,
countEndIndex));

 final int dateIndex = rect.indexOf("data-date="") + 11;
 try {
 date = sdf.parse(rect.substring(dateIndex, dateIndex +
11)).getTime();
 } catch(ParseException ignored) {}
 }

 @Override
 }
}

```

```

 public String toString() {
 return "ContributionsDay{" +
 "color=" + color +
 ", date=" + date +
 ", contributions=" + contributions +
 '}';
 }

 @Override
 public int describeContents() {
 return 0;
 }

 @Override
 public void writeToParcel(Parcel dest, int flags) {
 dest.writeInt(this.color);
 dest.writeLong(this.date);
 dest.writeInt(this.contributions);
 }

 protected ContributionsDay(Parcel in) {
 this.color = in.readInt();
 this.date = in.readLong();
 this.contributions = in.readInt();
 }

 public static final Creator<ContributionsDay> CREATOR = new
 Creator<ContributionsDay>() {
 @Override
 public ContributionsDay createFromParcel(Parcel source) {
 return new ContributionsDay(source);
 }

 @Override
 public ContributionsDay[] newArray(int size) {
 return new ContributionsDay[size];
 }
 };
}

interface ContributionsRequestListener {

 void onResponse(ArrayList<ContributionsDay> contributions);

 void onError(VolleyError error);

}
}

```

The interface `ContributionsRequestListener` is used as a callback once the contributions information is loaded.

As such, the `ContributionsLoader` constructor takes a `ContributionsLoader` and stores a `WeakReference` to it.

The `beginRequest` method formats the user login and performs a get request to the URL of the SVG image.

If the image is loaded successfully, the `parse` method is called.

`parse` uses two counters, `first` and `last` to iterate through each substring which is surrounded by the tags `<rect` and the closing tag `>`.

The `String indexOf` method searches a string for a substring from a given position, 0 in the simpler overloaded method, returning -1 if the substring is not present.

Each time the substring is found, the `parse` method adds a new `ContributionsDay` object to the contributions `ArrayList`.

`ContributionsDay` contains the integer value of the `fill` attribute, the long value of the `data-date` attribute, and the integer value of the `data-count` attribute.

Finally, if the `ContributionsRequestListener` is non null, the values are returned to it.

## Displaying contributions

The `ContributionsView` is a descendant of `View` used to draw the contributions heatmap.

### ContributionsView.java

```
package com(tpb.contributionsview;

import android.content.Context;
import android.content.res.TypedArray;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Rect;
import android.os.Parcel;
import android.os.Parcelable;
import android.util.AttributeSet;
import android.view.View;
import android.view.ViewGroup;

import com.android.volley.VolleyError;

import java.lang.ref.WeakReference;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.List;

/**
 * Created by theo on 12/01/17.
 */

public class ContributionsView extends View implements
ContributionsLoader.ContributionsRequestListener {

 private static final Calendar mCalendar = Calendar.getInstance();
```

```
private boolean mShouldDisplayMonths;
private int mTextColor;
private int mTextSize;
private int mBackGroundColor;
private ArrayList<ContributionsLoader.ContributionsDay> mContributions =
new ArrayList<>();

private Paint mDayPainter;
private Paint mTextPainter;

private Rect mRect;
private final Rect mTextBounds = new Rect();

private WeakReference<ContributionsLoadListener> mListener;

public ContributionsView(Context context) {
 super(context);
 initView(context, null, 0, 0);
}

public ContributionsView(Context context, AttributeSet attrs) {
 super(context, attrs);
 initView(context, attrs, 0, 0);
}

public ContributionsView(Context context, AttributeSet attrs, int defStyleAttr) {
 super(context, attrs, defStyleAttr);
 initView(context, attrs, defStyleAttr, 0);
}

private void initView(Context context, AttributeSet attrs, int defStyleAttr, int defStyleRes) {
 mRect = new Rect();

 final TypedArray attributes =
context.getTheme().obtainStyledAttributes(attrs,
 R.styleable.ContributionsView, defStyleAttr, defStyleRes
);
 useAttributes(attributes);

 mTextPainter = new Paint(Paint.ANTI_ALIAS_FLAG);
 mDayPainter = new Paint(Paint.ANTI_ALIAS_FLAG);
 mDayPainter.setStyle(Paint.Style.FILL);
}

private void useAttributes(TypedArray ta) {
 mShouldDisplayMonths =
ta.getBoolean(R.styleable.ContributionsView_showMonths, true);
 mBackGroundColor =
ta.getColor(R.styleable.ContributionsView_backgroundColor,
 0xD6E685
); //GitHub default color
 mTextColor = ta.getColor(R.styleable.ContributionsView_textColor,
Color.BLACK);
 mTextSize =
ta.getDimensionPixelSize(R.styleable.ContributionsView_textSize, 7);
```

```
 if(ta.getString(R.styleable.ContributionsView_username) != null &&
!isInEditMode()) {

loadContributions(ta.getString(R.styleable.ContributionsView_username));
 }
}

public void loadContributions(String user) {
 new ContributionsLoader(this).beginRequest(getContext(), user);
}

@Override
protected void onDraw(Canvas canvas) {
 super.onDraw(canvas);
 canvas.getClipBounds(mRect);

 final int w = mRect.width();
 final int h = mRect.height();

 final int hnum = mContributions.size() == 0 ? 52 : (int) Math
 .ceil(mContributions.size() / 7d); //The number of columns to
show horizontally

 final float bd = (w / (float) hnum) * 0.9f; //The dimension of a
single block
 final float m = (w / (float) hnum) - bd; //The margin around a block

 final float mth = mShouldDisplayMonths ? mTextSize : 0; //Height of
month text

 //Draw the background
 mDayPainter.setColor(mBackGroundColor);
 canvas.drawRect(0, (2 * mth), w, h + mth, mDayPainter);
 float x = 0;
 if(mContributions.size() > 0) {
 int dow = getDayOfWeek(mContributions.get(0).date) - 1;
 float y = (dow * (bd + m)) + 2 * mth;
 for(ContributionsLoader.ContributionsDay d : mContributions) {
 mDayPainter.setColor(d.color);
 canvas.drawRect(x, y, x + bd, y + bd, mDayPainter);
 dow = getDayOfWeek(d.date) - 1;
 if(dow == 6) { //We just drew the last day of the week
 x += bd + m;
 y = 2 * mth;
 } else {
 y += bd + m;
 }
 }
 } else {
 int dow = 0;
 float y = 2 * mth;
 mDayPainter.setColor(0xffffffff);
 for(int i = 0; i < 364; i++) {
 canvas.drawRect(x, y, x + bd, y + bd, mDayPainter);
 if(dow == 6) { //We just drew the last day of the week
 x += bd + m;
 y = 2 * mth;
 dow = 0;
 }
 }
 }
}
```

```

 } else {
 y += bd + m;
 dow++;
 }
 }
}

if(mShouldDisplayMonths) {
 mTextPainter.setColor(mTextColor);
 mTextPainter.setTextSize(mth);
 mCalendar.setTimeInMillis(System.currentTimeMillis());
 mCalendar.add(Calendar.MONTH, -12);
 x = 0;
 for(int i = 0; i < 12; i++) {
 final String month =
getMonthName(mCalendar.getTimeInMillis());
 mTextPainter.getTextBounds(month, 0, month.length(),
mTextBounds);
 if(w > x + mTextBounds.width()) {
 canvas.drawText(
 month,
 x,
 mth,
 mTextPainter
);
 } else {
 break;
 }
 mCalendar.add(Calendar.MONTH, 1);
 x += w / 12;
 }
}
final ViewGroup.LayoutParams lp = getLayoutParams();
lp.height = h;
setLayoutParams(lp);
}

private int getDayOfWeek(long stamp) {
 mCalendar.setTimeInMillis(stamp);
 //Day of week is indexed 1 to 7
 return mCalendar.get(Calendar.DAY_OF_WEEK);
}

private static final SimpleDateFormat month = new SimpleDateFormat("MMM");

private String getMonthName(long stamp) {
 return month.format(stamp);
}

@Override
public void onResponse(ArrayList<ContributionsLoader.ContributionsDay>
contributions) {
 mContributions = contributions;
 invalidate();
 if(mListener != null && mListener.get() != null) {
 mListener.get().contributionsLoaded(contributions);
 }
}

@Override

```

```
public void onError(VolleyError error) {
}

public void setListener(ContributionsLoadListener listener) {
 mListener = new WeakReference<>(listener);
}

public interface ContributionsLoadListener {

 void contributionsLoaded(List<ContributionsLoader.ContributionsDay>
contributions);
}

@Override
protected Parcelable onSaveInstanceState() {
 final Parcelable ss = super.onSaveInstanceState();
 final ContributionsState state = new ContributionsState(ss);
 state.contributions = mContributions.toArray(new
ContributionsLoader.ContributionsDay[0]);
 return state;
}

@Override
protected void onRestoreInstanceState(Parcelable state) {
 super.onRestoreInstanceState(state);

 if(!(state instanceof ContributionsState)) {
 super.onRestoreInstanceState(state);
 return;
 }
 final ContributionsState cs = (ContributionsState) state;
 super.onRestoreInstanceState(cs.getSuperState());
 this.mContributions = new
ArrayList<>(Arrays.asList(cs.contributions));
 invalidate();
}

private static class ContributionsState extends BaseSavedState {
 ContributionsLoader.ContributionsDay[] contributions;

 ContributionsState(Parcel source) {
 super(source);
 this.contributions =
source.createTypedArray(ContributionsLoader.ContributionsDay.CREATOR);
 }

 ContributionsState.Parcelable superState) {
 super(superState);
 }

 @Override
 public void writeToParcel(Parcel out, int flags) {
 super.writeToParcel(out, flags);
 out.writeTypedArray(contributions, flags);
 }

 public static final Parcelable.Creator<ContributionsState> CREATOR =
```

```

 new Parcelable.Creator<ContributionsState>() {
 public ContributionsState createFromParcel(Parcel in) {
 return new ContributionsState(in);
 }

 public ContributionsState[] newArray(int size) {
 return new ContributionsState[size];
 }
 };
 }
}

```

The ContributionsView has attributes for its background colour, text colour, text size, and whether months should be displayed. A default user login can also be set via XML.

As the view is being repeatedly drawn, it is important to ensure that as few objects as possible are allocated in the `onDraw` method.

The single `Paint` objects for text and day (rectangle) painting are therefore reused, as are the `Rectangle` used to store the `View` dimensions and the second `Rectangle` used to store text bounds.

The `Calendar` used for parsing dates is static and used in all instances of `ContributionsView`.

The `ContributionsView` implements `ContributionsRequestListener` and contains the `loadContributions` method, which creates an anonymous `ContributionsLoader` and begins loading the contributions for the given user.

In the `onResponse` method the `ContributionsView` stores the `ArrayList` of contributions and calls the `invalidate` method, forcing a redraw.

It also checks if its listener has been set, and notifies it of the newly loaded `ArrayList` of `ContributionsDay` objects.

The `onDraw` method needs to deal with two states, either the `ContributionsView` has a non empty list of `ContributionsDays` or it does not.

## **ContributionsView.java**

```

protected void onDraw(Canvas canvas) {
 super.onDraw(canvas);
 canvas.getClipBounds(mRect);

 final int w = mRect.width();
 final int h = mRect.height();

 final int hnum = mContributions.size() == 0 ? 52 : (int) Math
 .ceil(mContributions.size() / 7d); //The number of columns to
 show horizontally

 final float bd = (w / (float) hnum) * 0.9f; //The dimension of a
single block
 final float m = (w / (float) hnum) - bd; //The margin around a block
}

```

```

final float mth = mShouldDisplayMonths ? mTextSize : 0; //Height of
month text

//Draw the background
mDayPainter.setColor(mBackGroundColor);
canvas.drawRect(0, (2 * mth), w, h + mth, mDayPainter);
float x = 0;
if(mContributions.size() > 0) {
 int dow = getDayOfWeek(mContributions.get(0).date) - 1;
 float y = (dow * (bd + m)) + 2 * mth;
 for(ContributionsLoader.ContributionsDay d : mContributions) {
 mDayPainter.setColor(d.color);
 canvas.drawRect(x, y, x + bd, y + bd, mDayPainter);
 dow = getDayOfWeek(d.date) - 1;
 if(dow == 6) { //We just drew the last day of the week
 x += bd + m;
 y = 2 * mth;
 } else {
 y += bd + m;
 }
 }
} else {
 int dow = 0;
 float y = 2 * mth;
 mDayPainter.setColor(0xffffffff);
 for(int i = 0; i < 364; i++) {
 canvas.drawRect(x, y, x + bd, y + bd, mDayPainter);
 if(dow == 6) { //We just drew the last day of the week
 x += bd + m;
 y = 2 * mth;
 dow = 0;
 } else {
 y += bd + m;
 dow++;
 }
 }
}
if(mShouldDisplayMonths) {
 mTextPainter.setColor(mTextColor);
 mTextPainter.setTextSize(mth);
 mCalendar.setTimeInMillis(System.currentTimeMillis());
 mCalendar.add(Calendar.MONTH, -12);
 x = 0;
 for(int i = 0; i < 12; i++) {
 final String month =
getMonthName(mCalendar.getTimeInMillis());
 mTextPainter.getTextBounds(month, 0, month.length(),
mTextBounds);
 if(w > x + mTextBounds.width()) {
 canvas.drawText(
 month,
 x,
 mth,
 mTextPainter
);
 } else {
 break;
 }
 }
}

```

```

 }
 mCalendar.add(Calendar.MONTH, 1);
 x += w / 12;
 }
}
final ViewGroup.LayoutParams lp = getLayoutParams();
lp.height = h;
setLayoutParams(lp);
}

```

The `onDraw` method begins by measuring the size which it has to draw in. Next, the number of columns to show horizontally can be calculated, as either the number of contributions split across 7 day weeks, or 52 weeks. Each rectangle has an area of the view width over the horizontal number of contributions, however it must also account for the margin between each rectangle. For each rectangular segment of the canvas, 90% of the width will be filled with the block, and the remaining 10% left as a margin.

Next the month text height is set as either the text size, or 0 dependent on whether month text should be drawn.

The first draw call is to draw the background behind the image. This call draws the background colour behind everything but the space for the month text.

The next draw call is dependent on whether there are contributions to be drawn. The contributions are drawn based on the day of the week, which is calculated from the value stored in each `ContributionsDay` and the `Calendar`.

The y position is the day of the week multiplied by the total height of a rectangle plus the margin for drawing the month text.

After computing the initial y position, the `onDraw` method uses a for-each loop to iterate through each `ContributionsDay`, setting the colour of `mDayPainter`, drawing the rectangle, and calculating the day of the week.

If the day of the week is 6, the week has ended; the x position is incremented by the width of rectangle and the y value is reset to the month text margin.

Otherwise, the y value is incremented by the height of a rectangle.

If there are no contributions to draw `mDayPainter` is set to the default background colour, and 364 rectangles are drawn to fill the 52 weeks with full 7 day weeks.

The final set of draw calls occur if `mShouldDisplayMonths` is true.

In this case `mTextPainter` is setup with `mTextColor` and the month text height, and the calendar is initialised to the current time, before subtracting 12 months to return to the start of the year.

For each month in the year, the three letter month string is collected from the `Calendar` instance and the `mTextBounds` rectangle is used to store the measured bounds of the text,

ensuring that it doesn't extend past the end of the canvas.

After each draw call the calendar month is incremented, and the x position is incremented by one twelfth of the canvas width.

The final call in `onDraw` is to set the `LayoutParams` of the `ContributionsView` to ensure that the canvas is drawn across the full width available.

## Contributions statistics

Returning to the `UserInfoFragment` we have the callback for `contributionsLoaded`. This method computes numerous statistics about the user:

- Their total number of contributions
- Their average number of contributions per day
- Their average number of contributions per active day
- Their greatest number of contributions per day
- Their longest uninterrupted 'streak' of active days

## `UserInfoFragment.java`

```
public void contributionsLoaded(List<ContributionsLoader.ContributionsDay>
contributions) {
 if(!areViewsValid()) return;
 int totalContributions = 0;
 int daysActive = 0;
 int maxContributions = 0;
 int streak = 0;
 int maxStreak = 0;
 for(ContributionsLoader.ContributionsDay gd : contributions) {
 if(gd.contributions > 0) {
 totalContributions += gd.contributions;
 daysActive += 1;
 if(gd.contributions > maxContributions) {
 maxContributions = gd.contributions;
 }
 streak += 1;
 if(streak > maxStreak) {
 maxStreak = streak;
 }
 } else {
 streak = 0;
 }
 }
 if(totalContributions > 0) {
 final String info = getResources()
 .getQuantityString(R.plurals.text_user_contributions,
totalContributions, totalContributions) +
 "\n" +
 String.format(getString(R.string.text_user_average),
 (float) totalContributions / contributions.size())
 } +
 "\n" +
```

```

String.format(getString(R.string.text_user_average_active),
 ((float) totalContributions / daysActive)
) +
"\n" +

String.format(getString(R.string.text_user_max_contributions),
maxContributions) +
"\n" +
String.format(getString(R.string.text_user_streak),
maxStreak);
final boolean isEmpty =
mContributionsInfo.getText().toString().isEmpty();
mContributionsInfo.setText(info);
if(isEmpty) {
 ObjectAnimator.ofInt(
 mContributionsInfo,
 "maxLines",
 0,
 mContributionsInfo.getLineCount()
).setDuration(200).start();
}
} else {

mContributionsInfo.setText(getString(R.string.text_user_no_contributions));
}
}

```

5 counters are used for the computation.

Each iteration of the loop checks if any contributions have been made.

If any contributions have been made, the total number of contributions is incremented by the value, the number of active days is incremented, the number of contributions on that day

is checked against the current maximum number of contributions, and streak counter is incremented before being checked against the current maxStreak value.

If no contributions have been made, the streak counter is reset to 0.

Once the computations have been made, the total number of contributions is made.

If it is 0, the “No contributions” string resource is displayed.

Otherwise, a string is built from 5 different format strings to display the information.

Before setting the text of mContributionsInfo, a check is performed to see if it already contains any text.

If the TextView is empty, then it will have 0 height (other than its margin), and setting its text would cause both it and its parent CardView to jump in size.

Rather than allowing this, an ObjectAnimator is used to increment the maxLines count of the TextView from 0 to the required number over a period of 200 milliseconds.

This completes objectives 2.a.iv and 2.a.v

## Displaying user information

The level of information which a user provides is not constant. While some provide information about their location, company, email and bio, others provide no information.

The `displayUser` method in `Formatter` is used in both the `LoginActivity` and `UserInfoLayout` to bind data to the `Views` in an inflated `shared_user_info` layout.

### Formatter.java

```
public static void displayUser(ViewGroup userInfoParent, User user) {
 userInfoParent.setVisibility(View.VISIBLE);

 final NetworkImageView avatar = ButterKnife.findById(userInfoParent,
R.id.user_avatar);
 avatar.setImageUrl(user.getAvatarUrl());
 final TextView login = ButterKnife.findById(userInfoParent,
R.id.user_login);
 login.setText(user.getLogin());

 final Context context = userInfoParent.getContext();
 final LinearLayout infoList = ButterKnife.findById(userInfoParent,
R.id.user_info_layout);
 infoList.removeAllViews();
 TextView tv;
 final LinearLayout.LayoutParams params = new
LinearLayout.LayoutParams(
 ViewGroup.LayoutParams.MATCH_PARENT,
 ViewGroup.LayoutParams.WRAP_CONTENT
);
 params.setMargins(0, UI.pxFromDp(4), 0, UI.pxFromDp(4));
 if(user.getName() != null) {
 tv = getInfoTextView(context, R.drawable.ic_person);
 tv.setText(user.getName());
 infoList.addView(tv, params);
 }
 tv = getInfoTextView(context, R.drawable.ic_date);
 tv.setText(
 String.format(
 context.getString(R.string.text_user_created_at),
 Util.formatDateLocally(
 context,
 new Date(user.getCreatedAt())
)
)
);
 infoList.addView(tv, params);
 if(user.getEmail() != null) {
 tv = getInfoTextView(context, R.drawable.ic_email);
 tv.setAutoLinkMask(Linkify.EMAIL_ADDRESSES);
 tv.setText(user.getEmail());
 infoList.addView(tv, params);
 }
 if(user.getBlog() != null) {
```

```

 tv = getInfoTextView(context, R.drawable.ic_blog);
 tv.setAutoLinkMask(Linkify.WEB_URLS);
 tv.setText(user.getBlog());
 infoList.addView(tv, params);
 }
 if(user.getCompany() != null) {
 tv = getInfoTextView(context, R.drawable.ic_company);
 tv.setText(user.getCompany());
 infoList.addView(tv, params);
 }
 if(user.getLocation() != null) {
 tv = getInfoTextView(context, R.drawable.ic_location);
 tv.setText(user.getLocation());
 infoList.addView(tv, params);
 }
 if(user.getRepos() > 0) {
 tv = getInfoTextView(context, R.drawable.ic_repo);
 tv.setText(context.getResources().getQuantityString(
 R.plurals.text_user_repositories,
 user.getRepos(),
 user.getRepos()
));
 infoList.addView(tv, params);
 }
 if(user.getGists() > 0) {
 tv = getInfoTextView(context, R.drawable.ic_gist);
 tv.setText(context.getResources().getQuantityString(
 R.plurals.text_user_gists,
 user.getGists(),
 user.getGists()
));
 infoList.addView(tv, params);
 }
 if(user.getBio() != null) {
 tv = getInfoTextView(context, R.drawable.ic_bio);
 tv.setText(user.getBio());
 infoList.addView(tv, params);
 }
 UI.expand(infoList);
}

```

The method first finds the `NetworkImageView` to display the user's avatar, and the `TextView` to display their username.

Once the username and avatar URL have been bound, a `LayoutParams` instance is created to ensure that each `TextView` uses the same margins.

The `getInfoTextView` method takes the context required to instantiate a `View` and a drawable resource id to display at the start of the `TextView`.

## Formatter.java

```

private static TextView getInfoTextView(Context context, @DrawableRes int
drawableRes) {
 final TextView tv = new TextView(context);
 tv.setCompoundDrawablePadding(UI.pxFromDp(4));
 tv.setCompoundDrawablesRelativeWithIntrinsicBounds(drawableRes, 0, 0,
0);
 return tv;
}

```

`displayUser` checks each value string value to be displayed, and if it is non null, the string is added to its own row with a corresponding icon.

Numeric values greater than 0 are also displayed, however they require extra formatting.

Correct grammar is achieved using plural strings, string resources with multiple values for different quantities.

Once each `TextView` row has been added to the `LinearLayout`, the `LinearLayout` is expanded with the `UI expand` method.

This completes objectives 2.a.i, 2.a.ii, and 2.a.iii

## Following and unfollowing users

Objective 2.g requires the implementation of following and unfollowing users. This can be implemented as a `Button` below their information, displaying either “follow” if the user is not currently followed, or “unfollow” if they are currently followed.

The `Loader` method to check whether a user is followed requires an `ItemLoader<Boolean>` and the user’s login, while the `Editor` methods to follow or unfollow a user

require an `UpdateListener<Boolean>`, as well as the user’s login.

The `Button` should only be shown if the user being displayed is not the authenticated user.

This check is performed in the `userLoaded` method:

```
if(!GitHubSession.getSession(getContext()).getUserLogin().equals(user.getLogin()))
{
 new Loader(getContext()).checkIfFollowing(this, user.getLogin());
}
```

The `updated` method is used for binding the following information, and `loadComplete` passes its return value through to `updated`.

### **UserInfoFragment.java**

```
public void updated(Boolean isFollowing) {
 if(mFollowButton == null) {
 mFollowButton = new Button(getContext());
 mFollowButton.setBackground(null);
 mFollowButton.setLayoutParams(
 new
 LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
 ViewGroup.LayoutParams.WRAP_CONTENT
));
 mUserInfoParent.addView(mFollowButton);
 }

 if(isFollowing) {
 mFollowButton.setText(R.string.text_unfollow_user);
 } else {
 mFollowButton.setText(R.string.text_follow_user);
 }
 mFollowButton.setOnClickListener(v -> {
 mFollowButton.setEnabled(false);
```

```

 mRefresher.setRefreshing(true);
 if(isFollowing) {

Editor.getEditor(getContext()).unfollowUser(UserInfoFragment.this,
mUser.getLogin());
 } else {

Editor.getEditor(getContext()).followUser(UserInfoFragment.this,
mUser.getLogin());
 }
});

mFollowButton.setEnabled(true);
mRefresher.setRefreshing(false);
}

```

updated first checks if the Button has ben created, creating it if not. It then sets the appropriate resource string for the button and adds an onClickListener.

The onClickListener disables the button, to prevent multiple requests, starts theSwipeRefreshLayout to indicate loading, and then calls the Editor to follow or unfollow the user.

Finally, the Button is enabled, and the SwipeRefreshLayout is stopped. The UserInfoFragment completes all objectives in 2.a

## UserReposFragment

The UserReposFragment is used to fullfill objective 2.b, displaying the repositories which a user has contributed to.

As multiple different Fragments contain only a SwipeRefreshLayout and a RecyclerView there is no need to create an individual layout file for each of them.

Instead they can each use the same layout file:

### **fragment\_recycler.xml**

```

<?xml version="1.0" encoding="utf-8"?>

<android.support.v4.widget.SwipeRefreshLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 android:id="@+id/fragment_refresher"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 app:layout_behavior="@string/appbar_scrolling_view_behavior">

 <com(tpb).animatingrecyclerview.AnimatingRecyclerView
 android:id="@+id/fragment_recycler"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:layout_margin="8dp"/>

</android.support.v4.widget.SwipeRefreshLayout>

```

The UserReposFragment contains very little logic, as it only has to load the User and then pass this User to the RecyclerView adapter which contains logic for loading the user's repositories and binding them to a list of views.

### UserReposFragment.java

```
package com(tpb.projects.user.fragments;

import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com(tpb.animatingrecyclerview.AnimatingRecyclerView;
import com(tpb.github.data.models.User;
import com(tpb.projects.R;
import com(tpb.projects.common.FixedLinearLayoutManger;
import com(tpb.projects.common.RepositoriesAdapter;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.Unbinder;

/**
 * Created by theo on 10/03/17.
 */

public class UserReposFragment extends UserFragment {

 private Unbinder unbinder;

 @BindView(R.id.fragment_recycler) AnimatingRecyclerView mRecycler;
 @BindView(R.id.fragment_refresher) SwipeRefreshLayout mRefresher;
 private RepositoriesAdapter mAdapter;

 @Nullable
 @Override
 public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
 final View view = inflater.inflate(R.layout.fragment_recycler,
 container, false);
 unbinder = ButterKnife.bind(this, view);

 final LinearLayoutManager manager = new
FixedLinearLayoutManger(getApplicationContext());
 mRecycler.setLayoutManager(manager);
 mRecycler.enableLineDecoration();
 mAdapter = new RepositoriesAdapter(getActivity(), mRefresher);
 mRecycler.setAdapter(mAdapter);

 mRecycler.setOnScrollListener(new RecyclerView.OnScrollListener() {
 @Override
 public void onScrolled(RecyclerView recyclerView, int dx, int dy)
 {
```

```

 super.onScrolled(recyclerView, dx, dy);
 if(manager.findFirstVisibleItemPosition() + 20 >
manager.getItemCount()) {
 mAdapter.notifyBottomReached();
 }
 });
 mAreViewsValid = true;
 if(mUser != null) userLoaded(mUser);
 return view;
}

@Override
public void userLoaded(User user) {
 mUser = user;
 if(!areViewsValid()) return;
 mAdapter.setUser(user.getLogin(), false);
}

@Override
public void onDestroyView() {
 super.onDestroyView();
 unbinder.unbind();
}
}
}

```

Following the same View binding structure as other Fragments the UserRepoFragment inflates its layout in onCreateView. One function of the UserReposFragment is to listen for the RecyclerView scrolling. The GitHub API is paginated, returning a maximum of 30 items per request. In order to load further items a page number must be specified in the request. The RepositoriesAdapter tracks its page number when loading Repositories and is notified that its RecyclerView is approaching the bottom of its content by the UserReposFragment OnScrollListener. The second function of the UserReposFragment is to launch the RepoActivity for displaying a repository when an item in the RecyclerView is clicked. openRepo creates the Intent to launch the RepoActivity, begins preloading the data which will be used in RepoActvitiy, and then launches RepoActivity.

## RepositoriesAdapter

The RepositoriesAdapter is used in two places, displaying the repositories that a user contributes to and displaying the repositories that a user has starred. When displaying a user's repositories, the RepositoriesAdapter must also support pinning repositories (Objective [2.b.vi](#)).

### RepositoriesAdapter.java

```

package com(tpb.projects.common;

import android.annotation.SuppressLint;
import android.app.Activity;
import android.content.Context;

```

```
import android.content.Intent;
import android.content.SharedPreferences;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.widget.RecyclerView;
import android.text.format.DateUtils;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Loader;
import com(tpb.github.data.auth.GitHubSession;
import com(tpb.github.data.models.Repository;
import com(tpb.github.data.models.State;
import com(tpb.mdtext.Markdown;
import com(tpb.mdtext.views.MarkdownTextView;
import com(tpb.projects.R;
import com(tpb.projects.flow.IntentHandler;
import com(tpb.projects.repo.RepoActivity;
import com(tpb.projects.util.Util;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by theo on 14/12/16.
 */

public class RepositoriesAdapter extends
RecyclerView.Adapter<RepositoriesAdapter.RepoHolder> implements
Loader.ListLoader<Repository> {

 private final Loader mLoader;
 private final SwipeRefreshLayout mRefresher;
 private final ArrayList<Repository> mRepos = new ArrayList<>();
 private final String mAuthenticatedUser;
 private String mUser;
 private final RepoPinChecker mPinChecker;
 private final Activity mActivity;
 private int mPage = 1;
 private boolean mIsLoading = false;
 private boolean mMaxPageReached = false;

 private boolean mShowingStars = false;

 public RepositoriesAdapter(Activity activity, SwipeRefreshLayout
refresher) {
 mActivity = activity;
 mLoader = Loader.getLoader(activity);
 mRefresher = refresher;
 mRefresher.setRefreshing(true);
 mRefresher.setOnRefreshListener(() -> {
 mPage = 1;
 mMaxPageReached = false;
 });
 }

 @Override
 public void onBindViewHolder(RepoHolder holder, int position) {
 Repository repo = mRepos.get(position);
 holder.bind(repo);
 }

 @Override
 public void onCreateViewHolder(ViewGroup parent, int viewType) {
 View view = LayoutInflater.from(parent.getContext())
.inflate(R.layout.item_repository, parent, false);
 return new RepoHolder(view);
 }

 @Override
 public int getItemCount() {
 return mRepos.size();
 }

 @Override
 public void onLoadMore(int page, int totalItems, int itemsCount) {
 if (!mIsLoading && !mMaxPageReached) {
 mIsLoading = true;
 mLoader.loadMore();
 }
 }

 @Override
 public void onLoading(int progress) {
 mRefresher.setRefreshing(false);
 mRefresher.setProgress(progress);
 }

 @Override
 public void onFailure(Throwable t) {
 mRefresher.setRefreshing(false);
 mRefresher.setError(true);
 }

 @Override
 public void onCompleted() {
 mRefresher.setRefreshing(false);
 mRefresher.setError(false);
 }

 class RepoHolder extends RecyclerView.ViewHolder {
 TextView title;
 TextView description;
 TextView stars;
 TextView forks;
 TextView updated;
 TextView language;
 TextView owner;
 TextView commit;
 TextView pullRequest;
 TextView issue;
 TextView repository;
 TextView star;
 TextView fork;
 TextView refresh;
 TextView refreshIcon;

 RepoHolder(View itemView) {
 super(itemView);
 title = itemView.findViewById(R.id.title);
 description = itemView.findViewById(R.id.description);
 stars = itemView.findViewById(R.id.stars);
 forks = itemView.findViewById(R.id.forks);
 updated = itemView.findViewById(R.id.updated);
 language = itemView.findViewById(R.id.language);
 owner = itemView.findViewById(R.id.owner);
 commit = itemView.findViewById(R.id.commit);
 pullRequest = itemView.findViewById(R.id.pullRequest);
 issue = itemView.findViewById(R.id.issue);
 repository = itemView.findViewById(R.id.repository);
 star = itemView.findViewById(R.id.star);
 fork = itemView.findViewById(R.id.fork);
 refresh = itemView.findViewById(R.id.refresh);
 refreshIcon = itemView.findViewById(R.id.refreshIcon);
 }

 void bind(Repository repo) {
 title.setText(repo.getName());
 description.setText(repo.getDescription());
 stars.setText(repo.getStarCount());
 forks.setText(repo.getForkCount());
 updated.setText(DateUtils.getRelativeTimeSpanString(repo.getUpdatedAt().getTime()));
 language.setText(repo.getLanguage());
 owner.setText(repo.getOwner().getLogin());
 commit.setText(repo.getCommitCount());
 pullRequest.setText(repo.getPullRequestCount());
 issue.setText(repo.getIssueCount());
 repository.setText(repo.getRepositoryCount());
 star.setOnClickListener(v -> star(repo));
 fork.setOnClickListener(v -> fork(repo));
 refresh.setOnClickListener(v -> refresh(repo));
 }

 void star(Repository repo) {
 if (mAuthenticatedUser != null) {
 APIHandler.getStar(repo, mAuthenticatedUser);
 } else {
 IntentHandler.showLogin(mActivity);
 }
 }

 void fork(Repository repo) {
 if (mAuthenticatedUser != null) {
 APIHandler.getFork(repo, mAuthenticatedUser);
 } else {
 IntentHandler.showLogin(mActivity);
 }
 }

 void refresh(Repository repo) {
 mRefresher.setRefreshing(true);
 mLoader.loadMore();
 }
 }
}
```

```

 final int oldSize = mRepos.size();
 mRepos.clear();
 notifyItemRangeRemoved(0, oldSize);
 loadReposForUser(true);
 });
 mPinChecker = new RepoPinChecker(activity);
 mAuthenticatedUser =
GitHubSession.getSession(activity).getUserLogin();
}

public void setUser(String user, boolean isShowingStars) {
 mUser = user;
 mIsShowingStars = isShowingStars;
 mRepos.clear();
 loadReposForUser(true);
}

public void notifyBottomReached() {
 if(!mIsLoading && !mMaxPageReached) {
 mPage++;
 loadReposForUser(false);
 }
}

private void loadReposForUser(boolean resetPage) {
 mIsLoading = true;
 mRefresher.setRefreshing(true);
 if(resetPage) {
 mPage = 1;
 mMaxPageReached = false;
 }
 if(mIsShowingStars) {
 mLoader.loadStarredRepositories(this, mUser, mPage);
 } else if(mUser.equals(mAuthenticatedUser)) { //The session user
 mLoader.loadRepositories(this, mPage);
 } else {
 mLoader.loadRepositories(this, mUser, mPage);
 }
 mPinChecker.setKey(mUser);
}

@Override
public RepoHolder onCreateViewHolder(ViewGroup parent, int viewType) {
 return new RepoHolder(LayoutInflater.from(parent.getContext())
 .inflate(R.layout.viewholder_repo,
parent, false));
}

@SuppressWarnings("SetTextI18n")
@Override
public void onBindViewHolder(RepoHolder holder, int position) {
 final int pos = holder.getAdapterPosition();
 final Repository r = mRepos.get(pos);
 holder.mName.setText(
 (r.getUserLogin().equals(mUser) ? r.getName() :
r.getFullName()) + (r
 .isFork() ? " (Forked) " : ""))
}

```

```

 if(Util.isNotNullOrEmpty(r.getLanguage())) {
 holder.mLanguage.setVisibility(View.VISIBLE);
 holder.mLanguage.setText(r.getLanguage());
 } else {
 holder.mLanguage.setVisibility(View.GONE);
 }
 if(Util.isNotNullOrEmpty(r.getDescription())) {
 holder.mDescription.setVisibility(View.VISIBLE);

holder.mDescription.setMarkdown(Markdown.formatMD(r.getDescription(),
r.getFullName()));
 } else {
 holder.mDescription.setVisibility(View.GONE);
 }
 if(mIsShowingStars) {
 holder.mImage.resetImage();
 holder.mImage.setImageUrl(r.getUserAvatarUrl());
 IntentHandler.addOnClickHandler(mActivity, holder.mImage,
r.getUserLogin());
 } else {
 final boolean isPinned = mPinChecker.isPinned(r.getFullName());
 holder.isPinned = isPinned;
 holder.mImage
 .setImageResource(isPinned ? R.drawable.ic_pinned :
R.drawable.ic_not_pinned);
 }
 holder.mForks.setText(Integer.toString(r.getForks()));
 holder.mStars.setText(Integer.toString(r.getStarGazers()));

holder.mLastUpdated.setText(DateUtils.getRelativeTimeSpanString(r.getUpdatedAt()));
 }

private void togglePin(int pos) {
 final Repository r = mRepos.get(pos);
 if(mPinChecker.isPinned(r.getFullName())) {
 mRepos.remove(pos);
 final int newPos = mPinChecker.initialPosition(r.getFullName());
 mRepos.add(newPos, r);
 mPinChecker.unpin(r.getFullName());
 notifyItemMoved(pos, newPos);
 } else {
 mRepos.remove(pos);
 mRepos.add(0, r);
 mPinChecker.pin(r.getFullName());
 notifyItemMoved(pos, 0);
 }
}

@Override
public int getItemCount() {
 return mRepos.size();
}

@Override
public void listLoadComplete(List<Repository> repos) {
 mRefresher.setRefreshing(false);
 mIsLoading = false;
}

```

```

 if(!repos.isEmpty()) {
 final int oldLength = mRepos.size();
 if(mIsShowingStars) {
 mRepos.addAll(repos);
 } else {
 insertPinnedRepos(repos);
 }
 notifyItemRangeInserted(oldLength, mRepos.size());
 } else {
 mMaxPageReached = true;
 }
 }

private void insertPinnedRepos(List<Repository> repos) {
 if(mPage == 1) {
 for(Repository r : repos) {
 if(mPinChecker.isPinned(r.getFullName())) {
 mRepos.add(0, r);
 } else {
 mRepos.add(r);
 }
 }
 mPinChecker.setInitialPositions(mRepos);
 ensureLoadOfPinnedRepos();
 } else {
 for(Repository repo : repos) {
 if(!mRepos.contains(repo)) mRepos.add(repo);
 }
 mPinChecker.appendInitialPositions(repos);
 }
}

@Override
public void listLoadError(APIHandler.APIError error) {
 mIsLoading = false;
 mRefresher.setRefreshing(false);
}

private void ensureLoadOfPinnedRepos() {
 for(String repo : mPinChecker.findNonLoadedPinnedRepositories()) {
 mLoader.loadRepository(new Loader.ItemLoader<Repository>() {
 @Override
 public void loadComplete(Repository data) {
 if(!mRepos.contains(data)) {
 mRepos.add(0, data);
 mPinChecker.appendPinnedPosition(data.getFullName());
 notifyItemInserted(0);
 }
 }
 });

 @Override
 public void loadError(APIHandler.APIError error) {
 }
 }, repo);
 }
}

```

```

private void openItem(int pos) {
 final Repository repo = mRepos.get(pos);
 final Intent i = new Intent(mActivity, RepoActivity.class);
 i.putExtra(mActivity.getString(R.string.intent_repo), repo);
 Loader.getLoader(mActivity)
 .loadProjects(null, repo.getFullName())
 .loadIssues(null, repo.getFullName(), State.OPEN, null, null, 0)
 .loadProjects(null, repo.getFullName());
 mActivity.startActivity(i);
 mActivity.overridePendingTransition(R.anim.slide_up, R.anim.none);
}

class RepoHolder extends RecyclerView.ViewHolder {
 @BindView(R.id.repo_name) TextView mName;
 @BindView(R.id.repo_description) MarkdownTextView mDescription;
 @BindView(R.id.repo_forks) TextView mForks;
 @BindView(R.id.repo_stars) TextView mStars;
 @BindView(R.id.repo_language) TextView mLanguage;
 @BindView(R.id.repo_last_updated) TextView mLastUpdated;
 @BindView(R.id.repo_icon) NetworkImageView mImage;
 private boolean isPinned = false;

 RepoHolder(View view) {
 super(view);
 ButterKnife.bind(this, view);
 view.setOnClickListener(v -> openItem(getAdapterPosition()));
 mDescription.setOnClickListener(v ->
openItem(getAdapterPosition()));
 if(!mIsShowingStars) {
 mImage.setOnClickListener((v) -> {
 togglePin(getAdapterPosition());
 isPinned = !isPinned;
 mImage.setImageResource(
 isPinned ? R.drawable.ic_pinned :
R.drawable.ic_not_pinned);
 });
 }
 }
}

private class RepoPinChecker {

 private final SharedPreferences prefs;
 private static final String PREFS_KEY = "PINS";
 private String KEY;
 private final ArrayList<String> pins = new ArrayList<>();
 private final ArrayList<String> mInitialPositions = new ArrayList<>();

 RepoPinChecker(Context context) {
 prefs = context.getSharedPreferences(PREFS_KEY,
Context.MODE_PRIVATE);
 }

 void setKey(String key) {
 KEY = key;
 final String[] saved Pins =
Util.stringArrayFromPrefs(prefs.getString(KEY, ""));
 pins.clear();
 }
}

```

```

 pins.addAll(Arrays.asList(saved Pins));
 }

 void pin(String path) {
 if(!pins.contains(path)) {
 pins.add(path);
 prefs.edit().putString(KEY,
Util.stringArrayForPrefs(pins)).apply();
 }
 }

 void unpin(String path) {
 pins.remove(path);
 prefs.edit().putString(KEY,
Util.stringArrayForPrefs(pins)).apply();
 }

 void setInitialPositions(List<Repository> repos) {
 mInitialPositions.clear();
 for(Repository r : repos) mInitialPositions.add(r.getFullName());
 }

 void appendInitialPositions(List<Repository> repos) {
 for(Repository r : repos) mInitialPositions.add(r.getFullName());
 }

 void appendPinnedPosition(String key) {
 mInitialPositions.add(0, key);
 }

 int initialPosition(String key) {
 return mInitialPositions.indexOf(key);
 }

 boolean isPinned(String path) {
 return pins.contains(path);
 }

 List<String> findNonLoadedPinnedRepositories() {
 final List<String> repos = new ArrayList<>();
 for(String pin : pins) {
 if(!mInitialPositions.contains(pin)) repos.add(pin);
 }
 return repos;
 }

}

```

The RepositoriesAdapter is constructed with a parent Activity, used for launching the RepoActivity when a repository is clicked or a user when a user icon is clicked, and a SwipeRefreshLayout which is used to refresh the RecyclerView. The Loader is created, and the SwipeRefreshLayout is set to refreshing. The OnRefreshListener is set on the SwipeRefreshLayout to reset the adapter conditions when it is refreshed.

Finally, the RepoPinChecker for sorting Repositories by their pin status is created, and the authencitaed user login is loaded from GitHubSession.

## States

The adapter will begin loading Repositories once setUser is called.

This method sets the mUser login string, the mIsShowingStars flag, and then clears the current list of Repositories before calling loadReposForUser with the flag to reset the page.

When loadReposForUser is called, the mIsLoading flag is set to true, and the SwipeRefreshLayout is set to refreshing.

If resetPage is true, the mPage value is reset to 1 and mMaxPageReached is reset to false.

Next, there are three network calls which may be made:

1. loadStarredRepositories if mIsShowingStars is true.
2. loadRepositories without a user parameter if the current user login is the same as the authenticated user.
3. loadRepositories with a user parameter otherwise.

Finally, the RepoPinChecker key is set to the current user login.

When the Repositories are loaded, they are returned through the ListLoader interface method listLoadComplete.

The SwipeRefreshLayout is disabled, and mIsLoading is set to false.

If the list of Repositories is not empty, the old length is stored.

If mIsShowingStars is true, all of the new Repositories are added.

Otherwise insertPinnedRepos is called which uses the RepoPinChecker isPinned method when iterating through each Repository and if it is pinned, adds it to the start of the list.

Finally, notifyItemRangeInserted is called.

Otherwise, mMaxPageReached is set to true.

## insertPinnedRepos

insertPinnedRepos serves two purposes, first ensuring that the pinned Repositories which have been loaded are added to the start of the adapter, and second loading any pinned repositories which are not loaded in the first page of the the Repositories returned by GitHub.

If the page is 1, each Repository is added either to the start or end of the array, and these initial positions are then passed to

the RepoPinChecker before ensureLoadOfPinnedRepos is called.

Otherwise, the Repositories are added to the end of the array if they do not already exist in the array.

ensureLoadOfPinnedRepos iterates through each of the repository names returned by RepoPinChecker.findNonLoadedPinnedRepositories (That's a bit of a mouthful), and loads each of the Repositories individually, inserting them into the array and RepoPinChecker if they do not already exist there, and calling notifyItemInserted(0).

### **Objective 2.b.vi: RepoPinChecker**

The RepoPinChecker which has been reference above is a class which controls a SharedPreferences instance and manages the pinned repositories.

Each time a user is loaded in RepositoriesAdapter the SharedPreferences is opened with the "PINS" key, and from this map a delimited string is loaded using the user login as a key.

This allows different repositories to be pinned for each user.

The RepoPinChecker contains two ArrayLists, one containing the names of the pinned repositories, and the other containing the names of the repositories loaded in their initial order, allowing them to be returned to these positions if they are unpinned.

When setKey is called, the KEY is st, and a string array is loaded with util.stringArrayFromPrefs which loads the string and splits it around each comma.

When pin is called, if the repository is not already pinned, the repository name is added to the pins list and the new string from util.stringArrayForPrefs is written to the SharedPreferences.

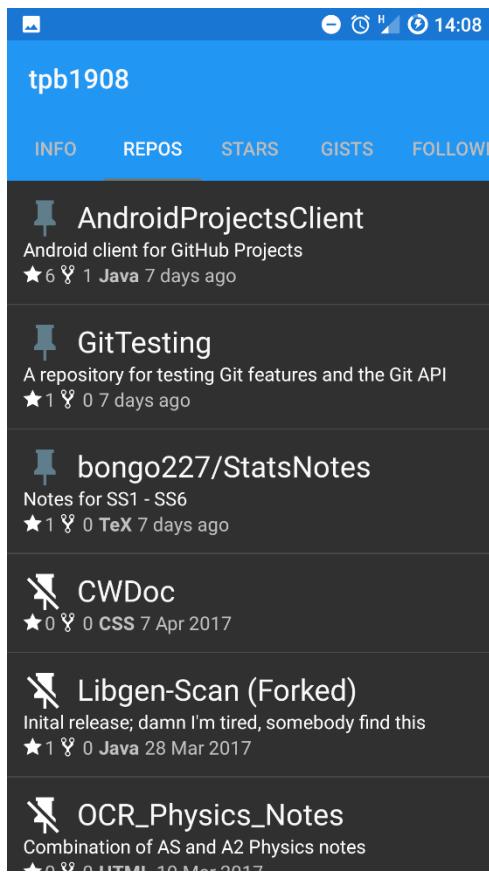
When unpin is called, the repository is removed from the pins list, and the new string is again written to the SharedPreferences.

findNonLoadedPinnedRepositories checks each value in pins and if it is not in mInitialPositions, it is added to the list of non loaded pinned repositories which is then returned.

### **Binding**

The RepoHolder sets OnClickListeners on its elements, and if the mIsShowingStars flag is false, it sets an OnClickListener on the NetworkImageView which would otherwise be used for displaying the user avatar. This listener calls togglePin, flips the isPinned flag and switches the image resource from the pinned to not pinned drawable.

The Fragment is shown in the screenshot below:



## Objective 2.c: UserStarsFragment

The `UserStarsFragment` is very simple as all of the view logic is handled in the `RepositoriesAdapter`.

The Fragment is only responsible for creating the layout, handling its own lifecycle, and notifying the adapter of scroll changes in the `RecyclerView`.

### `UserStarsFragment.java`

```
package com(tpb.projects.user.fragments;

import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com(tpb.animatingrecyclerview.AnimatingRecyclerView;
import com(tpb.github.data.models.User;
import com(tpb.projects.R;
import com(tpb.projects.common.FixedLinearLayoutManger;
import com(tpb.projects.common.RepositoriesAdapter;

import butterknife.BindView;
import butterknife.ButterKnife;
```

```
import butterknife.Unbinder;

/**
 * Created by theo on 10/03/17.
 */

public class UserStarsFragment extends UserFragment {

 private Unbinder unbinder;

 @BindView(R.id.fragment_recycler) AnimatingRecyclerView mRecycler;
 @BindView(R.id.fragment_refresher) SwipeRefreshLayout mRefresher;
 private RepositoriesAdapter mAdapter;

 @Nullable
 @Override
 public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
 final View view = inflater.inflate(R.layout.fragment_recycler,
 container, false);
 unbinder = ButterKnife.bind(this, view);

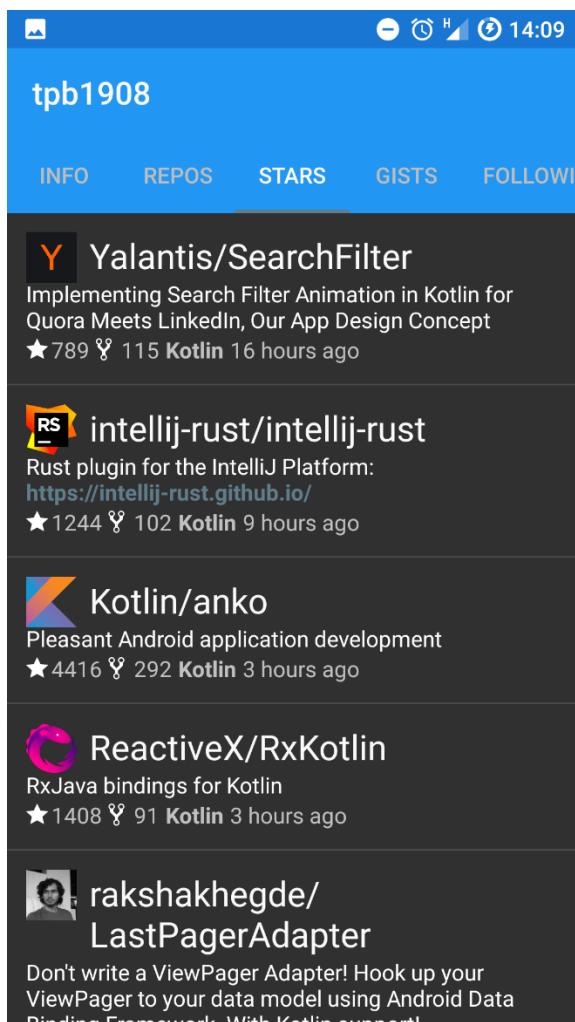
 final LinearLayoutManager manager = new
 FixedLinearLayoutManger(getApplicationContext());
 mRecycler.setLayoutManager(manager);
 mRecycler.enableLineDecoration();
 mAdapter = new RepositoriesAdapter(getActivity(), mRefresher);
 mRecycler.setAdapter(mAdapter);

 mRecycler.setOnScrollListener(new RecyclerView.OnScrollListener() {
 @Override
 public void onScrolled(RecyclerView recyclerView, int dx, int dy)
{
 super.onScrolled(recyclerView, dx, dy);
 if(manager.findFirstVisibleItemPosition() + 20 >
manager.getItemCount()) {
 mAdapter.notifyBottomReached();
 }
 });
 mAreViewsValid = true;
 if(mUser != null) userLoaded(mUser);
 return view;
 }

 @Override
 public void userLoaded(User user) {
 mUser = user;
 if(!areViewsValid()) return;
 mAdapter.setUser(user.getLogin(), true);
 }

 @Override
 public void onDestroyView() {
 super.onDestroyView();
 unbinder.unbind();
 }
}
```

The Fragment is shown in the screenshot below:



## Objective 2.d: UserGistsFragment

The UserGistsFragment is also very simple as it only deals with notifying the GistAdapter of scroll changes, and opening the FileActivity when a gist is clicked.

### UserGistsFragment.java

```
package com(tpb.projects.user.fragments;

import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com(tpb.animatingrecyclerview.AnimatingRecyclerView;
import com(tpb.github.data.models.Gist;
import com(tpb.github.data.models.User;
import com(tpb.projects.R;
import com(tpb.projects.common.FixedLinearLayoutManger;
import com(tpb.projects.repo.content.FileActivity;
import com(tpb.projects.user.GistAdapter;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.Unbinder;

/**
 * Created by theo on 10/03/17.
 */

public class UserGistsFragment extends UserFragment implements
GistAdapter.GistOpener {

 private Unbinder unbinder;

 @BindView(R.id.fragment_refresher) SwipeRefreshLayout mRefresher;
 @BindView(R.id.fragment_recycler) AnimatingRecyclerView mRecycler;

 private GistAdapter mAdapter;

 @Nullable
 @Override
 public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
 final View view = inflater.inflate(R.layout.fragment_recycler,
container, false);
 unbinder = ButterKnife.bind(this, view);

 final LinearLayoutManager manager = new
FixedLinearLayoutManger(getContext());
 mRecycler.setLayoutManager(manager);
 mRecycler.enableLineDecoration();
```

```

mAdapter = new GistAdapter(getContext(), this, mRefresher);
mRecycler.setAdapter(mAdapter);

mRecycler.setOnScrollListener(new RecyclerView.OnScrollListener() {
 @Override
 public void onScrolled(RecyclerView recyclerView, int dx, int dy)
{
 super.onScrolled(recyclerView, dx, dy);
 if(manager.findFirstVisibleItemPosition() + 20 >
manager.getItemCount()) {
 mAdapter.notifyBottomReached();
 }
 }
});
mAreViewsValid = true;
if(mUser != null) userLoaded(mUser);
return view;
}

@Override
public void userLoaded(User user) {
 mUser = user;
 if(!areViewsValid()) return;
 mAdapter.setUser(user.getLogin(), false);
}

@Override
public void openGist(Gist gist, View view) {
 final Intent i = new Intent(getContext(), FileActivity.class);
 i.putExtra(getString(R.string.intent_gist_url),
gist.getFiles().get(0).getRawUrl());
 startActivity(i);
}

@Override
public void onDestroyView() {
 super.onDestroyView();
 unbinder.unbind();
}
}

```

## GistAdapter.java

```

package com(tpb.projects.user;

import android.content.Context;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Loader;
import com(tpb.github.data.auth.GitHubSession;
import com(tpb.github.data.models.Gist;
import com(tpb.projects.R;
import com(tpb.projects.common.NetworkImageView;

```

```
import com(tpb.projects.util.Util;

import java.util.ArrayList;
import java.util.List;

import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by theo on 11/03/17.
 */

public class GistAdapter extends RecyclerView.Adapter<GistAdapter.GistHolder>
implements Loader.ListLoader<Gist> {

 private String mUser;
 private String mAuthenticatedUser;
 private boolean mIsShowingPublic;

 private ArrayList<Gist> mGists = new ArrayList<>();

 private int mPage = 1;
 private boolean mIsLoading = false;
 private boolean mMaxPageReached = false;

 private SwipeRefreshLayout mRefresher;
 private Loader mLoader;
 private GistOpener mOpener;

 public GistAdapter(Context context, GistOpener opener, SwipeRefreshLayout refresher) {
 mLoader = Loader.getLoader(context);
 mOpener = opener;
 mRefresher = refresher;
 mRefresher.setRefreshing(true);
 mRefresher.setOnRefreshListener(() -> {
 mPage = 1;
 mMaxPageReached = false;
 final int oldSize = mGists.size();
 mGists.clear();
 notifyItemRangeRemoved(0, oldSize);
 loadGistsForUser(true);
 });
 mAuthenticatedUser = GitHubSession.getSession(context).getUserLogin();
 }

 public void setUser(String user, boolean isShowingPublic) {
 mUser = user;
 mIsShowingPublic = isShowingPublic;
 mGists.clear();
 loadGistsForUser(true);
 }

 public void notifyBottomReached() {
 if(!mIsLoading && !mMaxPageReached) {
 mPage++;
 loadGistsForUser(false);
 }
 }
}
```

```
private void loadGistsForUser(boolean resetPage) {
 mIsLoading = true;
 mRefresher.setRefreshing(true);
 if(resetPage) {
 mPage = 1;
 mMaxPageReached = false;
 }
 if(mIsShowingPublic) {
 mLoader.loadGists(this, mUser, mPage);
 } else if(mUser.equals(mAuthenticatedUser)) { //The session user
 mLoader.loadGists(this, mPage);
 } else {
 mLoader.loadGists(this, mUser, mPage);
 }
}

private void open(View view, int pos) {
 mOpener.openGist(mGists.get(pos), view);
}

@Override
public void listLoadComplete(List<Gist> gists) {
 mRefresher.setRefreshing(false);
 mIsLoading = false;
 if(gists.size() > 0) {
 int oldLength = mGists.size();
 if(mPage == 1) mGists.clear();
 mGists.addAll(gists);
 notifyItemRangeInserted(oldLength, mGists.size());
 } else {
 mMaxPageReached = true;
 }
}

@Override
public void listLoadError(APIHandler.APIError error) {
 mIsLoading = false;
}

@Override
public GistHolder onCreateViewHolder(ViewGroup parent, int viewType) {
 return new GistHolder(LayoutInflater.from(parent.getContext())
 .inflate(R.layout.viewholder_gist,
parent, false));
}

@Override
public void onBindViewHolder(GistHolder holder, int position) {
 final Gist g = mGists.get(position);
 if(mIsShowingPublic) {
 holder.mTitle.setText(
 String.format(
 holder.itemView.getResources()
.getString(R.string.text_gist_viewholder_title),
 g.getOwner().getLogin(),
 g.getFiles().get(0).getName()
)
);
 }
}
```

```

)
);
 holder.mAvatar.setImageUrl(g.getOwner().getAvatarUrl());
} else {
 holder.mTitle.setText(g.getFiles().get(0).getName());
 holder.mAvatar.setVisibility(View.GONE);
}
if(Util.isNotNullOrEmpty(g.getDescription())) {
 holder.mInfo.setVisibility(View.VISIBLE);
 holder.mInfo.setText(g.getDescription());
} else {
 holder.mInfo.setVisibility(View.GONE);
}
}

@Override
public int getItemCount() {
 return mGists.size();
}

class GistHolder extends RecyclerView.ViewHolder {

 @BindView(R.id.gist_title) TextView mTitle;
 @BindView(R.id.gist_info) TextView mInfo;
 @BindView(R.id.gist_user_avatar) NetworkImageView mAvatar;

 GistHolder(View itemView) {
 super(itemView);
 ButterKnife.bind(this, itemView);
 itemView.setOnClickListener(v -> open(itemView,
getAdapterPosition()));
 }

}

public interface GistOpener {
 void openGist(Gist gist, View view);
}
}

```

The `GistAdapter` implements `Loader.ListLoader<Gist>` and deals with loading and binding a list of a user's gists.

The adapter inflates the `viewholder_gist` layout which contains a `NetworkImageView` and two `TextViews`.

It then binds the `Gist` owner's avatar to the `NetworkImageView`, the `Gist` name to the title `TextView`, and the `Gist` description to the second `TextView` if the description exists.

When a `gist` list item is clicked, the `FileActivity` is launched to display the `gist` file.

## Objectives 2.e and 2.f: UserFollowingFragment and UserFollowersFragment

The two fragments display a list of users that the authenticated user is following or that are following the authenticated user respectively.

Each list item consists of the user's login and their avatar, as other information is not guaranteed to exist and is superfluous.

### UserFollowingFragment.java

```
package com(tpb.projects.user.fragments;

import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com(tpb.animatingrecyclerview.AnimatingRecyclerView;
import com(tpb.github.data.models.User;
import com(tpb.projects.R;
import com(tpb.projects.common.FixedLinearLayoutManger;
import com(tpb.projects.user.UserAdapter;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.Unbinder;

/**
 * Created by theo on 19/03/17.
 */

public class UserFollowingFragment extends UserFragment {

 private Unbinder unbinder;

 private UserAdapter mAdapter;
 @BindView(R.id.fragment_refresher) SwipeRefreshLayout mRefresher;
 @BindView(R.id.fragment_recycler) AnimatingRecyclerView mRecycler;

 @Nullable
 @Override
 public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
 final View view = inflater.inflate(R.layout.fragment_recycler,
 container, false);
 unbinder = ButterKnife.bind(this, view);
 final LinearLayoutManager manager = new
 FixedLinearLayoutManger(getApplicationContext());
 mRecycler.setLayoutManager(manager);
 mAdapter = new UserAdapter(getActivity(), mRefresher);
 mRecycler.enableLineDecoration();
 mRecycler.setAdapter(mAdapter);
```

```

 mRecycler.setOnScrollListener(new RecyclerView.OnScrollListener() {
 @Override
 public void onScrolled(RecyclerView recyclerView, int dx, int dy)
{
 super.onScrolled(recyclerView, dx, dy);
 if(manager.findFirstVisibleItemPosition() + 20 >
manager.getItemCount()) {
 mAdapter.notifyBottomReached();
 }
 }
 });
 mAreViewsValid = true;
 if(mUser != null) userLoaded(mUser);
 return view;
 }

 @Override
 public void userLoaded(User user) {
 mUser = user;
 if(!areViewsValid()) return;
 mAdapter.setUser(user.getLogin(), false);
 }

 @Override
 public void onDestroyView() {
 super.onDestroyView();
 unbinder.unbind();
 }
}

```

## UserFollowersFragment.java

```

package com(tpb.projects.user.fragments;

import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com(tpb.animatingrecyclerview.AnimatingRecyclerView;
import com(tpb.github.data.models.User;
import com(tpb.projects.R;
import com(tpb.projects.common.FixedLinearLayoutManger;
import com(tpb.projects.user.UserAdapter;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.Unbinder;

/**
 * Created by theo on 19/03/17.
 */

public class UserFollowersFragment extends UserFragment {

```

```

private Unbinder unbinder;

private UserAdapter mAdapter;
@BindView(R.id.fragment_refresher) SwipeRefreshLayout mRefresher;
@BindView(R.id.fragment_recycler) AnimatingRecyclerView mRecycler;

@Nullable
@Override
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
 final View view = inflater.inflate(R.layout.fragment_recycler,
 container, false);
 unbinder = ButterKnife.bind(this, view);
 final LinearLayoutManager manager = new
FixedLinearLayoutManger(getContext());
 mRecycler.setLayoutManager(manager);
 mAdapter = new UserAdapter(getActivity(), mRefresher);
 mRecycler.enableLineDecoration();
 mRecycler.setAdapter(mAdapter);
 mRecycler.setOnScrollListener(new RecyclerView.OnScrollListener() {
 @Override
 public void onScrolled(RecyclerView recyclerView, int dx, int dy)
{
 super.onScrolled(recyclerView, dx, dy);
 if(manager.findFirstVisibleItemPosition() + 20 >
manager.getItemCount()) {
 mAdapter.notifyBottomReached();
 }
 }
});
 mAreViewsValid = true;
 if(mUser != null) userLoaded(mUser);
 return view;
}

@Override
public void userLoaded(User user) {
 mUser = user;
 if(!areViewsValid()) return;
 mAdapter.setUser(user.getLogin(), true);
}

@Override
public void onDestroyView() {
 super.onDestroyView();
 unbinder.unbind();
}
}

```

They each inflate the `fragment_recycler` layout and when the `User` is loaded they pass it to the `UserAdapter` which deals with loading and bind data.

### **UserAdapter.java**

```

package com(tpb.projects.user;

import android.app.Activity;
import android.support.v4.widget.SwipeRefreshLayout;

```

```
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Loader;
import com(tpb.github.data.models.User;
import com(tpb.projects.R;
import com(tpb.projects.common.NetworkImageView;
import com(tpb.projects.flow.IntentHandler;

import java.util.ArrayList;
import java.util.List;

import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by theo on 19/03/17.
 */

public class UserAdapter extends RecyclerView.Adapter<UserAdapter.UserHolder>
implements Loader.ListLoader<User> {

 private ArrayList<User> mUsers = new ArrayList<>();

 private boolean mIsShowingFollowers = false;
 private String mUser;

 private int mPage = 1;
 private boolean mIsLoading = false;
 private boolean mMaxPageReached = false;

 private SwipeRefreshLayout mRefresher;
 private Loader mLoader;

 private Activity mLauncher;

 public UserAdapter(Activity activity, SwipeRefreshLayout refresher) {
 mLauncher = activity;
 mLoader = Loader.getLoader(activity);
 mRefresher = refresher;
 mRefresher.setRefreshing(true);
 mRefresher.setOnRefreshListener(() -> {
 mPage = 1;
 mMaxPageReached = false;
 final int oldSize = mUsers.size();
 mUsers.clear();
 notifyDataSetChanged(0, oldSize);
 loadUsers(true);
 });
 }

 public void setUser(String user, boolean isShowingFollowers) {
 mUser = user;
 mIsShowingFollowers = isShowingFollowers;
 mUsers.clear();
 }

 @Override
 public UserHolder onCreateViewHolder(ViewGroup parent, int viewType) {
 View view = LayoutInflater.from(parent.getContext())
 .inflate(R.layout.item_user, parent, false);
 return new UserHolder(view);
 }

 @Override
 public void onBindViewHolder(UserHolder holder, int position) {
 User user = mUsers.get(position);
 holder.bind(user);
 }

 @Override
 public int getItemCount() {
 return mUsers.size();
 }

 private void loadUsers(boolean forceLoad) {
 if (mIsLoading || mMaxPageReached) {
 return;
 }
 mIsLoading = true;
 mLoader.loadListData(mUser, mIsShowingFollowers, mPage, forceLoad);
 }

 @Override
 public void onListDataLoaded(List<User> users) {
 mIsLoading = false;
 mUsers.addAll(users);
 notifyDataSetChanged();
 }

 @Override
 public void onListDataError(String error) {
 mIsLoading = false;
 mRefresher.setRefreshing(false);
 }

 @Override
 public void onListDataEmpty() {
 mIsLoading = false;
 mRefresher.setRefreshing(false);
 }

 static class UserHolder extends RecyclerView.ViewHolder {
 TextView name;
 NetworkImageView profileImage;
 TextView followerCount;
 TextView followingCount;

 UserHolder(View itemView) {
 super(itemView);
 name = itemView.findViewById(R.id.name);
 profileImage = itemView.findViewById(R.id.profileImage);
 followerCount = itemView.findViewById(R.id.followerCount);
 followingCount = itemView.findViewById(R.id.followingCount);
 }

 void bind(User user) {
 name.setText(user.getLogin());
 Picasso.with(itemView.getContext())
 .load(user.getProfileImage())
 .into(profileImage);
 followerCount.setText(user.getFollowers());
 followingCount.setText(user.getFollowing());
 }
 }
}
```

```
 loadUsers(true);
 }

 public void notifyBottomReached() {
 if(!mIsLoading && !mMaxPageReached) {
 mPage++;
 loadUsers(false);
 }
 }

 private void loadUsers(boolean resetPage) {
 mIsLoading = true;
 mRefresher.setRefreshing(true);
 if(resetPage) {
 mPage = 1;
 mMaxPageReached = false;
 }
 if(mIsShowingFollowers) {
 mLoader.loadFollowers(this, mUser, mPage);
 } else {
 mLoader.loadFollowing(this, mUser, mPage);
 }
 }

 @Override
 public void listLoadComplete(List<User> users) {
 mRefresher.setRefreshing(false);
 mIsLoading = false;
 if(users.size() > 0) {
 int oldLength = mUsers.size();
 mUsers.addAll(users);
 notifyItemRangeInserted(oldLength, mUsers.size());
 } else {
 mMaxPageReached = true;
 }
 }

 @Override
 public void listLoadError(APIHandler.APIError error) {
 mIsLoading = false;
 }

 @Override
 public UserHolder onCreateViewHolder(ViewGroup parent, int viewType) {
 return new UserHolder(LayoutInflater.from(parent.getContext())
 .inflate(R.layout.viewholder_user,
parent, false));
 }

 @Override
 public void onBindViewHolder(UserHolder holder, int position) {

 holder.mAvatar.setImageUrl(mUsers.get(position).getAvatarUrl());
 holder.mName.setText(mUsers.get(position).getLogin());
 }

 @Override
 public int getItemCount() {
```

```

 return mUsers.size();
 }

 private void openUser(int pos, NetworkImageView iv) {
 IntentHandler.openUser(mLauncher, iv, mUsers.get(pos).getLogin());
 }

 class UserHolder extends RecyclerView.ViewHolder {

 @BindView(R.id.user_avatar) NetworkImageView mAvatar;
 @BindView(R.id.user_name) TextView mName;

 UserHolder(View itemView) {
 super(itemView);
 ButterKnife.bind(this, itemView);
 itemView.setOnClickListener(v -> openUser(getAdapterPosition(),
mAvatar));
 }
 }
}

```

The two states for the `UserAdapter` are showing followers, or showing following. If `mIsShowingFollowers` is true, the `loadFollowersCall` is made, otherwise the `loadFollowing` call is made. (An amazingly complex piece of logic).

## Search

---

I implemented two different fuzzy string matching algorithms while implementing search features in different parts of the app.

The first algorithm uses the Levenshtein distance between two strings to score each string, allowing them to be sorted.

```

package com(tpb.projects.util.search;

import android.support.annotation.NonNull;
import android.support.v4.util.Pair;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

/**
 * Created by theo on 30/04/17.
 */

public class StringSearcher {
 private List<String> mItems = new ArrayList<>();

 public void setItems(List<String> items) {
 mItems = items;
 }
}

```

```

public List<Integer> matches(@NonNull String query, int maxResults) {
 if(query.isEmpty()) return Collections.emptyList();
 final List<Pair<Integer, Integer>> matches = new ArrayList<>();
 for(int i = 0; i < mItems.size(); i++) {
 matches.add(Pair.create(distance(query, mItems.get(i)), i));
 }
 Collections.sort(matches, (p1, p2) -> p1.first > p2.first ? 1 :
 p1.first.equals(p2.first) ? 0 : -1);

 final List<Integer> results = new ArrayList<>();
 for(int i = 0; i < matches.size() && i < maxResults; i++) {
 results.add(matches.get(i).second);
 }
 return results;
}

private static int distance(final String s1, final String s2) {
 final int len0 = s1.length() + 1;
 final int len1 = s2.length() + 1;

 // the array of distances
 int[] cost = new int[len0];
 int[] newcost = new int[len0];

 // initial cost of skipping prefix in String s0
 for(int i = 0; i < len0; i++) cost[i] = i;

 // dynamically computing the array of distances

 // transformation cost for each letter in s1
 for (int j = 1; j < len1; j++) {
 // initial cost of skipping prefix in String s1
 newcost[0] = j;

 // transformation cost for each letter in s0
 for(int i = 1; i < len0; i++) {
 // matching current letters in both strings
 int match = (s1.charAt(i - 1) == s2.charAt(j - 1)) ? 0 : 1;

 // computing cost for each transformation
 final int replaceCost = cost[i - 1] + match;
 final int insertCost = cost[i] + 1;
 final int deleteCost = newcost[i - 1] + 1;

 // keep minimum cost
 newcost[i] = Math.min(Math.min(insertCost, deleteCost),
replaceCost);
 }
 }

 // swap cost/newcost arrays
 int[] swap = cost; cost = newcost; newcost = swap;
}

// the distance is the cost for transforming all letters in both
strings
return cost[len0 - 1] / Math.max(s1.length(), s2.length());
}

```

The Levenshtein distance between two strings is found by creating a matrix of the distances between each character in each string, and finding the shortest path through it.

	m	e	i	l	e	n	s	t	e	i	n
0	1	2	3	4	5	6	7	8	9	10	11
1	1	2	3	3	4	5	6	7	8	9	10
e	2	2	1	2	3	3	4	5	6	7	8
v	3	3	2	2	3	4	4	5	6	7	8
e	4	4	3	3	3	3	4	5	6	6	7
n	5	5	4	4	4	4	3	4	5	6	7
s	6	6	5	5	5	5	4	3	4	5	6
h	7	7	6	6	6	6	5	4	4	5	6
t	8	8	7	7	7	7	6	5	4	5	6
e	9	9	8	8	8	7	7	6	5	4	5
i	10	10	9	8	9	8	8	7	6	5	4
n	11	11	10	9	9	9	8	8	7	6	5

The problem with this is that it returns short distances for short strings, even if they do not contain a subsequence remotely resembling the query.

Performance was improved slightly by dividing the distance by the length of the larger of the two strings, but small strings were still matched above larger ones which contained the entire substring.

The second algorithm I implemented is the Bitap algorithm.

The algorithm computes a set of bitmasks containing one bit for each element in the pattern.

The bitmask must be able to mask all characters which might occur, and as such it is  $2^{16}$  in length.

Rather than allocating this array on each search, which would be needlessly wasteful, `FuzzyStringSearcher` is a singleton which can be re-used. This will never be a problem as the user cannot be searching int two places at once.

Each item in the mask is an integer, which gives a limit of 31 characters for the query, as each character requires 1 bit.

### **FuzzyStringSearcher.java**

```
package com(tpb.projects.util.search;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

/**
 * Created by theo on 03/02/17.
 */

/*
```

```

Bitap algorithm fuzzy search
To perform fuzzy searching, we need a 2d bit array
Instead of having a single array which changes over the length of the text, we
have k arrays
R_i holds a representation of the prefixes of pattern that match any suffix of
the current string with i or fewer errors.
An error may be insertion, deletion, or substitution
 */

public class FuzzyStringSearcher {
 private List<String> mItems = new ArrayList<>();
 private final int[] queryMask = new int[65536];

 private static FuzzyStringSearcher instance;

 public FuzzyStringSearcher() {

 }

 private FuzzyStringSearcher(List<String> items) {
 mItems = items;
 }

 public static FuzzyStringSearcher getInstance(List<String> items) {
 if(instance == null) {
 instance = new FuzzyStringSearcher(items);
 } else {
 instance.setItems(items);
 }
 return instance;
 }

 public void setItems(List<String> items) {
 mItems = items;
 }

 public List<Integer> search(String query) {
 final List<Integer> positions = new ArrayList<>();
 final List<Integer> ranks = new ArrayList<>();
 int index, rank;
 for(int i = 0; i < mItems.size(); i++) {
 index = findIndex(mItems.get(i), query, 1);
 if(index >= 0) {
 rank = index;
 boolean added = false;
 for(int j = 0; j < ranks.size(); j++) {
 if(rank > ranks.get(j)) {
 added = true;
 ranks.add(j, rank);
 positions.add(j, i);
 break;
 }
 }
 if(!added) {
 ranks.add(rank);
 positions.add(i);
 }
 }
 }
 }
}

```

```

 return positions;
 }

 private int findIndex(String s, String query, int k) {
 int result = -1;
 int m = query.length();
 int[] R;
 int i, d;

 if(query.isEmpty()) return -1;
 if(m > 31) return -1;

 R = new int[k + 1];
 for(i = 0; i <= k; ++i) {
 R[i] = ~1; //Bitwise complement of 1
 }
 Arrays.fill(queryMask, ~0); //Fill the mask

 for(i = 0; i < m; ++i) {
 queryMask[query.charAt(i)] &= ~(1 << i);
 }
 for(i = 0; i < s.length(); ++i) {
 int oldRd1 = R[0];
 R[0] |= queryMask[s.charAt(i)];
 R[0] <= 1;

 for(d = 1; d <= k; ++d) {
 int tmp = R[d];

 R[d] = (oldRd1 & (R[d] | queryMask[s.charAt(i)])) << 1;
 oldRd1 = tmp;
 }

 if(0 == (R[k] & (1 << m))) {
 result = (i - m) + 1;
 break;
 }
 }
 return result;
 }

}

```

search iterates through the items given and finds the index of the query in them. If the index is valid (The query was matched), the current ranks are checked and the the rank is added to the first position which it is greater than. The index of the item is then added to the positions list which is returned once the searching is complete.

The ArrayFilter is a generic class used to filter a list for an ArrayAdapter which is the adapter type for dropdown search results.

## ArrayFilter.java

```

package com(tpb.projects.util.search;

import android.widget.ArrayAdapter;

```

```
import android.widget.Filter;

import java.util.ArrayList;
import java.util.List;

/**
 * Created by theo on 05/02/17.
 */

public class ArrayFilter<T> extends Filter {

 private final ArrayAdapter<T> mParent;
 private final FuzzyStringSearcher mSearcher;
 private final List<T> mData;
 private List<T> mFiltered;

 public ArrayFilter(ArrayAdapter<T> parent, FuzzyStringSearcher searcher,
List<T> data) {
 mParent = parent;
 mSearcher = searcher;
 mData = data;
 mFiltered = new ArrayList<>();
 }

 public List<T> getFiltered() {
 return mFiltered;
 }

 @Override
 protected FilterResults performFiltering(CharSequence charSequence) {
 final FilterResults results = new FilterResults();
 if(charSequence == null) {
 results.values = new ArrayList<T>();
 results.count = 0;
 } else {
 final List<Integer> positions =
mSearcher.search(charSequence.toString());
 final ArrayList<T> items = new ArrayList<>(positions.size());

 for(int i : positions) {
 items.add(mData.get(i));
 }
 results.values = items;
 results.count = items.size();
 }
 return results;
 }

 @Override
 protected void publishResults(CharSequence charSequence, FilterResults filterResults) {
 mFiltered = (List<T>) filterResults.values;
 if(filterResults.count > 0) {
 mParent.notifyDataSetChanged();
 } else {
 mParent.notifyDataSetInvalidated();
 }
 }
}
```

```
}
```

It uses the `FuzzyStringSearcher` to match a set of positions, and then creates the `FilterResults` object with the filtered items and their size.

## Objective 3: RepoActivity

The `RepoActivity` displays the Fragments showing information about a repository. It manages loading the Repository and attaching and re-attaching the Fragments on state changes such as rotation, as well as navigating to a particular Fragment if a page is included in the launch Intent.

### RepoActivity.java

```
package com(tpb.projects.repo;

import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.design.widget.TabLayout;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;
import android.support.v4.view.ViewPager;
import android.widget.TextView;

import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Loader;
import com(tpb.github.data.models.Repository;
import com(tpb.projects.R;
import com(tpb.projects.common BaseActivity;
import com(tpb.projects.common fab FloatingActionButton;
import com(tpb.projects.repo.fragments RepoCommitsFragment;
import com(tpb.projects.repo.fragments RepoFragment;
import com(tpb.projects.repo.fragments RepoInfoFragment;
import com(tpb.projects.repo.fragments RepoIssuesFragment;
import com(tpb.projects.repo.fragments RepoProjectsFragment;
import com(tpb.projects.repo.fragments RepoReadmeFragment;
import com(tpb.projects.util SettingsActivity;
import com(tpb.projects.util UI;
import com(tpb.projects.util Util;

import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by theo on 25/03/17.
 */

public class RepoActivity extends BaseActivity implements
Loader.ItemLoader<Repository> {

 public static final int PAGE_README = 1;
 public static final int PAGE_COMMITS = 2;
 public static final int PAGE_ISSUES = 3;
 public static final int PAGE_PROJECTS = 4;
```

```
private int mLaunchPage = 0;
private boolean mLaunchPageAttached = false; //When fragments are attached
during rotation

@BindView(R.id.title_repo) TextView mTitle;
@BindView(R.id.repo_fragment_tabs) TabLayout mTabs;
@BindView(R.id.repo_fragment_viewpager) ViewPager mPager;
@BindView(R.id.repo_fab) FloatingActionButton mFab;

private RepoFragmentAdapter mAdapter;
private Repository mRepo;

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 final SettingsActivity.Preferences prefs =
SettingsActivity.Preferences
 .getPreferences(this);
 setTheme(prefs.isDarkThemeEnabled() ? R.style.AppTheme_Dark :
R.style.AppTheme);
 UI.setStatusBarColor(getWindow(),
getResources().getColor(R.color.colorPrimaryDark));
 setContentView(R.layout.activity_repo);
 ButterKnife.bind(this);

 if(mAdapter == null) mAdapter = new
RepoFragmentAdapter(getSupportFragmentManager());
 mTabs.setupWithViewPager(mPager);
 mPager.setAdapter(mAdapter);
 mPager.setOffscreenPageLimit(5);
 mPager.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {
 @Override
 public void onPageScrolled(int position, float positionOffset, int
positionOffsetPixels) {

 }

 @Override
 public void onPageSelected(int position) {
 if(mAdapter.mFragments[position].areViewsValid()) {
 mAdapter.mFragments[position].handleFab(mFab);
 }
 }

 @Override
 public void onPageScrollStateChanged(int state) {

 }
});
 if(mLaunchPageAttached) mPager.setCurrentItem(mLaunchPage);

 final Intent launchIntent = getIntent();
 final Loader loader = Loader.getLoader(this);
 if(launchIntent.getParcelableExtra(getString(R.string.intent_repo)) !=
null) {
 mRepo =
launchIntent.getParcelableExtra(getString(R.string.intent_repo));
 if(mRepo.isFork()) {
 loader.loadRepository(this, mRepo.getFullName());
 }
 }
}
```

```
 } else {
 loadComplete(launchIntent.getParcelableExtra(getString(R.string.intent_repo)));
;
 }
} else {
 if(launchIntent.getStringExtra(getString(R.string.intent_page))) {
 mLaunchPage =
 launchIntent.getIntExtra(getString(R.string.intent_page), 0);
 }
 loader.loadRepository(this,
 launchIntent.getStringExtra(getString(R.string.intent_repo))
);
}
}

@Override
public void loadComplete(Repository repo) {
mRepo = repo;
mAdapter.notifyDataSetChanged();
mTitle.setText(repo.getName());
}

@Override
public void loadError(APIHandler.APIError error) {
}

@Override
public void onAttachFragment(Fragment fragment) {
super.onAttachFragment(fragment);
if(mAdapter == null) mAdapter = new
RepoFragmentAdapter(getSupportFragmentManager());
if(fragment instanceof RepoFragment)
mAdapter.ensureAttached((RepoFragment) fragment);
if(mAdapter.indexOf(fragment) == mLaunchPage) {
 if(mPager == null) {
 mLaunchPageAttached = true;
 } else {
 viewPager.setCurrentItem(mLaunchPage);
 }
}
}

@Override
public void onBackPressed() {
mAdapter.notifyBackPressed();
super.onBackPressed();
}

private class RepoFragmentAdapter extends FragmentPagerAdapter {

private RepoFragment[] mFragments = new RepoFragment[5];
RepoFragmentAdapter(FragmentManager fm) {
 super(fm);
}
```

```
 }

 @Override
 public int getCount() {
 return 5;
 }

 void ensureAttached(RepoFragment fragment) {
 if(fragment instanceof RepoInfoFragment) mFragments[0] = fragment;
 if(fragment instanceof RepoReadmeFragment) mFragments[1] =
fragment;
 if(fragment instanceof RepoCommitsFragment) mFragments[2] =
fragment;
 if(fragment instanceof RepoIssuesFragment) mFragments[3] =
fragment;
 if(fragment instanceof RepoProjectsFragment) mFragments[4] =
fragment;
 }

 void notifyRepoLoaded() {
 for(RepoFragment rf : mFragments) {
 if(rf != null) rf.repoLoaded(mRepo);
 }
 }

 void notifyBackPressed() {
 for(RepoFragment rf : mFragments) {
 if(rf != null) rf.notifyBackPressed();
 }
 }

 int indexOf(Fragment rf) {
 return Util.indexOf(mFragments, rf);
 }

 @Override
 public Fragment getItem(int position) {
 switch(position) {
 case 0:
 mFragments[0] = RepoInfoFragment.newInstance();
 break;
 case 1:
 mFragments[1] = RepoReadmeFragment.newInstance();
 break;
 case 2:
 mFragments[2] = RepoCommitsFragment.newInstance();
 break;
 case 3:
 mFragments[3] = RepoIssuesFragment.newInstance();
 break;
 case 4:
 mFragments[4] = RepoProjectsFragment.newInstance();
 break;
 }
 if(mRepo != null) mFragments[position].repoLoaded(mRepo);
 return mFragments[position];
 }

 @Override
```

```

 public CharSequence getPageTitle(int position) {
 switch(position) {
 case 0:
 return getString(R.string.title_repo_info);
 case 1:
 return getString(R.string.title_repo_readme);
 case 2:
 return getString(R.string.title_repo_commits);
 case 3:
 return getString(R.string.title_repo_issues);
 case 4:
 return getString(R.string.title_repo_projects);
 default:
 return "";
 }
 }
 }

}

```

## RepoFragment

The RepoFragment methods deal with loading the Repository, saving the Repository state, checking view validity, handling the click listener for the FloatingActionButton, and being notified of the back button being pressed.

### RepoFragment.java

```

package com(tpb.projects.repo.fragments;

import android.os.Bundle;
import android.support.annotation.Nullable;

import com(tpb.github.data.models.Repository;
import com(tpb.projects.R;
import com(tpb.projects.common.ViewSafeFragment;
import com(tpb.projects.common.fab.FloatingActionButton;
import com(tpb.projects.repo.RepoActivity;

/**
 * Created by theo on 25/03/17.
 */

public abstract class RepoFragment extends ViewSafeFragment {

 protected Repository mRepo;

 public abstract void repoLoaded(Repository repo);

 public abstract void handleFab(FloatingActionButton fab);

 public abstract void notifyBackPressed();

 public boolean areViewsValid() {
 return mAreViewsValid;
 }
}

```

```

 }

@Override
public void onActivityCreated(@Nullable Bundle savedInstanceState) {
 super.onActivityCreated(savedInstanceState);
 if(savedInstanceState != null && savedInstanceState
 .containsKey(getString(R.string.intent_repo))) {
 mRepo =
 savedInstanceState.getParcelable(getString(R.string.intent_repo));
 repoLoaded(mRepo);
 }
}

@Override
public void onSaveInstanceState(Bundle outState) {
 super.onSaveInstanceState(outState);
 outState.putParcelable(getString(R.string.intent_repo), mRepo);
}

protected RepoActivity getParent() {
 return (RepoActivity) getActivity();
}

}

```

## Objective 3.a: RepoInfoFragment

The RepoInfoFragment binds a set of information about the Repository:

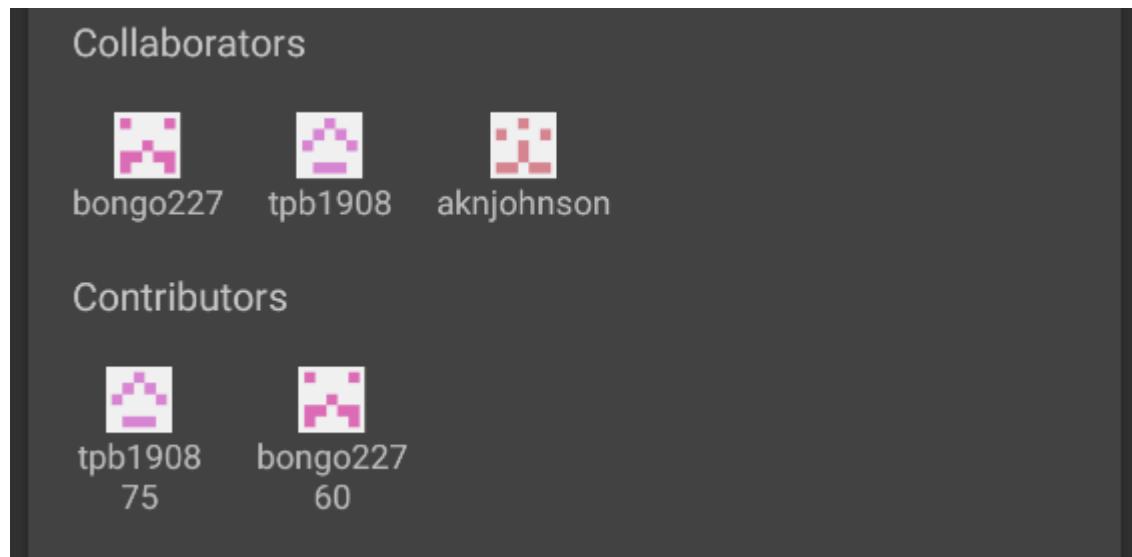
- Owner username
- Owner avatar
- Number of issues
- Number of forks
- Size
- Number of starts
- Description
- License

Much of this information is just text, and is handled in `repoLoaded` when the Views are valid.

`loadReleveantUsers` is used to load the contributors and collaborators on a repository.

Each of the `ListLoaders` then call either `displayCollaborators` or `displayContributors` which each fill a `HorizontalScrollView` with links to the users. The contributors list also includes the number of contributions.

When displayed the two lists look as shown below:



The collaborators can only be loaded by a user who are themselves a collaborator, so this layout will often be hidden.

The `FloatingActionButton` is handled by hiding it, as the `RepoInfoFragment` does not have any use for it.

The final three methods are `showLicense`, `openUser`, and `showFiles`:

`showLicense` first creates a `ProgressDialog` and shows it before loading the license body for the repository license type.

When the license is loaded, the `ProgressDialog` is dismissed, and an `AlertDialog` is displayed containing the license body.

`openUser` is called when the repository owner username or avatar is clicked. It launches the `UserActivity` using the username and avatar `Views` in the transition.

`showFiles` is called when the show files button is clicked. It launched the `ContentActivity` to display the files in the repository.

## RepoInfoFragment.java

```
package com(tpb.projects.repo.fragments;

import android.app.AlertDialog;
import android.app ProgressDialog;
import android.content Intent;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.ActivityOptionsCompat;
import android.support.v4.util.Pair;
import android.support.v4.widget SwipeRefreshLayout;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget ImageView;
import android.widget LinearLayout;
import android.widget TextView;
import android.widget.Toast;

import com(tpb.github.data.APIHandler;
```

```
import com(tpb.github.data.Loader;
import com(tpb.github.data.models.Repository;
import com(tpb.github.data.models.User;
import com(tpb.mdtext.Markdown;
import com(tpb.mdtext.views.MarkdownTextView;
import com(tpb.projects.R;
import com(tpb.projects.common.NetworkImageView;
import com(tpb.projects.common.fab.FloatingActionButton;
import com(tpb.projects.repo.content.ContentActivity;
import com(tpb.projects.user.UserActivity;
import com(tpb.projects.util.UI;
import com(tpb.projects.util.Util;

import java.util.List;
import java.util.Locale;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;
import butterknife.Unbinder;

/**
 * Created by theo on 25/03/17.
 */

public class RepoInfoFragment extends RepoFragment {
 private static final String TAG = RepoInfoFragment.class.getSimpleName();

 private Unbinder unbinder;

 private Loader mLoader;

 @BindView(R.id.repo_info_refresher) SwipeRefreshLayout mRefresher;
 @BindView(R.id.user_avatar) NetworkImageView mAvatar;
 @BindView(R.id.user_name) TextView mUserName;
 @BindView(R.id.repo_description) MarkdownTextView mDescription;
 @BindView(R.id.repo_collaborators) LinearLayout mCollaborators;
 @BindView(R.id.repo_contributors) LinearLayout mContributors;

 @BindView(R.id.repo_size) TextView mSize;
 @BindView(R.id.repo_stars) TextView mStars;
 @BindView(R.id.repo_issues) TextView mIssues;
 @BindView(R.id.repo_forks) TextView mForks;
 @BindView(R.id.repo_license) TextView mLicense;

 public static RepoInfoFragment newInstance() {
 return new RepoInfoFragment();
 }

 @Nullable
 @Override
 public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
 final View view = inflater.inflate(R.layout.fragment_repo_info,
 container, false);
 unbinder = ButterKnife.bind(this, view);
 mAreViewsValid = true;
 mRefresher.setRefreshing(true);
 mLoader = Loader.getLoader(getContext());
 }
}
```

```
mRefresher.setOnRefreshListener(() -> {
 Loader.getLoader(getContext()).loadRepository(new
Loader.ItemLoader<Repository>() {
 @Override
 public void loadComplete(Repository data) {
 repoLoaded(data);
 }

 @Override
 public void loadError(APIHandler.APIError error) {
 mRefresher.setRefreshing(false);
 }
}, mRepo.getFullName());
});
if(mRepo != null) repoLoaded(mRepo);
return view;
}

@Override
public void repoLoaded(Repository repo) {
 mRepo = repo;
 if(!areViewsValid()) return;
 mRefresher.setRefreshing(false);
 mAvatar.setImageUrl(repo.getUserAvatarUrl());
 mUserName.setText(repo.getUserLogin());
 mIssues.setText(String.valueOf(repo.getIssues()));
 mForks.setText(String.valueOf(repo.getForks()));
 mSize.setText(Util.formatKB(repo.getSize()));
 mStars.setText(String.valueOf(repo.getStarGazers()));
 if(Util.isNotNullOrEmpty(mRepo.getDescription())) {
 mDescription.setVisibility(View.VISIBLE);
 mDescription.setMarkdown(Markdown.formatMD(
 mRepo.getDescription(),
 mRepo.getFullName())
);
} else {
 mDescription.setVisibility(View.GONE);
}
if(mRepo.hasLicense()) {
 mLicense.setText(repo.getLicenseShortName());
} else {
 mLicense.setText(R.string.text_no_license);
}
loadRelevantUsers();
}

@Override
public void handleFab(FloatingActionButton fab) {
 fab.hide(true);
}

private void loadRelevantUsers() {
 mLoader.loadCollaborators(new Loader.ListLoader<User>() {
 @Override
 public void listLoadComplete(List<User> collaborators) {
 displayCollaborators(collaborators);
 }
 }

 @Override
```

```

 public void listLoadError(APIHandler.APIError error) {
 mCollaborators.setVisibility(View.GONE);
 ButterKnife.findById(getActivity(),
R.id.repo_collaborators_text)
 .setVisibility(View.GONE);
 }
 }, mRepo.getFullName());
 mLoader.loadContributors(new Loader.ListLoader<User>() {
 @Override
 public void listLoadComplete(List<User> contributors) {
 displayContributors(contributors);
 }

 @Override
 public void listLoadError(APIHandler.APIError error) {
 mContributors.setVisibility(View.GONE);
 ButterKnife.findById(getActivity(),
R.id.repo_contributors_text)
 .setVisibility(View.GONE);
 }
 }, mRepo.getFullName());
}

private void displayCollaborators(List<User> collaborators) {
 mCollaborators.removeAllViews();
 if(collaborators.size() > 1) {
 mCollaborators.setVisibility(View.VISIBLE);
 ButterKnife.findById(getActivity(), R.id.repo_collaborators_text)
 .setVisibility(View.VISIBLE);
 for(final User u : collaborators)
mCollaborators.addView(getUserView(u));
 } else {
 mCollaborators.setVisibility(View.GONE);
 ButterKnife.findById(getActivity(), R.id.repo_collaborators_text)
 .setVisibility(View.GONE);
 }
}

private void displayContributors(List<User> contributors) {
 if(!areViewsValid()) return;
 mContributors.removeAllViews();
 if(contributors.size() > 1) {
 mContributors.setVisibility(View.VISIBLE);
 ButterKnife.findById(getActivity(), R.id.repo_contributors_text)
 .setVisibility(View.VISIBLE);
 for(final User u : contributors)
mContributors.addView(getUserView(u));
 } else {
 mContributors.setVisibility(View.GONE);
 ButterKnife.findById(getActivity(), R.id.repo_contributors_text)
 .setVisibility(View.GONE);
 }
}

private View getUserView(User u) {
 final LinearLayout layout = (LinearLayout)
 getActivity()
 .getLayoutInflater()
 .inflate(R.layout.shard_user, mCollaborators, false);
}

```

```

 layout.setId(View.generateViewId());
 final NetworkImageView avatar = ButterKnife.findById(layout,
R.id.user_avatar);
 avatar.setId(View.generateViewId());
 avatar.setImageUrl(u.getAvatarUrl());
 avatar.setScaleType(ImageView.ScaleType.FIT_XY);
 final TextView login = ButterKnife.findById(layout, R.id.user_login);
 login.setId(View.generateViewId());
 if(u.getContributions() > 0) {
 login.setText(String.format(Locale.getDefault(), "%1$s\n%2$d",
u.getLogin(),
 u.getContributions()
));
 } else {
 login.setText(u.getLogin());
 }
 layout.setOnClickListener((v) -> {
 final Intent us = new Intent(getActivity(), UserActivity.class);
 us.putExtra(getString(R.string.intent_username), u.getLogin());
 UI.setDrawableForIntent(avatar, us);
 getActivity().startActivity(us,
 ActivityOptionsCompat.makeSceneTransitionAnimation(
 getActivity(),
 Pair.create(login,
 getString(R.string.transition_username)),
 Pair.create(avatar,
 getString(R.string.transition_user_image)
)
).toBundle()
);
 });
 return layout;
 }

 @OnClick({R.id.repo_license, R.id.repo_license_drawable,
R.id.repo_license_text})
 void showLicense() {
 if(mRepo.hasLicense()) {
 final ProgressDialog pd = new ProgressDialog(getContext());
 pd.setTitle(R.string.title_loading_license);
 pd.setMessage(mRepo.getLicenseName());
 pd.show();
 mLoader.loadLicenseBody(new Loader.ItemLoader<String>() {
 @Override
 public void loadComplete(String data) {
 pd.dismiss();
 new AlertDialog.Builder(getContext())
 .setTitle(mRepo.getLicenseName())
 .setMessage(data)
 .setPositiveButton(R.string.action_ok, null)
 .create()
 .show();
 }
 });

 @Override
 public void loadError(APIHandler.APIError error) {
 pd.dismiss();
 Toast.makeText(getContext(),
R.string.error_loading_license, Toast.LENGTH_SHORT)
 }
 }
 }
}

```

```

 .show();
 }
}, mRepo.getLicenseUrl());
}
}

@OnClick({R.id.user_avatar, R.id.user_name})
void openUser() {
 if(mRepo != null) {
 final Intent i = new Intent(getContext(), UserActivity.class);
 i.putExtra(getString(R.string.intent_username),
mRepo.getUserLogin());
 UI.setDrawableForIntent(mAvatar, i);
 startActivity(i,
 ActivityOptionsCompat.makeSceneTransitionAnimation(
 getActivity(),
 Pair.create(mUserName,
getString(R.string.transition_username)),
 Pair.create(mAvatar,
getString(R.string.transition_user_image))
 .toBundle()
);
 }
}

@OnClick(R.id.repo_show_files)
void showFiles() {
 if(mRepo != null) {
 final Intent i = new Intent(getContext(), ContentActivity.class);
 i.putExtra(getString(R.string.intent_repo), mRepo.getFullName());
 startActivity(i);
 }
}

@Override
public void notifyBackPressed() {

}

@Override
public void onDestroyView() {
 super.onDestroyView();
 unbinder.unbind();
}
}

```

## Objective 3.c: RepoReadmeFragment

The RepoReadmeFragment uses a MarkdownWebView to display the repository README. It first loads the README, and then uses the GitHub markdown API to render the markdown as it would be displayed on GitHub. It then fixes the relative links in the rendered HTML, and displays it in the MarkdownWebView.

RepoReadmeFragment uses notifyBackPressed to set the visibility of the MarkdownWebView to GONE.

This is because WebView extends AbsoluteLayout. As such it is not a transition group, and does not have a background which can be drawn during an animation.

This would result in the WebView remaining in place as the rest of the Activity layout performs an animation.

This undesirable effect is resolved by hiding the WebView prior to the animation starting.

## RepoReadmeFragment.java

```
package com(tpb.projects.repo.fragments;

import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.widget.SwipeRefreshLayout;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Loader;
import com(tpb.github.data.models.Repository;
import com(tpb.mdtext.Markdown;
import com(tpb.mdtext.webview.MarkdownWebView;
import com(tpb.projects.R;
import com(tpb.projects.common.fab.FloatingActionButton;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.Unbinder;

/**
 * Created by theo on 26/03/17.
 */

public class RepoReadmeFragment extends RepoFragment {

 private Unbinder unbinder;

 private Loader mLoader;

 @BindView(R.id.repo_readme_refresher) SwipeRefreshLayout mRefresher;
 @BindView(R.id.repo_readme) MarkdownWebView mReadme;

 public static RepoReadmeFragment newInstance() {
 return new RepoReadmeFragment();
 }

 @Nullable
 @Override
 public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
```

```
 final View view = inflater.inflate(R.layout.fragment_repo_readme,
container, false);
 unbinder = ButterKnife.bind(this, view);
 mAreViewsValid = true;
 mRefresher.setRefreshing(true);
 mLoader = Loader.getLoader(getContext());
 mRefresher.setOnRefreshListener(() -> {
 Loader.getLoader(getContext()).loadRepository(new
Loader.ItemLoader<Repository>() {
 @Override
 public void loadComplete(Repository data) {
 repoLoaded(data);
 }

 @Override
 public void loadError(APIHandler.APIError error) {

 }
 }, mRepo.getFullName());
 });
 mReadme.enableDarkTheme();
 if(mRepo != null) repoLoaded(mRepo);
 return view;
}

@Override
public void repoLoaded(Repository repo) {
 mRepo = repo;
 if(!areViewsValid()) return;
 mLoader.loadReadMe(new Loader.ItemLoader<String>() {
 @Override
 public void loadComplete(String data) {
 mLoader.renderMarkdown(new Loader.ItemLoader<String>() {
 @Override
 public void loadComplete(String data) {
 if(!areViewsValid()) return;
 mRefresher.setRefreshing(false);
 mReadme.setVisibility(View.VISIBLE);

mReadme.setMarkdown(Markdown.fixRelativeImageSrcs(data, mRepo.getFullName()));
 mReadme.reload();
 }
 }

 @Override
 public void loadError(APIHandler.APIError error) {
 Toast.makeText(getContext(),
R.string.error_rendering_readme,
 Toast.LENGTH_SHORT
).show();
 }
 }, data, mRepo.getDescription());
 }

 @Override
 public void loadError(APIHandler.APIError error) {
 if(!areViewsValid()) return;
 mRefresher.setRefreshing(false);
 if(error == APIHandler.APIError.NOT_FOUND) {
```

```

 Toast.makeText(getApplicationContext(),
R.string.error_readme_not_found,
 Toast.LENGTH_SHORT
).show();
 } else {
 Toast.makeText(getApplicationContext(), error.resId,
Toast.LENGTH_SHORT).show();
 }
 },
 mRepo.getFullName());
 }

@Override
public void handleFab(FloatingActionButton fab) {
 fab.hide(true);
}

@Override
public void notifyBackPressed() {
 mReadme.setVisibility(View.GONE);
}

@Override
public void onDestroyView() {
 super.onDestroyView();
 unbinder.unbind();
}
}

```

## Objective 3.d: RepoCommitsFragment

The RepoCommitsFragment primarily deals with the RepoCommitsAdapter, but it also manages the Spinner which is used to choose the branch to display commits for.

### Branch loading

The branches must be loaded separately from the repository. Unfortunately, the API returns the branches in the order that they were last committed to, and does not indicate which branch is the default. Although most repository's default branch is named "master", this is not guaranteed and even if a "master" branch is present it still may not be the default branch.

However, the API call to load the branches for a repository also returns the SHA hash of the last commit to each branch.

This means that when the API call to load commits (which returns commits on the default branch unless a branch is specified) returns, the hash of the first commit in the list can be checked against the hashes returned with the branches in order to determine the default branch.

Branch loading is managed in the same way as Repository loading, in order to fit the Fragment lifecycle.

When the `CommitsAdapter` finishes loading its commits, it calls `setLatestSHA` in the `RepoCommitsFragment` which stores the hash.

If the branches have already been loaded, `bindBranches` is called.

If the branches have not been loaded, and they are not being loaded, the call is made to load the branches.

`bindBranches` iterates through each `Pair` of branch name and hash, either adding the branch name to the start of a list if its hash is equal to the latest hash, or adding it to the end of the list.

An `ArrayAdapter` is then created to display the list of strings in the `Spinner`.

Finally, the `ArrayAdapter` is attached to the `Spinner` and an `OnItemSelectedListener` is added to it to notify the adapter when the branch is changed.

## RepoCommitsFragment.java

```
package com(tpb.projects.repo.fragments;

import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.util.Pair;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;

import com(tpb.animatingrecyclerview.AnimatingRecyclerView;
import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Loader;
import com(tpb.github.data.models.Repository;
import com(tpb.projects.R;
import com(tpb.projects.common.fab.FloatingActionButton;
import com(tpb.projects.repo.RepoCommitsAdapter;

import java.util.ArrayList;
import java.util.List;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.Unbinder;

/**
 * Created by theo on 29/03/17.
 */

public class RepoCommitsFragment extends RepoFragment implements
Loader.ListLoader<Pair<String, String>> {
 private static final String TAG =
RepoCommitsFragment.class.getSimpleName();
```

```
private Unbinder unbinder;

@BindView(R.id.repo_commits_branch_spinner) Spinner mBranchSpinner;
@BindView(R.id.repo_commits_recycler) AnimatingRecyclerView mRecyclerView;
@BindView(R.id.repo_commits_refresher) SwipeRefreshLayout mRefresher;

private RepoCommitsAdapter mAdapter;
private List<Pair<String, String>> mBranches;
private boolean mIsLoadingBranches = false;
private String mLATESTSHA;

public static RepoCommitsFragment newInstance() {
 return new RepoCommitsFragment();
}

@Override
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
 final View view = inflater.inflate(R.layout.fragment_repo_commits, container, false);
 unbinder = ButterKnife.bind(this, view);
 mRefresher.setRefreshing(true);
 mAdapter = new RepoCommitsAdapter(this, mRefresher);
 final LinearLayoutManager manager = new
LinearLayoutManager(getContext());
 mRecyclerView.enableLineDecoration();
 mRecyclerView.setLayoutManager(manager);
 mRecyclerView.setAdapter(mAdapter);
 mRecyclerView.addOnScrollListener(new RecyclerView.OnScrollListener()
{
 @Override
 public void onScrolled(RecyclerView recyclerView, int dx, int dy)
{
 super.onScrolled(recyclerView, dx, dy);
 if(manager.findFirstVisibleItemPosition() + 20 >
manager.getItemCount()) {
 mAdapter.notifyBottomReached();
 }
 }
});
 mAreViewsValid = true;
 if(mRepo != null) repoLoaded(mRepo);
 if(mBranches != null) listLoadComplete(mBranches);
 return view;
}

@Override
public void repoLoaded(Repository repo) {
 mRepo = repo;
 Loader.getLoader(getContext()).loadBranches(this,
mRepo.getFullName());
 mIsLoadingBranches = true;
 if(!areViewsValid()) return;
 mAdapter.setRepo(mRepo);
}

@Override
```

```
public void listLoadComplete(List<Pair<String, String>> branches) {
 mIsLoadingBranches = false;
 mBranches = branches;
 if(!areViewsValid()) return;
 if(mLatestSHA != null) bindBranches();
}

public void setLatestSHA(String sha) {
 if(mLatestSHA == null) {
 mLatestSHA = sha;
 if(mBranches != null && !mBranches.isEmpty()) {
 bindBranches();
 } else if(!mIsLoadingBranches) {
 Loader.getLoader(getContext()).loadBranches(this,
mRepo.getFullName());
 }
 }
}

public void bindBranches() {
 final List<String> branchNames = new ArrayList<>(mBranches.size());
 for(Pair<String, String> p : mBranches) {
 if(mLatestSHA.equals(p.second)) {
 branchNames.add(0, p.first);
 } else {
 branchNames.add(p.first);
 }
 }
 final ArrayAdapter<String> adapter = new ArrayAdapter<>(getContext(),
 android.R.layout.simple_spinner_item, branchNames
);
}

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)
;
 mBranchSpinner.setAdapter(adapter);
 if(mBranchSpinner.getOnItemSelectedListener() == null) {
 mBranchSpinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
 @Override
 public void onItemSelected(AdapterView<?> parent, View view,
int position, long id) {
 mAdapter.setBranch(branchNames.get(position));
 }

 @Override
 public void onNothingSelected(AdapterView<?> parent) {

 }
 });
 }
}

@Override
public void listLoadError(APIHandler.APIError error) {
 mIsLoadingBranches = false;
}

@Override
public void handleFab(FloatingActionButton fab) {
```

```

 fab.hide(true);
 }

 @Override
 public void notifyBackPressed() {
 }

 @Override
 public void onDestroyView() {
 super.onDestroyView();
 unbinder.unbind();
 }
}

```

The RepoCommitsAdapter manages its page as it is notified of new scroll positions and uses the SpanCache interface to cache SpannableStrings with their respective commits as they are created.

When `setBranch` is called, if the branch is not the current branch there are two possibilities:

The branch is being set for the first time after the default branch has been chosen (`mBranch` is null), in this case the branch is saved, but the content is not re-loaded as it is already displayed.

Otherwise, the branch is set, `mCommits` is cleared, `notifyDataSetChanged` is called, and `loadCommits` is called to reset the page and reload the commits.

## Binding

Each `CommitViewHolder` is bound with the following information:

- The committer avatar (If the committer is an actual user and not automated)
- The commit message
- The commit information (Committer username, short hash, and date)

`OnItemClickListener`s are then added to the avatar, title, and info views to launch either the `UserActivity` or `CommitActivity`.

## RepoCommitsAdapter.java

```

package com(tpb.projects.repo;

import android.content.res.Resources;
import android.support.v4.util.Pair;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.widget.RecyclerView;
import android.text.SpannableString;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com(tpb.github.data.APIHandler;

```

```
import com(tpb.github.data.Loader;
import com(tpb.github.data.models.Commit;
import com(tpb.github.data.models.Repository;
import com(tpb.mdtext.Markdown;
import com(tpb.mdtext.views.MarkdownTextView;
import com(tpb.projects.R;
import com(tpb.projects.common.NetworkImageView;
import com(tpb.projects.flow.IntentHandler;
import com(tpb.projects.repo.fragments.RepoCommitsFragment;
import com(tpb.projects.util.Util;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by theo on 29/03/17.
 */

public class RepoCommitsAdapter extends
RecyclerView.Adapter<RepoCommitsAdapter.CommitViewHolder> implements
Loader.ListLoader<Commit> {
 private static final String TAG =
RepoCommitsAdapter.class.getSimpleName();

 private RepoCommitsFragment mParent;
 private Repository mRepo;
 private String mBranch;
 private ArrayList<Pair<Commit, SpannableString>> mCommits = new
ArrayList<>();

 private int mPage = 1;
 private boolean mIsLoading = false;
 private boolean mMaxPageReached = false;

 private SwipeRefreshLayout mRefresher;
 private Loader mLoader;

 public RepoCommitsAdapter(RepoCommitsFragment parent, SwipeRefreshLayout
refresher) {
 mParent = parent;
 mRefresher = refresher;
 mLoader = Loader.getLoader(parent.getContext());
 mRefresher.setOnRefreshListener(() -> {
 final int oldSize = mCommits.size();
 mCommits.clear();
 notifyItemRangeRemoved(0, oldSize);
 loadCommits(true);
 });
 }

 public void setRepo(Repository repo) {
 mRepo = repo;
 final int oldSize = mCommits.size();
 mCommits.clear();
 notifyItemRangeRemoved(0, oldSize);
 }

 @Override
 public CommitViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
 View view = LayoutInflater.from(parent.getContext())
 .inflate(R.layout.item_commit, parent, false);
 return new CommitViewHolder(view);
 }

 @Override
 public void onBindViewHolder(CommitViewHolder holder, int position) {
 Commit commit = mCommits.get(position);
 holder.bind(commit);
 }

 @Override
 public int getItemCount() {
 return mCommits.size();
 }

 @Override
 public void onNewDataLoaded(List<Commit> data) {
 mCommits.addAll(data);
 notifyDataSetChanged();
 }

 @Override
 public void onListEmpty() {
 mRefresher.setRefreshing(false);
 }

 @Override
 public void onListFull() {
 mRefresher.setRefreshing(false);
 }

 @Override
 public void onListError() {
 mRefresher.setRefreshing(false);
 }

 private void loadCommits(boolean refresh) {
 if (mIsLoading || mMaxPageReached) {
 return;
 }
 mIsLoading = true;
 mRefresher.setRefreshing(true);
 mLoader.loadList(TAG, mPage, refresh);
 }
}
```

```

 loadCommits(true);
 }

 public void setBranch(String branch) {
 if(!branch.equals(mBranch)) {
 if(mBranch != null) {
 mBranch = branch;
 mCommits.clear();
 notifyDataSetChanged();
 loadCommits(true);
 } else {
 mBranch = branch;
 }
 }
 }

 public void notifyBottomReached() {
 if(!mIsLoading && !mMaxPageReached) {
 mPage++;
 loadCommits(false);
 }
 }

 private void loadCommits(boolean resetPage) {
 mIsLoading = true;
 mRefresher.setRefreshing(true);
 if(resetPage) {
 mPage = 1;
 mMaxPageReached = false;
 }
 mLoader.loadCommits(this, mRepo.getFullName(), mBranch, mPage);
 }

 @Override
 public void listLoadComplete(List<Commit> commits) {
 if(!mParent.areViewsValid()) return;
 mRefresher.setRefreshing(false);
 mIsLoading = false;
 Log.i("Loading", commits.size() + " Commits finished loading for page "
" + mPage);
 if(commits.size() > 0) {
 final int oldLength = mCommits.size();
 if(mPage == 1) {
 mParent.setLatestSHA(commits.get(0).getSha());
 }
 for(Commit c : commits) {
 mCommits.add(Pair.create(c, null));
 }
 Log.i("Loading", "Commits loaded: " + commits.toString());
 notifyDataSetChanged();
 } else {
 mMaxPageReached = true;
 }
 }

 @Override
 public void listLoadError(APIHandler.APIError error) {
 mRefresher.setRefreshing(false);
 }

```

```
 }

 @Override
 public CommitViewHolder onCreateViewHolder(ViewGroup parent, int viewType)
 {
 return new CommitViewHolder(LayoutInflater.from(parent.getContext())
 .inflate(R.layout.viewholder_commit, parent,
 false
));
 }

 @Override
 public void onBindViewHolder(CommitViewHolder holder, int position) {
 final Commit c = mCommits.get(position).first;
 if(c.getCommitter() != null) {
 holder.mAvatar.setImageUrl(c.getCommitter().getAvatarUrl());
 }
 holder.mTitle.setMarkdown(Markdown.formatMD(c.getMessage(),
mRepo.getFullName()));
 final String userName;
 final String userUrl;

 if(c.getCommitter() != null) {
 userName = c.getCommitter().getLogin();
 userUrl = c.getCommitter().getHtmlUrl();
 } else {
 userName = c.getCommitterName();
 userUrl = IntentHandler.getUserUrl(userName);
 }
 if(mCommits.get(position).second == null) {
 final StringBuilder builder = new StringBuilder();
 final Resources res = holder.itemView.getResources();

 builder.append(
 String.format(
 res.getString(R.string.text_committed_by_hash_at),
 String.format(
 res.getString(R.string.text_md_link),
 userName,
 userUrl
),
 String.format(
 res.getString(R.string.text_md_link),
 com(tpb.github.data.Util.shortenSha(c.getSha())),
 c.getHtmlUrl()
),
 Util.formatDateLocally(holder.itemView.getContext(),
 new Date(c.getCreatedAt()))
)
);
 holder.mInfo.setMarkdown(
 Markdown.formatMD(builder.toString(),
mRepo.getFullName()),
 null,
);
 }
 }
}
```

```
 text -> mCommits.set(position, Pair.create(c, text))
);
} else {
 holder.mInfo.setText(mCommits.get(position).second);
}
IntentHandler.addOnClickHandler(mParent.getActivity(), holder.mAvatar,
userName);
IntentHandler.addOnClickHandler(mParent.getActivity(), holder.mTitle,
c);
IntentHandler.addOnClickHandler(mParent.getActivity(), holder.mInfo,
c);
}

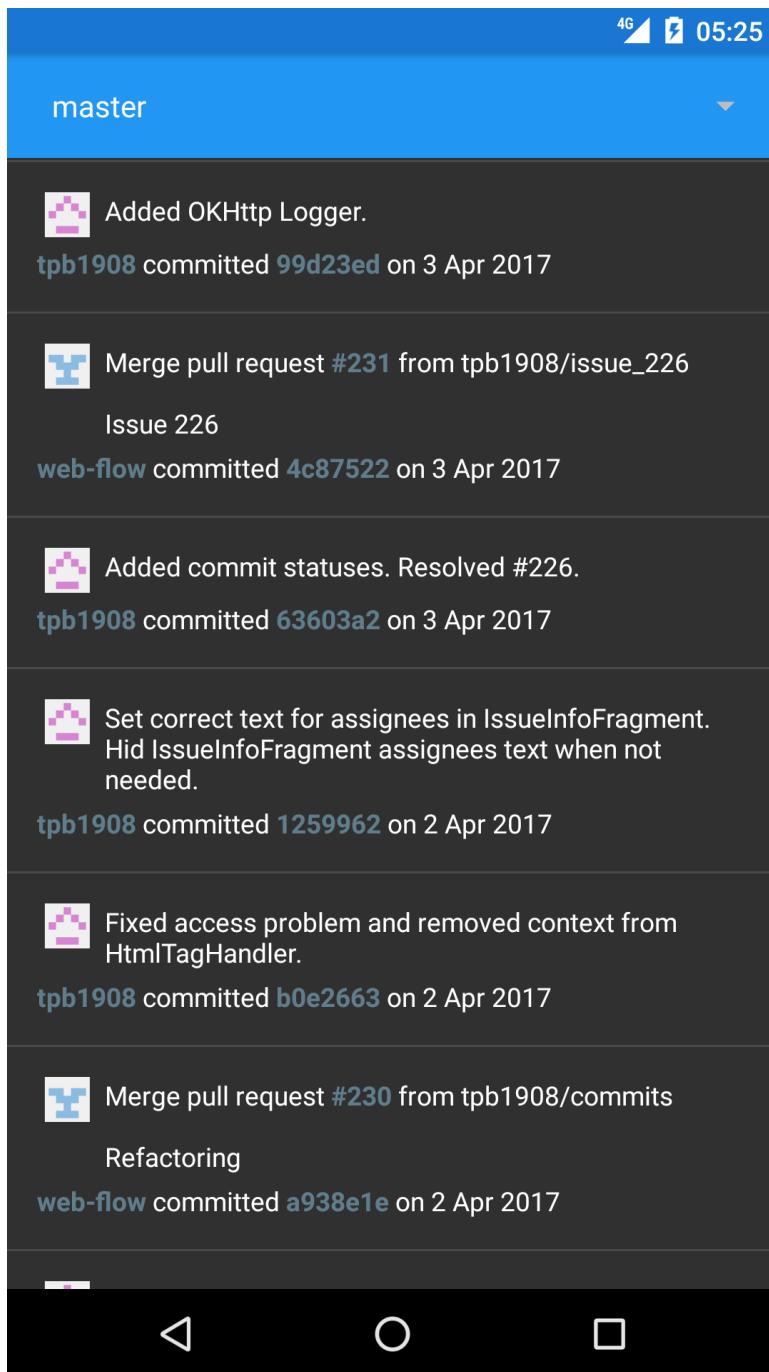
@Override
public int getItemCount() {
 return mCommits.size();
}

static class CommitViewHolder extends RecyclerView.ViewHolder {

 @BindView(R.id.commit_user_avatar) NetworkImageView mAvatar;
 @BindView(R.id.commit_title) MarkdownTextView mTitle;
 @BindView(R.id.commit_info) MarkdownTextView mInfo;

 CommitViewHolder(View itemView) {
 super(itemView);
 ButterKnife.bind(this, itemView);
 }
}
}
```

The screenshot below shows commits made from my account and the web-flow:



## Objective 3.e: RepolssuesFragment

The RepoIssuesFragment is the first RepoFragment which actually uses the FloatingActionButton.

It also manages filtering and searching the Issues.

### RepolssuesFragment.java

```
package com(tpb.projects.repo.fragments;

import android.app.ProgressDialog;
import android.content.Intent;
```

```
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.app.AlertDialog;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.SearchView;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.View;
import android.view.ViewGroup;
import android.widget.PopupMenu;
import android.widget.Toast;

import com(tpb.animatingrecyclerview.AnimatingRecyclerView;
import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Editor;
import com(tpb.github.data.Loader;
import com(tpb.github.data.models.Comment;
import com(tpb.github.data.models.Issue;
import com(tpb.github.data.models.Label;
import com(tpb.github.data.models.Repository;
import com(tpb.github.data.models.State;
import com(tpb.github.data.models.User;
import com(tpb.mdtext.views.MarkdownTextview;
import com(tpb.projects.R;
import com(tpb.projects.common.fab.FabHideScrollListener;
import com(tpb.projects.common.fab.FloatingActionButton;
import com(tpb.projects.editors.CommentEditor;
import com(tpb.projects.editors.IssueEditor;
import com(tpb.projects.editors.MultiChoiceDialog;
import com(tpb.projects.flow.IntentHandler;
import com(tpb.projects.repo.RepoIssuesAdapter;
import com(tpb.projects.util.UI;

import java.util.ArrayList;
import java.util.List;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;
import butterknife.Unbinder;

import static android.content.ContentValues.TAG;
import static com(tpb.github.data.models.State.ALL;
import static com(tpb.github.data.models.State.CLOSED;
import static com(tpb.github.data.models.State.OPEN;

/**
 * Created by theo on 25/03/17.
 */

public class RepoIssuesFragment extends RepoFragment {

 private Unbinder unbinder;

 @BindView(R.id.repo_issues_recycler) AnimatingRecyclerView m RecyclerView;
 private FabHideScrollListener mFabHideScrollListener;
 @BindView(R.id.repo_issues_refresher) SwipeRefreshLayout mRefresher;
```

```
@BindView(R.id.issues_search_view) SearchView mSearchView;
private RepoIssuesAdapter mAdapter;

private State mFilter = State.OPEN;
private String mAssigneeFilter;
private final ArrayList<String> mLabelsFilter = new ArrayList<>();

private Editor mEditor;

public static RepoIssuesFragment newInstance() {
 return new RepoIssuesFragment();
}

@Override
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
 final View view = inflater.inflate(R.layout.fragment_repo_issues, container, false);
 unbinder = ButterKnife.bind(this, view);
 mEditor = Editor.getEditor(getContext());
 mAdapter = new RepoIssuesAdapter(this, mRefresher);
 final LinearLayoutManager manager = new
 LinearLayoutManager(getContext());
 mRecyclerView.enableLineDecoration();
 mRecyclerView.setLayoutManager(manager);
 mRecyclerView.setAdapter(mAdapter);
 mRecyclerView.addOnScrollListener(new RecyclerView.OnScrollListener()
{
 @Override
 public void onScrolled(RecyclerView recyclerView, int dx, int dy)
{
 super.onScrolled(recyclerView, dx, dy);
 if(manager.findFirstVisibleItemPosition() + 20 >
manager.getItemCount()) {
 mAdapter.notifyBottomReached();
 }
 }
});
 mSearchView.setOnQueryTextListener(new
SearchView.OnQueryTextListener() {
 @Override
 public boolean onQueryTextSubmit(String query) {
 return false;
 }

 @Override
 public boolean onQueryTextChange(String newText) {
 mAdapter.search(newText);
 return false;
 }
});
 mSearchView.setOnCloseListener(() -> {
 mAdapter.closeSearch();
 return false;
 });
 mAreViewsValid = true;
 if(mRepo != null) repoReloaded(mRepo);
 return view;
}
```

```
}

@Override
public void repoLoaded(Repository repo) {
 mRepo = repo;
 if(!areViewsValid()) return;
 mAdapter.setRepo(repo);
}

@Override
public void handleFab(FloatingActionButton fab) {
 fab.show(true);
 fab.setOnClickListener(v -> {
 final Intent intent = new Intent(getContext(), IssueEditor.class);
 intent.putExtra(getString(R.string.intent_repo),
mRepo.getFullName());
 UI.setViewPositionForIntent(intent, fab);
 startActivityForResult(intent,
IssueEditor.REQUEST_CODE_NEW_ISSUE);
 });

 if(mFabHideScrollListener == null) {
 mFabHideScrollListener = new FabHideScrollListener(fab);
 mRecyclerView.addOnScrollListener(mFabHideScrollListener);
 }
}

@OnClick(R.id.issues_filter_button)
void filter(View v) {
 final PopupMenu menu = new PopupMenu(getContext(), v);
 menu.inflate(R.menu.menu_issues_filter);
 switch(mFilter) {
 case ALL:
 menu.getMenu().getItem(2).setChecked(true);
 break;
 case OPEN:
 menu.getMenu().getItem(0).setChecked(true);
 break;
 case CLOSED:
 menu.getMenu().getItem(1).setChecked(true);
 break;
 }
 menu.setOnMenuItemClickListener(menuItem -> {
 switch(menuItem.getItemId()) {
 case R.id.menu_filter_assignees:
 showAssigneesDialog();
 break;
 case R.id.menu_filter_labels:
 showLabelsDialog();
 break;
 case R.id.menu_filter_all:
 mFilter = ALL;
 refresh();
 break;
 case R.id.menu_filter_closed:
 mFilter = CLOSED;
 refresh();
 break;
 case R.id.menu_filter_open:
 break;
 }
 });
}
```

```
 mFilter = OPEN;
 refresh();
 break;
 }
 return false;
});
menu.show();
}

private void refresh() {
 mAdapter.applyFilter(mFilter, mAssigneeFilter, mLabelsFilter);
}

private void showLabelsDialog() {
 final ProgressDialog pd = new ProgressDialog(getContext());
 pd.setTitle(R.string.text_loading_labels);
 pd.setCancelable(false);
 pd.show();
 Loader.getLoader(getContext()).loadLabels(new
Loader.ListLoader<Label>() {
 @Override
 public void listLoadComplete(List<Label> labels) {
 final MultiChoiceDialog mcd = new MultiChoiceDialog();

 final Bundle b = new Bundle();
 b.putInt(getString(R.string.intent_title_res),
R.string.title_choose_labels);
 mcd.setArguments(b);

 final String[] labelTexts = new String[labels.size()];
 final int[] colors = new int[labels.size()];
 final boolean[] choices = new boolean[labels.size()];
 for(int i = 0; i < labels.size(); i++) {
 labelTexts[i] = labels.get(i).getName();
 colors[i] = labels.get(i).getColor();
 choices[i] =
mLabelsFilter.indexOf(labels.get(i).getName()) != -1;
 }

 mcd.setChoices(labelTexts, choices);
 mcd.setBackgroundColors(colors);
 mcd.setListener(new
MultiChoiceDialog.MultiChoiceDialogListener() {
 @Override
 public void choicesComplete(String[] choices, boolean[]
checked) {
 mLabelsFilter.clear();
 for(int i = 0; i < choices.length; i++) {
 if(checked[i]) {
 mLabelsFilter.add(choices[i]);
 }
 }
 refresh();
 }

 @Override
 public void choicesCancelled() {

```

```
 }
 });
 pd.dismiss();

 mcd.show(getActivity().getSupportFragmentManager(), TAG);
}

@Override
public void listLoadError(APIHandler.APIError error) {
 Toast.makeText(getContext(), error.resId,
Toast.LENGTH_SHORT).show();
}
}, mRepo.getFullName()));

}

private void showAssigneesDialog() {
 final ProgressDialog pd = new ProgressDialog(getContext());
 pd.setTitle(R.string.text_loading_contributors);
 pd.setCancelable(false);
 pd.show();
 Loader.getLoader(getContext()).loadContributors(new
Loader.ListLoader<User>() {
 @Override
 public void listLoadComplete(List<User> contributors) {
 final String[] collabNames = new String[contributors.size() +
2];
 collabNames[0] = getString(R.string.text_assignee_all);
 collabNames[1] = getString(R.string.text_assignee_none);
 int pos = 0;
 for(int i = 2; i < collabNames.length; i++) {
 collabNames[i] = contributors.get(i - 2).getLogin();
 if(collabNames[i].equals(mAssigneeFilter)) {
 pos = i;
 }
 }
 final String oldAssigneeFilter = mAssigneeFilter;
 final AlertDialog.Builder builder = new
AlertDialog.Builder(getContext());
 builder.setTitle(R.string.title_choose_assignee);
 builder.setSingleChoiceItems(collabNames, pos,
 (dialogInterface, i) -> mAssigneeFilter =
collabNames[i]
);
 builder.setPositiveButton(R.string.action_ok,
(dialogInterface, i) -> refresh());
 builder.setNegativeButton(R.string.action_cancel, (d, i) ->
mAssigneeFilter = oldAssigneeFilter);
 builder.create().show();
 pd.dismiss();
 }

 @Override
 public void listLoadError(APIHandler.APIError error) {
 pd.dismiss();
 Toast.makeText(getContext(), error.resId,
Toast.LENGTH_SHORT).show();
 }
}, mRepo.getFullName());
```

```
}

public void openMenu(View view, final Issue issue) {
 final PopupMenu menu = new PopupMenu(getContext(), view);
 menu.inflate(R.menu.menu_issue);
 menu.getMenu().add(0, R.id.menu_toggle_issue_state, Menu.NONE,
 issue.isClosed() ? R.string.menu_reopen_issue :
 R.string.menu_close_issue
);
 menu.getMenu().add(0, R.id.menu_edit_issue, Menu.NONE,
 getString(R.string.menu_edit_issue));

 menu.setOnMenuItemClickListener(menuItem -> {
 switch(menuItem.getItemId()) {
 case R.id.menu_toggle_issue_state:
 toggleIssueState(issue);
 break;
 case R.id.menu_edit_issue:
 editIssue(view, issue);
 break;
 case R.id.menu_fullscreen:
 IntentHandler.showFullScreen(getContext(),
issue.getBody(),
 issue.getRepoFullName(), getSupportFragmentManager()
);
 break;
 }
 return false;
 });
 menu.show();
}

private void editIssue(View view, Issue issue) {
 final Intent intent = new Intent(getContext(), IssueEditor.class);
 intent.putExtra(getString(R.string.intent_repo), mRepo.getFullName());
 intent.putExtra(getString(R.string.parcel_issue), issue);
 final Loader loader = Loader.getLoader(getContext());
 loader.loadLabels(null, issue.getRepoFullName());
 loader.loadContributors(null, issue.getRepoFullName());
 if(view instanceof MarkdownTextView) {
 UI.setClickPositionForIntent(getActivity(), intent,
 ((MarkdownTextView) view).getLastClickPosition()
);
 } else {
 UI.setViewPositionForIntent(intent, view);
 }
 startActivityForResult(intent, IssueEditor.REQUEST_CODE_EDIT_ISSUE);
}

private void toggleIssueState(Issue issue) {
 final Editor.UpdateListener<Issue> listener = new
Editor.UpdateListener<Issue>() {
 @Override
 public void updated(Issue toggled) {
 mAdapter.updateIssue(toggled);
 mRefresher.setRefreshing(false);
 }
 }
}
```

```
 @Override
 public void updateError(APIHandler.APIError error) {
 mRefresher.setRefreshing(false);
 Toast.makeText(getContext(), error.resId,
Toast.LENGTH_SHORT).show();
 }

 final AlertDialog.Builder builder = new
AlertDialog.Builder(getContext());
 builder.setTitle(R.string.title_state_change_comment);
 builder.setPositiveButton(R.string.action_ok, (dialog, which) -> {
 mRefresher.setRefreshing(true);
 final Intent i = new Intent(getContext(), CommentEditor.class);
 i.putExtra(getString(R.string.parcel_issue), issue);
 startActivityForResult(i,
CommentEditor.REQUEST_CODE_COMMENT_FOR_STATE);
 if(issue.isClosed()) {
 mEditor.openIssue(listener, issue.getRepoFullName(),
issue.getNumber());
 } else {
 mEditor.closeIssue(listener, issue.getRepoFullName(),
issue.getNumber());
 }
 });
 builder.setNegativeButton(R.string.action_no, (dialog, which) -> {
 mRefresher.setRefreshing(true);
 if(issue.isClosed()) {
 mEditor.openIssue(listener, issue.getRepoFullName(),
issue.getNumber());
 } else {
 mEditor.closeIssue(listener, issue.getRepoFullName(),
issue.getNumber());
 }
 });
 builder.setNeutralButton(R.string.action_cancel, null);
 builder.create().show();
}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data)
{
 super.onActivityResult(requestCode, resultCode, data);
 if(resultCode == IssueEditor.RESULT_OK) {
 final String[] assignees;
 final String[] labels;
 if(data.getStringExtra(getString(R.string.intent_issue_assignees))) {
 assignees =
data.getStringArrayExtra(getString(R.string.intent_issue_assignees));
 } else {
 assignees = null;
 }
 if(data.getStringExtra(getString(R.string.intent_issue_labels))) {
 labels =
data.getStringArrayExtra(getString(R.string.intent_issue_labels));
 } else {
 labels = null;
 }
 }
}
```

```
 final Issue issue =
data.getParcelableExtra(getString(R.string.parcel_issue));
 if(requestCode == IssueEditor.REQUEST_CODE_NEW_ISSUE) {
 createIssue(issue, assignees, labels);
 } else if(requestCode == IssueEditor.REQUEST_CODE_EDIT_ISSUE) {
 updateIssue(issue, assignees, labels);
 } else if(requestCode ==
CommentEditor.REQUEST_CODE_COMMENT_FOR_STATE) {
 final Comment comment =
data.getParcelableExtra(getString(R.string.parcel_comment));
 mEditor.createIssueComment(new
Editor.CreationListener<Comment>() {
 @Override
 public void created(Comment comment) {
 mRefresher.setRefreshing(false);
 }

 @Override
 public void creationError(APIHandler.APIError error) {
 mRefresher.setRefreshing(false);
 if(error == APIHandler.APIError.NO_CONNECTION) {
 Toast.makeText(getContext(), error.resId,
Toast.LENGTH_SHORT).show();
 }
 }
 }, issue.getRepoFullName(), issue.getNumber(),
comment.getBody());
 }
 }

 private void createIssue(Issue issue, String[] assignees, String[] labels)
{
 mRefresher.setRefreshing(true);
 mEditor.createIssue(new Editor.CreationListener<Issue>() {
 @Override
 public void created(Issue issue) {
 mRefresher.setRefreshing(false);
 mAdapter.addIssue(issue);
 mRecyclerView.scrollToPosition(0);
 }

 @Override
 public void creationError(APIHandler.APIError error) {
 mRefresher.setRefreshing(false);
 }
 }, mRepo.getFullName(), issue.getTitle(), issue.getBody(), assignees,
labels);
}

 private void updateIssue(Issue issue, String[] assignees, String[] labels)
{
 mRefresher.setRefreshing(true);
 mEditor.updateIssue(new Editor.UpdateListener<Issue>() {
 int issueCreationAttempts = 0;

 @Override
 public void updated(Issue issue) {
 mAdapter.updateIssue(issue);
 }
 });
}
```

```

 mRefresher.setRefreshing(false);
 }

 @Override
 public void updateError(APIHandler.APIError error) {
 if(error == APIHandler.APIError.NO_CONNECTION) {
 mRefresher.setRefreshing(false);
 Toast.makeText(getApplicationContext(), error.resId,
Toast.LENGTH_SHORT).show();
 } else {
 if(issueCreationAttempts < 5) {
 issueCreationAttempts++;
 mEditor.updateIssue(this, mRepo.getFullName(), issue,
assignees,
 labels
);
 } else {
 Toast.makeText(getApplicationContext(), error.resId,
Toast.LENGTH_SHORT)
 .show();
 mRefresher.setRefreshing(false);
 }
 }
 }
}, mRepo.getFullName(), issue, assignees, labels);
}

@Override
public void notifyBackPressed() {
}

@Override
public void onDestroyView() {
 super.onDestroyView();
 unbinder.unbind();
}
}

```

## Filtering

Objective 3.v.c is to filter issues by:

- Their state
- Their labels
- The user assigned to them

This is implemented through a filter menu alongside the search bar.

The menu is inflated from the following resource:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
```

```

<group android:checkableBehavior="single">
 <item
 android:id="@+id/menu_filter_open"
 android:title="@string/menu_filter_open"/>

 <item
 android:id="@+id/menu_filter_closed"
 android:title="@string/menu_filter_closed"/>

 <item
 android:id="@+id/menu_filter_all"
 android:title="@string/menu_filter_all"/>
</group>

<item
 android:id="@+id/menu_filter_labels"
 android:title="@string/menu_filter_labels"/>

<item
 android:id="@+id/menu_filter_assignees"
 android:title="@string/menu_filter_assignee"/>

</menu>

```

The first item is a group of radio buttons for the issue state. The second two items are buttons to show dialogs for choosing the labels or assigned user.

In the `onClick` method for the issues filter button, the menu is inflated.

One of the items in the set of radio buttons is ticked based upon the current filter state.

The `OnMenuItemClickListener` for the `PopupMenu` is set, which switches over the item id and either calls a method to show the appropriate dialog, or sets the filter and then calls `refresh` to apply the filter to the adapter.

`showLabelsDialog` first creates a `ProgressDialog` while the labels are loaded, and then creates a `MultiChoiceDialog` with the labels.

Items are checked when the dialog is shown if they are already present in the `mLabelsFilter` list.

The `MultiChoiceDialogListener` clears the `mLabelsFilter` list and adds the new labels before calling `refresh` to update the adapter.

`showAssigneesDialog` also displays a `ProgressDialog` while it loads the contributors. It then shows an `AlertDialog` with a set of single choice items.

When an item is chosen, the `mAssigneeFilter` is set to this new value. If the positive button is selected, `refresh` is called, otherwise the assignee filter is reset to its previous state.

## Editing

The `RepoIssuesFragment` also manages toggling of issue states, as well as creating and updating issues.

`editIssue` adds the repository name and `Issue` to an `Intent`, pre-loads the labels and collaborators and then launched the `IssueEditor` with the `REQUEST_CODE_EDIT_ISSUE` request code.

If the edited `Issue` is returned in `onActivityResult` the assignees and labels arrays are extracted from the `Intent` and passes to `updateIssue` which performs the `Editor` call to update the `Issue`, and then notifies the adapter of the change.

## Creation

When the `RepoIssuesFragment` is passed the `FloatingActionButton` it sets the `OnClickListener` to open the `IssueEditor` with the flag to create a new issue. If this issue is returned in `onActivityResult`, `createIssue` is called which performs the `Editor` call to create the `Issue`, notifies the adapter of the change, and scrolls the `RecyclerView` to position 0, displaying the new `Issue`.

## State changes

When the menu item for opening or closing an issue is selected, `toggleIssueState` is called.

This method first creates the `Editor.UpdateListener` which will update the `Issue` in the adapter and stop the `SwipeRefreshLayout`.

It then shows an `AlertDialog` asking the user if they wish to leave a comment explaining why they are opening or closing the issue.

If the user selects the positive action, the `CommentEditor` is shown with the request code `REQUEST_CODE_COMMENT_FOR_ISSUE_STATE`, and the issue state is changed.

If the user selects the negative button, the issue state is toggled without showing the `CommentEditor`.

If the user selects the third, neutral, option, the dialog is cancelled.

## Adapter

The `RepoIssuesAdapter` manages loading and displaying `Issues`, managing updates to `Issues`, and applying a search to the dataset.

Binding works in the standard manner, with each `Issue` stored in a `Pair` alongside its cached `SpannableString`.

The filters described above are set in `applyFilter` and passed to the `Loader` in `loadIssues`.

Search filters are applied using the `FuzzyStringSearcher`.

When `search` is called a list of the information about each `Issue` is created and passed with the query to the `FuzzyStringSearcher`.

The positions returned are set in `mSearchFilter`, and `notifyDataSetChanged` is called.

In `onBindViewHolder`, if `mIsSearching` is true, the actual position of the data is found from the value in `mSearchFilter` at the position being bound.

### **RepoIssuesAdapter.java**

```
package com(tpb.projects.repo;

import android.support.v4.util.Pair;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.widget.RecyclerView;
import android.text.SpannableString;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageButton;
import android.widget.ImageView;

import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Loader;
import com(tpb.github.data.models.Issue;
import com(tpb.github.data.models.Label;
import com(tpb.github.data.models.Repository;
import com(tpb.github.data.models.State;
import com(tpb.mdtext.Markdown;
import com(tpb.mdtext.views.MarkdownTextView;
import com(tpb.projects.R;
import com(tpb.projects.common.NetworkImageView;
import com(tpb.projects.flow.IntentHandler;
import com(tpb.projects.markdown.Formatter;
import com(tpb.projects.repo.fragments.RepoIssuesFragment;
import com(tpb.projects.util.Util;
import com(tpb.projects.util.search.FuzzyStringSearcher;

import java.util.ArrayList;
import java.util.List;

import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by theo on 25/03/17.
 */

public class RepoIssuesAdapter extends
RecyclerView.Adapter<RepoIssuesAdapter.IssueHolder> implements
Loader.ListLoader<Issue> {

 private final RepoIssuesFragment mParent;
 private final SwipeRefreshLayout mRefresher;
 private final ArrayList<Pair<Issue, SpannableString>> mIssues = new
ArrayList<>();
 private FuzzyStringSearcher mSearcher = new FuzzyStringSearcher();
 private boolean mIsSearching = false;
 private List<Integer> mSearchFilter = new ArrayList<>();

 private Loader mLoader;
 private Repository mRepo;
 private int mPage = 1;
 private boolean mIsLoading = false;
```

```
private boolean mMaxPageReached = false;

private State mFilter = State.OPEN;
private String mAssigneeFilter;
private final ArrayList<String> mLabelsFilter = new ArrayList<>();

public RepoIssuesAdapter(RepoIssuesFragment parent, SwipeRefreshLayout
refresher) {
 mParent = parent;
 mLoader = Loader.getLoader(mParent.getContext());
 mRefresher = refresher;
 mRefresher.setOnRefreshListener(() -> {
 final int oldSize = mIssues.size();
 mIssues.clear();
 mIsSearching = false;
 notifyItemRangeRemoved(0, oldSize);
 loadIssues(true);
 });
}

public void setRepo(Repository repo) {
 mRepo = repo;
 mIssues.clear();
 loadIssues(true);
}

public void search(String query) {
 if(mIsLoading) return;
 mIsSearching = true;
 final List<String> issues = new ArrayList<>();
 final StringBuilder builder = new StringBuilder();
 for(Pair<Issue, SpannableString> p : mIssues) {
 builder.append(p.first.getNumber());
 builder.append(" ");
 builder.append(p.first.getTitle());
 if(p.first.getOpenedBy() != null)
builder.append(p.first.getOpenedBy().getLogin());
 if(p.first.getLabels() != null) {
 for(Label l : p.first.getLabels())
builder.append(l.getName());
 }
 builder.append(" ");
 builder.append(p.first.getBody());
 issues.add(builder.toString());
 builder.setLength(0);
 }
 mSearcher.setItems(issues);
 mSearchFilter = mSearcher.search(query);
 notifyDataSetChanged();
}

public void closeSearch() {
 mIsSearching = false;
 mSearchFilter.clear();
}

public void applyFilter(State state, String assignee, ArrayList<String>
labels) {
 mIsSearching = false;
```

```
mSearchFilter.clear();
mFilter = state;
mAssigneeFilter = assignee;
mLabelsFilter.clear();
mLabelsFilter.addAll(labels);
final int oldSize = mIssues.size();
mIssues.clear();
notifyItemRangeRemoved(0, oldSize);
loadIssues(true);
}

@Override
public void listLoadComplete(List<Issue> issues) {
 mRefresher.setRefreshing(false);
 mIsLoading = false;
 if(issues.size() > 0) {
 final int oldLength = mIssues.size();
 for(Issue i : issues) {
 mIssues.add(Pair.create(i, null));
 }
 notifyItemRangeInserted(oldLength, mIssues.size());
 } else {
 mMaxPageReached = true;
 }
}

@Override
public void listLoadError(APIHandler.APIError error) {
 mRefresher.setRefreshing(false);
 loadIssues(false);
}

public void notifyBottomReached() {
 if(!mIsLoading && !mMaxPageReached) {
 mPage++;
 loadIssues(false);
 }
}

private void loadIssues(boolean resetPage) {
 mIsLoading = true;
 mRefresher.setRefreshing(true);
 if(resetPage) {
 mPage = 1;
 mMaxPageReached = false;
 }
 mLoader.loadIssues(this, mRepo.getFullName(), mFilter,
mAssigneeFilter, mLabelsFilter,
 mPage
);
}

public void addIssue(Issue issue) {
 mIssues.add(0, Pair.create(issue, null));
 notifyItemInserted(0);
}

public void updateIssue(Issue issue) {
 int index = Util.indexOf(mIssues, issue);
```

```

 if(index != -1) {
 mIssues.set(index, Pair.create(issue, null));
 notifyItemChanged(index);
 }
 }

@Override
public IssueHolder onCreateViewHolder(ViewGroup parent, int viewType) {
 return new IssueHolder(LayoutInflater.from(parent.getContext()))

.inflate(R.layout.viewholder_issue, parent, false));
}

@Override
public void onBindViewHolder(IssueHolder holder, int position) {
 final int pos = mIsSearching ? mSearchFilter.get(position) :
holder.getAdapterPosition();
 final Issue issue = mIssues.get(pos).first;
 holder.mTitle.setMarkdown(Formatter.bold(issue.getTitle()));
 holder.mIssueIcon.setImageResource(
 issue.isClosed() ? R.drawable.ic_state_closed :
R.drawable.ic_state_open);
 holder.mUserAvatar.setImageUrl(issue.getOpenedBy().getAvatarUrl());
 IntentHandler.addOnClickHandler(mParent.getActivity(),
holder.mUserAvatar,
 issue.getOpenedBy().getLogin()
);
 IntentHandler
 .addOnClickHandler(mParent.getActivity(), holder.mContent,
holder.mUserAvatar, null,
 issue
);
 if(mIssues.get(pos).second == null) {
 holder.mContent.setMarkdown(Markdown.formatMD(
Formatter.buildCombinedIssueSpan(holder.itemView.getContext(),
issue).toString(),
 issue.getRepoFullName()
),
 null,
 text -> mIssues.set(pos, Pair.create(issue, text))
);
}
else {
 holder.mContent.setText(mIssues.get(pos).second);
}
IntentHandler.addOnClickHandler(mParent.getActivity(),
holder.mContent, issue);
IntentHandler.addOnClickHandler(mParent.getActivity(), holder.mTitle,
issue);
IntentHandler.addOnClickHandler(mParent.getActivity(),
holder.itemView, issue);
holder.mMenuButton.setOnClickListener((v) -> mParent.openMenu(v,
issue));
}

@Override
public int getItemCount() {
 return mIsSearching ? mSearchFilter.size() : mIssues.size();
}

```

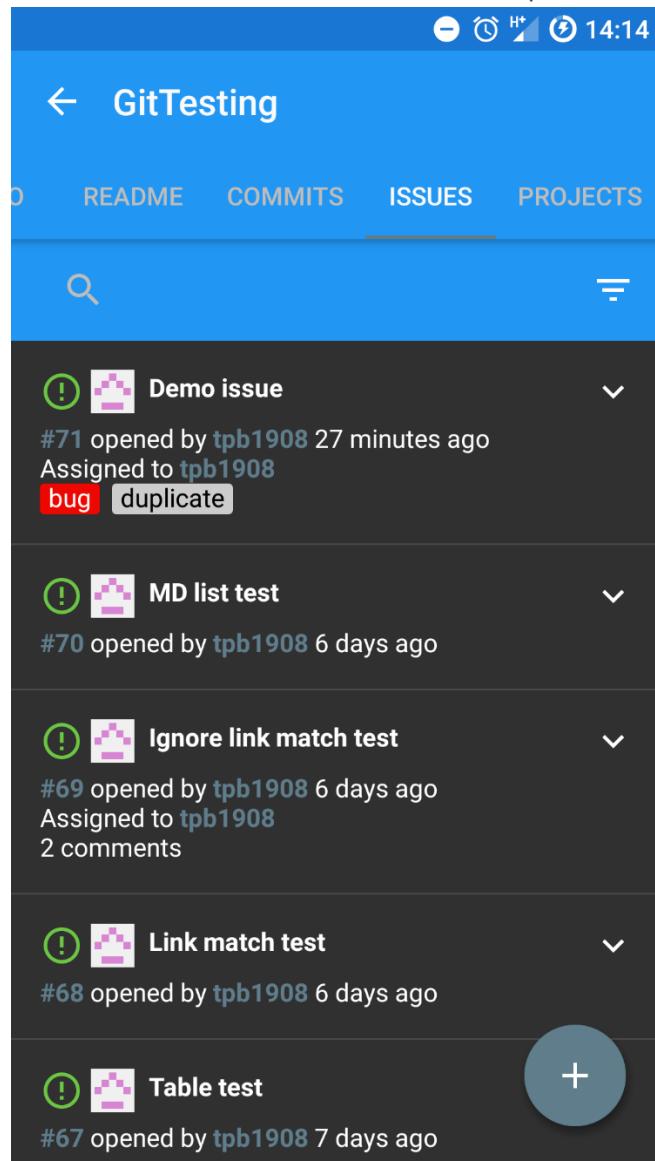
```
}

static class IssueHolder extends RecyclerView.ViewHolder {

 @BindView(R.id.issue_title) MarkdownTextView mTitle;
 @BindView(R.id.issue_content_markdown) MarkdownTextView mContent;
 @BindView(R.id.issue_menu_button) ImageButton mMenuButton;
 @BindView(R.id.issue_state_drawable) ImageView mIssueIcon;
 @BindView(R.id.issue_user_avatar) NetworkImageView mUserAvatar;

 IssueHolder(View view) {
 super(view);
 ButterKnife.bind(this, view);
 }
}
```

The screenshot below shows the RepoIssuesFragment:



## Objective 3.f: RepoProjectsFragment

The final RepoFragment displayed in RepoActivity is the RepoProjectsFragment which displays the projects associated with a repository, as well as managing their state, editing and deleting them.

### RepoProjectsFragment.java

```
package com(tpb.projects.repo.fragments;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.app.AlertDialog;
import android.support.v7.widget.LinearLayoutManager;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.PopupMenu;

import com(tpb.animatingrecyclerview.AnimatingRecyclerView;
import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Editor;
import com(tpb.github.data.models.Project;
import com(tpb.github.data.models.Repository;
import com(tpb.github.data.models.State;
import com(tpb.projects.R;
import com(tpb.projects.common.fab.FabHideScrollListener;
import com(tpb.projects.common.fab.FloatingActionButton;
import com(tpb.projects.editors.ProjectEditor;
import com(tpb.projects.repo.RepoProjectsAdapter;
import com(tpb.projects.util.UI;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.Unbinder;

/**
 * Created by theo on 25/03/17.
 */

public class RepoProjectsFragment extends RepoFragment {

 private Unbinder unbinder;

 @BindView(R.id.fragment_refresher) SwipeRefreshLayout mRefresher;
 @BindView(R.id.fragment_recycler) AnimatingRecyclerView mRecycler;
 private FabHideScrollListener mFabHideScrollListener;
 private RepoProjectsAdapter mAdapter;

 public static RepoProjectsFragment newInstance() {
 return new RepoProjectsFragment();
 }

 @Nullable
 @Override
```

```
 public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
 final View view = inflater.inflate(R.layout.fragment_recycler,
container, false);
 unbinder = ButterKnife.bind(this, view);
 mAdapter = new RepoProjectsAdapter(this, mRefresher);
 mRecycler.enableLineDecoration();
 mRecycler.setLayoutManager(new LinearLayoutManager(getContext()));
 mRecycler.setAdapter(mAdapter);
 mRefresher.setOnRefreshListener(() -> mAdapter.reload());
 mAreViewsValid = true;
 if(mRepo != null) repoLoaded(mRepo);
 return view;
 }

 @Override
 public void repoLoaded(Repository repo) {
 mRepo = repo;
 if(!mAreViewsValid) return;
 mAdapter.setRepository(repo);

 }

 @Override
 public void handleFab(FloatingActionButton fab) {
 fab.show(true);
 if(mFabHideScrollListener == null) {
 mFabHideScrollListener = new FabHideScrollListener(fab);
 mRecycler.addOnScrollListener(mFabHideScrollListener);
 }
 fab.setOnClickListener(v -> {
 final Intent i = new Intent(getContext(), ProjectEditor.class);
 UI.setViewPositionForIntent(i, fab);
 startActivityForResult(i, ProjectEditor.REQUEST_CODE_NEW_PROJECT);
 });
 }

 private void toggleProjectState(Project project) {
 mRefresher.setRefreshing(true);
 final Editor.UpdateListener<Project> listener = new
Editor.UpdateListener<Project>() {
 @Override
 public void updated(Project updated) {
 mRefresher.setRefreshing(false);
 mAdapter.updateProject(updated);
 }

 @Override
 public void updateError(APIHandler.APIError error) {
 mRefresher.setRefreshing(false);
 }
 };
 if(project.getState() == State.OPEN) {
 Editor.getEditor(getContext()).closeProject(listener,
project.getId());
 } else {
 Editor.getEditor(getContext()).openProject(listener,
project.getId());
 }
 }
}
```

```
}

private void deleteProject(Project project) {
 new AlertDialog.Builder(getContext())
 .setTitle(R.string.title_delete_project)
 .setMessage(R.string.text_delete_project_warning)
 .setPositiveButton(R.string.action_ok, (dialog, which) -> {
 mRefresher.setRefreshing(true);
 Editor.getEditor(getContext()).deleteProject(
 new Editor.DeletionListener<Project>() {
 @Override
 public void deleted(Project deleted) {
 mRefresher.setRefreshing(false);
 mAdapter.removeProject(deleted);
 }
 @Override
 public void deletionError(APIHandler.APIError
error) {
 mRefresher.setRefreshing(false);
 }
 }, project);
 })
 .setNegativeButton(R.string.action_cancel, null)
 .show();
}

private void editProject(Project project, View view) {
 final Intent i = new Intent(getContext(), ProjectEditor.class);
 i.putExtra(getString(R.string.parcel_project), project);
 UI.setViewPositionForIntent(i, view);
 startActivityForResult(i, ProjectEditor.REQUEST_CODE_EDIT_PROJECT);
}

public void showMenu(View view, Project project) {
 final PopupMenu pm = new PopupMenu(getContext(), view);
 pm.inflate(R.menu.menu_project);
 if(project.getState() == State.OPEN) {
 pm.getMenu().add(0, R.id.menu_toggle_project_state, 0,
R.string.menu_close_project);
 } else {
 pm.getMenu().add(0, R.id.menu_toggle_project_state, 0,
R.string.menu_reopen_project);
 }
 pm.setOnMenuItemClickListener(item -> {
 switch(item.getItemId()) {
 case R.id.menu_toggle_project_state:
 toggleProjectState(project);
 break;
 case R.id.menu_edit_project:
 editProject(project, view);
 break;
 case R.id.menu_delete_project:
 deleteProject(project);
 break;
 }
 return true;
 });
 pm.show();
}
```

```
}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data)
{
 super.onActivityResult(requestCode, resultCode, data);
 if(resultCode == Activity.RESULT_OK) {

 if(requestCode == ProjectEditor.REQUEST_CODE_NEW_PROJECT) {
 final String name =
data.getStringExtra(getString(R.string.intent_name));
 final String body =
data.getStringExtra(getString(R.string.intent_markdown));
 mRefresher.setRefreshing(true);
 Editor.getEditor(getContext()).createProject(
 new Editor.CreationListener<Project>() {
 @Override
 public void created(Project project) {
 mRefresher.setRefreshing(false);
 mAdapter.addProject(project);
 }
 }

 @Override
 public void creationError(APIHandler.APIError
error) {
 mRefresher.setRefreshing(false);
 }
 }, name, body, mRepo.getFullName());
 } else if(requestCode == ProjectEditor.REQUEST_CODE_EDIT_PROJECT)
{
 mRefresher.setRefreshing(true);
 final int id =
data.getIntExtra(getString(R.string.intent_project_number), -1);
 final String name =
data.getStringExtra(getString(R.string.intent_name));
 final String body =
data.getStringExtra(getString(R.string.intent_markdown));
 Editor.getEditor(getContext()).updateProject(
 new Editor.UpdateListener<Project>() {
 @Override
 public void updated(Project project) {
 mRefresher.setRefreshing(false);
 mAdapter.updateProject(project);
 }
 }

 @Override
 public void updateError(APIHandler.APIError error)
{
 mRefresher.setRefreshing(false);
 }
 }, name, body, id);
 }
 }

 @Override
 public void notifyBackPressed() {
}
```

```

@Override
public void onDestroyView() {
 super.onDestroyView();
 unbinder.unbind();
}
}

```

In handleFab, if a FabHideScrollListener has not already been created, it is created and added to the RecyclerView.

The FloatingActionButton OnClickListener is then set to launch the ProjectEditor with the REQUEST\_CODE\_NEW\_PROJECT request code. showMenu displays a PopupMenu with items for editing deleting and opening or closing a project.

toggleProjectState calls Editor.closeProject or Editor.openProject and updates the adapter in the UpdateListener callback.

deleteProject displays a warning dialog, and if the user confirms their action, it calls Editor.deleteProject with a callback to remove the project from the adapter. editProject launches ProjectEditor with the REQUEST\_CODE\_EDIT\_PROJECT request code.

If a successfull result is returned to onActivityResult the request code is checked, the project name and body are extracted and

either createProject or updateProject are called, the latter requiring the id of the pre-existing project.

The RepoProjectsAdapter manages binding Project information and passes click events back to the RepoProjectsFragment.

### **RepoProjectsAdapter.java**

```

package com(tpb.projects.repo;

import android.content.Intent;
import android.support.v4.app.ActivityOptionsCompat;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.widget.RecyclerView;
import android.text.format.DateUtils;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageButton;
import android.widget.TextView;

import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Loader;
import com(tpb.github.data.models.Project;
import com(tpb.github.data.models.Repository;
import com(tpb.github.data.models.State;
import com(tpb.mdtext.imagegetter.HttpImageGetter;
import com(tpb.mdtext.views.MarkdownTextView;
import com(tpb.projects.R;
import com(tpb.projects.project.ProjectActivity;
import com(tpb.projects.repo.fragments.RepoProjectsFragment;
import com(tpb.projects.util.Util;
}

```

```
import java.util.ArrayList;
import java.util.List;

import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by theo on 25/03/17.
 */

public class RepoProjectsAdapter extends
RecyclerView.Adapter<RepoProjectsAdapter.ProjectViewHolder> implements
Loader.ListLoader<Project> {

 private ArrayList<Project> mProjects = new ArrayList<>();
 private Loader mLoader;
 private Repository mRepo;
 private RepoProjectsFragment mParent;
 private SwipeRefreshLayout mRefresher;

 public RepoProjectsAdapter(RepoProjectsFragment parent, SwipeRefreshLayout
refresher) {
 mLoader = Loader.getLoader(parent.getContext());
 mParent = parent;
 mRefresher = refresher;
 }

 public void setRepository(Repository repo) {
 mRefresher.setRefreshing(true);
 mLoader.loadProjects(this, repo.getFullName());
 mRepo = repo;
 }

 public void reload() {
 final int oldSize = mProjects.size();
 mProjects.clear();
 notifyItemRangeRemoved(0, oldSize);
 mLoader.loadProjects(this, mRepo.getFullName());
 }

 @Override
 public void listLoadComplete(List<Project> projects) {
 mProjects.clear();
 mProjects.addAll(projects);
 notifyItemRangeChanged(0, mProjects.size());
 mRefresher.setRefreshing(false);
 }

 @Override
 public void listLoadError(APIHandler.APIError error) {
 mRefresher.setRefreshing(false);
 }

 public void updateProject(Project project) {
 final int index = mProjects.indexOf(project);
 if(index != -1) {
 mProjects.set(index, project);
 notifyItemChanged(index);
 }
 }
}
```

```
 }

 }

 public void addProject(Project project) {
 mProjects.add(0, project);
 notifyItemInserted(0);
 }

 public void removeProject(Project project) {
 final int index = mProjects.indexOf(project);
 if(index != -1) {
 mProjects.remove(index);
 notifyItemRemoved(index);
 }
 }

 @Override
 public ProjectViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
 return new ProjectViewHolder(LayoutInflater.from(parent.getContext()))
 .inflate(R.layout.viewholder_project, parent,
 false
);
 }

 @Override
 public void onBindViewHolder(ProjectViewHolder holder, int position) {
 final Project p = mProjects.get(position);
 holder.mName.setText(p.getName());
 holder.mName.setCompoundDrawablesWithIntrinsicBounds(
 p.getState() == State.OPEN ? R.drawable.ic_state_open :
R.drawable.ic_state_closed,
 0, 0, 0
);
 holder.mLastUpdate.setText(
 String.format(
 holder.itemView.getContext().getString(R.string.text_last_updated),
 DateUtils.getRelativeTimeSpanString(p.getUpdatedAt())
)
);
 if(Util.isNotNullOrEmpty(p.getBody())) {
 holder.mBody.setVisibility(View.VISIBLE);
 holder.mBody.setMarkdown(
 p.getBody(),
 new HttpImageGetter(holder.mBody),
 null
);
 }
 holder.itemView.setOnClickListener(v -> {
 final Intent i = new Intent(mParent.getContext(),
ProjectActivity.class);
 i.putExtra(mParent.getString(R.string.parcel_project), p);
 mParent.startActivity(i,
 ActivityOptionsCompat.makeSceneTransitionAnimation(
 mParent.getActivity(),
 holder.mName,
 mParent.getString(R.string.transition_title)
)
);
 });
 }
}
```

```

).toBundle()
);
 i.putExtra(mParent.getString(R.string.intent_project_number),
p.getNumber());
}
holder.mMenu.setOnClickListener(v -> mParent.showMenu(holder.mMenu,
p));
}

@Override
public int getItemCount() {
 return mProjects.size();
}

static class ProjectViewHolder extends RecyclerView.ViewHolder {

 @BindView(R.id.project_name) TextView mName;
 @BindView(R.id.project_last_updated) TextView mLastUpdate;
 @BindView(R.id.project_body) MarkdownTextView mBody;
 @BindView(R.id.project_menu_button) ImageButton mMenu;

 ProjectViewHolder(View view) {
 super(view);
 ButterKnife.bind(this, view);
 }
}
}

```

In onBindViewHolder it sets binds the ProjectViewHolder Views with:

- The name of the project
- The state drawable for the project
- The last time that the project was updated
- The project description if it exists

It then adds an OnClickListener to the itemView to open the ProjectActivity with a shared element transition using the project name (Which contains the state drawable).

The menu button OnClickListener is set to call showMenu on the RepoProjectsFragment.

## Objective 3.b: ContentActivity

---

The ContentActivity is used for displaying the content of a repository (whoever would have guessed?).

It uses the same method for displaying a branch Spinner as the RepoCommitsFragment except that the default branch HEAD hash comes from one of the Nodes loaded.

The file data is loaded with the GitHub contents API.

This API returns the contents of a directory within a repository.

If the path is a directory, JSON of the following format is returned:

```
[
 {
 "type": "file",
 "size": 625,
 "name": "octokit.rb",
 "path": "lib/octokit.rb",
 "sha": "fff6fe3a23bf1c8ea0692b4a883af99bee26fd3b",
 "url":
 "https://api.github.com/repos/octokit/octokit.rb/contents/lib/octokit.rb",
 "git_url":
 "https://api.github.com/repos/octokit/octokit.rb/git/blobs/fff6fe3a23bf1c8ea06
92b4a883af99bee26fd3b",
 "html_url":
 "https://github.com/octokit/octokit.rb/blob/master/lib/octokit.rb",
 "download_url":
 "https://raw.githubusercontent.com/octokit/octokit.rb/master/lib/octokit.rb",
 "_links": {
 "self": "https://api.github.com/repos/octokit/octokit.rb/contents/lib/octokit.rb",
 "git": "https://api.github.com/repos/octokit/octokit.rb/git/blobs/fff6fe3a23bf1c8ea06
92b4a883af99bee26fd3b",
 "html": "https://github.com/octokit/octokit.rb/blob/master/lib/octokit.rb"
 }
 },
 {
 "type": "dir",
 "size": 0,
 "name": "octokit",
 "path": "lib/octokit",
 "sha": "a84d88e7554fc1fa21bc4efae3c782a70d2b9d",
 "url":
 "https://api.github.com/repos/octokit/octokit.rb/contents/lib/octokit",
 "git_url":
 "https://api.github.com/repos/octokit/octokit.rb/git/trees/a84d88e7554fc1fa21b
cbc4efae3c782a70d2b9d",
 "html_url":
 "https://github.com/octokit/octokit.rb/tree/master/lib/octokit",
 "download_url": null,
 "_links": {
 "self": "https://api.github.com/repos/octokit/octokit.rb/contents/lib/octokit",
 "git": "https://api.github.com/repos/octokit/octokit.rb/git/trees/a84d88e7554fc1fa21b
cbc4efae3c782a70d2b9d",
 "html": "https://github.com/octokit/octokit.rb/tree/master/lib/octokit"
 }
 }
]
```

The array contains a single JSON object for each item in the directory.

The item types can be:

- File
- Directory
- Symbolic link
- Sum-module

Each item in the JSON is parsed into a Node model, which is separate from the DateModel used elsewhere.

### **Node.java**

```
package com(tpb.github.data.models.content;

import android.os.Parcel;
import android.os.Parcelable;
import android.support.annotation.NonNull;
import android.util.Log;

import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.List;

/**
 * Created by theo on 17/02/17.
 */

public class Node implements Parcelable {

 private NodeType type;
 private int size;
 private String encoding;
 private String name;
 private String path;
 private String content;
 private String sha;
 private String url;
 private String gitUrl;
 private String htmlUrl;
 private String downloadUrl;
 private String submoduleGitUrl;

 private Node parent;
 private List<Node> children = new ArrayList<>();

 private static final String TYPE_KEY = "type";
 private static final String SIZE_KEY = "size";
 private static final String ENCODING_KEY = "encoding";
 private static final String NAME_KEY = "name";
 private static final String PATH_KEY = "path";
 private static final String CONTENT_KEY = "content";
 private static final String SHA_KEY = "sha";
 private static final String URL_KEY = "url";
 private static final String GIT_URL_KEY = "git_url";
}
```

```

private static final String HTML_URL_KEY = "html_url";
private static final String DOWNLOAD_URL_KEY = "download_url";
private static final String SUBMODULE_GIT_URL_KEY = "submodule_git_url";

public Node(JSONObject obj) {
 try {
 type = NodeType.fromString(obj.getString(TYPE_KEY));
 size = obj.getInt(SIZE_KEY);
 if(obj.has(ENCODING_KEY)) encoding = obj.getString(ENCODING_KEY);
 name = obj.getString(NAME_KEY);
 path = obj.getString(PATH_KEY);
 if(obj.has(CONTENT_KEY)) content = obj.getString(CONTENT_KEY);
 sha = obj.getString(SHA_KEY);
 url = obj.getString(URL_KEY);
 gitUrl = obj.getString(GIT_URL_KEY);
 htmlUrl = obj.getString(HTML_URL_KEY);
 downloadUrl = obj.getString(DOWNLOAD_URL_KEY);
 if(obj.has(SUBMODULE_GIT_URL_KEY)) {
 submoduleGitUrl = obj.getString(SUBMODULE_GIT_URL_KEY);
 type = NodeType.SUBMODULE;
 }
 if(isSubmodule(url, gitUrl)) type = NodeType.SUBMODULE;
 } catch(JSONException jse) {
 Log.e("Node", "Node: Exception: ", jse);
 }
}

private boolean isSubmodule(@NonNull String url, @NonNull String gitUrl) {
 try {
 int start = url.indexOf("com/") + 4;
 int repoStart = url.indexOf('/', url.indexOf('/', url.indexOf('/', start + 1) + 1));
 int repoEnd = url.indexOf('/', repoStart + 1) + 1;
 final String repo = url.substring(repoStart, repoEnd);
 start = gitUrl.indexOf("com/") + 4;
 repoStart = gitUrl
 .indexOf('/', gitUrl.indexOf('/', gitUrl.indexOf('/', start + 1)));
 repoEnd = gitUrl.indexOf('/', repoStart + 1) + 1;
 return !repo.equals(gitUrl.substring(repoStart, repoEnd));
 } catch(IndexOutOfBoundsException iob) {
 return false;
 }
}

public NodeType getType() {
 return type;
}

public int getSize() {
 return size;
}

public String getEncoding() {
 return encoding;
}

public String getName() {
 return name;
}

```

```
}

public String getPath() {
 return path;
}

public String getContent() {
 return content;
}

public String getSha() {
 return sha;
}

public String getUrl() {
 return url;
}

public String getGitUrl() {
 return gitUrl;
}

public String getHtmlUrl() {
 return htmlUrl;
}

public String getDownloadUrl() {
 return downloadUrl;
}

public String getSubmoduleGitUrl() {
 return submoduleGitUrl;
}

public Node getParent() {
 return parent;
}

public List<Node> getChildren() {
 return children;
}

public String getRef() {
 if(htmlUrl.contains("/tree/")) {
 final int index = htmlUrl.indexOf("/tree/") + 6;
 return htmlUrl.substring(index, htmlUrl.indexOf('/', index));
 } else {
 final int index = htmlUrl.indexOf("/blob/") + 6;
 return htmlUrl.substring(index, htmlUrl.indexOf('/', index));
 }
}

public void setParent(Node parent) {
 this.parent = parent;
}

public void setChildren(List<Node> children) {
 this.children = children;
}
```

```
@Override
public boolean equals(Object obj) {
 return obj instanceof Node && sha.equals(((Node) obj).getSha());
}

@Override
public String toString() {
 return "Node{" +
 "type=" + type +
 ", size=" + size +
 ", encoding='" + encoding + '\'' +
 ", name='" + name + '\'' +
 ", path='" + path + '\'' +
 ", content='" + content + '\'' +
 ", sha='" + sha + '\'' +
 ", url='" + url + '\'' +
 ", gitUrl='" + gitUrl + '\'' +
 ", htmlUrl='" + htmlUrl + '\'' +
 ", downloadUrl='" + downloadUrl + '\'' +
 ", submoduleGitUrl='" + submoduleGitUrl + '\'' +
 '}';
}

public enum NodeType {

 FILE("file"),
 DIRECTORY("dir"),
 SYMLINK("symlink"),
 SUBMODULE("submodule");

 private final String type;

 NodeType(String type) {
 this.type = type;
 }

 public static NodeType fromString(String type) {
 for(NodeType nt : NodeType.values()) {
 if(nt.type.equals(type)) return nt;
 }
 throw new IllegalArgumentException("No NodeType with String value
" + type);
 }
}

@Override
public int describeContents() {
 return 0;
}

@Override
public void writeToParcel(Parcel dest, int flags) {
 dest.writeInt(this.type == null ? -1 : this.type.ordinal());
 dest.writeInt(this.size);
 dest.writeString(this.encoding);
 dest.writeString(this.name);
 dest.writeString(this.path);
 dest.writeString(this.content);
```

```

 dest.writeString(this.sha);
 dest.writeString(this.url);
 dest.writeString(this.gitUrl);
 dest.writeString(this.htmlUrl);
 dest.writeString(this.downloadUrl);
 dest.writeString(this.submoduleGitUrl);
 dest.writeParcelable(this.parent, flags);
 dest.writeTypedList(this.children);
 }

protected Node(Parcel in) {
 int tmpType = in.readInt();
 this.type = tmpType == -1 ? null : NodeType.values()[tmpType];
 this.size = in.readInt();
 this.encoding = in.readString();
 this.name = in.readString();
 this.path = in.readString();
 this.content = in.readString();
 this.sha = in.readString();
 this.url = in.readString();
 this.gitUrl = in.readString();
 this.htmlUrl = in.readString();
 this.downloadUrl = in.readString();
 this.submoduleGitUrl = in.readString();
 this.parent = in.readParcelable(Node.class.getClassLoader());
 this.children = in.createTypedArrayList(Node.CREATOR);
}

public static final Creator<Node> CREATOR = new Creator<Node>() {
 @Override
 public Node createFromParcel(Parcel source) {
 return new Node(source);
 }

 @Override
 public Node[] newArray(int size) {
 return new Node[size];
 }
};
}

```

The Nodemodel contains the following:

- A NodeType enum which may be FILE, DIRECTORY, SYMLINK, or SUBMODULE
- The size of the node, if applicable
- The encoding of the node, if applicable
- The name of the node
- The node path
- The node content, if applicable
- The SHA hash of the node
- The node URL, which is the API URL for the node
- The Git URL, which is the URL to the tree state for this version of the node

- The HTML URL, which is the URL to view the node online
- The download url, which is the raw.githubusercontent.com URL to download the node, if applicable
- The submodule Git URL, which is the URL to another repository if a submodule has been imported into the repository being viewed

`getRef` and `isSubmodule` use the `Node` variables to calculate other information about the `Node`.

The GitHub API warns that when the contents of a directory are listed, submodules have their type specified as "file" for backwards compatibility purposes.

`isSubmodule` extracts the repository name from both the `url` and `gitUrl`, and compares them.

When the repository used for this documentation is embedded in the project repository, it has the following URL:

["https://api.github.com/repos/tpb1908/AndroidProjectsClient/contents/CWDoc?ref=master"](https://api.github.com/repos/tpb1908/AndroidProjectsClient/contents/CWDoc?ref=master)

start is found as the index of "/" in "com/".

The first index found gives the substring

"tpb1908/AndroidProjectsClient/contents/CWDoc?ref=master"

The second index found gives the substring

"/tpb1908/AndroidProjectsClient/contents/CWDoc?ref=master"

The third index found gives the substring

"/AndroidProjectsClient/contents/CWDoc?ref=master"

The end index is the next "/" in the final substring, giving the repository as "/AndroidProjectsClient/".

The `gitUrl` is

["https://api.github.com/repos/tpb1908/CWDoc/git/trees/9d14a93dbb9592f948bcf29be1a8697c3e3c3395"](https://api.github.com/repos/tpb1908/CWDoc/git/trees/9d14a93dbb9592f948bcf29be1a8697c3e3c3395)

The first index found gives the substring

"tpb1908/CWDoc/git/trees/9d14a93dbb9592f948bcf29be1a8697c3e3c3395"

The second index found gives the substring

"/tpb1908/CWDoc/git/trees/9d14a93dbb9592f948bcf29be1a8697c3e3c3395"

The third index found gives the substring

"/CWDoc/git/trees/9d14a93dbb9592f948bcf29be1a8697c3e3c3395"

The repository is then extracted as "/CWDoc/".

As the two repository strings are not equal, `isSubmodule` returns true.

`getRef` is used to extract the SHA hash for the directory or file from its `htmlUrl`.

If the Node is a directory, the SHA is between the "/tree/" substring and the next "/".

Otherwise, the Node is a file, and the SHA is between the "/blob/" substring and the next "/".

## ContentActivity.java

```
package com(tpb.projects.repo.content;

import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.util.Pair;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.widget.LinearLayoutManager;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.HorizontalScrollView;
import android.widget.LinearLayout;
import android.widget.Spinner;
import android.widget.TextView;

import com(tpb.animatingrecyclerview.AnimatingRecyclerView;
import com(tpb.github.data.APIHandler;
import com(tpb.github.data.FileLoader;
import com(tpb.github.data.Loader;
import com(tpb.github.data.models.content.Node;
import com(tpb.projects.R;
import com(tpb.projects.common BaseActivity;
import com(tpb.projects.util SettingsActivity;
import com(tpb.projects.util UI;

import java.util.ArrayList;
import java.util.List;

import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by theo on 17/02/17.
 */

public class ContentActivity extends BaseActivity implements
Loader.ListLoader<Pair<String, String>> {

 @BindView(R.id.content_title) TextView mTitle;
 @BindView(R.id.content_ribbon_scrollview) HorizontalScrollView
mRibbonScrollView;
 @BindView(R.id.content_file_ribbon) LinearLayout mRibbon;
 @BindView(R.id.content_recycler) AnimatingRecyclerView mRecycler;
 @BindView(R.id.content_refresher) SwipeRefreshLayout mRefresher;
 @BindView(R.id.content_branch_spinner) Spinner mBranchSpinner;

 public static Node mLaunchNode;

 private ContentAdapter mAdapter;
 private List<Pair<String, String>> mBranches;
```

```

private String mDefaultRef;

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 final SettingsActivity.Preferences prefs =
SettingsActivity.Preferences
 .getPreferences(this);
 setTheme(prefs.isDarkThemeEnabled() ? R.style.AppTheme_Dark :
R.style.AppTheme);
 UI.setStatusBarColor(getWindow(),
getResources().getColor(R.color.colorPrimaryDark));
 setContentView(R.layout.activity_content);
 ButterKnife.bind(this);

 initRibbon();

 final Intent launchIntent = getIntent();
 final String repo =
launchIntent.getStringExtra(getString(R.string.intent_repo));
 mTitle.setText(repo.substring(repo.indexOf('/') + 1));

 mAdapter = new ContentAdapter(new FileLoader(this), this, repo, null);
 mRecycler.enableLineDecoration();
 mRecycler.setAdapter(mAdapter);
 mRecycler.setLayoutManager(new LinearLayoutManager(this));
 mRefresher.setOnRefreshListener(() -> mAdapter.reload());
 Loader.getLoader(this).loadBranches(this, repo);
}

@Override
public void listLoadComplete(List<Pair<String, String>> branches) {
 mBranches = branches;
 if(mDefaultRef != null) bindBranches();
}

public void setDefaultRef(String ref) {
 if(mDefaultRef == null) {
 mDefaultRef = ref;
 if(mBranches != null && !mBranches.isEmpty()) {
 bindBranches();
 }
 }
}

public void bindBranches() {
 final List<String> branchNames = new ArrayList<>(mBranches.size());
 for(Pair<String, String> p : mBranches) {
 if(mDefaultRef.equals(p.first)) {
 branchNames.add(0, p.first);
 } else {
 branchNames.add(p.first);
 }
 }
 final ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
 android.R.layout.simple_spinner_item, branchNames
);
}

```

```
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
;
mBranchSpinner.setAdapter(adapter);
if(mBranchSpinner.getOnItemSelectedListener() == null) {
 mBranchSpinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
 @Override
 public void onItemSelected(AdapterView<?> parent, View view,
int position, long id) {
 mAdapter.setRef(branchNames.get(position));
 }

 @Override
 public void onNothingSelected(AdapterView<?> parent) {

 }
 });
}
}

@Override
public void listLoadError(APIHandler.APIError error) {
}

private void initRibbon() {
 final TextView view = (TextView) getLayoutInflater()
 .inflate(R.layout.shard_ribbon_item, mRibbon, false);
 view.setText(R.string.text_ribbon_root);
 view.setOnClickListener((v) -> {
 mRibbon.removeAllViews();
 mRibbon.addView(view);
 mAdapter.moveToStart();
 });
 mRibbon.addView(view);
}

void addRibbonItem(final Node node) {
 final TextView view = (TextView) getLayoutInflater()
 .inflate(R.layout.shard_ribbon_item, mRibbon, false);
 view.setText(node.getName());
 view.setFocusable(false);
 view.setOnClickListener(v -> {
 final ArrayList<View> views = new ArrayList<>();
 for(int i = 0; i <= mRibbon.indexOfChild(view); i++) {
 views.add(mRibbon.getChildAt(i));
 }

 mRibbon.removeAllViews();
 for(View item : views) mRibbon.addView(item);
 mAdapter.moveTo(node);
 });
}

mRibbon.addView(view);
mRibbon.post(() -> mRibbonScrollView.fullScroll(View.FOCUS_RIGHT));

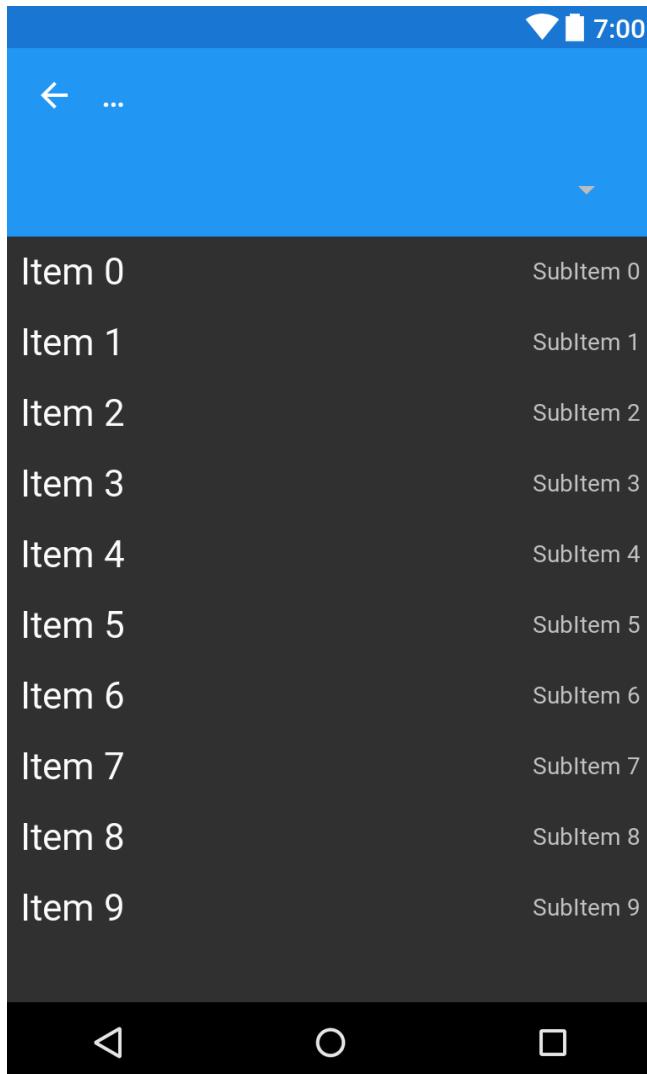
}
```

```
@Override
public void onBackPressed() {
 if(mRibbon.getChildCount() > 1) {
 final ArrayList<View> views = new ArrayList<>();
 for(int i = 0; i < mRibbon.getChildCount() - 1; i++) {
 views.add(mRibbon.getChildAt(i));
 }
 mRibbon.removeAllViews();
 for(View v : views) mRibbon.addView(v);
 mAdapter.moveBack();
 } else {
 super.onBackPressed();
 }
}

@Override
public void onToolbarBackPressed(View view) {
 super.onBackPressed();
}
}
```

initRibbon, addRibbonItem, and onBackPressed deal with navigation through the path.

The Activity layout contains the title bar, the branch spinner, and below both of these the path ribbon.



`initRibbon` adds the base item to the ribbon.

Each item in the ribbon `HorizontalScrollView` is a `TextView` with a chevron at the end.

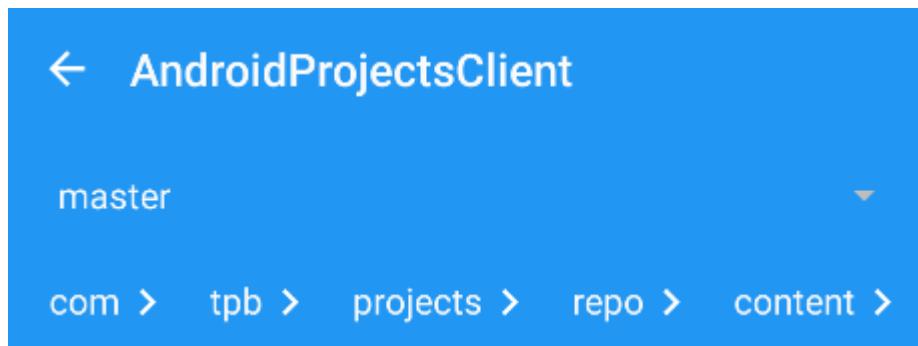
The text of the root item is a "/", indicating the base of the path.

The `OnClickListener` for this `TextView` removes all of the `Views` from the `LinearLayout`, re-adds the base item, and calls `moveToStart` on the adapter. `addRibbonItem` is used to add a new item to the ribbon when a directory is entered.

The method takes a `Node` as a parameter, adds the new `TextView` with an `OnClickListener` to save all of the `TextViews` upto the clicked `TextView`, and remove everything else before calling `moveTo` on the adapter.

A post call is made to scroll the `HorizontalScrollView` to the right once the `TextView` has been added.

When viewing the content directory in this project, the header appears as below:



## ContentAdapter

The ContentAdapter manages loading and traversing the repository file tree. It stores Lists of the root Nodes, the current Nodes, and a single Node which was the last Node to be opened.

### ContentAdapter.java

```
package com(tpb.projects.repo.content;

import android.content.Intent;
import android.support.annotation.Nullable;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import com(tpb.github.data.APIHandler;
import com(tpb.github.data.FileLoader;
import com(tpb.github.data.Loader;
import com(tpb.github.data.models.content.Node;
import com(tpb.projects.R;
import com(tpb.projects.repo.RepoActivity;
import com(tpb.projects.util.Util;

import java.util.ArrayList;
import java.util.List;
import java.util.Stack;

import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by theo on 17/02/17.
 */

public class ContentAdapter extends
RecyclerView.Adapter<ContentAdapter.NodeViewHolder> implements
Loader.ListLoader<Node> {
 private static final String TAG = ContentAdapter.class.getSimpleName();

 private List<Node> mRootNodes = new ArrayList<>();
 private List<Node> mCurrentNodes = new ArrayList<>();
 private Node mPreviousNode;
```

```
private final ContentActivity mParent;
private final String mRepo;
private final FileLoader mLoader;
private boolean mIsLoading = false;
private String mRef;

ContentAdapter(FileLoader loader, ContentActivity parent, String repo,
@Nullable String path) {
 mLoader = loader;
 mParent = parent;
 mRepo = repo;
 mParent.mRefresher.setRefreshing(true);
 mIsLoading = true;
 mLoader.loadDirectory(this, repo, path, null, mRef);
}

@Override
public NodeViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
 return new NodeViewHolder(LayoutInflater.from(parent.getContext()))

.inflate(R.layout.viewholder_node, parent, false));
}

@Override
public void onBindViewHolder(NodeViewHolder holder, int position) {
 if(mCurrentNodes.get(position).getType() == Node.NodeType.SYMLINK) {
 holder.mText.setText(mCurrentNodes.get(position).getPath());
 } else {
 holder.mText.setText(mCurrentNodes.get(position).getName());
 }
 if(mCurrentNodes.get(position).getType() == Node.NodeType.FILE) {
 holder.mText

.setCompoundDrawablesRelativeWithIntrinsicBounds(R.drawable.ic_file, 0, 0, 0);

holder.mSize.setText(Util.formatBytes(mCurrentNodes.get(position).getSize()));
 } else {
 holder.mText

.setCompoundDrawablesRelativeWithIntrinsicBounds(R.drawable.ic_folder, 0, 0,
0);
 holder.mSize.setText("");
 }
 holder.itemView.setOnClickListener(v) ->
loadNode(holder.getAdapterPosition()));
}

void setRef(String ref) {
 if(mRef == null) {
 mRef = ref;
 } else {
 mRef = ref;
 mPreviousNode = null;
 reload();
 }
}

void reload() {
```

```

mCurrentNodes.clear();
notifyDataSetChanged();
if(mPreviousNode == null) {
 mLoader.loadDirectory(this, mRepo, null, null, mRef);
} else {
 mLoader.loadDirectory(this, mRepo, mPreviousNode.getPath(),
mPreviousNode, mRef);
}
}

void moveToStart() {
 mCurrentNodes = mRootNodes;
 mPreviousNode = null;
 notifyDataSetChanged();
}

void moveTo(Node node) {
 mPreviousNode = node;
 mCurrentNodes = node.getChildren();
 notifyDataSetChanged();
}

void moveBack() {
/*
If we are at the root, mPreviousNode is null
If we are one layer down, mPreviousNode is non null, but its parent is
null
If we are further down, both mPreviousNode and its parent are non null
*/
if(mPreviousNode != null) {
 mPreviousNode = mPreviousNode.getParent();
 if(mPreviousNode.getParent() == null) {
 mCurrentNodes = mRootNodes;
 } else {
 mCurrentNodes = mPreviousNode.getChildren();
 }
 notifyDataSetChanged();
}
}

private void loadNode(int pos) {
 if(mIsLoading) return;
 final Node node = mCurrentNodes.get(pos);
 if(node.getType() == Node.NodeType.FILE) {
 ContentActivity.mLaunchNode = node;
 final Intent file = new Intent(mParent, FileActivity.class);
 mParent.startActivity(file);
 } else if(node.getType() == Node.NodeType.SUBMODULE) {
 final Intent i = new Intent(mParent, RepoActivity.class);
 String path = node.getHtmlUrl();
 path = path.substring(path.indexOf("com/") + 4,
path.indexOf("/tree"));
 i.putExtra(mParent.getString(R.string.intent_repo), path);
 mParent.startActivity(i);
 } else {
 mParent.addRibbonItem(node);
 mPreviousNode = node;
 mParent.mRefresher.setRefreshing(true);
 }
}

```

```

 mIsLoading = true;
 if(node.getChildren().size() == 0) {
 mLoader.loadDirectory(this, mRepo, node.getPath(), node,
mRef);
 } else {
 listLoadComplete(node.getChildren());
 }
 }

private Loader.ListLoader<Node> backgroundLoader = new
Loader.ListLoader<Node>() {
 @Override
 public void listLoadComplete(List<Node> directory) {
 if(directory.size() == 0) return;
 final Node parent = directory.get(0).getParent();
 for(Node n : mCurrentNodes) { //Most likely here
 if(parent.equals(n)) {
 n.setChildren(directory);
 return;
 }
 }

 final Stack<Node> stack = new Stack<>();
 Node current;
 for(Node n : mRootNodes) {
 stack.push(n);
 while(!stack.isEmpty()) {
 current = stack.pop();
 for(Node child : current.getChildren()) {
 if(parent.equals(child)) {
 parent.setChildren(directory);
 return;
 } else if(child.getType() == Node.NodeType.DIRECTORY)
{
 stack.push(child);
 }
 }
 }
 }
 }

 @Override
 public void listLoadError(APIHandler.APIError error) {
 }
};

@Override
public void listLoadComplete(List<Node> directory) {
 if(mPreviousNode == null) { //We are at the root
 mRootNodes = directory;
 mCurrentNodes = directory;
 notifyItemRangeInserted(0, mCurrentNodes.size());
 if(mCurrentNodes.size() > 0)
mParent.setDefaultRef(mCurrentNodes.get(0).getRef());
 } else {
 mPreviousNode.setChildren(directory);
 mCurrentNodes = directory;
 }
}

```

```

 notifyDataSetChanged();
 }
 mIsLoading = false;
 mParent.mRefresher.setRefreshing(false);
 for(Node n : directory) {
 if(n.getType() == Node.NodeType.DIRECTORY &&
n.getChildren().size() == 0) {
 mLoader.loadDirectory(backgroundLoader, mRepo, n.getPath(), n,
mRef);
 }
 }
}

@Override
public void listLoadError(APIHandler.APIError error) {
 mIsLoading = false;
 mParent.mRefresher.setRefreshing(false);
}

@Override
public int getItemCount() {
 return mCurrentNodes.size();
}

static class NodeViewHolder extends RecyclerView.ViewHolder {

 @BindView(R.id.node_text) TextView mText;
 @BindView(R.id.node_size) TextView mSize;

 NodeViewHolder(View itemView) {
 super(itemView);
 ButterKnife.bind(this, itemView);
 }
}
}

```

When `loadNode` is called the `Node` type is checked.

## Loading files

If it is a file, `ContentActivity.mLaunchNode` is set to the clicked `Node`.

This may appear strange, given that everywhere else throughout the application, data has been sent between Activities as extras on an Intent.

The problem with this is that `Intents` have a size limit.

The limit has remained nearly constant for the last 6 years, decreasing from around 518600 bytes in Android Gingerbread (API 10) to 517700 bytes in Android Marshmallow.

The GitHub API states that content up to 1MB in size may be included in the JSON, which cannot be passed through an Intent.

The `Node` is therefore passed through a public static member on the `FileActivity`.

This is unlikely to occur anyway, as when a FILE Node is loaded as part of a directory, its content is not included.

It will only be loaded as a full FILE Node if it is loaded separately, which may happen if it is pre-loaded.

## Loading submodule

If a SUBMODULE Node is clicked, the repository that the submodule refers to is launched.

## Loading directories

When a DIRECTORY Node is clicked, the Node is added to the ribbon, and mPreviousNode is set to the Node.

The Node children may already have been loaded, in which case listLoadComplete is called directly with the children.

Otherwise, Loader.loadDirectory is called with the ContentAdapter, repository, node path, the node itself, and the current HEAD reference.

When listLoadComplete is called, there are two possible states:

First, this we may be at the root of the file tree.

In this case mPreviousNode is null.

mRootNodes and mCurrentNodes are set to the loaded List.

notifyItemRangeInserted is then called, and the setDefaultRef is called on the ContentActivity.

Second, we are somewhere else in the tree.

In this case, the children of mPreviousNode are set, and mCurrentNodes is set before notifyDataSetChanged is called.

The loading state is then reset, and background loading begins:

For each Node in the new directory, we check that the Node is a DIRECTORY Node, and that it has no children.

If so, Loader.loadDirectory is called, with the backgroundLoader, an instance of ListLoader<Node> which deals with inserting children in the background.

## Background loading

When listLoadComplete is called on the backgroundLoader, it first checks each of the current Nodes, as the network calls generally return fast enough that the user is yet to navigate to another directory.

If one of the current Nodes is the parent of the first Node in the new directory, its children are set to the new directory and listLoadComplete returns.

Otherwise, the tree must be traversed to find the parent Node.

A Stack is created, and each of the Nodes in mRootNodes are traversed:

- Each Node is added to the Stack.

- While the stack is not empty, the top Node is popped.
- For each child of the popped Node:
  - If the child is the parent, its children are set and listLoadComplete returns
  - Otherwise the if Node is a directory it is pushed to the stack

This background-loading generally ensures that the next level of a directory is loaded before the user clicks on an item.

## Moving to and from nodes

The purposes `moveToStart`, `moveTo`, and `moveBack` are hopefully evident from their names.

`moveToStart` sets `mCurrentNodes` to `mRootNodes`, sets `mPreviousNode` to null, and calls `notifyDataSetChange`, resetting the adapter to its original state.

`moveTo` sets `mPreviousNode` to the Node passed to it, `mCurrentNodes` to the children of the Node passed, and then calls `NotifyDataSetChanged`.

`moveBack` is slightly more complex, as it has to deal with a greater number of states.

If `mPreviousNode` is null, we are already at the root and there is nowhere to move to.

Otherwise, `mPreviousNode` is set to its own parent.

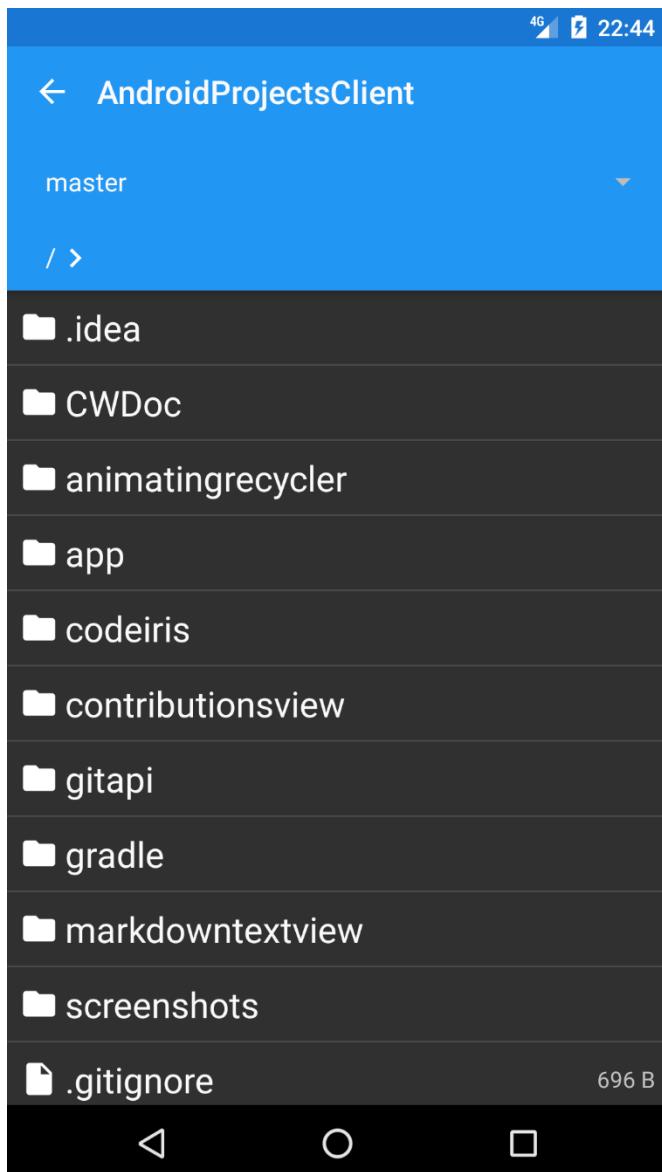
If the parent of `mPreviousNode` is null, we are one step away from the root:

- `mCurrentNodes` is set to `mRootNodes`  
Otherwise, we are deeper in the tree:
- `mCurrentNodes` is set to the children of `mPreviousNode` (Which currently refers to the parent of the Node which was `mPreviousNode` at the start of the method)

`notifyDataSetChanged` is then called.

When displaying the root of the repository for this project,

the `ContentActivity` appears as shown below:



## FileActivity

The `FileActivity` is used to display a file with proper highlighting using `HighlightJS`.

It can be launched with a blob path, a gist URL, or a Node.

When it is launched with a blob path, the repository and blob are split from the full blob path.

The `HighlightJS` language is then set based on the blob string.

`getFileType` first finds the index of the first question mark in the path, which may specify the ref being viewed.

The substring of the path from the last index of ".", to either the index of the question mark or the length of the string is then returned.

The `FileLoader` is then used to load the raw resource.

If the `FileActivity` is started with a gist path, the gist path is passed straight to the `FileLoader`.

Finally, the `FileActivity` may have been launched with a `Node`.

In this case the `Node` encoding is checked. If the encoding exists, the `Node` content is decoded and passed to the `StringRequestListener`.

Otherwise, the `FileLoader` is passed the `Node` download URL to download the file.

## FileActivity.java

```
package com(tpb.projects.repo.content;

import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v7.app.AppCompatActivity;
import android.util.Base64;
import android.view.View;
import android.widget.ProgressBar;
import android.widget.TextView;

import com.androidnetworking.error.ANError;
import com.androidnetworking.interfaces.StringRequestListener;
import com.pddstudio.highlightjs.HighlightJsView;
import com.pddstudio.highlightjs.models.Language;
import com.pddstudio.highlightjs.models.Theme;
import com(tpb.github.data.FileLoader;
import com(tpb.github.data.models.Node;
import com(tpb.projects.R;
import com(tpb.projects.util.SettingsActivity;

import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by theo on 19/02/17.
 */

public class FileActivity extends AppCompatActivity {
 @BindView(R.id.file_name) TextView mName;
 @BindView(R.id.file_webview) HighlightJsView mWebView;
 @BindView(R.id.file_loading_spinner) ProgressBar mSpinner;

 @Override
 protected void onCreate(@Nullable Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 final SettingsActivity.Preferences prefs =
SettingsActivity.Preferences
 .getPreferences(this);
 setTheme(prefs.isDarkThemeEnabled() ? R.style.AppTheme_Dark :
R.style.AppTheme);
 setContentView(R.layout.activity_file);
 ButterKnife.bind(this);

 if(prefs.isDarkThemeEnabled()) {
 mWebView.setTheme(Theme.ANDROID_STUDIO);
 }
 mWebView.setZoomSupportEnabled(true);
 mWebView.setShowLineNumbers(true);
 mWebView.getSettings().setLoadWithOverviewMode(true);
 }
}
```

```

 mWebView.setOnContentChangedListener(() -> {
 mSpinner.setVisibility(View.GONE);
 mWebView.setVisibility(View.VISIBLE);
 });
 final StringRequestListener fileLoadListener = new
StringRequestListener() {
 @Override
 public void onResponse(String response) {
 mWebView.setSource(response.replace("&", "&#"));
 }

 @Override
 public void onError(ANError anError) {
 mSpinner.setVisibility(View.GONE);
 }
 };
 if(getIntent().hasExtra(getString(R.string.intent_blob_path))) {
 final String repo =
getIntent().getStringExtra(getString(R.string.intent_repo));
 final String blob =
getIntent().getStringExtra(getString(R.string.intent_blob_path));
 final int nameStart = blob.lastIndexOf('/') + 1;
 if(nameStart < blob.length()) {
 mName.setText(blob.substring(nameStart));
 }
 mWebView.setHighlightLanguage(getLanguage(getFileType(blob)));
 new FileLoader(this).loadRawFile(fileLoadListener,
 "https://raw.githubusercontent.com/" + repo + blob
);
 } else if(getIntent().hasExtra(getString(R.string.intent_gist_url))) {
 final String url =
getIntent().getStringExtra(getString(R.string.intent_gist_url));
 mWebView.setHighlightLanguage(getLanguage(getFileType(url)));
 new FileLoader(this).loadRawFile(fileLoadListener, url);
 } else if(ContentActivity.mLaunchNode != null) {
 final Node node = ContentActivity.mLaunchNode;
 mName.setText(node.getName());

 mWebView.setHighlightLanguage(getLanguage(getFileType(node.getUrl())));
 if("base64".equals(node.getEncoding())) {
 fileLoadListener.onResponse(new
String(Base64.decode(node.getContent(), Base64.DEFAULT)));
 } else {
 new FileLoader(this).loadRawFile(fileLoadListener,
node.getDownloadUrl());
 }
 } else {
 finish();
 }
}

private static String getFileType(String path) {
 final int qIndex = path.lastIndexOf('?');
 return path.substring(path.lastIndexOf('.') + 1, qIndex > 0 ? qIndex :
path.length());
}

private static Language getLanguage(String lang) {
 for(Language l : Language.values()) {

```

```

 if(l.toString().equalsIgnoreCase(lang)) return 1;
 }
 return Language.AUTO_DETECT;
}

public void onToolbarBackPressed(View view) {
 onBackPressed();
}

}

```

## Objective 5: CommitActivity

---

The `CommitActivity` is used to show detailed information about a commit, as well as handling displaying and commenting upon the commit.

Most of the logic is within the two `Fragments` used.

The `CommitActivity` itself deals only with the initial loading of the `Commit`, which comes either from a `Commit` parceled with the launch `Intent`, or a `Commit` loaded from a repository and a hash.

The `CommitActivity` also manages showing and hiding the `FloatingActionButton` used when adding comments.

The actual displaying of information is handled by two `Fragments`, the `CommitInfoFragment` and the `CommitCommentsFragment`.

The `CommitInfoFragment` displays the information which is stored in the `Commit` model.

A small problem is faced here, as not all `Commit` objects are created equal.

Those loaded in an array within the `RepoCommitsAdapter` do not contain information about the changes that were made during the commit.

To access this information, the `commit` must be re-loaded wherever it originated from.

### `CommitActivity.java`

```

package com(tpb.projects.commits;

import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.design.widget.TabLayout;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;
import android.support.v4.view.ViewPager;
import android.widget.TextView;

import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Loader;
import com(tpb.github.data.models.Commit;
import com(tpb.projects.R;
import com(tpb.projects.commits.fragments.CommitCommentsFragment;
import com(tpb.projects.commits.fragments.CommitInfoFragment;

```

```
import com(tpb.projects.common.CircularRevealActivity;
import com(tpb.projects.common.fab.FloatingActionButton;
import com(tpb.projects.util.SettingsActivity;
import com(tpb.projects.util.UI;

import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by theo on 30/03/17.
 */

public class CommitActivity extends CircularRevealActivity implements
Loader.ItemLoader<Commit> {

 @BindView(R.id.commit_hash) TextView mHash;
 @BindView(R.id.commit_comment_fab) FloatingActionButton mFab;
 @BindView(R.id.commit_fragment_tabs) TabLayout mTabs;
 @BindView(R.id.commit_content_viewpager) ViewPager mPager;

 private CommitPagerAdapter mAdapter;

 private Commit mCommit;

 @Override
 protected void onCreate(@Nullable Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 final SettingsActivity.Preferences prefs =
SettingsActivity.Preferences
 .getPreferences(this);
 setTheme(prefs.isDarkThemeEnabled() ? R.style.AppTheme_Dark :
R.style.AppTheme);
 UI.setStatusBarColor(getWindow(),
getResources().getColor(R.color.colorPrimaryDark));
 setContentView(R.layout.activity_commit);
 ButterKnife.bind(this);

 mAdapter = new CommitPagerAdapter(getSupportFragmentManager());

 mPager.setOffscreenPageLimit(2);
 mPager.setAdapter(mAdapter);
 mTabs.setupWithViewPager(mPager);
 mPager.addOnPageChangeListener(new
ViewPager.SimpleOnPageChangeListener() {
 @Override
 public void onPageSelected(int position) {
 super.onPageSelected(position);
 if(position == 1) {
 mFab.show(true);
 } else {
 mFab.hide(true);
 }
 }
);
 final Intent launchIntent = getIntent();
 if(launchIntent.getStringExtra(getString(R.string.transition_card))) {
 postponeEnterTransition();
 }
 }
}
```

```
 }
 if(launchIntent.hasExtra(getString(R.string.parcel_commit))) {
 mCommit =
 launchIntent.getParcelableExtra(getString(R.string.parcel_commit));
 loadComplete(mCommit);
 Loader.getLoader(this).loadCommit(this, mCommit.getFullRepoName(),
mCommit.getSha());
 } else
 if(launchIntent.getStringExtra(getString(R.string.intent_commit_sha))) {
 Loader.getLoader(this).loadCommit(this,
 launchIntent.getStringExtra(getString(R.string.intent_repo)),
 launchIntent.getStringExtra(getString(R.string.intent_commit_sha)))
);
 }

 }

@Override
public void loadComplete(Commit data) {
 mCommit = data;
 mHash.setText(com.tpb.github.data.Util.shortenSha(mCommit.getSha()));
 mAdapter.notifyCommitLoaded();
}

@Override
public void loadError(APIHandler.APIError error) {
}

private class CommitPagerAdapter extends FragmentPagerAdapter {

 private CommitInfoFragment mInfoFragment;
 private CommitCommentsFragment mCommentsFragment;

 CommitPagerAdapter(FragmentManager fm) {
 super(fm);
 }

 @Override
 public Fragment getItem(int position) {
 if(position == 0) {
 mInfoFragment = CommitInfoFragment.getInstance();
 return mInfoFragment;
 } else {
 mCommentsFragment = CommitCommentsFragment.getInstance();
 if(mFab != null) mCommentsFragment.setFab(mFab);
 return mCommentsFragment;
 }
 }

 void notifyCommitLoaded() {
 if(mInfoFragment != null) mInfoFragment.commitLoaded(mCommit);
 if(mCommentsFragment != null)
mCommentsFragment.commitLoaded(mCommit);
 }

 @Override
```

```

public CharSequence getPageTitle(int position) {
 if(position == 0) {
 return getString(R.string.title_commit_info);
 } else {
 return getString(R.string.title_commit_comments);
 }
}

@Override
public int getCount() {
 return 2;
}
}

```

## Objective 5.a, 5.b, and 5.d: CommitInfoFragment

When a commit is loaded, the CommitInfoFragment displays the commit title, and commiter information.

If the Commit contains a list of files, the number of additions and deletions are also displayed, and the files are passed to the CommitDiffAdapter.

### Objective 5.d: Statuses

If an integration is present for a repository, the status of a commit can be loaded and displayed.

The CompleteStatus model incorporates an overall state, and the individual states of the multiple systems which may exist.

When a non-empty model is returned the status image is set to reflect whether the integrations were successfull, unsuccessfull, or are still working.

The overall status text is set, and the description is built from each of the individual statuses of the different integrations.

### CommitInfoFragment.java

```

package com(tpb.projects.commits.fragments;

import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.widget.NestedScrollView;
import android.support.v4.widget.SwipeRefreshLayout;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.ViewTreeObserver;
import android.widget.ImageView;
import android.widget.TextView;

import com(tpb.animatingrecyclerview.AnimatingRecyclerView;

```

```
import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Loader;
import com(tpb.github.data.models.Commit;
import com(tpb.github.data.models.CompleteStatus;
import com(tpb.github.data.models.Status;
import com(tpb.mdtext.Markdown;
import com(tpb.mdtext.views.MarkdownTextView;
import com(tpb.projects.R;
import com(tpb.projects.commits.CommitDiffAdapter;
import com(tpb.projects.common.FixedLinearLayoutManger;
import com(tpb.projects.common.NetworkImageView;
import com(tpb.projects.flow.IntentHandler;
import com(tpb.projects.markdown.Formatter;
import com(tpb.projects.util.Util;

import java.util.Date;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.Unbinder;

/**
 * Created by theo on 30/03/17.
 */

public class CommitInfoFragment extends CommitFragment {

 private Unbinder unbinder;

 @BindView(R.id.commit_header_card) View mHeader;
 @BindView(R.id.commit_title) MarkdownTextView mTitle;
 @BindView(R.id.commit_user_avatar) NetworkImageView mAvatar;
 @BindView(R.id.commit_info) MarkdownTextView mInfo;
 @BindView(R.id.commit_info_refresher) SwipeRefreshLayout mRefresher;
 @BindView(R.id.commit_info_scrollview) NestedScrollView mScrollView;
 @BindView(R.id.commit_diff_recycler) AnimatingRecyclerView mRecyclerView;

 private CommitDiffAdapter mAdapter;

 public static CommitInfoFragment getInstance() {
 return new CommitInfoFragment();
 }

 @Nullable
 @Override
 public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
 final View view = inflater.inflate(R.layout.fragment_commit_info,
 container, false);
 unbinder = ButterKnife.bind(this, view);
 mRefresher.setRefreshing(true);
 mAdapter = new CommitDiffAdapter();
 mRecyclerView.setLayoutManager(new
FixedLinearLayoutManger(getApplicationContext()));
 mRecyclerView.setAdapter(mAdapter);
 mRefresher.setOnRefreshListener(() -> {
 ButterKnife.findById(getActivity(),
R.id.commit_status).setVisibility(View.GONE);
 mAdapter.clear();
 });
 }
}
```

```
 Loader.getLoader(getContext()).loadCommit(new
Loader.ItemLoader<Commit>() {
 @Override
 public void loadComplete(Commit commit) {
 commitLoaded(commit);
 }

 @Override
 public void loadError(APIHandler.APIError error) {
 mRefresher.setRefreshing(false);
 }
}, mCommit.getFullRepoName(), mCommit.getSha());
});
checkSharedElementEntry();
mAreViewsValid = true;
if(mCommit != null) commitLoaded(mCommit);
return view;
}

@Override
public void commitLoaded(Commit commit) {
 mCommit = commit;
 if(!areViewsValid()) return;
 mTitle.setMarkdown(Formatter.bold(mCommit.getMessage()));
 final String user;
 if(mCommit.getCommitter() != null) {
 mAvatar.setImageUrl(mCommit.getCommitter().getAvatarUrl());
 IntentHandler
 .addOnClickHandler(getActivity(), mAvatar,
mCommit.getCommitter().getLogin());
 user = String.format(getString(R.string.text_md_link),
 mCommit.getCommitter().getLogin(),
 mCommit.getCommitter().getHtmlUrl()
);
 } else {
 user = mCommit.getCommitterName();
 IntentHandler.addOnClickHandler(getActivity(), mAvatar, user);
 }
 if(mCommit.GetFiles() != null) {
 final String commitText =
 "
" +
 getResources()

 .getQuantityString(R.plurals.text_commit_additions,
 mCommit.getAdditions(),
 mCommit.getAdditions()
) +
 "
" +
 getResources().getQuantityString(R.plurals.text_commit_deletions,
 mCommit.getDeletions(),
 mCommit.getDeletions()
) +
 "

" +
 String.format(
 getString(R.string.text_committed_by),
 user,
 Util.formatDateLocally(getContext(),
 new Date(mCommit.getCreatedAt())

```

```

)
);
 mInfo.setMarkdown(Markdown.formatMD(commitText,
mCommit.getFullRepoName()));
 mRefresher.setRefreshing(false);
 mAdapter.setDiffs(mCommit.GetFiles());
}
Loader.getLoader(getContext()).loadCommitStatuses(new
Loader.ItemLoader<CompleteStatus>() {
 @Override
 public void loadComplete(CompleteStatus data) {
 if(data.getTotalCount() == 0) return; //We don't care if there
is no integration
 ButterKnife.findById(getActivity(),
R.id.commit_status).setVisibility(View.VISIBLE);
 final ImageView niv = ButterKnife.findById(getActivity(),
R.id.status_image);
 final TextView status = ButterKnife.findById(getActivity(),
R.id.status_state);
 final TextView desc = ButterKnife.findById(getActivity(),
R.id.status_context);
 if("success".equals(data.getState())) {
 niv.setImageResource(R.drawable.ic_check);
 } else if("pending".equals(data.getState())) {
 niv.setImageResource(R.drawable.ic_loading);
 } else {
 niv.setImageResource(R.drawable.ic_failure);
 }

status.setText(String.format(getString(R.string.text_ci_status),
data.getState()));
 final StringBuilder builder = new StringBuilder();
 if(data.getStatuses() != null) {
 for(Status s : data.getStatuses()) {
 builder.append(
String.format(getString(R.string.text_ci_info),
 s.getContext(),
 s.getDescription()
)
);
 builder.append('\n');
 }
 desc.setText(builder.toString());
 }

 @Override
 public void loadError(APIHandler.APIError error) {
 }
 }, mCommit.getFullRepoName(), mCommit.getSha());
}

private void checkSharedElementEntry() {
 final Intent i = getActivity().getIntent();
 if(i.getStringExtra(getString(R.string.transition_card))) {
 mHeader.getViewTreeObserver()

```

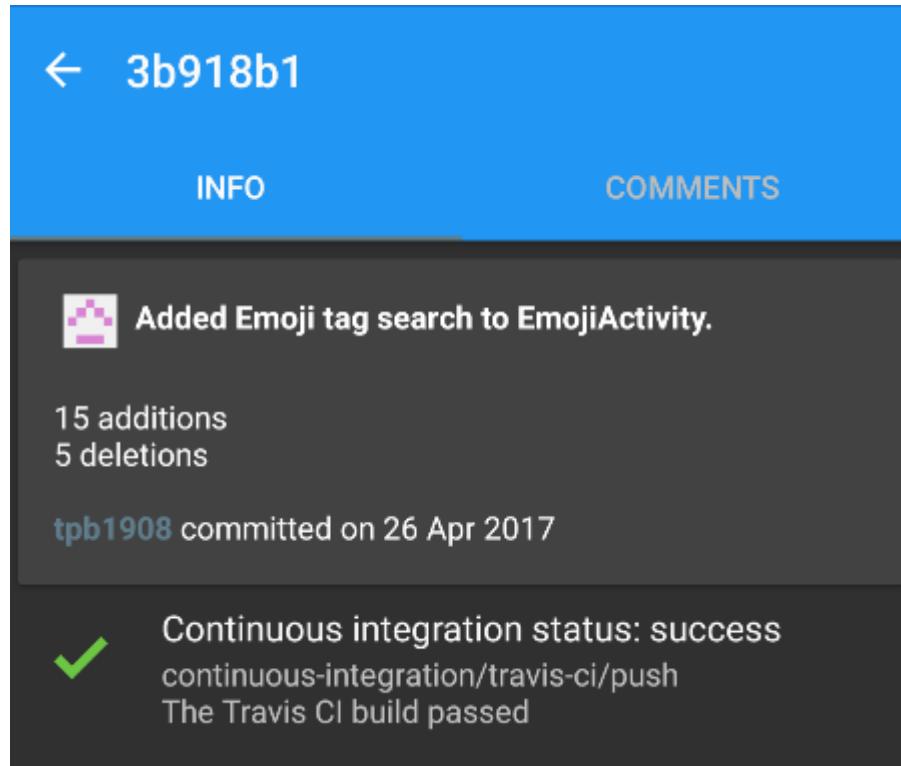
```

 .addOnPreDrawListener(new
ViewTreeObserver.OnPreDrawListener() {
 @Override
 public boolean onPreDraw() {

mHeader.getViewTreeObserver().removeOnPreDrawListener(this);
 if(i.getStringExtra(getString(R.string.intent_drawable)))
{
 mAvatar.setImageBitmap(
i.getParcelableExtra(getString(R.string.intent_drawable)));
 }
 getActivity().startPostponedEnterTransition();
 return true;
 }
});
}
@Override
public void onDestroyView() {
 super.onDestroyView();
 unbinder.unbind();
}
}

```

The primary body of information about a commit may appear as shown below:



### Objective 5.b: CommitDiffAdapter

The `CommitDiffAdapter` is used to display the actual changes made by a `Commit`.

Unlike other adapters, it does not have to deal with repeatedly loading new data, or even caching intensive work, as the information displayed is relatively simple.

Each `DiffFile` model contains a file name, and information about the changes. This information includes:

- The status of the change; whether the file was created, updated, or deleted
- The number of lines added
- The number of lines removed

The `DiffFile` also contains a patch string which displays the changed lines. These strings can be quite large, and as such it would not make sense to display the entire string immediately.

Instead, only the first 3 lines are shown by default and the rest of the patch string can be shown or hidden by clicking on the list item.

The span is built in the `Formatter` class in `buildDiffSpan`.

### **Formatter.java**

```
public static SpannableStringBuilder buildDiffSpan(@NonNull String diff) {
 final SpannableStringBuilder builder = new SpannableStringBuilder();

 int oldLength = 0;
 for(String line : diff.split("\n")) {
 oldLength = builder.length();
 if(line.startsWith("+")) {
 builder.append(line);
 builder.setSpan(new
FullWidthBackgroundColorSpan(Color.parseColor("#8BC34A")),
 oldLength, builder.length(),
 Spanned.SPAN_EXCLUSIVE_EXCLUSIVE
);
 } else if(line.startsWith("-")) {
 builder.append(line);
 builder.setSpan(new
FullWidthBackgroundColorSpan(Color.parseColor("#F44336")),
 oldLength, builder.length(),
 Spanned.SPAN_EXCLUSIVE_EXCLUSIVE
);
 } else {
 builder.append(line);
 builder.setSpan(new
FullWidthBackgroundColorSpan(Color.parseColor("#9E9E9E")),
 oldLength, builder.length(),
 Spanned.SPAN_EXCLUSIVE_EXCLUSIVE
);
 }
 builder.append("\n");
 }
 builder.setSpan(new TypefaceSpan("monospace"), 0, builder.length(),
 Spanned.SPAN_EXCLUSIVE_EXCLUSIVE
);
 return builder;
}
```

This splits the string into individual lines and colours each line depending on whether it begins with a "+" or a "-".

The `FullWidthBackgroundColorSpan` used is not part of the the `markdown` package as it is only used here.

It extends `LineBackgroundSpan` and draws an opaque rectangle across the entire line.

```
private static class FullWidthBackgroundColorSpan implements
LineBackgroundSpan {
 private final int color;

 FullWidthBackgroundColorSpan(int color) {
 this.color = color;
 }

 @Override
 public void drawBackground(Canvas c, Paint p, int left, int right, int
top, int baseline,
 int bottom, CharSequence text, int start, int
end, int lnum) {
 final int paintColor = p.getColor();
 p.setColor(color);
 p.setAlpha(128);
 c.drawRect(new Rect(left, top, right, bottom), p);
 p.setColor(paintColor);
 }
}
```

The show and hide animations are created using `ObjectAnimators` which change the `maxLines` attribute of the `TextViews` displaying the patch strings.

### **CommitDiffAdapter.java**

```
package com(tpb.projects.commits;

import android.animation.ObjectAnimator;
import android.content.res.Resources;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import com(tpb.github.data.models.DiffFile;
import com(tpb.projects.R;
import com(tpb.projects.markdown.Formatter;

import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by theo on 01/04/17.
 */

public class CommitDiffAdapter extends
RecyclerView.Adapter<CommitDiffAdapter.DiffHolder> {

 private DiffFile[] mDiffs = new DiffFile[0];
```

```
public void setDiffs(DiffFile[] diffs) {
 mDiffs = diffs;
 notifyItemRangeInserted(0, mDiffs.length);
}

public void clear() {
 final int size = mDiffs.length;
 mDiffs = new DiffFile[0];
 notifyItemRangeRemoved(0, size);
}

@Override
public DiffHolder onCreateViewHolder(ViewGroup parent, int viewType) {
 return new DiffHolder(LayoutInflater.from(parent.getContext())
 .inflate(R.layout.viewholder_diff,
parent, false));
}

@Override
public void onBindViewHolder(DiffHolder holder, int position) {
 holder.mFileName.setText(mDiffs[position].getFileName());
 final Resources res = holder.itemView.getResources();
 holder.mInfo.setText(
 String.format(
 res.getString(R.string.text_diff_changes),
 mDiffs[position].getStatus(),
 res.getQuantityString(R.plurals.text_commit_additions,
 mDiffs[position].getAdditions(),
 mDiffs[position].getAdditions()
),
 res.getQuantityString(R.plurals.text_commit_deletions,
 mDiffs[position].getDeletions(),
 mDiffs[position].getDeletions()
)
)
);
 if(mDiffs[position].getPatch() != null) {
 holder.mDiff.setVisibility(View.VISIBLE);

 holder.mDiff.setText(Formatter.buildDiffSpan(mDiffs[position].getPatch()));
 holder.mDiff.post(() -> {
 final int maxLines = holder.mDiff.getLineCount();
 holder.mDiff.setMaxLines(3);
 holder.itemView.setOnClickListener(v -> {
 if(holder.mDiff.getLineCount() < maxLines) {
 ObjectAnimator.ofInt(
 holder.mDiff,
 "maxLines",
 3,
 maxLines
).setDuration(res.getInteger(android.R.integer.config_mediumAnimTime()))
 .start();
 } else {
 ObjectAnimator.ofInt(
 holder.mDiff,
 "maxLines",
 maxLines,
).setDuration(res.getInteger(android.R.integer.config_mediumAnimTime()))
 .start();
 }
 });
 });
 }
}
```

3

```

).setDuration(res.getInteger(android.R.integer.config_mediumAnimTime))
 .start();
 }
});

} else {
 holder.mDiff.setVisibility(View.GONE);
}
}

@Override
public int getItemCount() {
 return mDiffs.length;
}

static class DiffHolder extends RecyclerView.ViewHolder {

 @BindView(R.id.diff_filename) TextView mFileName;
 @BindView(R.id.diff_info) TextView mInfo;
 @BindView(R.id.diff_diff) TextView mDiff;

 DiffHolder(View itemView) {
 super(itemView);
 ButterKnife.bind(this, itemView);
 }
}
}

```

Short span	Expanded span
<pre> app/src/main/java/com/tpb/projects/editors/ EmojiActivity.java modified with 10 additions and 0 deletions @@ -87,12 +87,22 @@ void filter(String query) {     mFilteredEmojis.addAll(mE... </pre>	<pre> app/src/main/java/com/tpb/projects/editors/ EmojiActivity.java modified with 10 additions and 0 deletions @@ -87,12 +87,22 @@ void filter(String query) {     mFilteredEmojis.addAll(mE... +         boolean added = false; +         for(String s : e.getAliases()) { +             if(s.contains(query)) { +                 mFilteredEmojis.add(e); +                 added = true; +                 break; +             } +         } +         if(!added) { +             for(String s : e.getTags()) { +                 if(s.contains(query)) { +                     mFilteredEmojis.add(e); +                     break; +                 } +             } } notifyDataSetChanged(); </pre>

## Objective 5.c: CommitCommentsFragment

The `CommitCommentsFragment` inflates the `fragment_recycler` layout to display a `Recyclerview` showing any comments on the commit. It also manages launching the `CommentEditor` to allow the user to create or edit comments, and then to perform the relevant requests.

### CommitCommentsFragment.java

```
package com(tpb.projects.commits.fragments;

import android.app.Activity;
import android.app.Dialog;
import android.content.ClipData;
import android.content.ClipboardManager;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.View;
import android.view.ViewGroup;
import android.widget.PopupMenu;
import android.widget.Toast;

import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Editor;
import com(tpb.github.data.auth.GitHubSession;
import com(tpb.github.data.models.Comment;
import com(tpb.github.data.models.Commit;
import com(tpb.projects.R;
import com(tpb.projects.commits.CommitCommentsAdapter;
import com(tpb.projects.common.FixedLinearLayoutManger;
import com(tpb.projects.common.fab.FloatingActionButton;
import com(tpb.projects.editors.CommentEditor;
import com(tpb.projects.flow.IntentHandler;
import com(tpb.projects.util.UI;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.Unbinder;

/**
 * Created by theo on 30/03/17.
 */

public class CommitCommentsFragment extends CommitFragment {

 private Unbinder unbinder;
 @BindView(R.id.fragment_recycler) RecyclerView mRecycler;
```

```
@BindView(R.id.fragment_refresher) SwipeRefreshLayout mRefresher;
private FloatingActionButton mFab;

private Editor mEditor;

private CommitCommentsAdapter mAdapter;

public static CommitCommentsFragment getInstance() {
 return new CommitCommentsFragment();
}

@Override
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
 final View view = inflater.inflate(R.layout.fragment_recycler,
 container, false);
 unbinder = ButterKnife.bind(this, view);
 mEditor = Editor.getEditor(getContext());
 mAdapter = new CommitCommentsAdapter(this, mRefresher);
 mRecycler.setLayoutManager(new FixedLinearLayoutManger(getContext()));
 mRecycler.setAdapter(mAdapter);

 mAreViewsValid = true;
 if(mFab != null) addListeners();
 if(mCommit != null) commitLoaded(mCommit);
 return view;
}

public void setFab(FloatingActionButton fab) {
 mFab = fab;
 if(mAreViewsValid) addListeners();
}

private void addListeners() {
 final LinearLayoutManager manager = (LinearLayoutManager)
 mRecycler.getLayoutManager();
 mRecycler.addOnScrollListener(new RecyclerView.OnScrollListener() {
 @Override
 public void onScrolled(RecyclerView recyclerView, int dx, int dy)
 {
 super.onScrolled(recyclerView, dx, dy);
 if(manager.findFirstVisibleItemPosition() + 20 >
 manager.getItemCount())
 mAdapter.notifyBottomReached();
 }
 if(dy > 10) {
 mFab.hide(true);
 } else if(dy < -10) {
 mFab.show(true);
 }
}
});
mFab.setOnClickListener(v -> {
 final Intent i = new Intent(getContext(), CommentEditor.class);
 UI.setViewPositionForIntent(i, mFab);
 startActivityForResult(i, CommentEditor.REQUEST_CODE_NEW_COMMENT);
});
}
```

```
@Override
public void onAttach(Activity activity) {
 super.onAttach(activity);
 if(mFab != null && mAreViewsValid) addListeners();
}

@Override
public void commitLoaded(Commit commit) {
 mCommit = commit;
 if(!areViewsValid()) return;
 mAdapter.setCommit(mCommit);
}

private void createComment(Comment comment) {
 mRefresher.setRefreshing(true);
 mEditor.createCommitComment(new Editor.CreationListener<Comment>() {
 @Override
 public void created(Comment comment) {
 mRefresher.setRefreshing(false);
 mAdapter.addComment(comment);
 mRecycler.post(() ->
mRecycler.smoothScrollToPosition(mAdapter.getItemCount()));
 }

 @Override
 public void creationError(APIHandler.APIError error) {
 mRefresher.setRefreshing(false);
 }
 }, mCommit.getFullRepoName(), mCommit.getSha(), comment.getBody());
}

private void editComment(Comment comment) {
 mRefresher.setRefreshing(true);
 mEditor.updateCommitComment(new Editor.UpdateListener<Comment>() {
 @Override
 public void updated(Comment comment) {
 mRefresher.setRefreshing(false);
 mAdapter.updateComment(comment);
 }

 @Override
 public void updateError(APIHandler.APIError error) {
 mRefresher.setRefreshing(false);
 }
 }, mCommit.getFullRepoName(), comment.getId(), comment.getBody());
}

void removeComment(Comment comment) {
 final AlertDialog.Builder builder = new
AlertDialog.Builder(getContext());
 builder.setTitle(R.string.title_delete_comment);
 builder.setPositiveButton(R.string.action_yes, (dialogInterface, i) ->
{
 mRefresher.setRefreshing(true);
 mEditor.deleteCommitComment(new Editor.DeletionListener<Integer>()
{
 @Override
 public void deleted(Integer id) {
```

```
 mRefresher.setRefreshing(false);
 mAdapter.removeComment(id);
 }

 @Override
 public void deletionError(APIHandler.APIError error) {
 mRefresher.setRefreshing(false);
 }
}, mCommit.getFullRepoName(), comment.getId());
});
builder.setNegativeButton(R.string.action_no, null);
final Dialog deleteDialog = builder.create();
deleteDialog.getWindow().getAttributes().windowAnimations =
R.style.DialogAnimation;
deleteDialog.show();
}

public void displayCommentMenu(View view, Comment comment) {
 final PopupMenu menu = new PopupMenu(getContext(), view);
 menu.inflate(R.menu.menu_comment);
 if(comment.getUser().getLogin().equals(
 GitHubSession.getSession(getContext()).getUserLogin())) {
 menu.getMenu()
 .add(0, R.id.menu_edit_comment, Menu.NONE,
getString(R.string.menu_edit_comment));
 menu.getMenu().add(0, R.id.menu_delete_comment, Menu.NONE,
 getString(R.string.menu_delete_comment))
);
}
menu.setOnMenuItemClickListener(menuItem -> {
 switch(menuItem.getItemId()) {
 case R.id.menu_edit_comment:
 final Intent i = new Intent(getContext(),
CommentEditor.class);
 i.putExtra(getString(R.string.parcel_comment), comment);
 UI.setViewPositionForIntent(i, view);
 startActivityForResult(i,
CommentEditor.REQUEST_CODE_EDIT_COMMENT);
 break;
 case R.id.menu_delete_comment:
 removeComment(comment);
 break;
 case R.id.menu_copy_comment_text:
 final ClipboardManager cm = (ClipboardManager)
getActivity().getSystemService(
 Context.CLIPBOARD_SERVICE);
 cm.setPrimaryClip(ClipData.newPlainText("Comment",
comment.getBody()));
 Toast.makeText(getContext(),
getString(R.string.text_copied_to_board),
 Toast.LENGTH_SHORT
).show();
 break;
 case R.id.menu_fullscreen:
 IntentHandler.showFullScreen(getContext(),
comment.getBody(),
 mCommit.getFullRepoName(), getFragmentManager()
);
 break;
 }
}
```

```

 }
 return false;
 });
 menu.show();
}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data)
{
 super.onActivityResult(requestCode, resultCode, data);
 if(resultCode == AppCompatActivity.RESULT_OK) {
 final Comment comment =
data.getParcelableExtra(getString(R.string.parcel_comment));
 if(requestCode == CommentEditor.REQUEST_CODE_NEW_COMMENT) {
 createComment(comment);
 } else if(requestCode == CommentEditor.REQUEST_CODE_EDIT_COMMENT)
{
 editComment(comment);
 }
 }
}

@Override
public void onDestroyView() {
 super.onDestroyView();
 unbinder.unbind();
}
}

```

`displayCommentMenu` is called from the `CommitCommentsAdapter` and inflates a `PopupMenu` with options to copy the comment text to the clipboard or show the comment in fullscreen.

If the authenticated user is the same as the creator of the comment, options are also displayed to edit or delete the comment.

`removeComment` is called if the delete option is clicked.

A confirmation dialog is shown, and if the user confirms their action the comment is deleted and removed from the adapter.

If the edit comment option is selected, the `CommentEditor` is created with the `REQUEST_CODE_EDIT_COMMENT` request code.

If the result is `RESULT_OK` `editComment` is called, which makes the call to update the comment, and updates the adapter with the new `Comment`.

`createComment` is called when a result is returned after the `CommentEditor` was launched from the `FloatingActionButton`.

It makes a call to `Editor.createCommitComment`, adds the result to the adapter, and posts to the `RecyclerView` to scroll to the bottom.

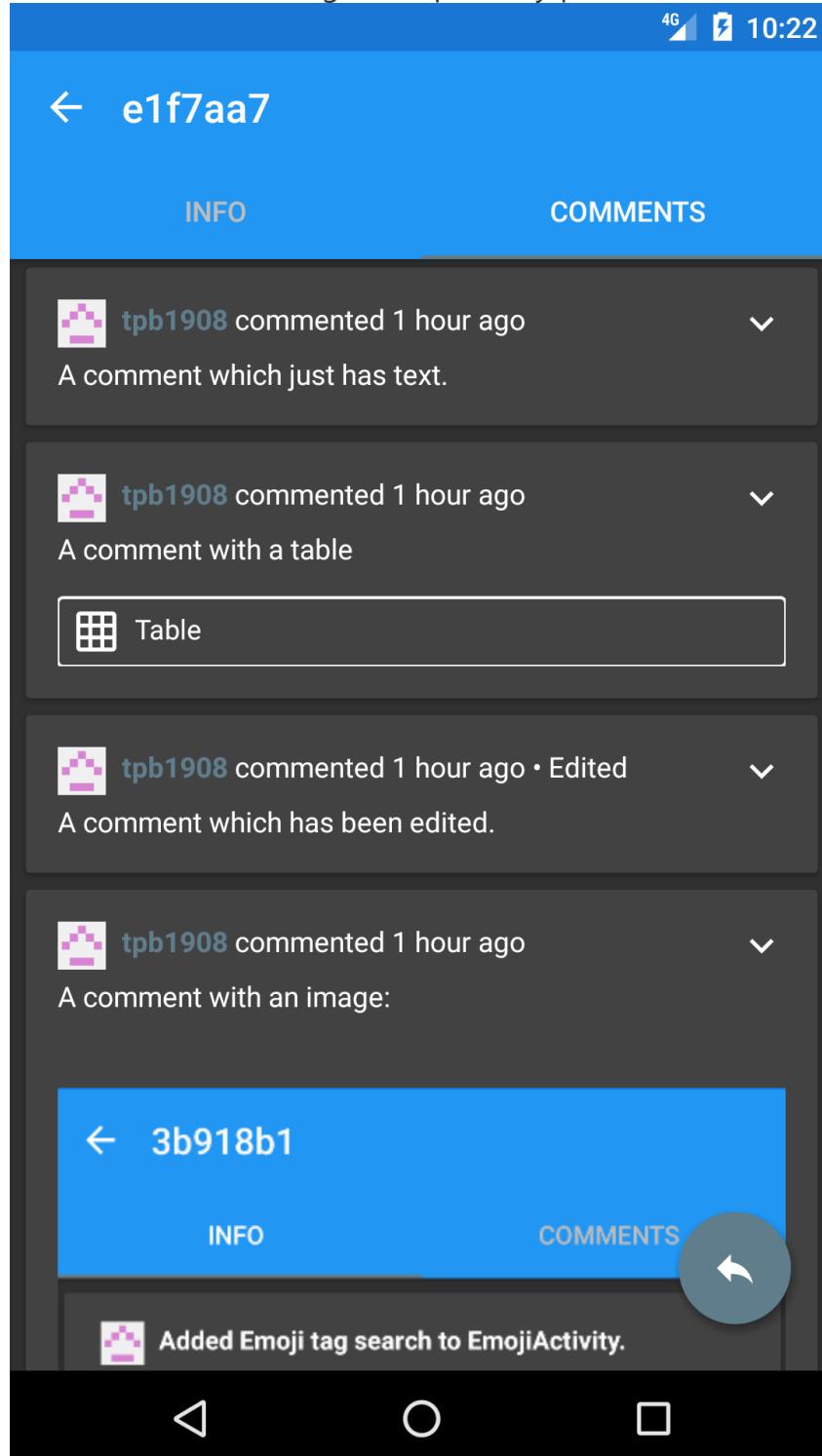
## CommitCommentsAdapter

The `CommitCommentsAdapter` loads and binds each comment, as well as adding, updating, and removing user created comments.

onBindViewHolder builds the span with information about the comment, such as the commenter, the comment time, whether the comment has been edited, and reactions to the comment.

The result is then cached with the Comment.

Comments in the CommitCommentAdapter are formatted in the same way as any other markdown, using the repository path for the Commit.



```
package com(tpb.projects.commits;

import android.support.v4.util.Pair;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.widget.RecyclerView;
import android.text.SpannableString;
import android.text.format.DateUtils;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageButton;

import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Loader;
import com(tpb.github.data.models.Comment;
import com(tpb.github.data.models.Commit;
import com(tpb.mdtext.Markdown;
import com(tpb.mdtext.imagegetter.HttpImageGetter;
import com(tpb.mdtext.views.MarkdownTextView;
import com(tpb.projects.R;
import com(tpb.projects.commits.fragments.CommitCommentsFragment;
import com(tpb.projects.common.NetworkImageView;
import com(tpb.projects.flow.IntentHandler;
import com(tpb.projects.markdown.Formatter;

import java.util.ArrayList;
import java.util.List;

import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by theo on 01/04/17.
 */

public class CommitCommentsAdapter extends
RecyclerView.Adapter<CommitCommentsAdapter.CommentHolder> implements
Loader.ListLoader<Comment> {

 private final ArrayList<Pair<Comment, SpannableString>> mComments = new
ArrayList<>();
 private Commit mCommit;
 private final CommitCommentsFragment mParent;

 private int mPage = 1;
 private boolean mIsLoading = false;
 private boolean mMaxPageReached = false;

 private SwipeRefreshLayout mRefresher;
 private Loader mLoader;

 public CommitCommentsAdapter(CommitCommentsFragment parent,
SwipeRefreshLayout refresher) {
 mParent = parent;
 mRefresher = refresher;
 mLoader = Loader.getLoader(mParent.getContext());
 mRefresher.setOnRefreshListener(() -> {
 mPage = 1;
 mMaxPageReached = false;
 });
 }
}
```

```
 clear();
 loadComments(true);
 });

public void clear() {
 final int oldSize = mComments.size();
 mComments.clear();
 notifyItemRangeRemoved(0, oldSize);
}

public void setCommit(Commit commit) {
 mCommit = commit;
 clear();
 mPage = 1;
 mLoader.loadCommitComments(this, mCommit.getFullRepoName(),
mCommit.getSha(), mPage);
}

@Override
public void listLoadComplete(List<Comment> comments) {
 mRefresher.setRefreshing(false);
 mIsLoading = false;
 if(comments.size() > 0) {
 final int oldLength = mComments.size();
 for(Comment c : comments) {
 mComments.add(Pair.create(c, null));
 }
 notifyItemRangeInserted(oldLength, mComments.size());
 } else {
 mMaxPageReached = true;
 }
}

@Override
public void listLoadError(APIHandler.APIError error) {
 mRefresher.setRefreshing(false);
}

public void notifyBottomReached() {
 if(!mIsLoading && !mMaxPageReached) {
 mPage++;
 loadComments(false);
 }
}

private void loadComments(boolean resetPage) {
 mIsLoading = true;
 mRefresher.setRefreshing(true);
 if(resetPage) {
 mPage = 1;
 mMaxPageReached = false;
 }
 mLoader.loadCommitComments(this, mCommit.getFullRepoName(),
mCommit.getSha(), mPage);
}

public void addComment(Comment comment) {
```

```
 mComments.add(Pair.create(comment, null));
 notifyItemInserted(mComments.size());
 }

 public void removeComment(int commentId) {
 for(int i = 0; i < mComments.size(); i++) {
 if(mComments.get(i).first.getId() == commentId) {
 mComments.remove(i);
 notifyItemRemoved(i);
 break;
 }
 }
 }

 public void updateComment(Comment comment) {
 for(int i = 0; i < mComments.size(); i++) {
 if(mComments.get(i).first.getId() == comment.getId()) {
 mComments.set(i, Pair.create(comment, null));
 notifyItemChanged(i);
 break;
 }
 }
 }

 @Override
 public CommentHolder onCreateViewHolder(ViewGroup parent, int viewType) {
 return new CommentHolder(LayoutInflater.from(parent.getContext()))

 .inflate(R.layout.viewholder_comment, parent,
 false
));
 }

 @Override
 public void onBindViewHolder(CommentHolder holder, int position) {
 final int pos = holder.getAdapterPosition();
 final Comment comment = mComments.get(pos).first;
 if(mComments.get(pos).second == null) {
 holder.mAvatar.setImageUrl(comment.getUser().getAvatarUrl());
 final StringBuilder builder = new StringBuilder();
 builder.append(String.format(
 holder.itemView.getResources().getString(R.string.text_comment_by),
 String.format(
 holder.itemView.getResources().getString(R.string.text_href),
 comment.getUser().getHtmlUrl(),
 comment.getUser().getLogin()
),
 DateUtils.getRelativeTimeSpanString(comment.getCreatedAt())
));
 if(comment.getUpdatedAt() != comment.getCreatedAt()) {
 builder.append(" • ");
 builder.append(holder.itemView.getResources()
 .getString(R.string.text_comment_edited));
 }
 }
 }
}
```

```

 holder.mCommenter.setMarkdown(builder.toString());
 builder.setLength(0);
 builder.append(Markdown.formatMD(comment.getBody(),
mCommit.getFullRepoName()));
 if(comment.hasReaction()) {
 builder.append("\n");
 builder.append(Formatter.reactions(comment.getReaction()));
 }

 holder.mBody.setMarkdown(
 builder.toString(),
 new HttpImageGetter(holder.mBody),
 text -> mComments.set(pos, Pair.create(comment, text))
);
 } else {
 holder.mAvatar.setImageUrl(comment.getUser().getAvatarUrl());
 holder.mBody.setText(mComments.get(pos).second);
 }
 IntentHandler.addOnClickHandler(mParent.getActivity(), holder.mBody);
 IntentHandler.addOnClickHandler(mParent.getActivity(), holder.mAvatar,
 comment.getUser().getLogin()
);
 holder.mMenu.setOnClickListener((v) -> displayMenu(v,
holder.getAdapterPosition()));
}

@Override
public int getItemCount() {
 return mComments.size();
}

private void displayMenu(View view, int pos) {
 mParent.displayCommentMenu(view, mComments.get(pos).first);
}

static class CommentHolder extends RecyclerView.ViewHolder {
 @BindView(R.id.event_comment_avatar) NetworkImageView mAvatar;
 @BindView(R.id.comment_commenter) MarkdownTextView mCommenter;
 @BindView(R.id.comment_text) MarkdownTextView mBody;
 @BindView(R.id.comment_menu_button) ImageButton mMenu;

 CommentHolder(View view) {
 super(view);
 ButterKnife.bind(this, view);
 }
}
}

```

## Objective 4: IssueActivity

---

The IssueActivity and its Fragments are structured in a similar manner to the CommitActivity, except that the adapter for Events is more complicated than the adapter for DiffFiles.

## IssueActivity.java

```
package com(tpb.projects.issues;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.design.widget.AppBarLayout;
import android.support.design.widget.TabLayout;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;
import android.support.v4.view.ViewPager;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;

import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Loader;
import com(tpb.github.data.auth.GitHubSession;
import com(tpb.github.data.models.Issue;
import com(tpb.github.data.models.Repository;
import com(tpb.projects.R;
import com(tpb.projects.common.CircularRevealActivity;
import com(tpb.projects.common.LockableViewPager;
import com(tpb.projects.common.ShortcutDialog;
import com(tpb.projects.common.fab.FloatingActionButton;
import com(tpb.projects.editors.CommentEditor;
import com(tpb.projects.issues.fragments.IssueCommentsFragment;
import com(tpb.projects.issues.fragments.IssueInfoFragment;
import com(tpb.projects.util.SettingsActivity;
import com(tpb.projects.util.UI;

import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by theo on 15/03/17.
 */

public class IssueActivity extends CircularRevealActivity implements
Loader.ItemLoader<Issue>{
 private static final String TAG = IssueActivity.class.getSimpleName();
 private static final String URL =
"https://raw.githubusercontent.com/tpb1908/AndroidProjectsClient/master/app/src/main/java/com/tpb/projects/issues/IssueActivity.java";

 @BindView(R.id.issue_appbar) AppBarLayout mAppbar;
 @BindView(R.id.issue_toolbar) Toolbar mToolbar;
 @BindView(R.id.issue_content_viewpager) LockableViewPager mPager;
 @BindView(R.id.issue_fragment_tabs) TabLayout mTabs;
 @BindView(R.id.issue_number) TextView mNumber;
```

```
@BindView(R.id.issue_comment_fab) FloatingActionButton mFab;

private Loader mLoader;

private IssueFragmentAdapter mAdapter;

private Issue mIssue;
public Repository.AccessLevel mAccessLevel = Repository.AccessLevel.NONE;

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 final SettingsActivity.Preferences prefs =
SettingsActivity.Preferences
 .getPreferences(this);
 setTheme(prefs.isDarkThemeEnabled() ? R.style.AppTheme_Dark :
R.style.AppTheme);
 UI.setStatusBarColor(getWindow(),
getResources().getColor(R.color.colorPrimaryDark));
 setContentView(R.layout.activity_issue);
 ButterKnife.bind(this);
 setSupportActionBar(mToolbar);
 getSupportActionBar().setDisplayHomeAsUpEnabled(false);

 mLoader = Loader.getLoader(this);
 mAdapter = new IssueFragmentAdapter(getSupportFragmentManager());
 final Intent launchIntent = getIntent();
 if(launchIntent.getStringExtra(getString(R.string.transition_card))) {
 postponeEnterTransition();
 }
 if(launchIntent.getExtras() != null &&
launchIntent.getExtras().containsKey(
 getString(R.string.parcel_issue))) {
 mIssue =
 launchIntent.getParcelable(getString(R.string.parcel_issue));

 loadComplete(mIssue);
 } else {
 final int issueNumber = launchIntent
 .getIntExtra(getString(R.string.intent_issue_number), -1);
 final String repoFullName = launchIntent
 .getStringExtra(getString(R.string.intent_repo));
 mLoader.loadIssue(this, repoFullName, issueNumber, true);
 }

 mPager.setOffscreenPageLimit(2);
 mPager.setAdapter(mAdapter);
 mTabs.setupWithViewPager(mPager);
 mPager.addOnPageChangeListener(new
ViewPager.SimpleOnPageChangeListener() {
 @Override
 public void onPageSelected(int position) {
 super.onPageSelected(position);
 if(position == 1 && mIssue != null && !mIssue.isLocked()) {
 mFab.show(true);
 } else {
 mFab.hide(true);
 }
 }
 })
}
```

```
 });

 }

@Override
public void loadComplete(Issue issue) {
 mIssue = issue;
 mNumber.setText(String.format("#%1$s", issue.getNumber()));
 final String login =
GitHubSession.getSession(IssueActivity.this).getUserLogin();
 if(mIssue.getOpenedBy().getLogin().equals(login)) {
 mAccessLevel = Repository.AccessLevel.ADMIN;
 if(mAdapter.mInfoFragment != null)
mAdapter.mInfoFragment.setAccessLevel(mAccessLevel);
 } else {
 mLoader.checkIfCollaborator(new
Loader.ItemLoader<Repository.AccessLevel>() {
 @Override
 public void loadComplete(Repository.AccessLevel data) {
 mAccessLevel = data;
 if(mAdapter.mInfoFragment != null) {
 mAdapter.mInfoFragment.setAccessLevel(mAccessLevel);
 }
 }
 });
 }
}

@Override
public void loadError(APIHandler.APIError error) {

 },
GitHubSession.getSession(this).getUserLogin(),
mIssue.getRepoFullName());
}
mAdapter.setIssue();
}

@Override
public void loadError(APIHandler.APIError error) {

}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
 super.onActivityResult(requestCode, resultCode, data);
 if(resultCode == AppCompatActivity.RESULT_OK) {
 if(requestCode == CommentEditor.REQUEST_CODE_COMMENT_FOR_STATE) {
 mAdapter.mCommentsFragment.createCommentForState(
 data.getParcelableExtra(
 getString(R.string.parcel_comment)
)
);
 }
 }
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
 getMenuInflater().inflate(R.menu.menu_activity, menu);
 return true;
}
```

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
 switch(item.getItemId()) {
 case R.id.menu_settings:
 startActivity(new Intent(IssueActivity.this,
SettingsActivity.class));
 break;
 case R.id.menu_source:
 startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse(URL)));
 break;
 case R.id.menu_share:
 if(mIssue != null) {
 final Intent share = new Intent();
 share.setAction(Intent.ACTION_SEND);
 share.putExtra(Intent.EXTRA_TEXT,
 "https://github.com/" + mIssue.getRepoFullName() +
"/issues/" + mIssue
 .getNumber()
);
 share.setType("text/plain");
 startActivity(share);
 }
 break;
 case R.id.menu_save_to_homescreen:
 final ShortcutDialog dialog = new ShortcutDialog();
 final Bundle args = new Bundle();
 args.putInt(getString(R.string.intent_title_res),
 R.string.title_save_issue_shortcut
);
 args.putBoolean(getString(R.string.intent_drawable), false);
 args.putString(getString(R.string.intent_name), "#" +
mIssue.getNumber());

 dialog.setArguments(args);
 dialog.setListener((name, iconFlag) -> {
 final Intent i = new Intent(getApplicationContext(),
IssueActivity.class);
 i.putExtra(getString(R.string.intent_repo),
mIssue.getRepoFullName());
 i.putExtra(getString(R.string.intent_issue_number),
mIssue.getNumber());

 final Intent add = new Intent();
 add.putExtra(Intent.EXTRA_SHORTCUT_INTENT, i);
 add.putExtra(Intent.EXTRA_SHORTCUT_NAME, name);
 add.putExtra(Intent.EXTRA_SHORTCUT_ICON_RESOURCE,
Intent.ShortcutIconResource
 .fromContext(getApplicationContext(),
R.mipmap.ic_launcher));
 add.putExtra("duplicate", false);

 add.setAction("com.android.launcher.action.INSTALL_SHORTCUT");
 getApplicationContext().sendBroadcast(add);
 });
 dialog.show(getSupportFragmentManager(), TAG);
 break;
 }
 return true;
}
```

```

}

private class IssueFragmentAdapter extends FragmentPagerAdapter {

 IssueCommentsFragment mCommentsFragment;
 IssueInfoFragment mInfoFragment;

 IssueFragmentAdapter(FragmentManager fm) {
 super(fm);
 }

 @Override
 public Fragment getItem(int position) {
 if(position == 0) {
 mInfoFragment = IssueInfoFragment.getInstance();
 if(mIssue != null) mInfoFragment.issueLoaded(mIssue);
 return mInfoFragment;
 } else {
 mCommentsFragment = IssueCommentsFragment.getInstance(mFab);
 if(mIssue != null) mCommentsFragment.issueLoaded(mIssue);
 return mCommentsFragment;
 }
 }

 void setIssue() {
 if(mCommentsFragment != null)
mCommentsFragment.issueLoaded(mIssue);
 if(mInfoFragment != null) mInfoFragment.issueLoaded(mIssue);
 }

 @Override
 public CharSequence getPageTitle(int position) {
 if(position == 0) {
 return getString(R.string.title_issue_info_fragment);
 } else {
 return getString(R.string.title_issue_comments_fragment);
 }
 }

 @Override
 public int getCount() {
 return 2;
 }
}
}

```

The `IssueActivity` is launched with either a parcelled `Issue`, or an issue number and a repository name.

Once an `Issue` has been loaded, a request is made to check the level of access that a user has to a particular issue.

If it is in their repository, they are able to edit other user's comments.

The `IssueActivity` also manages its overflow menu, with the ability to save a shortcut to a particular issue to the homescreen.

## Objective 4.a: IssueInfoFragment

The `IssueInfoFragment` is responsible for displaying the following information:

- The issue title (4.a.i)
- The issue state (4.a.iii)
- The issue body (4.a.iv)
- The user that created the issue (4.a.v)
- The date that the issue was opened ([4.a.vi](#))
- The user(s) assigned to the issue, if they exist (4.a.vii)
- The labels added to the issue (4.a.viii)
- The milestone that the issue is attached to, if applicable

It should also display the events which have occurred on the issue, via the `IssueEventsAdapter`.

The `IssueInfoFragment` also implements objective 4.c, editing the issue.

## **IssueInfoFragment.java**

```
package com(tpb.projects.issues.fragments;

import android.app.Activity;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.ActivityOptionsCompat;
import android.support.v4.util.Pair;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.app.AlertDialog;
import android.support.v7.widget.CardView;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.text.format.DateUtils;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.View;
import android.view.ViewGroup;
import android.view.ViewTreeObserver;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.PopupMenu;
import android.widget.TextView;
import android.widget.Toast;

import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Editor;
import com(tpb.github.data.Loader;
import com(tpb.github.data.models.Issue;
import com(tpb.github.data.models.Milestone;
import com(tpb.github.data.models.Repository;
import com(tpb.github.data.models.State;
import com(tpb.github.data.models.User;
import com(tpb.mdtext.Markdown;
```

```
import com(tpb).mdtext.imagegetter.HttpImageGetter;
import com(tpb).mdtext.views.MarkdownTextView;
import com(tpb).projects.R;
import com(tpb).projects.common.FixedLinearLayoutManger;
import com(tpb).projects.common.NetworkImageView;
import com(tpb).projects.editors.CommentEditor;
import com(tpb).projects.editors.IssueEditor;
import com(tpb).projects.flow.IntentHandler;
import com(tpb).projects.issues.IssueActivity;
import com(tpb).projects.issues.IssueEventsAdapter;
import com(tpb).projects.markdown.Formatter;
import com(tpb).projects.user.UserActivity;
import com(tpb).projects.util.UI;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;
import butterknife.Unbinder;

/**
 * Created by theo on 14/03/17.
 */

public class IssueInfoFragment extends IssueFragment {

 private Unbinder unbinder;

 @BindView(R.id.issue_header_card) CardView mHeader;
 @BindView(R.id.issue_events_recycler) RecyclerView mRecycler;
 @BindView(R.id.issue_assignees) LinearLayout mAssigneesLayout;
 @BindView(R.id.text_issue_assignees) View mAssigneesTitle;
 @BindView(R.id.issue_menu_button) ImageButton mOverflowButton;
 @BindView(R.id.issue_user_avatar) NetworkImageView mUserAvatar;
 @BindView(R.id.issue_state) ImageView mImageState;
 @BindView(R.id.issue_title) MarkdownTextView mTitle;
 @BindView(R.id.issue_info) MarkdownTextView mInfo;
 @BindView(R.id.issue_events_refresher) SwipeRefreshLayout mRefresher;
 @BindView(R.id.viewholder_milestone_card) CardView mMilestoneCard;

 private Issue mIssue;

 private Repository.AccessLevel mAccessLevel = Repository.AccessLevel.NONE;
 private Editor mEditor;

 private IssueEventsAdapter mAdapter;

 public static IssueInfoFragment getInstance() {
 return new IssueInfoFragment();
 }

 @Nullable
 @Override
 public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
 final View view = inflater.inflate(R.layout.fragment_issue_info,
 container, false);
 unbinder = ButterKnife.bind(this, view);
 mAccessLevel = ((IssueActivity) getActivity()).mAccessLevel;
 mEditor = Editor.getEditor(getContext());
 }
}
```

```

 mAdapter = new IssueEventsAdapter(this, mRefresher);
 final LinearLayoutManager manager = new
FixedLinearLayoutManger(getContext());
 mRecycler.setOnScrollListener(new RecyclerView.OnScrollListener() {
 @Override
 public void onScrolled(RecyclerView recyclerView, int dx, int dy)
{
 super.onScrolled(recyclerView, dx, dy);
 if(manager.findFirstVisibleItemPosition() + 20 >
manager.getItemCount()) {
 mAdapter.notifyBottomReached();
 }
 }
 });
 mRecycler.setAdapter(mAdapter);
 mRecycler.setLayoutManager(manager);

 mRefresher.setOnRefreshListener(() -> {
 mAdapter.clear();
 Loader.getLoader(getContext()).loadIssue(new
Loader.ItemLoader<Issue>() {
 @Override
 public void loadComplete(Issue issue) {
 issueLoaded(issue);
 }

 @Override
 public void loadError(APIHandler.APIError error) {
 mRefresher.setRefreshing(false);
 }
 }, mIssue.getRepoFullName(), mIssue.getNumber(), true);
 });
 checkSharedElementEntry();
 if(mIssue != null) issueLoaded(mIssue);
 return view;
 }

 @Override
 public void issueLoaded(Issue issue) {
 mIssue = issue;
 if(mAdapter != null) {
 mAdapter.setIssue(issue);
 displayIssue();
 displayAssignees();
 displayMilestone();
 }
 }

 private void displayIssue() {
 mTitle.setMarkdown(Formatter.header(mIssue.getTitle(), 1));
 mInfo.setMarkdown(
 Formatter.buildIssueSpan(
 getContext(),
 mIssue,
 false, //Show title
 false, //No numbered link
 false, //No assignees

```

```
 true, //Closed at
 false //No comment count
).toString(),
 new HttpImageGetter(mInfo), null
);
mUserAvatar.setImageUrl(mIssue.getOpenedBy().getAvatarUrl());
IntentHandler.addOnClickHandler(getActivity(), mUserAvatar,
mIssue.getOpenedBy().getLogin());
mImageState.setOnClickListener(v -> toggleIssueState());
if(mIssue.isClosed()) {
 mImageState.setImageResource(R.drawable.ic_state_closed);
} else {
 mImageState.setImageResource(R.drawable.ic_state_open);
}
}

private void displayAssignees() {
 mAssigneesLayout.removeAllViews();
 if(mIssue != null && mIssue.getAssignees() != null &&
mIssue.getAssignees().length > 0) {
 mAssigneesLayout.setVisibility(View.VISIBLE);
 mAssigneesTitle.setVisibility(View.VISIBLE);
 for(User u : mIssue.getAssignees()) {
 final LinearLayout user = (LinearLayout)
getActivity().getLayoutInflater()
.inflate(R.layout.shard_user,
mAssigneesLayout,
false
);
 user.setId(View.generateViewId());
 mAssigneesLayout.addView(user);
 final NetworkImageView avatar = ButterKnife.findById(user,
R.id.user_avatar);
 avatar.setId(View.generateViewId());
 avatar.setImageUrl(u.getAvatarUrl());
 avatar.setScaleType(ImageView.ScaleType.FIT_XY);
 final TextView login = ButterKnife.findById(user,
R.id.user_login);
 login.setId(View.generateViewId());
 login.setText(u.getLogin());
 user.setOnClickListener((v) -> {
 final Intent us = new Intent(getActivity(),
UserActivity.class);
 us.putExtra(getString(R.string.intent_username),
u.getLogin());
 UI.setDrawableForIntent(avatar, us);
 getActivity().startActivity(us,
ActivityOptionsCompat.makeSceneTransitionAnimation(
 getActivity(),
 Pair.create(login,
getString(R.string.transition_username)),
 Pair.create(avatar,
getString(R.string.transition_user_image))
 .toBundle()
));
 });
 }
 }
}
```

```

 });
 }
} else {
 mAssigneesLayout.setVisibility(View.GONE);
 mAssigneesTitle.setVisibility(View.GONE);
}
}

private void displayMilestone() {
 final Milestone milestone = mIssue.getMilestone();
 if(milestone != null) {
 mMilestoneCard.setVisibility(View.VISIBLE);
 final MarkdownTextView tv = ButterKnife
 .findById(mMilestoneCard,
R.id.milestone_content_markdown);
 final ImageView status = ButterKnife.findById(mMilestoneCard,
R.id.milestone_drawable);
 final NetworkImageView user = ButterKnife
 .findById(mMilestoneCard, R.id.milestone_user_avatar);
 IntentHandler
 .addOnClickHandler(getActivity(), tv, user,
milestone.getCreator().getLogin());
 IntentHandler.addOnClickHandler(getActivity(), user,
milestone.getCreator().getLogin());
 status.setImageResource(milestone
 .getState() == State.OPEN ? R.drawable.ic_state_open :
R.drawable.ic_state_closed);
 user.setImageUrl(milestone.getCreator().getAvatarUrl());

 final StringBuilder builder = new StringBuilder();

 Formatter.bold(Markdown.escape(milestone.getTitle()));
 builder.append("
");
 if(milestone.getOpenIssues() > 0 || milestone.getClosedIssues() >
0) {
 builder.append("
");

builder.append(String.format(getString(R.string.text_milestone_completion),
 milestone.getOpenIssues(),
 milestone.getClosedIssues(),
 Math.round(100f * milestone.getClosedIssues() /
(milestone
 .getOpenIssues() +
milestone.getClosedIssues())))
);
 }
 builder.append("
");
 builder.append(
 String.format(
 getString(R.string.text_milestone_opened_by),
 String.format(getString(R.string.text_href),
 "https://github.com/" +
milestone.getCreator()
 .getLogin(),
 milestone.getCreator().getLogin()
),
 DateUtils.getRelativeTimeSpanString(milestone.getCreatedAt())
)
);
}
}

```

```
)
);
 if(milestone.getUpdatedAt() != milestone.getCreatedAt()) {
 builder.append("
");
 builder.append(
 String.format(
 getString(R.string.text_last_updated),
 DateUtils.getRelativeTimeSpanString(milestone.getUpdatedAt())
)
);
 }
 if(milestone.getClosedAt() > 0) {
 builder.append("
");
 builder.append(
 String.format(
 getString(R.string.text_milestone_closed_at),
 DateUtils.getRelativeTimeSpanString(milestone.getClosedAt())
)
);
 }
 if(milestone.getDueOn() > 0) {
 builder.append("
");
 if(System.currentTimeMillis() < milestone.getDueOn() ||
 (milestone.getClosedAt() != 0 &&
milestone.getClosedAt() < milestone
 .getDueOn())))
 builder.append(
 String.format(
 getString(R.string.text_milestone_due_on),
 DateUtils.getRelativeTimeSpanString(milestone.getDueOn())
)
);
 } else {
 builder.append("<font color=\"");
 builder.append(String.format("#%06X", (0xFFFF &
Color.RED)));
 builder.append("\">");
 builder.append(
 String.format(
 getString(R.string.text_milestone_due_on),
 DateUtils.getRelativeTimeSpanString(milestone.getDueOn())
)
);
 builder.append("");
 }
 }
 tv.setMarkdown(Markdown.formatMD(builder.toString(),
mIssue.getRepoFullName()));
}
} else {
 mMilestoneCard.setVisibility(View.GONE);
}
}
```

```

public void updateIssue(Issue issue, String[] assignees, String[] labels)
{
 mRefresher.setRefreshing(true);
 mEditor.updateIssue(new Editor.UpdateListener<Issue>() {
 int issueCreationAttempts = 0;

 @Override
 public void updated(Issue issue) {
 int matchCount = 0;
 final Issue old = mIssue;
 mIssue = issue;
 if(old.getAssignees() != null && mIssue.getAssignees() != null) {
 for(User u : old.getAssignees()) {
 for(User v : old.getAssignees()) {
 if(u.equals(v)) matchCount++;
 }
 }
 if(matchCount != old.getAssignees().length ||
 matchCount != mIssue.getAssignees().length) {
 displayAssignees();
 }
 }
 displayIssue();
 displayMilestone();
 mRefresher.setRefreshing(false);
 }

 @Override
 public void updateError(APIHandler.APIError error) {
 if(error == APIHandler.APIError.NO_CONNECTION) {
 mRefresher.setRefreshing(false);
 Toast.makeText(getContext(), error.resId,
Toast.LENGTH_SHORT).show();
 } else {
 if(issueCreationAttempts < 5) {
 issueCreationAttempts++;
 mEditor.updateIssue(this, mIssue.getRepoFullName(),
issue, assignees,
 labels
);
 } else {
 Toast.makeText(getContext(), error.resId,
Toast.LENGTH_SHORT).show();
 mRefresher.setRefreshing(false);
 }
 }
 }
 }, mIssue.getRepoFullName(), issue, assignees, labels);
}

private void editIssue(View view) {
 final Intent i = new Intent(getContext(), IssueEditor.class);
 i.putExtra(getString(R.string.intent_repo), mIssue.getRepoFullName());
 i.putExtra(getString(R.string.parcel_issue), mIssue);
 UI.setViewPositionForIntent(i, view);
 startActivityForResult(i, IssueEditor.REQUEST_CODE_EDIT_ISSUE);
}

```

```
private void toggleIssueState() {
 if(mIssue == null || mAccessLevel != Repository.AccessLevel.ADMIN)
 return;
 final Editor.UpdateListener<Issue> listener = new
Editor.UpdateListener<Issue>() {
 @Override
 public void updated(Issue issue) {
 mIssue = issue;
 mImageState.setImageResource(
 mIssue.isClosed() ? R.drawable.ic_state_closed :
R.drawable.ic_state_open);
 }

 @Override
 public void updateError(APIHandler.APIError error) {
 mRefresher.setRefreshing(false);
 if(error == APIHandler.APIError.NO_CONNECTION) {
 mRefresher.setRefreshing(false);
 Toast.makeText(getContext(), error.resId,
Toast.LENGTH_SHORT).show();
 }
 }
 };

 final AlertDialog.Builder builder = new
AlertDialog.Builder(getContext());
 builder.setTitle(R.string.title_state_change_comment);
 builder.setPositiveButton(R.string.action_ok, (dialog, which) -> {
 final Intent i = new Intent(getContext(), CommentEditor.class);
 startActivityForResult(i,
CommentEditor.REQUEST_CODE_COMMENT_FOR_STATE);
 mRefresher.setRefreshing(true);
 if(mIssue.isClosed()) {
 mEditor.openIssue(listener, mIssue.getRepoFullName(),
mIssue.getNumber());
 } else {
 mEditor.closeIssue(listener, mIssue.getRepoFullName(),
mIssue.getNumber());
 }
 });
 builder.setNeutralButton(R.string.action_no, (dialog, which) -> {
 mRefresher.setRefreshing(true);
 if(mIssue.isClosed()) {
 mEditor.openIssue(listener, mIssue.getRepoFullName(),
mIssue.getNumber());
 } else {
 mEditor.closeIssue(listener, mIssue.getRepoFullName(),
mIssue.getNumber());
 }
 });
 builder.setNegativeButton(R.string.action_cancel, null);
 builder.create().show();
}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data)
{
 super.onActivityResult(requestCode, resultCode, data);
```

```

 if(resultCode == Activity.RESULT_OK) {
 if(requestCode == IssueEditor.REQUEST_CODE_EDIT_ISSUE) {
 final Issue issue =
data.getParcelableExtra(getString(R.string.parcel_issue));
 final String[] assignees;
 final String[] labels;
 if(data.hasExtra(getString(R.string.intent_issue_assignees)))
{
 assignees = data
.getStringArrayExtra(getString(R.string.intent_issue_assignees));
 } else {
 assignees = null;
 }
 if(data.hasExtra(getString(R.string.intent_issue_labels))) {
 labels =
data.getStringArrayExtra(getString(R.string.intent_issue_labels));
 } else {
 labels = null;
 }
 updateIssue(issue, assignees, labels);
 }
 }
 }

 @OnClick({R.id.issue_header_card, R.id.issue_info, R.id.issue_title})
 void onHeaderClick() {
 if(mIssue != null && mAccessLevel == Repository.AccessLevel.ADMIN)
editIssue(mInfo);
 }

 @OnClick(R.id.issue_menu_button)
 public void displayIssueMenu(View view) {
 final PopupMenu menu = new PopupMenu(getContext(), view);
 menu.inflate(R.menu.menu_issue);
 if(mAccessLevel == Repository.AccessLevel.ADMIN) {
 menu.getMenu().add(0, 1, Menu.NONE,
 mIssue.isClosed() ? R.string.menu_reopen_issue :
R.string.menu_close_issue
);
 menu.getMenu().add(0, 2, Menu.NONE, R.string.menu_edit_issue);
 }
 menu.setOnMenuItemClickListener(menuItem -> {
 switch(menuItem.getItemId()) {
 case 1:
 toggleIssueState();
 break;
 case 2:
 editIssue(view);
 break;
 case R.id.menu_fullscreen:
 IntentHandler.showFullScreen(getContext(),
mIssue.getBody(),
 mIssue.getRepoFullName(), getFragmentManager()
);
 break;
 }
 return false;
 });
 }
}

```

```

 menu.show();
 }

@Override
public void setAccessLevel(Repository.AccessLevel level) {
 mAccessLevel = level;
}

private void checkSharedElementEntry() {
 final Intent i = getActivity().getIntent();
 if(i.getStringExtra(getString(R.string.transition_card))) {
 mHeader.getViewTreeObserver()
 .addOnPreDrawListener(new
ViewTreeObserver.OnPreDrawListener() {
 @Override
 public boolean onPreDraw() {

mHeader.getViewTreeObserver().removeOnPreDrawListener(this);
 if(i.getStringExtra(getString(R.string.intent_drawable)))
{
 mUserAvatar.setImageBitmap(
i.getParcelableExtra(getString(R.string.intent_drawable)));
 }
 getActivity().startPostponedEnterTransition();
 return true;
 }
 });
 }
}

@Override
public void onDestroyView() {
 super.onDestroyView();
 unbinder.unbind();
}
}

```

When issueLoaded is called, the Issue is passed to the adapter, before displayIssue, displayAssignees, and displayMilestone are called.

## displayIssue

displayIssue first sets the the title text, using Formatter to wrap the title in a level 1 header element.

### Formatter.java

```

public static String header(String s, @IntRange(from = 0, to = 6) int depth) {
 String header = "<h" + depth + ">";
 header += Markdown.escape(s).replace("\n", "</h" + depth + "><h" +
depth + ">");
 return header + "</h" + depth + ">";
}

```

It then uses Formatter again to build the span:

### Formatter.java

```

public static StringBuilder buildIssueSpan(Context context, Issue issue,
 boolean showTitle,
 boolean showNumberedLink,
 boolean showAssignees,
 boolean showClosedAt,
 boolean showCommentCount) {
 final StringBuilder builder = new StringBuilder();
 if(showTitle) {
 builder.append(header(Markdown.escape(issue.getTitle()), 1));
 }

 if(issue.getBody() != null && !issue.getBody().isEmpty()) {

builder.append(Markdown.formatMD(issue.getBody().replaceFirst("\s++$", ""),
 issue.getRepoFullName()
));
 }
 builder.append("\n\n");
 if(showNumberedLink) {

builder.append(String.format(context.getString(R.string.text_issue_opened_by),
 String.format(context.getString(R.string.text_md_link),
 "#" + Integer.toString(issue.getNumber()),
 "https://github.com/" + issue.getRepoFullName() +
"issues/" + Integer
 .toString(issue.getNumber())
),
 String.format(context.getString(R.string.text_md_link),
 issue.getOpenedBy().getLogin(),
 issue.getOpenedBy().getHtmlUrl()
),
 DateUtils.getRelativeTimeSpanString(issue.getCreatedAt())
)
);
 builder.append("
");
} else {
 builder.append(
 String.format(
context.getString(R.string.text_opened_this_issue),
String.format(context.getString(R.string.text_href),
 "https://github.com/" +
issue.getOpenedBy().getLogin(),
 issue.getOpenedBy().getLogin()
),
DateUtils.getRelativeTimeSpanString(issue.getCreatedAt())
)
);
 builder.append("
");
}

if(issue.getLabels() != null && issue.getLabels().length > 0) {
 appendLabels(builder, issue.getLabels(), " ");
 builder.append("
");
}
if(showAssignees && issue.getAssignees() != null) {
 builder.append(context.getString(R.string.text_assigned_to));
}

```

```

 builder.append(' ');
 for(User u : issue.getAssignees()) {

builder.append(String.format(context.getString(R.string.text_md_link),
 u.getLogin(),
 u.getHtmlUrl()
));
 builder.append(' ');
 }
 builder.append("
");
}
if(showCommentCount && issue.getComments() > 0) {
 builder.append(context.getResources()

.getQuantityString(R.plurals.text_issue_comment_count,
 issue.getComments(),
issue.getComments()
));
 builder.append("
");
}
if(showClosedAt && issue.getClosedAt() != 0 && issue.getClosedBy() !=
null) {

builder.append(String.format(context.getString(R.string.text_closed_by_link),
 issue.getClosedBy().getLogin(),
 issue.getClosedBy().getHtmlUrl(),
 DateUtils.getRelativeTimeSpanString(issue.getClosedAt())
));
 builder.append("
");
}
return builder;
}

```

This method is used to format an `Issue` model for displaying.

It first adds the title, if the `showTitle` flag is true.

Next it adds the body if it is non-empty, removing any leading whitespace.

If `showNumberedLink` is true, it adds a line containing the issue number, creator, and the time that it was created. Otherwise the issue number is omitted.

Next, the labels are added.

If `showAssignees` is true, and the `Issue` has assignees, a link is added to each assignee.

If `showCommentCount` is true, a line is added showing the number of comments.

Finally, if `showClosedAt` is true, and the issue is closed, a line is added with a link to the user that closed the issue, and when they closed it.

Returning to `displayIssue`, the user avatar is set and an `OnClickListener` is added to open the user.

An `OnClickListener` is then added to the state `ImageView` and its resource is set.

## displayAssignees

`displayAssignees` clears the `Views` from the `mAssigneesLayout LinearLayout`.

It then checks if there are assignees to add, hiding the `LinearLayout` and its header `TextView` otherwise.

If there are assignees, the shard\_user layout is inflated for each of them, the Views are bound with the user login and avatar, and the OnClickListener is set to open the UserActivity with a shared element transition.

### **displayMilestone**

displayMilestone populates a MarkdownTextView with information about a Milestone to which the Issue is attached (If it is attached to one).

The information lines are as follows:

- Milestone title
- Number of open and closed issues attached to milestone, and percentage complete
- The date that the milestone was created at, and the user that created the milestone
- The last time that the milestone was updated, if it has been updated
- The date that the milestone was closed, if it has been closed
- The date that the milestone is due, if it is due at a specific time. This is displayed in red if the deadline has been reached

### **updateIssue**

updateIssue is called onActivityResult from the IssueEditor.

It first checks whether any of the assignees have changed, calling displayAssignees if they have.

Next displayIssue and displayMilestone are called to update the issue information.

### **toggleIssueState**

toggleIssueState checks that the Issue exists, and that the user has access to modify its state.

If these conditions pass, an AlertDialog is created to allow the user to choose whether they wish to add a comment when changing the state.

### **IssueEventsAdapter**

The IssueEventsAdapter has to handle 24 different events, as listed in objective 4.a.xi.

It also has to manage these events when they occur at the same time, as a singular event.

### **IssueEventsAdapter.java**

```
package com(tpb.projects.issues;

import android.content.res.Resources;
import android.support.v4.util.Pair;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.widget.RecyclerView;
import android.text.SpannableString;
import android.text.format.DateUtils;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Loader;
import com(tpb.github.data.Util;
import com(tpb.github.data.models.DataModel;
import com(tpb.github.data.models.Issue;
import com(tpb.github.data.models.IssueEvent;
import com(tpb.github.data.models.MergedModel;
import com(tpb.mdtext.imagegetter.HttpImageGetter;
import com(tpb.mdtext.views.MarkdownTextView;
import com(tpb.projects.BuildConfig;
import com(tpb.projects.R;
import com(tpb.projects.common.NetworkImageView;
import com(tpb.projects.flow.IntentHandler;
import com(tpb.projects.issues.fragments.IssueInfoFragment;
import com(tpb.projects.markdown.Formatter;

import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;

import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by theo on 15/03/17.
 */

public class IssueEventsAdapter extends
RecyclerView.Adapter<IssueEventsAdapter.EventHolder> implements
Loader.ListLoader<IssueEvent> {

 private final ArrayList<Pair<DataModel, SpannableString>> mEvents = new
ArrayList<>();
 private Issue mIssue;
 private final IssueInfoFragment mParent;

 private int mPage = 1;
 private boolean mIsLoading = false;
 private boolean mMaxPageReached = false;

 private SwipeRefreshLayout mRefresher;
 private Loader mLoader;

 public IssueEventsAdapter(IssueInfoFragment parent, SwipeRefreshLayout
refresher) {
 mParent = parent;
 mLoader = Loader.getLoader(parent.getContext());
 }
}
```

```
mRefresher = refresher;
mRefresher.setRefreshing(true);
mRefresher.setOnRefreshListener(() -> {
 mPage = 1;
 mMaxPageReached = false;
 notifyDataSetChanged();
 loadEvents(true);
});
}

public void clear() {
 mEvents.clear();
 notifyDataSetChanged();
}

public void setIssue(Issue issue) {
 mIssue = issue;
 mEvents.clear();
 mPage = 1;
 mLoader.loadEvents(this, issue.getRepoFullName(), issue.getNumber(),
mPage);
}

@Override
public void listLoadComplete(List<IssueEvent> events) {
 mRefresher.setRefreshing(false);
 mIsLoading = false;
 if(events.size() > 0) {
 int oldLength = mEvents.size();
 if(mPage == 1) mEvents.clear();
 for(DataModel dm : Util.mergeModels(events, comparator)) {
 mEvents.add(Pair.create(dm, null));
 }
 notifyDataSetChangedRangeInserted(oldLength, mEvents.size());
 } else {
 mMaxPageReached = true;
 }
}

private static Comparator<DataModel> comparator = (o1, o2) ->
 o1 instanceof IssueEvent &&
 o2 instanceof IssueEvent &&
 o1.getCreatedAt() == o2.getCreatedAt() &&
 ((IssueEvent) o1).getEvent() == ((IssueEvent)
o2).getEvent()
 ? 0 : -1;

@Override
public void listLoadError(APIHandler.APIError error) {

}

public void notifyBottomReached() {
 if(!mIsLoading && !mMaxPageReached) {
 mPage++;
 loadEvents(false);
 }
}
```

```

private void loadEvents(boolean resetPage) {
 mIsLoading = true;
 mRefresher.setRefreshing(true);
 if(resetPage) {
 mPage = 1;
 mMaxPageReached = false;
 }
 mLoader.loadEvents(this, mIssue.getRepoFullName(), mIssue.getNumber(),
mPage);
}

@Override
public EventHolder onCreateViewHolder(ViewGroup parent, int viewType) {
 return new EventHolder(LayoutInflater.from(parent.getContext()))

.inflate(R.layout.viewholder_event, parent, false));
}

@SuppressWarnings("unchecked")
@Override
public void onBindViewHolder(EventHolder holder, int position) {
 if(mEvents.get(position).first instanceof IssueEvent) {
 bindEvent(holder, (IssueEvent) mEvents.get(position).first);
 } else if(mEvents.get(position).first instanceof MergedModel) {
 bindMergedEvent(holder, (MergedModel<IssueEvent>)
mEvents.get(position).first);
 }
}

private void bindMergedEvent(EventHolder eventHolder,
MergedModel<IssueEvent> me) {
 String text;
 final Resources res = eventHolder.itemView.getResources();
 final StringBuilder builder = new StringBuilder();
 switch(me.getData().get(0).getEvent()) {
 case ASSIGNED:
 for(IssueEvent e : me.getData()) {

builder.append(String.format(res.getString(R.string.text_href),
 e.getActor().getHtmlUrl(),
 e.getActor().getLogin()
));
 builder.append(", ");
 }
 builder.setLength(builder.length() - 2); //Remove final comma
 text =
String.format(res.getString(R.string.text_event_assigned_multiple),
 builder.toString()
);
 break;
 case UNASSIGNED:
 for(IssueEvent e : me.getData()) {

builder.append(String.format(res.getString(R.string.text_href),
 e.getActor().getHtmlUrl(),
 e.getActor().getLogin()
));
}
 }
}

```

```
 builder.append(", ");
 }
 builder.setLength(builder.length() - 2); //Remove final comma
 text =
String.format(res.getString(R.string.text_event_unassigned_multiple),
 builder.toString())
);
break;
case REVIEW_REQUESTED:
 for(IssueEvent e : me.getData()) {

builder.append(String.format(res.getString(R.string.text_href),
 e.getRequestedReviewer().getHtmlUrl(),
 e.getRequestedReviewer().getLogin()
));
 builder.append(", ");
}
 builder.setLength(builder.length() - 2);
 text =
String.format(res.getString(R.string.text_event_review_requested_multiple),
 String.format(res.getString(R.string.text_href),
me.getData().get(0).getReviewRequester().getHtmlUrl(),
me.getData().get(0).getReviewRequester().getLogin()
),
 builder.toString()
);
break;
case REVIEW_REQUEST_REMOVED:
 for(IssueEvent e : me.getData()) {

builder.append(String.format(res.getString(R.string.text_href),
 e.getReviewRequester().getHtmlUrl(),
 e.getReviewRequester().getLogin()
));
 builder.append(", ");
}
 builder.setLength(builder.length() - 2);
 text = String

.format(res.getString(R.string.text_event_review_request_removed_multiple),
String.format(res.getString(R.string.text_href),
me.getData().get(0).getActor().getHtmlUrl(),
me.getData().get(0).getActor().getLogin()
),
 builder.toString()
);
break;
case LABELED:
 for(IssueEvent e : me.getData()) {
 builder.append(Formatter.getLabelString(e.getLabelName(),
e.getLabelColor()));
 builder.append(" ");
 }
 builder.setLength(builder.length() - " ".length());
```

```

 text = String.format(
 res.getString(R.string.text_event_labels_added),
 String.format(res.getString(R.string.text_href),
 me.getData().get(0).getActor().getHtmlUrl(),
 me.getData().get(0).getActor().getLogin()
),
 builder.toString()
);
 break;
 case UNLABELED:
 for(IssueEvent e : me.getData()) {
 builder.append(Formatter.getLabelString(e.getLabelName(),
e.getLabelColor()));
 builder.append(" ");
 }
 builder.setLength(builder.length() - 2);
 text =
String.format(res.getString(R.string.text_event_labels_removed),
 String.format(res.getString(R.string.text_href),
 me.getData().get(0).getActor().getHtmlUrl(),
 me.getData().get(0).getActor().getLogin()
),
 builder.toString()
);
 break;
 case REFERENCED:
 for(IssueEvent e : me.getData()) {
 builder.append("
");

builder.append(String.format(res.getString(R.string.text_href),
 "https://github.com/" + mIssue.getRepoFullName() +
"/commit/" + e
 .getCommitId(),
 String.format(res.getString(R.string.text_commit),
e.getShortCommitId())
));
 }
 builder.append("
");
 text =
String.format(res.getString(R.string.text_event_referenced_multiple),
 builder.toString()
);
 break;
 case MENTIONED:
 for(IssueEvent e : me.getData()) {
 builder.append("
");

builder.append(String.format(res.getString(R.string.text_href),
 e.getActor().getHtmlUrl(),
 e.getActor().getLogin()
));
 }
 text =
String.format(res.getString(R.string.text_event_mentioned_multiple),
 builder.toString()
);
 break;
 case RENAMED:
 for(IssueEvent e : me.getData()) {

```

```

 builder.append("
");
 builder.append(
 String.format(
 res.getString(R.string.text_event_rename_multiple),
 e.getRenameFrom(),
 e.getRenameTo()
)
);
 }
 text =
String.format(res.getString(R.string.text_event_renamed_multiple),
 builder.toString()
);
break;
case MOVED_COLUMNS_IN_PROJECT:
 text =
res.getString(R.string.text_event_moved_columns_in_project_multiple);
 break;
default:
 bindEvent(eventHolder, me.getData().get(0));
 return;
}
text += " • " +
DateUtils.getRelativeTimeSpanString(me.getCreatedAt());
eventHolder.mText.setMarkdown(
 text,
 new HttpImageGetter(eventHolder.mText),
 null
);
if(me.getData().get(0).getActor() != null) {
 eventHolder.mAvatar.setVisibility(View.VISIBLE);

eventHolder.mAvatar.setImageUrl(me.getData().get(0).getActor().getAvatarUrl())
;
 IntentHandler.addOnClickHandler(
 mParent.getActivity(),
 eventHolder.mAvatar,
me.getData().get(0).getActor().getLogin()
);
 IntentHandler.addOnClickHandler(mParent.getActivity(),
 eventHolder.mText,
 eventHolder.mAvatar,
 me.getData().get(0).getActor().getLogin()
);
} else {
 eventHolder.mAvatar.setVisibility(View.GONE);
}
}

private void bindEvent(EventHolder eventHolder, IssueEvent event) {
String text;
final Resources res = eventHolder.itemView.getResources();
switch(event.getEvent()) {
 case CLOSED:
 if(event.getShortCommitId() != null) {
 text =
String.format(res.getString(R.string.text_event_closed_in),
 String.format(res.getString(R.string.text_href),

```

```
 event.getActor().getHtmlUrl(),
 event.getActor().getLogin()
),
 String.format(res.getString(R.string.text_href),
 "https://github.com/" + mIssue
 .getRepoFullName() + "/commit/" +
event.getCommitId(),

String.format(res.getString(R.string.text_commit),
 event.getShortCommitId()
)
)
);
} else {
 text =
String.format(res.getString(R.string.text_event_closed),
 String.format(res.getString(R.string.text_href),
 event.getActor().getHtmlUrl(),
 event.getActor().getLogin()
)
)
};
}
break;
case REOPENED:
 text =
String.format(res.getString(R.string.text_event_reopened),
 String.format(res.getString(R.string.text_href),
 event.getActor().getHtmlUrl(),
 event.getActor().getLogin()
)
)
);
}
break;
case SUBSCRIBED:
 text =
String.format(res.getString(R.string.text_event_subscribed),
 String.format(res.getString(R.string.text_href),
 event.getActor().getHtmlUrl(),
 event.getActor().getLogin()
)
)
);
}
break;
case MERGED:
 text =
String.format(res.getString(R.string.text_event_merged),
 String.format(res.getString(R.string.text_href),
 event.getActor().getHtmlUrl(),
 event.getActor().getLogin()
),
 String.format(res.getString(R.string.text_href),
 "https://github.com/" + mIssue
 .getRepoFullName() + "/commit/" +
event.getCommitId(),

String.format(res.getString(R.string.text_commit),
 event.getShortCommitId()
)
)
);
}
break;
```

```
 case REFERENCED:
 text =
String.format(res.getString(R.string.text_event_referenced),
 String.format(res.getString(R.string.text_href),
 "https://github.com/" + mIssue
 .getRepoFullName() + "/commit/" +
event.getCommitId(),

String.format(res.getString(R.string.text_commit),
 event.getShortCommitId()
)
);
 break;
 case MENTIONED:
 text =
String.format(res.getString(R.string.text_event_mentioned),
 String.format(res.getString(R.string.text_href),
 event.getActor().getHtmlUrl(),
 event.getActor().getLogin()
)
);
 break;
 case ASSIGNED:
 if(event.getAssignee() != null &&
event.getActor().equals(event.getAssignee())) {
 text =
String.format(res.getString(R.string.text_event_assigned_themselves),
 String.format(res.getString(R.string.text_href),
 event.getActor().getHtmlUrl(),
 event.getActor().getLogin()
)
);
 } else {
 text =
String.format(res.getString(R.string.text_event_assigned),
 String.format(res.getString(R.string.text_href),
 event.getActor().getHtmlUrl(),
 event.getActor().getLogin()
)
);
 }
 break;
 case UNASSIGNED:
 if(event.getAssignee() != null &&
event.getActor().equals(event.getAssignee())) {
 text =
String.format(res.getString(R.string.text_event_unassigned_themselves),
 String.format(res.getString(R.string.text_href),
 event.getActor().getHtmlUrl(),
 event.getActor().getLogin()
)
);
 } else {
 text =
String.format(res.getString(R.string.text_event_unassigned),
 String.format(res.getString(R.string.text_href),
 event.getActor().getHtmlUrl(),
 event.getActor().getLogin()

```

```
)
);
}
break;
case LABELED:
text =
String.format(res.getString(R.string.text_event_labeled),
 String.format(res.getString(R.string.text_href),
 event.getActor().getHtmlUrl(),
 event.getActor().getLogin()
),
 Formatter.getLabelString(event.getLabelName(),
event.getLabelColor())
);
break;
case UNLABELED:
text =
String.format(res.getString(R.string.text_event_unlabeled),
 String.format(res.getString(R.string.text_href),
 event.getActor().getHtmlUrl(),
 event.getActor().getLogin()
),
 Formatter.getLabelString(event.getLabelName(),
event.getLabelColor())
);
break;
case MILESTONED:
text =
String.format(res.getString(R.string.text_event_milestoned),
 String.format(res.getString(R.string.text_href),
 event.getMilestone().getHtmlUrl(),
 event.getMilestone().getTitle()
),
 String.format(res.getString(R.string.text_href),
 event.getActor().getHtmlUrl(),
 event.getActor().getLogin()
)
);
break;
case DEMILESTONED:
text =
String.format(res.getString(R.string.text_event_demilestoned),
 String.format(res.getString(R.string.text_href),
 event.getMilestone().getHtmlUrl(),
 event.getMilestone().getTitle()
),
 String.format(res.getString(R.string.text_href),
 event.getActor().getHtmlUrl(),
 event.getActor().getLogin()
)
);
break;
case RENAMED:
text =
String.format(res.getString(R.string.text_event_renamed),
 String.format(res.getString(R.string.text_href),
 event.getActor().getHtmlUrl(),
 event.getActor().getLogin()
),
),
```

```
 event.getRenameFrom(),
 event.getRenameTo()
);
 break;
case LOCKED:
 text =
String.format(res.getString(R.string.text_event_locked),
 String.format(res.getString(R.string.text_href),
 event.getActor().getHtmlUrl(),
 event.getActor().getLogin()
)
);
 break;
case UNLOCKED:
 text =
String.format(res.getString(R.string.text_event_unlocked),
 String.format(res.getString(R.string.text_href),
 event.getActor().getHtmlUrl(),
 event.getActor().getLogin()
)
);
 break;
case HEAD_REF_DELETED:
 text = res.getString(R.string.text_event_head_ref_deleted);
 break;
case HEAD_REF_RESTORED:
 text = res.getString(R.string.text_event_head_ref_restored);
 break;
case REVIEW_DISMISSED:
 text =
String.format(res.getString(R.string.text_event_review_dismissed),
 String.format(res.getString(R.string.text_href),
 event.getActor().getHtmlUrl(),
 event.getActor().getLogin()
)
);
 break;
case REVIEW_REQUESTED:
 if(event.getReviewRequester().getId() ==
event.getRequestedReviewer().getId()) {
 text =
String.format(res.getString(R.string.text_event_own_review_request),
 String.format(res.getString(R.string.text_href),
 event.getReviewRequester().getHtmlUrl(),
 event.getReviewRequester().getLogin()
)
);
 } else {
 text =
String.format(res.getString(R.string.text_event_review_requested),
 String.format(res.getString(R.string.text_href),
 event.getReviewRequester().getHtmlUrl(),
 event.getReviewRequester().getLogin()
),
 String.format(res.getString(R.string.text_href),
 event.getRequestedReviewer().getHtmlUrl(),
 event.getRequestedReviewer().getLogin()
)
);
 };
};
```

```
 }
 break;
 case REVIEW_REQUEST_REMOVED:
 if(event.getReviewRequester().getId() ==
event.getRequestedReviewer().getId()) {
 text = String
.format(res.getString(R.string.text_event_removed_own_review_request),
String.format(res.getString(R.string.text_href),
event.getReviewRequester().getHtmlUrl(),
event.getReviewRequester().getLogin()
)
);
 } else {
 text =
String.format(res.getString(R.string.text_review_request_removed),
String.format(res.getString(R.string.text_href),
event.getActor().getHtmlUrl(),
event.getActor().getLogin()
),
String.format(res.getString(R.string.text_href),
event.getRequestedReviewer().getHtmlUrl(),
event.getRequestedReviewer().getLogin()
)
)
);
 }
 break;
 case REMOVED_FROM_PROJECT:
 text =
String.format(res.getString(R.string.text_event_removed_from_project),
String.format(res.getString(R.string.text_href),
event.getActor().getHtmlUrl(),
event.getActor().getLogin()
)
)
);
 break;
 case ADDED_TO_PROJECT:
 text =
String.format(res.getString(R.string.text_event_added_to_project),
String.format(res.getString(R.string.text_href),
event.getActor().getHtmlUrl(),
event.getActor().getLogin()
)
)
);
 break;
 case MOVED_COLUMNS_IN_PROJECT:
 text =
res.getString(R.string.text_event_moved_columns_in_project);
 break;
 default:
 text = "An event type hasn't been implemented " +
event.getEvent()
 + "\nTell me here " + BuildConfig.BUG_EMAIL;
 }
 text += " • " +
DateUtils.getRelativeTimeSpanString(event.getCreatedAt());
```

```

 eventHolder.mText.setMarkdown(
 text,
 new HttpImageGetter(eventHolder.mText),
 null
);
 if(event.getActor() != null) {
 eventHolder.mAvatar.setVisibility(View.VISIBLE);
 eventHolder.mAvatar.setImageUrl(event.getActor().getAvatarUrl());
 IntentHandler.addOnClickHandler(mParent.getActivity(),
eventHolder.mAvatar,
 event.getActor().getLogin()
);
 IntentHandler.addOnClickHandler(mParent.getActivity(),
eventHolder.mText,
 eventHolder.mAvatar, event.getActor().getLogin()
);
 } else {
 eventHolder.mAvatar.setVisibility(View.GONE);
 }
 }

@Override
public int getItemCount() {
 return mEvents.size();
}

static class EventHolder extends RecyclerView.ViewHolder {
 @BindView(R.id.event_text) MarkdownTextView mText;
 @BindView(R.id.event_user_avatar) NetworkImageView mAvatar;

 EventHolder(View view) {
 super(view);
 ButterKnife.bind(this, view);
 }
}

}

```

Ignoring the `ViewHolder` bindings for now, I will first explain the loading and merging of events.

The `IssueEvent` models are loaded from the API

through `ListLoader<IssueEvent>` and passed to `listLoadComplete` in the adapter.

An example of a singular `IssueEvent` type is `CLOSED`.

An issue can clearly only be closed once consecutively as in order to be closed again it must first be re-opened.

This event should be displayed in its own `ViewHolder` telling the user something along the lines of, "The issue was renamed from title0 to title1".

An example of a merged event type is `LABLED`.

This event occurs when a label is applied to the issue.

The API returns individual events for each label that is added to the issue, each with the same timestamp.

The website however, displays these events as a single merged event:

 tpb1908 added **bug** **duplicate** **enhancement** **help wanted** **invalid** **question** **wontfix** labels  
on 30 Dec 2016

In order to implement the merging of events I added the `MergedModel` class.

### MergedModel.java

```
package com(tpb.github.data.models;

import android.os.Parcel;
import android.os.Parcelable;
import android.util.Log;

import java.util.ArrayList;
import java.util.List;

/**
 * Created by theo on 08/01/17.
 */

public class MergedModel<T extends DataModel> extends DataModel implements Parcelable {

 private List<T> data = new ArrayList<>();

 public MergedModel(ArrayList<T> data) {
 this.data.addAll(data);
 }

 public MergedModel(List<T> data) {
 this.data.addAll(data);
 }

 public List<T> getData() {
 return data;
 }

 @Override
 public long getCreatedAt() {
 return data.size() > 0 ? data.get(0).createdAt : 0;
 }

 @Override
 public String toString() {
 return "MergedModel{" +
 "data=" + data +
 '}';
 }

 @Override
 public int describeContents() {
 return 0;
 }
}
```

```

 }

@Override
public void writeToParcel(Parcel dest, int flags) {
 dest.writeString(data.getClass().getName());
 dest.writeList(this.data);
}

protected MergedModel(Parcel in) {
 final String c = in.readString();
 this.data = new ArrayList<>();
 try {
 in.readList(this.data, Class.forName(c).getClassLoader());
 } catch(ClassNotFoundException cnfe) {
 Log.e(MergedModel.class.getSimpleName(), "Error finding class",
cnfe);
 }
}

public static final Creator<MergedModel> CREATOR = new
Creator<MergedModel>() {
 @Override
 public MergedModel createFromParcel(Parcel source) {
 return new MergedModel(source);
 }

 @Override
 public MergedModel[] newArray(int size) {
 return new MergedModel[size];
 }
};
}
}

```

This generic class stores a list of a generic type extending DataModel.

The merging of DataModels is done in the utility method Util.mergeModels.

## Util.java

```

public static List<DataModel> mergeModels(List<? extends DataModel> models,
Comparator<DataModel> comparator) {
 final List<DataModel> merged = new ArrayList<>();
 List<DataModel> toMerge = new ArrayList<>();
 DataModel last = null;
 for(int i = 0; i < models.size(); i++) {
 //If we have two of the same event, happening at the same time
 if(comparator.compare(models.get(i), last) == 0) {
 if(merged.size() > 1) merged.remove(merged.size() - 1);
 /* If multiple events (labels or assignees) were added as the
first event,
 * then we need to stop the first item being duplicated
 */
 if(merged.size() == 1 && merged.get(0).equals(last))
 merged.remove(0);
 toMerge.add(models.get(i - 1)); //Add the previous event
 int j = i;
 //Loop until we find an event which shouldn't be merged
 while(j < models.size() && comparator.compare(models.get(j),
last) == 0) {
 toMerge.add(models.get(j++));
 }
 }
 }
}

```

```

 }
 i = j - 1; //Jump to the end of the merged positions
 merged.add(new MergedModel<>(toMerge));
 toMerge = new ArrayList<>(); //Reset the list of merged events
 } else {
 merged.add(models.get(i));
 }
 last = models.get(i); //Set the last event
}
return merged;
}

```

This method takes a list of any type of `DataModel`, and a comparator for the models.

It merges the `DataModels` into a mixed list of the models and `MergedModels`, maintaining their original order.

The merged list is the output list to be returned.

The `toMerge` list is the working list which builds up a list of `DataModels` to be added to each `MergedModel`.

The models list is iterated through, and each model is compared to the last.

If the models are not equal, the current model is added to merged.

However, if the comparator indicates that the models are equal, a `MergedModel` is built.

In the case that there is more than one `DataModel` currently in the merged list, the last item is removed because it needs to be added to the `MergedModel`.

If there is exactly one item in merged, a specific case where the models begin with a merged set of events must be dealt with.

The previous item in models is added, and we begin adding the `DataModels` from models to `toMerge`.

The while loop runs while we are within the size of models, and the current model is still equal to the model that began the `MergedModel`.

Once the loop completes, the outer counter is jumped to the end of the merged positions, a `MergedModel` containing the items from `toMerge` is created, and `toMerge` is reset.

Returning to `IssueEventsAdapter` the comparator used checks that both models are instances of `IssueEvent`, that they were created at the same time, and that their event type is the same.

`onBindViewHolder` calls either `bindEvent` or `bindMergedEvent` depending on the type of the `DataModel` at the position being bound.

`bindEvent` declares a string for the item text and then switches over the event type.

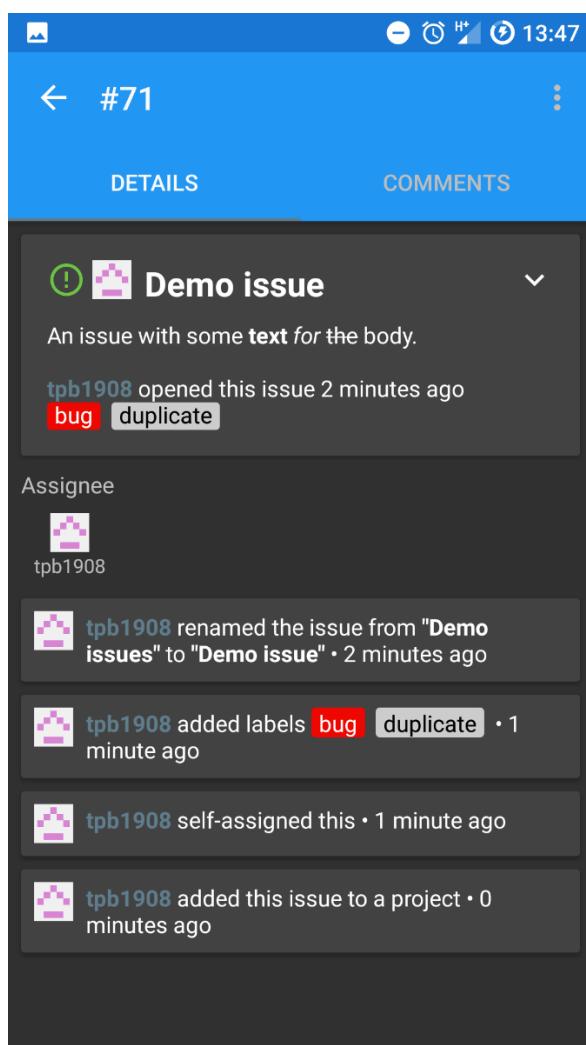
Each event type has a corresponding format string which is populated with the event information.

If a new event type is added and it has not been implemented, the `GitIssueEvent` enum will default to UNKNOWN and the event string will be added to the `EventHolder`.

Finally, if the event has an actor, their login and avatar are displayed.

bindMergedEvent deals with the following event types:

- ASSIGNED- Multiple users were assigned to the issue
- UNASSIGNED - Multiple users were unassigned from the issue
- REVIEW\_REQUESTED- Multiple users were requested to review the issue
- REVIEW\_REQUEST\_REMOVED- Multiple review requests were removed
- LABELED- Multiple labels were added
- UNLABELED- Multiple labels were removed
- REFERENCED- The issue was reference in multiple commits
- MENTIONED- Multiple users were mentioned
- RENAMED- The issue was rename multiple times
- MOVED\_COLUMNS\_IN\_PROJECT- The issue was moved around in a project multiple times



## IssueCommentsFragment

The IssueCommentsFragment is used to display the comments on a particular issue.

### IssueCommentsFragment.java

```
package com(tpb.projects.issues.fragments;

import android.app.Dialog;
import android.content.ClipData;
import android.content.ClipboardManager;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.View;
import android.view.ViewGroup;
import android.widget.PopupMenu;
import android.widget.Toast;

import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Editor;
import com(tpb.github.data.auth.GitHubSession;
import com(tpb.github.data.models.Comment;
import com(tpb.github.data.models.Issue;
import com(tpb.github.data.models.Repository;
import com(tpb.projects.R;
import com(tpb.projects.common.fab.FloatingActionButton;
import com(tpb.projects.editors.CommentEditor;
import com(tpb.projects.issues.IssueCommentsAdapter;
import com(tpb.projects.util.UI;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.Unbinder;

/**
 * Created by theo on 14/03/17.
 */

public class IssueCommentsFragment extends IssueFragment {

 private Unbinder unbinder;

 @BindView(R.id.fragment_recycler) RecyclerView mRecycler;
 @BindView(R.id.fragment_refresher) SwipeRefreshLayout mRefresher;
 private FloatingActionButton mFab;

 private IssueCommentsAdapter mAdapter;
```

```
private Editor mEditor;

private Repository.AccessLevel mAccessLevel;
private Issue mIssue;

public static IssueCommentsFragment getInstance(FloatingActionButton fab)
{
 final IssueCommentsFragment frag = new IssueCommentsFragment();
 frag.mFab = fab;
 return frag;
}

@Override
public void onCreate(@Nullable Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 mEditor = Editor.getEditor(getContext());
}

@Nullable
@Override
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
 final View view = inflater.inflate(R.layout.fragment_recycler,
 container, false);
 unbinder = ButterKnife.bind(this, view);
 final LinearLayoutManager manager = new
 LinearLayoutManager(getContext());
 mRecycler.addOnScrollListener(new RecyclerView.OnScrollListener() {
 @Override
 public void onScrolled(RecyclerView recyclerView, int dx, int dy)
 {
 super.onScrolled(recyclerView, dx, dy);
 if(manager.findFirstVisibleItemPosition() + 20 >
 manager.getItemCount()) {
 mAdapter.notifyBottomReached();
 }
 if(dy > 10) {
 mFab.hide(true);
 } else if(dy < -10 && mIssue != null && !mIssue.isLocked()) {
 mFab.show(true);
 }
 }
 });
 mFab.setOnClickListener(v -> {
 final Intent i = new Intent(getContext(), CommentEditor.class);
 UI.setViewPositionForIntent(i, mFab);
 startActivityForResult(i, CommentEditor.REQUEST_CODE_NEW_COMMENT);
 });
 mAdapter = new IssueCommentsAdapter(this, mRefresher);
 if(mIssue != null) mAdapter.setIssue(mIssue);
 if(mAccessLevel != null && mAccessLevel ==
Repository.AccessLevel.ADMIN) {
 mFab.setVisibility(View.VISIBLE);
 }
 mRecycler.setAdapter(mAdapter);
 mRecycler.setLayoutManager(manager);
 mRefresher.setOnRefreshListener(() -> {
 mAdapter.clear();
 });
}
```

```
 mAdapter.setIssue(mIssue);
 });

 return view;
}

@Override
public void issueLoaded(Issue issue) {
 mIssue = issue;
 if(mAdapter != null) mAdapter.setIssue(issue);
}

@Override
public void setAccessLevel(Repository.AccessLevel level) {
 mAccessLevel = level;
 if(mAccessLevel == Repository.AccessLevel.ADMIN && mIssue != null &&
!mIssue.isLocked()) {
 mFab.setVisibility(View.VISIBLE);
 mFab.show(true);
 }
}

private void createComment(Comment comment) {
 mRefresher.setRefreshing(true);
 mEditor.createIssueComment(new Editor.CreationListener<Comment>() {
 @Override
 public void created(Comment comment) {
 mRefresher.setRefreshing(false);
 mAdapter.addComment(comment);
 mRecycler.post(() ->
mRecycler.smoothScrollToPosition(mAdapter.getItemCount()));
 }

 @Override
 public void creationError(APIHandler.APIError error) {
 mRefresher.setRefreshing(false);
 }
 }, mIssue.getRepoFullName(), mIssue.getNumber(), comment.getBody());
}

public void createCommentForState(Comment comment) {
 createComment(comment);
}

private void editComment(Comment comment) {
 mEditor.updateIssueComment(new Editor.UpdateListener<Comment>() {
 @Override
 public void updated(Comment comment) {
 mRefresher.setRefreshing(false);
 mAdapter.updateComment(comment);
 }

 @Override
 public void updateError(APIHandler.APIError error) {
 mRefresher.setRefreshing(false);
 }
 }, mIssue.getRepoFullName(), comment.getId(), comment.getBody());
}
```

```

 void removeComment(Comment comment) {
 final AlertDialog.Builder builder = new
AlertDialog.Builder(getContext());
 builder.setTitle(R.string.title_delete_comment);
 builder.setPositiveButton(R.string.action_yes, (dialogInterface, i) ->
{
 mRefresher.setRefreshing(true);
 mEditor.deleteCommitComment(new Editor.DeletionListener<Integer>()
{
 @Override
 public void deleted(Integer id) {
 mRefresher.setRefreshing(false);
 mAdapter.removeComment(id);
 }

 @Override
 public void deletionError(APIHandler.APIError error) {
 mRefresher.setRefreshing(false);
 }
 }, mIssue.getRepoFullName(), comment.getId());
 });
 builder.setNegativeButton(R.string.action_no, null);
 final Dialog deleteDialog = builder.create();
 deleteDialog.getWindow().getAttributes().windowAnimations =
R.style.DialogAnimation;
 deleteDialog.show();
 }

 public void displayCommentMenu(View view, Comment comment) {
 final PopupMenu menu = new PopupMenu(getContext(), view);
 menu.inflate(R.menu.menu_comment);
 if(comment.getUser().getLogin().equals(
 GitHubSession.getSession(getContext()).getUserLogin()) &&
!mIssue.isLocked()) {
 menu.getMenu()
 .add(0, R.id.menu_edit_comment, Menu.NONE,
getString(R.string.menu_edit_comment));
 menu.getMenu().add(0, R.id.menu_delete_comment, Menu.NONE,
 getString(R.string.menu_delete_comment))
 };
 menu.setOnMenuItemClickListener(menuItem -> {
 switch(menuItem.getItemId()) {
 case R.id.menu_edit_comment:
 final Intent i = new Intent(getContext(),
CommentEditor.class);
 i.putExtra(getString(R.string.parcel_comment), comment);
 UI.setViewPositionForIntent(i, view);
 startActivityForResult(i,
CommentEditor.REQUEST_CODE_EDIT_COMMENT);
 break;
 case R.id.menu_delete_comment:
 removeComment(comment);
 break;
 case R.id.menu_copy_comment_text:
 final ClipboardManager cm = (ClipboardManager)
getActivity()
 .getSystemService(Context.CLIPBOARD_SERVICE);

```

```

 cm.setPrimaryClip(ClipData.newPlainText("Comment",
comment.getBody())));
 Toast.makeText(getApplicationContext(),
getString(R.string.text_copied_to_board),
Toast.LENGTH_SHORT
).show();
break;
}
return false;
});
menu.show();
}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data)
{
 super.onActivityResult(requestCode, resultCode, data);
 if(resultCode == AppCompatActivity.RESULT_OK) {
 final Comment comment =
data.getParcelableExtra(getString(R.string.parcel_comment));
 if(requestCode == CommentEditor.REQUEST_CODE_NEW_COMMENT) {
 createComment(comment);
 } else if(requestCode == CommentEditor.REQUEST_CODE_EDIT_COMMENT)
{
 mRefresher.setRefreshing(true);
 editComment(comment);
 } else if(requestCode ==
CommentEditor.REQUEST_CODE_COMMENT_FOR_STATE) {
 createCommentForState(comment);
 }
 }
}

@Override
public void onDestroyView() {
 super.onDestroyView();
 unbinder.unbind();
}
}

```

## IssueCommentsAdapter

### IssueCommentsAdapter.java

```

package com(tpb.projects.issues;

import android.support.v4.util.Pair;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.widget.RecyclerView;
import android.text.SpannableString;
import android.text.format.DateUtils;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageButton;

import com(tpb.github.data.APIHandler;

```

```
import com(tpb.github.data.Loader;
import com(tpb.github.data.models.Comment;
import com(tpb.github.data.models.Issue;
import com(tpb.mdtext.Markdown;
import com(tpb.mdtext.imagegetter.HttpImageGetter;
import com(tpb.mdtext.views.MarkdownTextView;
import com(tpb.projects.R;
import com(tpb.projects.common.NetworkImageView;
import com(tpb.projects.flow.IntentHandler;
import com(tpb.projects.issues.fragments.IssueCommentsFragment;
import com(tpb.projects.markdown.Formatter;

import java.util.ArrayList;
import java.util.List;

import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by theo on 14/03/17.
 */

public class IssueCommentsAdapter extends
RecyclerView.Adapter<IssueCommentsAdapter.CommentHolder> implements
Loader.ListLoader<Comment> {

 private final ArrayList<Pair<Comment, SpannableString>> mComments = new
ArrayList<>();
 private Issue mIssue;
 private final IssueCommentsFragment mParent;

 private int mPage = 1;
 private boolean mIsLoading = false;
 private boolean mMaxPageReached = false;

 private SwipeRefreshLayout mRefresher;
 private Loader mLoader;

 public IssueCommentsAdapter(IssueCommentsFragment parent,
SwipeRefreshLayout refresher) {
 mParent = parent;
 mLoader = Loader.getLoader(parent.getContext());
 mRefresher = refresher;
 mRefresher.setRefreshing(true);
 mRefresher.setOnRefreshListener(() -> {
 mPage = 1;
 mMaxPageReached = false;
 clear();
 loadComments(true);
 });
 }

 public void clear() {
 final int oldSize = mComments.size();
 mComments.clear();
 notifyItemRangeRemoved(0, oldSize);
 }

 public void setIssue(Issue issue) {
```

```
mIssue = issue;
clear();
mPage = 1;
mLoader.loadIssueComments(this, mIssue.getRepoFullName(),
mIssue.getNumber(), mPage);
}

@Override
public void listLoadComplete(List<Comment> data) {
 mRefresher.setRefreshing(false);
 mIsLoading = false;
 if(data.size() > 0) {
 int oldLength = mComments.size();
 for(Comment c : data) {
 mComments.add(Pair.create(c, null));
 }
 notifyDataSetChanged(oldLength, mComments.size());
 } else {
 mMaxPageReached = true;
 }
}

@Override
public void listLoadError(APIHandler.APIError error) {

}

public void notifyBottomReached() {
 if(!mIsLoading && !mMaxPageReached) {
 mPage++;
 loadComments(false);
 }
}

private void loadComments(boolean resetPage) {
 mIsLoading = true;
 mRefresher.setRefreshing(true);
 if(resetPage) {
 mPage = 1;
 mMaxPageReached = false;
 }
 mLoader.loadIssueComments(this, mIssue.getRepoFullName(),
mIssue.getNumber(), mPage);
}

public void addComment(Comment comment) {
 mComments.add(Pair.create(comment, null));
 notifyDataSetChanged(mComments.size());
}

public void removeComment(int commentId) {
 int index = -1;
 for(int i = 0; i < mComments.size(); i++) {
 if(mComments.get(i).first.getId() == commentId) {
 index = i;
 break;
 }
 }
 if(index != -1) {
```

```
 mComments.remove(index);
 notifyItemRemoved(index);
 }
}

public void updateComment(Comment comment) {
 int index = -1;
 for(int i = 0; i < mComments.size(); i++) {
 if(mComments.get(i).first.getId() == comment.getId()) {
 index = i;
 break;
 }
 }
 if(index != -1) {
 mComments.set(index, Pair.create(comment, null));
 notifyDataSetChanged(index);
 }
}

@Override
public CommentHolder onCreateViewHolder(ViewGroup parent, int viewType) {
 return new CommentHolder(LayoutInflater.from(parent.getContext()))

.inflate(R.layout.viewholder_comment, parent,
 false
));
}

@Override
public void onBindViewHolder(CommentHolder holder, int position) {
 final int pos = holder.getAdapterPosition();
 final Comment comment = mComments.get(pos).first;
 if(mComments.get(pos).second == null) {
 holder.mAvatar.setImageUrl(comment.getUser().getAvatarUrl());
 final StringBuilder builder = new StringBuilder();
 builder.append(String.format(
holder.itemView.getResources().getString(R.string.text_comment_by),
 String.format(
holder.itemView.getResources().getString(R.string.text_href),
 comment.getUser().getHtmlUrl(),
 comment.getUser().getLogin()
),
DateUtils.getRelativeTimeSpanString(comment.getCreatedAt())
));
 if(comment.getUpdatedAt() != comment.getCreatedAt()) {
 builder.append(" • ");
 builder.append(holder.itemView.getResources()
.getString(R.string.text_comment_edited));
 }
 holder.mCommenter.setMarkdown(builder.toString());
 builder.setLength(0);
 builder.append(Markdown.formatMD(comment.getBody(),
mIssue.getRepoFullName()));
 if(comment.hasReaction()) {
```

```
 builder.append("\n");
 builder.append(Formatter.reactions(comment.getReaction()));
 }

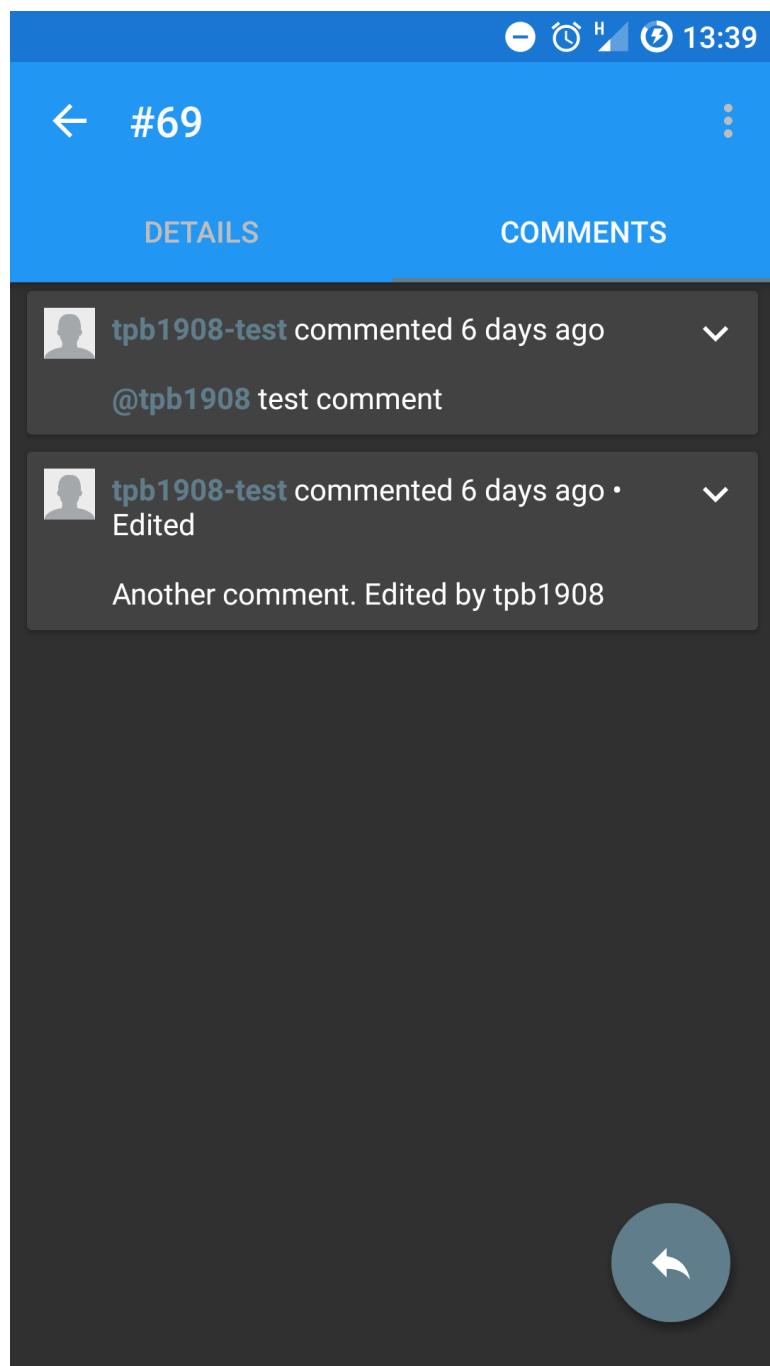
 holder.mBody.setMarkdown(
 builder.toString(),
 new HttpImageGetter(holder.mBody),
 text -> mComments.set(pos, Pair.create(comment, text))
);
} else {
 holder.mAvatar.setImageUrl(comment.getUser().getAvatarUrl());
 holder.mBody.setText(mComments.get(pos).second);
}
IntentHandler.addOnClickHandler(mParent.getActivity(), holder.mBody);
IntentHandler.addOnClickHandler(mParent.getActivity(), holder.mAvatar,
 comment.getUser().getLogin()
);
holder.mMenu.setOnClickListener(v -> displayMenu(v,
holder.getAdapterPosition()));
}

@Override
public int getItemCount() {
 return mComments.size();
}

private void displayMenu(View view, int pos) {
 mParent.displayCommentMenu(view, mComments.get(pos).first);
}

static class CommentHolder extends RecyclerView.ViewHolder {
 @BindView(R.id.event_comment_avatar) NetworkImageView mAvatar;
 @BindView(R.id.comment_commenter) MarkdownTextView mCommenter;
 @BindView(R.id.comment_text) MarkdownTextView mBody;
 @BindView(R.id.comment_menu_button) ImageButton mMenu;

 CommentHolder(View view) {
 super(view);
 ButterKnife.bind(this, view);
 }
}
}
```



# ProjectActivity

The ProjectActivity and ColumnFragment are the most complicated of the app as they have to deal with the most possible states, and a limited API.

The ProjectActivity deals with:

- Managing the the loading of the Project
- Managing the loading of each column
- Managing the priority of loading Issues
- Managing the refreshing of content
- Managing the creation of:
  - New columns
  - New cards
  - New issue and the cards created from them
- Deletion of cards
- Searching of loaded content
- Checking access to the repository
- Dragging and dropping of cards between columns
- Dragging and dropping of columns

## ProjectActivity.java

```
package com(tpb.projects.project;

import android.app.Dialog;
import android.content.Context;
import android.content.Intent;
import android.graphics.Rect;
import android.net.Uri;
import android.os.Bundle;
import android.os.Handler;
import android.os.Parcel;
import android.support.annotation.Nullable;
import android.support.design.widget.Snackbar;
import android.support.v4.app.FragmentManager;
import android.support.v4.view.ViewPager;
import android.support.v4.widget.NestedScrollView;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.SearchView;
import android.support.v7.widget.Toolbar;
import android.util.DisplayMetrics;
import android.view.DragEvent;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.WindowManager;
```

```
import android.view.inputmethod.InputMethodManager;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.commonsware.cwac.page.PageDescriptor;
import com.commonsware.cwac.page.v4.ArrayPagerAdapter;
import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Editor;
import com(tpb.github.data.Loader;
import com(tpb.github.data.auth.GitHubSession;
import com(tpb.github.data.models.Card;
import com(tpb.github.data.models.Column;
import com(tpb.github.data.models.Comment;
import com(tpb.github.data.models.Issue;
import com(tpb.github.data.models.Project;
import com(tpb.github.data.models.Repository;
import com(tpb.github.data.models.State;
import com(tpb.projects.R;
import com(tpb.projects.common BaseActivity;
import com(tpb.projects.common ShortcutDialog;
import com(tpb.projects.common.fab FloatingActionButton;
import com(tpb.projects.common.fab FloatingActionMenu;
import com(tpb.projects.editors.CardEditor;
import com(tpb.projects.editors.CommentEditor;
import com(tpb.projects.editors.IssueEditor;
import com(tpb.projects.util Analytics;
import com(tpb.projects.util SettingsActivity;
import com(tpb.projects.util UI;

import java.util.ArrayList;
import java.util.List;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;

import static com(tpb.projects.flow ProjectsApplication.mAnalytics;

/**
 * Created by theo on 19/12/16.
 */

public class ProjectActivity extends BaseActivity implements
Loader.ItemLoader<Project> {
 private static final String TAG = ProjectActivity.class.getSimpleName();
 private static final String URL =
"https://github.com/tpb1908/AndroidProjectsClient/blob/master/app/src/main/java/com/tpb/projects/project/ProjectActivity.java";

 @BindView(R.id.project_toolbar) Toolbar mToolbar;
 @BindView(R.id.project_name) TextView mName;
 @BindView(R.id.project_refresher) SwipeRefreshLayout mRefresher;
 @BindView(R.id.project_column_pager) ViewPager mColumnPager;
 @BindView(R.id.project_fab_menu) FloatingActionMenu mMenu;
 @BindView(R.id.project_add_card) FloatingActionButton mAddCard;
 @BindView(R.id.project_add_column) FloatingActionButton mAddColumn;
 @BindView(R.id.project_add_issue) FloatingActionButton mAddIssue;
```

```
private SearchView mSearchView;
private MenuItem mSearchItem;

private ColumnPagerAdapter mAdapter;
private int mCurrentPosition = -1;
private Loader mLoader;
Project mProject;
private Editor mEditor;
private NavigationDragListener mNavListener;
private Repository.AccessLevel mAccessLevel =
Repository.AccessLevel.ADMIN;
private int mLaunchCardId = -1;
private int mLoadCount;

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 final SettingsActivity.Preferences prefs =
SettingsActivity.Preferences
 .getPreferences(this);
 setTheme(prefs.isDarkThemeEnabled() ? R.style.AppTheme_Dark :
R.style.AppTheme);
 UI.setStatusBarColor(getWindow(),
getResources().getColor(R.color.colorPrimaryDark));
 setContentView(R.layout.activity_project);
 ButterKnife.bind(this);

 final Intent launchIntent = getIntent();
 mLoader = Loader.getLoader(this);
 mEditor = Editor.getEditor(this);

 setSupportActionBar(mToolbar);
 getSupportActionBar().setDisplayShowTitleEnabled(false);

 if(launchIntent.getStringExtra(getString(R.string.parcel_project))) {

loadComplete(launchIntent.getParcelableExtra(getString(R.string.parcel_project
)));
 if(launchIntent.getStringExtra(getString(R.string.intent_access_level)))
{
 mAccessLevel = (Repository.AccessLevel) launchIntent
 .getSerializableExtra(getString(R.string.intent_access_level));
 if(mAccessLevel == Repository.AccessLevel.ADMIN ||
mAccessLevel == Repository.AccessLevel.WRITE) {
 new Handler().postDelayed(() ->
mMenu.showMenuButton(true), 400);
 }
 } else {
 checkAccess(mProject);
 }
 } else {
 final String repo =
launchIntent.getStringExtra(getString(R.string.intent_repo));
 final int number = launchIntent
 .getIntExtra(getString(R.string.intent_project_number),
1);
 if(launchIntent.getStringExtra(getString(R.string.intent_card_id))) {
```

```

 mLaunchCardId =
 launchIntent.getIntExtra(getString(R.string.intent_card_id), -1);
 }
 loadFromId(repo, number);
}
//Ensure that the keyboard does not show

getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_HIDDEN);

 initialiseListeners();
}

private void initialiseListeners() {
 mAdapter = new ColumnPagerAdapter(getSupportFragmentManager(), new
ArrayList<>());
 mColumnPager.setAdapter(mAdapter);
 mColumnPager.addOnPageChangeListener(new
ViewPager.OnPageChangeListener() {
 @Override
 public void onPageScrolled(int position, float positionOffset, int
positionOffsetPixels) {

 }

 @Override
 public void onPageSelected(int position) {
 mCurrentPosition = position;
 if(mAccessLevel == Repository.AccessLevel.ADMIN ||
mAccessLevel == Repository.AccessLevel.WRITE) {
 showFab();
 }
 }

 @Override
 public void onPageScrollStateChanged(int state) {
 if(state == ViewPager.SCROLL_STATE_DRAGGING) {
 mRefresher.setEnabled(false);
 } else if(state == ViewPager.SCROLL_STATE_IDLE) {
 mRefresher.setEnabled(true);
 }
 }
});
});

mRefresher.setRefreshing(true);
mMenu.hideMenuButton(false); //Hide the button so that we can show it
later
mMenu.setClosedOnTouchOutside(true);
mRefresher.setOnRefreshListener(() -> {
 mLoader.loadProject(ProjectActivity.this, mProject.getId());
});
mRefresher.setOnChildScrollUpCallback((parent, child) -> {
 mAdapter.getCurrentFragment().notifyScroll();
 return false;
});
mNavListener = new NavigationDragListener();
mRefresher.setOnDragListener(mNavListener);
}

```

```
private void loadFromId(String repo, int number) {
 //We have to load all of the projects to get the id that we want
 //This is because we have the number and need the id
 mLoader.loadProjects(new Loader.ListLoader<Project>() {

 @Override
 public void listLoadComplete(List<Project> projects) {
 for(Project p : projects) {
 if(number == p.getNumber()) {
 ProjectActivity.this.loadComplete(p);
 checkAccess(p);
 return;
 }
 }
 Toast.makeText(ProjectActivity.this,
R.string.error_project_not_found,
 Toast.LENGTH_LONG
).show();
 finish();
 }

 @Override
 public void listLoadError(APIHandler.APIError error) {
 mRefresher.setRefreshing(false);
 Toast.makeText(ProjectActivity.this, error.resId,
Toast.LENGTH_SHORT).show();
 }
 }, repo);
}

private void checkAccess(Project project) {
 mLoader.checkAccessToRepository(new
Loader.ItemLoader<Repository.AccessLevel>() {

 @Override
 public void loadComplete(Repository.AccessLevel data) {
 mAccessLevel = data;
 if(mAccessLevel == Repository.AccessLevel.ADMIN ||
mAccessLevel == Repository.AccessLevel.WRITE) {
 mMenu.showMenuButton(true);
 } else {
 mMenu.hideMenuButton(false);
 }
 for(int i = 0; i < mAdapter.getCount(); i++) {
 if(mAdapter.getExistingFragment(i) != null) {

mAdapter.getExistingFragment(i).setAccessLevel(mAccessLevel);
 }
 }
 }

 @Override
 public void loadError(APIHandler.APIError error) {
 mRefresher.setRefreshing(false);
 Toast.makeText(ProjectActivity.this, error.resId,
Toast.LENGTH_SHORT).show();
 }
 }
}
```

```
 }, GitHubSession.getSession(this).getUserLogin(),
project.getRepoPath());
}

void showFab() {
 mMenu.showMenuButton(true);
}

void hideFab() {
 mMenu.hideMenuButton(true);
}

@Override
public void loadComplete(Project project) {
 mProject = project;
 mName.setText(mProject.getName());
 mName.setCompoundDrawablesRelativeWithIntrinsicBounds(
 project.getState() == State.OPEN ? R.drawable.ic_state_open :
R.drawable.ic_state_closed,
 0, 0, 0
);

 mLoadCount = 0;
 mLoader.loadColumns(new Loader.ListLoader<Column>() {
 @Override
 public void listLoadComplete(List<Column> columns) {
 if(columns.size() > 0) {
 mAddCard.setVisibility(View.INVISIBLE);
 mAddIssue.setVisibility(View.INVISIBLE);

 int id = 0;
 if(mCurrentPosition != -1) {
 id = mAdapter.getCurrentFragment().mColumn.getId();
 }
 mCurrentPosition = 0;
 mAdapter.columns = new ArrayList<>(columns);
 if(mAdapter.getCount() != 0) {
 for(int i = mAdapter.getCount() - 1; i >= 0; i--)
mAdapter.remove(i);
 }
 for(int i = 0; i < columns.size(); i++) {
 mAdapter.add(new
ColumnPageDescriptor(columns.get(i)));
 if(columns.get(i).getId() == id) {
 mCurrentPosition = i;
 }
 }
 mColumnPager.setOffscreenPageLimit(mAdapter.getCount());
 if(mCurrentPosition >= mAdapter.getCount()) {
 mCurrentPosition = mAdapter.getCount() - 1;
 //If the end column has been deleted
 }
 mColumnPager.setCurrentItem(mCurrentPosition, true);
 mColumnPager.postDelayed(() ->
mColumnPager.setVisibility(View.VISIBLE), 300);
 } else {
 mRefresher.setRefreshing(false);
 mAddCard.setVisibility(View.GONE);
 mAddIssue.setVisibility(View.GONE);
 }
 }
 });
}
```

```

 }

 }

 @Override
 public void listLoadError(APIHandler.APIError error) {
 mRefresher.setRefreshing(false);
 Toast.makeText(ProjectActivity.this, error.resId,
 Toast.LENGTH_SHORT)
 .show();
 }
}, mProject.getId());

}

@Override
public void loadError(APIHandler.APIError error) {
 final Bundle bundle = new Bundle();
 bundle.putString(Analytics.KEY_LOAD_STATUS, Analytics.VALUE_FAILURE);
 mAnalytics.logEvent(Analytics.TAG_PROJECT_LOADED, bundle);
}

void loadIssue(Loader.ItemLoader<Issue> loader, int issueId, Column
column) {
 mLoader.loadIssue(loader, mProject.getRepoPath(), issueId,
 mAdapter.indexOf(column.getId()) == mCurrentPosition
);
}

@OnClick(R.id.project_add_column)
void addColumn() {
 mMenu.close(true);
 final AlertDialog dialog = new AlertDialog.Builder(this)
 .setView(R.layout.dialog_new_column)
 .setTitle(R.string.title_new_column)
 .setNegativeButton(R.string.action_cancel, null)
 .create();
 dialog.setButton(AlertDialog.BUTTON_POSITIVE,
getString(R.string.action_ok), (di, w) -> {
 });
 dialog.getWindow().getAttributes().windowAnimations =
R.style.DialogAnimation;
 dialog.show();
 dialog.getButton(AlertDialog.BUTTON_POSITIVE).setOnClickListener(v ->
{
 final EditText editor = (EditText)
dialog.findViewById(R.id.project_new_column);
 final String text = editor.getText().toString();
 final InputMethodManager imm = (InputMethodManager)
getSystemService(
 Context.INPUT_METHOD_SERVICE);
 if(imm.isActive()) {
 imm.toggleSoftInput(InputMethodManager.HIDE_IMPLICIT_ONLY, 0);
 }
 if(!text.isEmpty()) {
 mRefresher.setRefreshing(true);
 mEditor.addColumn(new Editor.CreationListener<Column>() {

 @Override
 public void created(Column column) {

```

```
 mAddCard.setVisibility(View.INVISIBLE);
 mAddIssue.setVisibility(View.INVISIBLE);
 mAdapter.columns.add(column);
 if(mAdapter.columns.size() == 0) {
 mAdapter.notifyDataSetChanged();
 } else {
 mAdapter.add(new ColumnPageDescriptor(column));
 mColumnPager.setCurrentItem(mAdapter.getCount(),
true);
 }
 mRefresher.setRefreshing(false);
 }

 @Override
 public void creationError(APIHandler.APIError error) {
 mRefresher.setRefreshing(false);
 Toast.makeText(ProjectActivity.this, error.resId,
Toast.LENGTH_SHORT)
 .show();
 }
 }, mProject.getId(), text);
 dialog.dismiss();
} else {
 Toast.makeText(this, R.string.error_no_column_title,
Toast.LENGTH_SHORT).show();
}
});
dialog.getButton(AlertDialog.BUTTON_NEGATIVE).setOnClickListener(v ->
{
 final InputMethodManager imm = (InputMethodManager)
getSystemService(
 Context.INPUT_METHOD_SERVICE);
 if(imm.isActive()) {
 imm.toggleSoftInput(InputMethodManager.HIDE_IMPLICIT_ONLY, 0);
 }
 dialog.dismiss();
});
final InputMethodManager imm = (InputMethodManager) getSystemService(
 Context.INPUT_METHOD_SERVICE);
imm.toggleSoftInput(InputMethodManager.SHOW_FORCED, 0);
}

@OnClick(R.id.project_add_issue)
void addIssue() {
 final Intent intent = new Intent(ProjectActivity.this,
IssueEditor.class);
 intent.putExtra(getString(R.string.intent_repo),
mProject.getRepoPath());
 UI.setViewPositionForIntent(intent, mAddIssue);
 startActivityForResult(intent, IssueEditor.REQUEST_CODE_NEW_ISSUE);
}

@OnClick(R.id.project_add_card)
void addCard() {
 final Intent intent = new Intent(this, CardEditor.class);

 final ArrayList<Integer> ids = new ArrayList<>();
 for(int i = 0; i < mAdapter.getCount(); i++) {
 for(Card c : mAdapter.getExistingFragment(i).getCards()) {
```

```
 if(c.hasIssue()) ids.add(c.getIssue().getId());
 }
}
UI.setViewPositionForIntent(intent, mAddCard);
intent.putExtra(getString(R.string.intent_repo),
mProject.getRepoPath());

intent.putIntegerArrayListExtra(getString(R.string.intent_int_arraylist),
ids);
startActivityForResult(intent, CardEditor.REQUEST_CODE_NEW_CARD);
}

void deleteColumn(Column column) {
 new AlertDialog.Builder(this, R.style.DialogAnimation)
 .setTitle(R.string.title_delete_column)
 .setMessage(R.string.text_delete_column_warning)
 .setNegativeButton(R.string.action_cancel, null)
 .setPositiveButton(R.string.action_ok, (dialogInterface, i) ->
{
 mRefresher.setRefreshing(true);
 mEditor.deleteColumn(new
Editor.DeletionListener<Integer>() {

 @Override
 public void deleted(Integer integer) {
 mAdapter.remove(mCurrentPosition);
 mAdapter.columns.remove(mCurrentPosition);
 mRefresher.setRefreshing(false);
 if(mAdapter.columns.size() == 0) {
 mAddCard.setVisibility(View.GONE);
 mAddIssue.setVisibility(View.GONE);
 }
 }
 }

 @Override
 public void deletionError(APIHandler.APIError error) {
 mRefresher.setRefreshing(false);
 Toast.makeText(ProjectActivity.this, error.resId,
 Toast.LENGTH_SHORT
).show();
 }
 }, column.getId());
}).show();
}

void deleteCard(Card card, boolean showWarning) {
 final Editor.DeletionListener<Card> listener = new
Editor.DeletionListener<Card>() {
 @Override
 public void deleted(Card card) {
 mRefresher.setRefreshing(false);
 mAdapter.getCurrentFragment().removeCard(card);
 Snackbar.make(findViewById(R.id.project_coordinator),
 getString(R.string.text_note_deleted),
 Snackbar.LENGTH_LONG
)
 .setAction(getString(R.string.action_undo),
```

```
 view ->
mAdapter.getCurrentFragment().recreateCard(card)
)
 .show();
}

@Override
public void deletionError(APIHandler.APIError error) {
}

};

if(showWarning) {
 final Dialog dialog = new AlertDialog.Builder(this)
 .setTitle(R.string.title_delete_card)
 .setMessage(R.string.text_delete_note_warning)
 .setNegativeButton(R.string.action_cancel, null)
 .setPositiveButton(R.string.action_ok, (dialogInterface,
i) -> {
 mRefresher.setRefreshing(true);
 mEditor.deleteCard(listener, card);
 }).create();
 dialog.getWindow().getAttributes().windowAnimations =
R.style.DialogAnimation;
 dialog.show();
} else {
 mRefresher.setRefreshing(true);
 mEditor.deleteCard(listener, card);
}
}

void notifyFragmentLoaded() {
 mLoadCount++;
 if(mLoadCount == mAdapter.getCount()) {
 mRefresher.setRefreshing(false);
 if(mLaunchCardId != -1) {
 new Handler().postDelayed(() ->
mAdapter.moveTo(mLaunchCardId), 500);
 }
 }
}

@Override
public void onBackPressed() {
 if(mMenu.isOpened()) {
 mMenu.close(true);
 } else {
 mColumnPagerAdapter.setAdapter(null);
 mMenu.hideMenuButton(true);
 super.onBackPressed();
 }
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
 getMenuInflater().inflate(R.menu.menu_activity_search, menu);
 mSearchItem = menu.findItem(R.id.menu_action_search);

 if(mSearchItem != null) {
 mSearchView = (SearchView) mSearchItem.getActionView();
 }
}
```

```
 }

 return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
 switch(item.getItemId()) {
 case R.id.menu_settings:
 startActivity(new Intent(ProjectActivity.this,
SettingsActivity.class));
 break;
 case R.id.menu_source:
 startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse(URL)));
 break;
 case R.id.menu_share:
 final Intent share = new Intent();
 share.setAction(Intent.ACTION_SEND);
 share.putExtra(Intent.EXTRA_TEXT, "https://github.com/" +
 mProject.getRepoPath() +
 "/projects/" +
 Integer.toString(mProject.getNumber()));
 share.setType("text/plain");
 startActivity(share);
 break;
 case R.id.menu_save_to_homescreen:
 final ShortcutDialog dialog = new ShortcutDialog();
 final Bundle args = new Bundle();
 args.putInt(getString(R.string.intent_title_res),
 R.string.title_save_project_shortcut
);
 args.putString(getString(R.string.intent_name),
mProject.getName());

 dialog.setArguments(args);
 dialog.setListener((name, iconFlag) -> {
 final Intent i = new Intent(getApplicationContext(),
ProjectActivity.class);
 i.putExtra(getString(R.string.intent_repo),
mProject.getRepoPath());
 i.putExtra(getString(R.string.intent_project_number),
mProject.getNumber());

 final Intent add = new Intent();
 add.putExtra(Intent.EXTRA_SHORTCUT_INTENT, i);
 add.putExtra(Intent.EXTRA_SHORTCUT_NAME, name);
 add.putExtra("duplicate", false);
 add.putExtra(Intent.EXTRA_SHORTCUT_ICON_RESOURCE,
Intent.ShortcutIconResource
 .fromContext(getApplicationContext(),
R.mipmap.ic_launcher));

 add.setAction("com.android.launcher.action.INSTALL_SHORTCUT");
 getApplicationContext().sendBroadcast(add);
 });
 dialog.show(getSupportFragmentManager(), TAG);
 break;
 case R.id.menu_action_search:
 if(mAdapter.getCount() > 0) {
```

```

 final SearchView.SearchAutoComplete searchSrc =
(SearchView.SearchAutoComplete) mSearchView

.findViewById(android.support.v7.appcompat.R.id.search_src_text);
 searchSrc.setThreshold(1);
 final ProjectSearchAdapter searchAdapter = new
ProjectSearchAdapter(this,
 mAdapter.getAllCards()
);
 searchSrc.setAdapter(searchAdapter);
 searchSrc.setOnItemClickListener((adapterView, view, i, l)
-> {
 mSearchItem.collapseActionView();
 mAdapter.moveTo(searchAdapter.getItem(i).getId());
});
}

return true;
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
 super.onActivityResult(requestCode, resultCode, data);
 if(resultCode == AppCompatActivity.RESULT_OK) {
 mMenu.close(true);
 mRefresher.setRefreshing(true);
 if(requestCode == IssueEditor.REQUEST_CODE_NEW_ISSUE) {
 String[] assignees = null;
 String[] labels = null;
 if(data.getStringExtra(getString(R.string.intent_issue_assignees)))
{
 assignees = data
.getStringArrayExtra(getString(R.string.intent_issue_assignees));
 }
 if(data.getStringExtra(getString(R.string.intent_issue_labels))) {
 labels =
data.getStringArrayExtra(getString(R.string.intent_issue_labels));
 }
 final Issue issue =
data.getParcelableExtra(getString(R.string.parcel_issue));
 mEditor.createIssue(new Editor.CreationListener<Issue>() {
 @Override
 public void created(Issue issue) {
 mAdapter.getCurrentFragment().createIssueCard(issue);
 final Bundle bundle = new Bundle();
 bundle.putString(Analytics.KEY_EDIT_STATUS,
Analytics.VALUE_SUCCESS);
 mAnalytics.logEvent(Analytics.TAG_ISSUE_CREATED,
bundle);
 mRefresher.setRefreshing(false);
 }
 }
 }
 }
}
public void creationError(APIHandler.APIError error) {
 final Bundle bundle = new Bundle();
}

```

```
 bundle.putString(Analytics.KEY_EDIT_STATUS,
Analytics.VALUE_FAILURE);
 mAnalytics.logEvent(Analytics.TAG_ISSUE_CREATED,
bundle);
 mRefresher.setRefreshing(false);
 }
 }, mProject.getRepoPath(), issue.getTitle(), issue.getBody(),
assignees, labels);

 } else if(requestCode == CardEditor.REQUEST_CODE_NEW_CARD) {
 final Card card =
data.getParcelableExtra(getString(R.string.parcel_card));
 if(card.hasIssue()) {

mAdapter.getCurrentFragment().createIssueCard(card.getIssue());
 } else {
 mAdapter.getCurrentFragment().newCard(card);
 }
} else if(requestCode == CardEditor.REQUEST_CODE_EDIT_CARD) {

mAdapter.getCurrentFragment().editCard(data.getParcelableExtra(getString(R.str
ing.parcel_card)));
} else if(requestCode ==
CommentEditor.REQUEST_CODE_COMMENT_FOR_STATE) {
 final Comment comment =
data.getParcelableExtra(getString(R.string.parcel_comment));
 final Issue issue =
data.getParcelableExtra(getString(R.string.parcel_issue));
 mEditor.createIssueComment(new
Editor.CreationListener<Comment>() {
 @Override
 public void created(Comment comment) {
 Toast.makeText(ProjectActivity.this,
R.string.text_comment_created,
 Toast.LENGTH_SHORT
).show();
 mRefresher.setRefreshing(false);
 }

 @Override
 public void creationError(APIHandler.APIError error) {
 mRefresher.setRefreshing(false);
 }
 },
 issue.getRepoFullName(), issue.getNumber(),
comment.getBody());
} else if(requestCode == IssueEditor.REQUEST_CODE_EDIT_ISSUE) {
 mAdapter.getCurrentFragment().onActivityResult(requestCode,
resultCode, data);
 }
}
}

/**
 * @param tag id of the column being moved
 * @param dropTag id of the column being dropped onto
 * @param direction side of the drop column to drop to true=left
false=right
 */
void moveColumn(int tag, int dropTag, boolean direction) {
```

```

 final int from = mAdapter.indexOf(tag);
 final int to;
 if(direction) {
 to = Math.max(0, mAdapter.indexOf(dropTag) - 1);
 } else {
 to = Math.min(mAdapter.getCount() - 1, mAdapter.indexOf(dropTag) +
1);
 }
 mAdapter.move(from, to);
 mAdapter.columns.add(to, mAdapter.columns.remove(from));
 mColumnPager.setCurrentItem(to, true);
 mEditor.moveColumn(new Editor.UpdateListener<Integer>() {
 @Override
 public void updated(Integer integer) {

 }

 @Override
 public void updateError(APIHandler.APIError error) {

 }
 }, tag, dropTag, to);
 }

 private void dragLeft() {
 if(mCurrentPosition > 0) {
 mColumnPager.setCurrentItem(mCurrentPosition - 1, true);
 }
 }

 private void dragRight() {
 if(mCurrentPosition < mAdapter.getCount()) {
 mColumnPager.setCurrentItem(mCurrentPosition + 1, true);
 }
 }

 private void dragUp() {
 mAdapter.getCurrentFragment().scrollUp();
 }

 private void dragDown() {
 mAdapter.getCurrentFragment().scrollDown();
 }

 class NavigationDragListener implements View.OnDragListener {

 private long mLastPageChange = 0;

 @Override
 public boolean onDrag(View view, DragEvent event) {
 final DisplayMetrics metrics = getResources().getDisplayMetrics();
 if(event.getAction() == DragEvent.ACTION_DRAG_ENTERED && view
 .getId() == R.id.viewholder_card) {
 final RecyclerView rv = (RecyclerView) view.getParent();
 final CardAdapter ca = (CardAdapter) rv.getAdapter();
 final Rect r = new Rect();
 ((NestedScrollView) rv.getParent().getParent()).getHitRect(r);
 int first = -1;
 int last = -1;

```

```

 for(int i = 0; i < rv.getAdapter().getItemCount(); i++) {
 if(rv.getChildAt(i).getLocalVisibleRect(r)) {
 if(first == -1) {
 first = i;
 } else if(i == rv.getAdapter().getItemCount() - 1) {
 last = i;
 }
 } else if(first != -1) {
 last = i - 1;
 break;
 }
 }
 final int tp = ca.indexOf((int) view.getTag());
 final float relativePos = event.getY() - view.getY();
 final int[] pos = new int[2];
 if(tp == first) {
 rv.getChildAt(first).getLocationOnScreen(pos);
 if(pos[1] + relativePos < 0.1 * metrics.heightPixels) {
 dragUp();
 }
 }

 } else if(tp == last) {
 rv.getChildAt(last).getLocationOnScreen(pos);
 if(pos[1] + relativePos > 0.9 * metrics.heightPixels) {
 dragDown();
 }
 }
 }

} else if(event.getAction() == DragEvent.ACTION_DRAG_LOCATION) {
 if(event.getX() / metrics.widthPixels > 0.85f && System
 .nanoTime() - mLastPageChange > 5E8) {
 dragRight();
 mLastPageChange = System.nanoTime();
 } else if(event.getX() / metrics.widthPixels < 0.15f && System
 .nanoTime() - mLastPageChange > 5E8) {
 dragLeft();
 mLastPageChange = System.nanoTime();
 }
}
return true;
}

private class ColumnPagerAdapter extends ArrayPagerAdapter<ColumnFragment>
{
 private List<Column> columns = new ArrayList<>();

 ColumnPagerAdapter(FragmentManager manager, List<PageDescriptor>
descriptors) {
 super(manager, descriptors);
 }

 int indexOf(int id) {
 for(int i = 0; i < columns.size(); i++) {
 if(columns.get(i).getId() == id) return i;
 }
 return -1;
 }
}

```

```
 }

 List<Card> getAllCards() {
 final List<Card> cards = new ArrayList<>();
 for(int i = 0; i < getCount(); i++) {
 cards.addAll(getExistingFragment(i).getCards());
 }
 return cards;
 }

 void moveTo(int cardId) {
 for(int i = 0; i < getCount(); i++) {
 if(getExistingFragment(i).attemptMoveTo(cardId)) {
 mColumnPagerAdapter.setCurrentItem(i, true);
 break;
 }
 }
 }

 @Override
 protected ColumnFragment createFragment(PageDescriptor pageDescriptor)
 {
 return ColumnFragment.getInstance(
 ((ColumnPageDescriptor) pageDescriptor).mColumn,
 mNavListener,
 mAccessLevel
);
 }

}

private static class ColumnPageDescriptor implements PageDescriptor {
 private final Column mColumn;

 ColumnPageDescriptor(Column column) {
 mColumn = column;
 }

 @Override
 public String getFragmentTag() {
 return Integer.toString(mColumn.getId());
 }

 @Override
 public String getTitle() {
 return mColumn.getName();
 }

 @Override
 public int describeContents() {
 return 0;
 }

 @Override
 public void writeToParcel(Parcel dest, int flags) {
 dest.writeParcelable(mColumn, flags);
 }

 ColumnPageDescriptor(Parcel in) {
```

```

 this.mColumn = in.readParcelable(Column.class.getClassLoader());
 }

 public static final Creator<ColumnPageDescriptor> CREATOR = new
Creator<ColumnPageDescriptor>() {
 @Override
 public ColumnPageDescriptor createFromParcel(Parcel source) {
 return new ColumnPageDescriptor(source);
 }

 @Override
 public ColumnPageDescriptor[] newArray(int size) {
 return new ColumnPageDescriptor[size];
 }
};

}
}

```

## Loading the Project

The Project model can be passed as a parcel, or passed as the project number when launched from the Interceptor.

The first case is trivial, however the second is a problem because although the URL for a project only gives the project number, there is no API endpoint to load the project from its number and repository.

Instead, `loadFromId` is called.

This method loads all of the Project models for a repository, and checks their numbers against the number passed from the Interceptor.

When `LoadComplete` is called, the title and its state drawable are set, and the next stage of loading can begin.

## Loading the columns

The Column models are loaded with a call to `Loader.loadColumns`.

If there are columns to be shown the FloatingActionButtons for adding cards and issues are set to the invisible state, and may be made visible if the authenticated user has access to edit the project.

If the Columns have already been loaded and displayed, the id of the currently visible column is saved before the ColumnFragments are removed.

Each of the new Columns are then added, and if the same Column id is found again, `mCurrentPosition` is saved.

Once the Fragments have been created, the position is checked to ensure that the column still exists, and the adapter is moved to this column.

## Adding a new column

`addColumn` is the `onClick` method for one of the `FloatingActionButtons`. It shows a dialog to input the name of a new column. A listener is then added to the dialog to capture the input if the positive button is pressed, and create the new column before adding it to the adapter and moving to the new position.

## **Deleting columns**

When a user attempts to delete a column, an `AlertDialog` is shown asking them to confirm that they wish to delete the column. If the user confirms their action, the request is made to delete the column and the `ColumnFragment` is removed from the adapter.

## **Dragging and dropping Fragments in a ViewPager**

In order to have the same functionality as the desktop website, users should be able to re-order the columns in a project.

This is done by allowing the `Fragments` to be dragged and dropped when their header is long pressed.

While the `Fragment` layouts themselves are far too complex to move around the screen, the header card containing the column title can be easily moved.

The `moveColumn` method is called from the `ColumnFragment` in an `OnDragListener` set on the header card. This will be explained in further detail later.

The `tag` parameter is the tag set on the `ColumnFragment` to be moved, and the `dropTag` is the tag set on the `View` which the detached header card is being held over.

The new position is calculated from the position of the dragging `Fragment` in the adapter, and the `Fragment` is moved from one position to the other.

The `ViewPager` is then set to the new position, and the API call is made to perform the movement.

Of course, prior to the `Fragment` being moved, the user must already have performed the dragging action.

In the `ColumnFragment` an instance of `ColumnDragListener` is attached to the header card containing information about the column.

The `OnLongClickListener` is then set on the card to create a `DrawShadowBuilder` and start a drag and drop action with the shadowed version of the `View`.

The `OnDragListener` is called when the shadow is placed over another `view`.

In the event of the shadow being released, the object returned from `DragEvent.getLocalState` is the `View` which the shadow is being held over. If the tag of this `View` is not equal to the tag of the `View` being dragged, and the `View` has the `column_card` id, the `moveColumn` method is called.

This still does not explain how the user drags their shadowed view onto another Fragment, they are currently only able to drag the view around the Fragment which it belongs to, not triggering any action.

The action of dragging the shadowed view to the edge of the screen and performing a scroll is handled by the `NavigationDragListener`, another implementation of `OnDragListener`.

There is only one instance of `NavigationDragListener` and it is passed to each `ColumnFragment` which is created.

This listener is then attached to the `RecyclerView` in each `ColumnFragment` via a `CardDragListener`.

The `CardDragListener` is the third and final implementation of `OnDragListener` and will be explained later in the context of dragging and dropping cards.

For now, it is only necessary to know that the `CardDragListener` may have a reference to a parent `OnDragListener` and it passes `onDrag` events to this parent without having to set one huge `OnDragListener` on every View.

To recap:

- The `ColumnDragListener` tells the `ProjectActivity` when to move a `ColumnFragment` from one position to another. It does this when the view being dragged is released by the user over another header card view.
- The `NavigationDragListener` manages the `onDrag` events when the shadowed view is dragged over the views contained within the `RecyclerView` in each `ColumnFragment` and the `RecyclerView` itself. It determines whether the user is trying to drag the shadowed view to the next `ColumnFragment` or up or down the `RecyclerView`.

## **NavigationDragListener**

In the `onDrag` method of `NavigationDragListener` an instance of `DisplayMetrics` is collected.

The event action is then checked.

The two actions that we are concerned with are `ACTION_DRAG_ENTERED`, which occurs when the shadowed view enters the bounds of a new view with an `OnDragListener`, and `ACTION_DRAG_LOCATION` which is fired while the shadow view is *inside* the bounds of a view with an `OnDragListener`.

I will not yet explain the process behind `ACTION_DRAG_ENTERED` as it is only used when dragging cards around, not column headers.

`ACTION_DRAG_LOCATION` on the other hand is used in both cases in exactly the same way.

We know that the `ColumnFragment` fills the entire width of the screen, so `DisplayMetrics.widthPixels` is the same value without requiring a reference to a `FragmentLayout`.

The x value of the DragEvent is compared to the screen width and if it is in the outer 15% of either side of the screen, either dragLeft or dragRight are called. Of course, this would result in near instantaneous scrolling to the final ColumnFragment, so the page change time is stored and used in the comparisons to only allow a page drag every 500ms. dragLeft and dragRight simply check that the movement is possible, and set the ViewPagerposition.

At this point it is necessary to explain the structure of the ColumnFragment and the Viewscontained within it prior to any further discussion about drag shadows.

## ColumnFragment

The ColumnFragment manages displaying the information about a particular column, as well as the cards present in the column.

It also manages creating, deleting, and moving cards.

### ColumnFragment.java

```
package com(tpb.projects.project;

import android.app.Dialog;
import android.content.ClipData;
import android.content.ClipboardManager;
import android.content.Context;
import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.util.Pair;
import android.support.v4.widget.NestedScrollView;
import android.support.v7.app.AlertDialog;
import android.support.v7.widget.CardView;
import android.support.v7.widget.LinearLayoutManager;
import android.text.format.DateUtils;
import android.view.DragEvent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.inputmethod.EditorInfo;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import com(tpb.animatingrecyclerview.AnimatingRecyclerView;
import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Editor;
import com(tpb.github.data.Loader;
import com(tpb.github.data.models.Card;
import com(tpb.github.data.models.Column;
import com(tpb.github.data.models.Issue;
import com(tpb.github.data.models.Repository;
import com(tpb.mdtext.views.MarkdownTextView;
import com(tpb.projects.R;
```

```
import com(tpb.projects.common.ViewSafeFragment;
import com(tpb.projects.editors.CardEditor;
import com(tpb.projects.editors.CommentEditor;
import com(tpb.projects.editors.IssueEditor;
import com(tpb.projects.flow.IntentHandler;
import com(tpb.projects.util.SettingsActivity;
import com(tpb.projects.util.UI;

import java.util.List;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;
import butterknife.Unbinder;

import static
com(tpb.projects.util.SettingsActivity.Preferences.CardAction.COPY;

/**
 * Created by theo on 19/12/16.
 */

public class ColumnFragment extends ViewSafeFragment {

 private Unbinder unbinder;

 Column mColumn;

 @BindView(R.id.column_card) CardView mCard;
 @BindView(R.id.column_name) EditText mName;
 @BindView(R.id.column_last_updated) TextView mLastUpdate;
 @BindView(R.id.column_card_count) TextView mCardCount;
 @BindView(R.id.column_scrollview) NestedScrollView mNestedScroller;
 @BindView(R.id.column_recycler) AnimatingRecyclerView mRecycler;

 ProjectActivity mParent;
 private ProjectActivity.NavigationDragListener mNavListener;
 private Editor mEditor;
 private Repository.AccessLevel mAccessLevel;

 private CardAdapter mAdapter;

 public static ColumnFragment getInstance(Column column,
ProjectActivity.NavigationDragListener navListener, Repository.AccessLevel
accessLevel) {
 final ColumnFragment cf = new ColumnFragment();
 cf.mColumn = column;
 cf.mNavListener = navListener;
 cf.mAccessLevel = accessLevel;
 return cf;
 }

 @Nullable
 @Override
 public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
 final View view = inflater.inflate(R.layout.fragment_column,
container, false);
 unbinder = ButterKnife.bind(this, view);
 }
}
```

```
 if(mColumn == null && savedInstanceState != null) {
 mColumn =
 savedInstanceState.getParcelable(getString(R.string.parcel_column));
 }
 mName.setText(mColumn.getName());

 mAdapter = new CardAdapter(this, mNavListener, mAccessLevel,
mParent.mRefresher);
 mAdapter.setColumn(mColumn.getId());
 mRecycler.setAdapter(mAdapter);
 mRecycler.setLayoutManager(new LinearLayoutManager(getContext()));
 mAreViewsValid = true;
 if(mAccessLevel == Repository.AccessLevel.ADMIN || mAccessLevel ==
Repository.AccessLevel.WRITE) {
 enableAccess(view);
 } else {
 disableAccess(view);
 }
 mName.clearFocus();

 displayLastUpdate();
 return view;
}

@Override
public void onActivityCreated(@Nullable Bundle savedInstanceState) {
 super.onActivityCreated(savedInstanceState);

 mEditor = Editor.getEditor(getContext());
 mName.setOnEditorActionListener((textView, i, keyEvent) -> {
 if(i == EditorInfo.IME_ACTION_DONE) {
 if(mName.getText().toString().isEmpty()) {
 Toast.makeText(getContext(),
R.string.error_no_column_title, Toast.LENGTH_SHORT)
 .show();
 mName.setText(mColumn.getName());
 } else {
 mEditor.updateColumnName(new
Editor.UpdateListener<Column>() {

 @Override
 public void updated(Column column) {
 if(mAreViewsValid) {
 mColumn.setName(mName.getText().toString());
 resetLastUpdate();
 }
 }
 }

 @Override
 public void updateError(APIHandler.APIError error) {
 Toast.makeText(getContext(),
R.string.error_title_change_failed,
 Toast.LENGTH_SHORT
).show();
 mName.setText(mColumn.getName());
 }
 }, mColumn.getId(), mName.getText().toString());
 }
 })
}
```

```
 return false;
 }
 return false;
});

}

void setAccessLevel(Repository.AccessLevel accessLevel) {
 mAccessLevel = accessLevel;
 if(mAccessLevel == Repository.AccessLevel.ADMIN || mAccessLevel ==
Repository.AccessLevel.WRITE) {
 enableAccess(getView());
 } else {
 disableAccess(getView());
 }
 mAdapter.setAccessLevel(accessLevel);
}

private void enableAccess(View view) {
 mRecycler.setOnDragListener(new CardDragListener(getContext(),
mNavListener));
 mCard.setTag(mColumn.getId());
 mCard.setOnLongClickListener(v -> {
 final ClipData data = ClipData.newPlainText("", "");
 final View.DragShadowBuilder shadowBuilder = new
View.DragShadowBuilder(v);
 if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
 v.startDragAndDrop(data, shadowBuilder, v, 0);
 } else {
 v.startDrag(data, shadowBuilder, v, 0);
 }
 // v.setVisibility(View.INVISIBLE);
 return true;
 });
 final ColumnDragListener listener = new ColumnDragListener((int)
mCard.getTag());
 mName.setOnDragListener(listener);
 mLastUpdate.setOnDragListener(listener);
 mCard.setOnDragListener(listener);
 ((NestedScrollView) view.findViewById(R.id.column_scrollview))
 .setOnScrollChangeListener(new
NestedScrollView.OnScrollChangeListener() {
 @Override
 public void onScrollChange(NestedScrollView v, int
scrollX, int scrollY, int oldScrollX, int oldScrollY) {
 if(scrollY - oldScrollY > 10) {
 mParent.hideFab();
 } else if(scrollY - oldScrollY < -10) {
 mParent.showFab();
 }
 }
 });
}

private void disableAccess(View view) {
 mName.setEnabled(false);
 view.findViewById(R.id.column_delete).setVisibility(View.GONE);
}
```

```
private void resetLastUpdate() {
 mColumn.setUpdatedAt(System.currentTimeMillis());
 displayLastUpdate();
}

private void displayLastUpdate() {
 mLastUpdate.setText(
 String.format(
 getContext().getString(R.string.text_last_updated),
 DateUtils.getRelativeTimeSpanString(mColumn.getUpdatedAt())
)
);
 mCardCount.setText(Integer.toString(mAdapter.getItemCount()));
}

@OnClick(R.id.column_delete)
void deleteColumn() {
 mParent.deleteColumn(mColumn);
}

void loadIssue(Loader.ItemLoader<Issue> loader, int issueId) {
 mParent.loadIssue(loader, issueId, mColumn);
}

private void addCard(Card card) {
 mAdapter.addCard(card);
 resetLastUpdate();
}

void removeCard(Card card) {
 mAdapter.removeCard(card);
 resetLastUpdate();
}

void recreateCard(Card card) {
 mParent.mRefresher.setRefreshing(true);
 mEditor.createCard(new Editor.CreationListener<Pair<Integer, Card>>()
{
 @Override
 public void created(Pair<Integer, Card> integerCardPair) {
 addCard(card);
 mParent.mRefresher.setRefreshing(false);
 }

 @Override
 public void creationError(APIHandler.APIError error) {
 Toast.makeText(getContext(), error.resId,
 Toast.LENGTH_SHORT).show();
 mParent.mRefresher.setRefreshing(false);
 }
}, mColumn.getId(), card.getNote());
}

boolean attemptMoveTo(int cardId) {
 final int index = mAdapter.indexOf(cardId);
 if(index == -1) return false;
```

```
 UI.flashViewBackground(
 mRecycler.findViewHolderForAdapterPosition(index).itemView,
 getResources().getColor(R.color.md_grey_800),
 getResources().getColor(R.color.colorAccent)
);

 //Initial height is the top cardview
 int height = ((LinearLayout)
mNestedScroller.getChildAt(0)).getChildAt(0).getHeight();
 for(int i = 0; i < mRecycler.getChildCount() && i < index; i++) {
 height += mRecycler.getChildAt(i).getHeight();
 }
 mNestedScroller.scrollTo(0, height);
 return true;
 }

 List<Card> getCards() {
 return mAdapter.getCards();
 }

 @Override
 public void onActivityResult(int requestCode, int resultCode, Intent data)
{
 super.onActivityResult(requestCode, resultCode, data);
 if(resultCode != IssueEditor.RESULT_OK) return;
 String[] assignees = null;
 String[] labels = null;
 if(data.hasExtra(getString(R.string.intent_issue_assignees))) {
 assignees =
data.getStringArrayExtra(getString(R.string.intent_issue_assignees));
 }
 if(data.hasExtra(getString(R.string.intent_issue_labels))) {
 labels =
data.getStringArrayExtra(getString(R.string.intent_issue_labels));
 }
 switch(requestCode) {
 case IssueEditor.REQUEST_CODE_EDIT_ISSUE:
 final Card card =
data.getParcelableExtra(getString(R.string.parcel_card));
 final Issue edited =
data.getParcelableExtra(getString(R.string.parcel_issue));
 editIssue(card, edited, assignees, labels);
 break;
 case IssueEditor.REQUEST_CODE_ISSUE_FROM_CARD:
 final Card oldCard =
data.getParcelableExtra(getString(R.string.parcel_card));
 final Issue issue =
data.getParcelableExtra(getString(R.string.parcel_issue));
 mEditor.createIssue(
 new Editor.CreationListener<Issue>() {
 @Override
 public void created(Issue issue) {
 convertCardToIssue(oldCard, issue);
 }
 }

 @Override
 public void creationError(APIHandler.APIError
error) {

```

```
 }
 },
 mParent.mProject.getRepoPath(), issue.getTitle(),
issue.getBody(), assignees, labels
);
break;
}
}

void openMenu(View view, Card card) {
//We use the non AppCompat popup as the AppCompat version has a bug
which scrolls the RecyclerView up
final android.widget.PopupMenu popup = new
android.widget.PopupMenu(getContext(), view);
popup.setOnMenuItemClickListener(menuItem -> {
switch(menuItem.getItemId()) {
case R.id.menu_edit_note:
final Intent i = new Intent(getContext(),
CardEditor.class);
i.putExtra(getString(R.string.parcel_card), card);
UI.setViewPositionForIntent(i, view);
getActivity().startActivityForResult(i,
CardEditor.REQUEST_CODE_EDIT_CARD);
break;
case R.id.menu_delete_note:
mParent.deleteCard(card, true);
break;
case R.id.menu_copy_card_note:
copyToClipboard(card.getNote());
break;
case R.id.menu_copy_card_url:

copyToClipboard(String.format(mParent.getString(R.string.text_card_url),
mParent.mProject.getRepoPath(),
mParent.mProject.getNumber(),
mCard.getId()
));
break;
case R.id.menu_copy_issue_url:

copyToClipboard(String.format(mParent.getString(R.string.text_issue_url),
mParent.mProject.getRepoPath(),
card.getIssue().getNumber()
));
break;
case R.id.menu_fullscreen:
showFullscreen(card);
break;
case R.id.menu_convert_to_issue:
final Intent intent = new Intent(getContext(),
IssueEditor.class);
intent.putExtra(getString(R.string.intent_repo),
mParent.mProject.getRepoPath()
);
intent.putExtra(getString(R.string.parcel_card), card);
UI.setViewPositionForIntent(intent, view);
getActivity().startActivityForResult(intent,
IssueEditor.REQUEST_CODE_ISSUE_FROM_CARD
);
}
```

```
 break;
 case R.id.menu_edit_issue:
 showIssueEditor(view, card);
 break;
 case R.id.menu_delete_issue_card:
 if(!card.getIssue().isClosed()) {
 final AlertDialog.Builder builder = new
AlertDialog.Builder(getContext());
 builder.setTitle(R.string.title_close_issue);

builder.setMessage(R.string.text_close_issue_on_delete);
 builder.setPositiveButton(R.string.action_yes,
(dialogInterface, j) -> {
 mEditor.closeIssue(null,
mParent.mProject.getRepoPath(),
 card.getIssue().getNumber()
);
 mParent.deleteCard(card, false);
 });
 builder.setNeutralButton(R.string.action_cancel,
null);
 builder.setNegativeButton(R.string.action_no,
 (dialogInterface, j) ->
mParent.deleteCard(card, false)
);
 final Dialog deleteDialog = builder.create();
 deleteDialog.getWindow()
 .getAttributes().windowAnimations =
R.style.DialogAnimation;
 deleteDialog.show();
 } else {
 mParent.deleteCard(card, false);
 }
 break;
 case 1:
 toggleIssueState(card);
 break;
 }

 return true;
});
if(card.hasIssue()) {
 popup.inflate(R.menu.menu_card_issue);
 popup.getMenu().add(0, 1, 0, card.getIssue()
 .isClosed() ?
R.string.menu_reopen_issue : R.string.menu_close_issue);
} else {
 popup.inflate(R.menu.menu_card);
}

popup.show();
}

void newCard(Card card) {
 mParent.mRefresher.setRefreshing(true);
 mEditor.createCard(new Editor.CreationListener<Pair<Integer, Card>>()
{
 @Override
```

```
 public void created(Pair<Integer, Card> pair) {
 addCard(pair.second);
 mParent.mRefresher.setRefreshing(false);
 }

 @Override
 public void creationError(APIHandler.APIError error) {
 mParent.mRefresher.setRefreshing(false);
 Toast.makeText(getApplicationContext(), error.resId,
Toast.LENGTH_SHORT).show();
 }
}, mColumn.getId(), card.getNote());
}

void editCard(Card card) {
 mParent.mRefresher.setRefreshing(true);
 mEditor.updateCard(new Editor.UpdateListener<Card>() {

 @Override
 public void updated(Card card) {
 mAdapter.updateCard(card);
 resetLastUpdate();
 mParent.mRefresher.setRefreshing(false);
 }

 @Override
 public void updateError(APIHandler.APIError error) {
 Toast.makeText(getApplicationContext(), error.resId,
Toast.LENGTH_SHORT).show();
 mParent.mRefresher.setRefreshing(false);
 }
}, card.getId(), card.getNote()));

}

private void toggleIssueState(Card card) {
 final Editor.UpdateListener<Issue> listener = new
Editor.UpdateListener<Issue>() {

 @Override
 public void updated(Issue issue) {
 card.setFromIssue(issue);
 mAdapter.updateCard(card);
 }

 @Override
 public void updateError(APIHandler.APIError error) {
 Toast.makeText(getApplicationContext(), error.resId,
Toast.LENGTH_SHORT).show();
 mParent.mRefresher.setRefreshing(false);
 }
};

 final AlertDialog.Builder builder = new
AlertDialog.Builder(getApplicationContext());
 builder.setTitle(R.string.title_state_change_comment);
 builder.setPositiveButton(R.string.action_ok, (dialog, which) -> {
 if(card.getIssue().isClosed()) {
 mEditor.openIssue(listener, card.getIssue().getRepoFullName(),

```

```

 card.getIssue().getNumber()
);
} else {
 mEditor.closeIssue(listener,
card.getIssue().getRepoFullName(),
 card.getIssue().getNumber()
);
}
final Intent i = new Intent(getContext(), CommentEditor.class);
i.putExtra(getString(R.string.parcel_issue), card.getIssue());
getActivity().startActivityForResult(i,
CommentEditor.REQUEST_CODE_COMMENT_FOR_STATE);

});
builder.setNegativeButton(R.string.action_no, (dialog, which) -> {
 if(card.getIssue().isClosed()) {
 mEditor.openIssue(listener, card.getIssue().getRepoFullName(),
 card.getIssue().getNumber()
);
} else {
 mEditor.closeIssue(listener,
card.getIssue().getRepoFullName(),
 card.getIssue().getNumber()
);
}
});
builder.setNeutralButton(R.string.action_cancel, null);
builder.create().show();
}

private void showIssueEditor(View view, Card card) {
 final Intent i = new Intent(getContext(), IssueEditor.class);
 i.putExtra(getString(R.string.intent_repo),
mParent.mProject.getRepoPath());
 i.putExtra(getString(R.string.parcel_card), card);
 i.putExtra(getString(R.string.parcel_issue), card.getIssue());
 if(view instanceof MarkdownTextView) {
 UI.setClickPositionForIntent(getContext(), i,
 ((MarkdownTextView) view).getLastClickPosition()
);
 } else {
 UI.setViewPositionForIntent(i, view);
 }
 getActivity().startActivityForResult(i,
IssueEditor.REQUEST_CODE_EDIT_ISSUE);
}

public void editIssue(Card card, Issue issue, @Nullable String[]
assignees, @Nullable String[] labels) {
 mParent.mRefresher.setRefreshing(true);
 mEditor.updateIssue(new Editor.UpdateListener<Issue>() {

 @Override
 public void updated(Issue issue) {
 card.setFromIssue(issue);
 mAdapter.updateCard(card);
 mParent.mRefresher.setRefreshing(false);
 resetLastUpdate();
 }
 }
}

```

```
 @Override
 public void updateError(APIHandler.APIError error) {
 Toast.makeText(getApplicationContext(), error.resId,
Toast.LENGTH_SHORT).show();
 mParent.mRefresher.setRefreshing(false);
 }
}, card.getIssue().getRepoFullName(), issue, assignees, labels);
}

private void convertCardToIssue(Card oldCard, Issue issue) {
 mEditor.deleteCard(new Editor.DeletionListener<Card>() {

 @Override
 public void deleted(Card card) {
 createIssueCard(issue, oldCard.getId());
 resetLastUpdate();
 }

 @Override
 public void deletionError(APIHandler.APIError error) {
 mParent.mRefresher.setRefreshing(false);
 Toast.makeText(getApplicationContext(), error.resId,
Toast.LENGTH_SHORT).show();
 }
 }, oldCard);
}

void createIssueCard(Issue issue) {
 createIssueCard(issue, -1);
}

private void createIssueCard(Issue issue, int oldCardId) {
 mParent.mRefresher.setRefreshing(true);
 mEditor.createCard(new Editor.CreationListener<Pair<Integer, Card>>()
{
 @Override
 public void created(Pair<Integer, Card> val) {
 mParent.mRefresher.setRefreshing(false);
 if(oldCardId == -1) {
 mAdapter.addCard(val.second);
 } else {
 mAdapter.updateCard(val.second, oldCardId);
 }
 resetLastUpdate();
 }

 @Override
 public void creationError(APIHandler.APIError error) {
 Toast.makeText(getApplicationContext(), error.resId,
Toast.LENGTH_SHORT).show();
 mParent.mRefresher.setRefreshing(false);
 }
}, mColumn.getId(), issue.getId());
}

private void copyToClipboard(String text) {
 final ClipboardManager cm = (ClipboardManager) getContext()
```

```
 .getSystemService(Context.CLIPBOARD_SERVICE);
 cm.setPrimaryClip(ClipData.newPlainText("Card", text));
 Toast.makeText(mParent, getString(R.string.text_copied_to_board),
 LENGTH_SHORT)
 .show();
 }

 private void showFullscreen(Card card) {
 IntentHandler.showFullScreen(getApplicationContext(), card.getNote(),
 mParent.mProject.getRepoPath(),
 getFragmentManager()
);
}

void cardClick(View view, Card card) {
 final SettingsActivity.Preferences.CardAction action;
 if(mAccessLevel == Repository.AccessLevel.NONE || mAccessLevel ==
Repository.AccessLevel.READ) {
 action = COPY;
 } else {
 action =
SettingsActivity.Preferences.getPreferences(getApplicationContext()).getCardAction();
 }
 switch(action) {
 case EDIT:
 if(card.hasIssue()) {
 showIssueEditor(view, card);
 } else {
 final Intent i = new Intent(getApplicationContext(),
CardEditor.class);
 i.putExtra(getString(R.string.parcel_card), card);
 if(view instanceof MarkdownTextView) {
 UI.setClickPositionForIntent(getApplicationContext(), i,
 ((MarkdownTextView)
view).getLastClickPosition()
);
 } else {
 UI.setViewPositionForIntent(i, view);
 }
 getActivity().startActivityForResult(i,
CardEditor.REQUEST_CODE_EDIT_CARD);
 }
 break;
 case FULLSCREEN:
 showFullscreen(card);
 break;
 case COPY:
 copyToClipboard(card.getNote());
 break;
 }
}

void notifyScroll() {
 mAdapter.notifyBottomReached();
}

void scrollUp() {
 final LinearLayoutManager lm = (LinearLayoutManager)
mRecycler.getLayoutManager();
```

```
 final int pos = lm.findFirstVisibleItemPosition();
 final int height = mRecycler.getChildAt(pos).getHeight();
 mNestedScroller.smoothScrollBy(0, -height);
}

void scrollDown() {
 final LinearLayoutManager lm = (LinearLayoutManager)
mRecycler.getLayoutManager();
 final int pos = lm.findLastVisibleItemPosition();
 final int height = mRecycler.getChildAt(pos).getHeight();
 mNestedScroller.smoothScrollBy(0, height);
}

@Override
public void onAttach(Context context) {
 super.onAttach(context);
 try {
 mParent = (ProjectActivity) context;
 } catch(ClassCastException cce) {
 throw new IllegalArgumentException("Parent of ColumnFragment must
be ProjectActivity");
 }
}

@Override
public void onDestroyView() {
 super.onDestroyView();
 unbinder.unbind();
 mAreViewsValid = false;
}

@Override
public void onSaveInstanceState(Bundle outState) {
 super.onSaveInstanceState(outState);
 outState.putParcelable(getString(R.string.parcel_column), mColumn);
}

private class ColumnDragListener implements View.OnDragListener {
 private int mTargetTag;

 ColumnDragListener(int targetTag) {
 mTargetTag = targetTag;
 }

 @Override
 public boolean onDrag(View view, DragEvent event) {
 if(event.getAction() == DragEvent.ACTION_DROP) {
 final View sourceView = (View) event.getLocalState();
 view.setVisibility(View.VISIBLE);

 final int sourceTag = (int) sourceView.getTag();
 if(sourceTag != mTargetTag && sourceView.getId() ==
R.id.column_card) {
 mParent.moveColumn(sourceTag, mTargetTag, event.getX() <
view.getWidth() / 2);
 }
 }
 return true;
 }
}
```

```
 }
}
```

enableAccess is called once the access level is determined.

If the access level is sufficient to allow the authenticated user to edit the project, listeners for drag and drop actions are added.

addCard and removeCard are called from the ProjectActivity and add or remove cards from the adapter before calling resetLastUpdate to display the time that a card was last changed.

recreateCard is called if the user accidentally deletes a card. When a card is deleted, a Snackbar is shown with an undo button, allowing the card to be recreated.

attemptMoveTo is called in order to scroll the RecyclerView to a particular position when the ProjectActivity is launched with a card id.

The adapter is checked to determine whether it contains the card, and if so the View is flashed.

As the RecyclerView is within a NestedScrollView it cannot scroll by itself. Instead, the NestedScrollView must be scrolled.

In order to find the scroll position, the sum of the ViewHolders below the position of the launched card is found, and the NestedScrollView is moved to this position. In onActivityResult, issue cards are managed. In the case of an existing issue card, editIssue is called. Otherwise, an Issue is created, and convertCardToIssue is called.

editCard calls Editor.updateCard to update the card model before notifying the adapter and calling resetLastUpdate.

newCard calls createdCard with the id of the current column and the note text for the card, calling addCard with the resulting card.

toggleIssueState is called when the options menu item to change the state of the issue attached to a particular card.

convertCardToIssue deletes the card and then calls createIssueCard to create the new Issue from the card.

## CardAdapter

The CardAdapter manages the loading and binding of cards as well as adding the OnDragListeners to them for moving cards between columns.

### CardAdapter.java

```
package com(tpb.projects.project;

import android.content.ClipData;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.os.HandlerThread;
import android.support.v4.util.Pair;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.widget.CardView;
import android.support.v7.widget.RecyclerView;
import android.text.SpannableString;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.ProgressBar;

import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Editor;
import com(tpb.github.data.Loader;
import com(tpb.github.data.models.Card;
import com(tpb.github.data.models.Issue;
import com(tpb.github.data.models.Repository;
import com(tpb.mdtext.Markdown;
import com(tpb.mdtext.imagegetter.HttpImageGetter;
import com(tpb.mdtext.views.MarkdownTextView;
import com(tpb.projects.R;
import com(tpb.projects.common.NetworkImageView;
import com(tpb.projects.flow.IntentHandler;
import com(tpb.projects.markdown.Formatter;
import com(tpb.projects.util.Analytics;

import java.util.ArrayList;
import java.util.List;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;

import static com(tpb.projects.flow.ProjectsApplication.mAnalytics;

/**
 * Created by theo on 20/12/16.
 */

class CardAdapter extends RecyclerView.Adapter<CardAdapter.CardHolder>
implements Loader.ListLoader<Card> {
 private static final String TAG = CardAdapter.class.getSimpleName();
```

```
private final ArrayList<Pair<Card, SpannableString>> mCards = new
ArrayList<>();

private final ColumnFragment mParent;
private int mColumn;
private final Editor mEditor;
private static final HandlerThread parseThread = new
HandlerThread("card_parser");

private int mPage = 1;
private boolean mIsLoading = false;
private boolean mMaxPageReached = false;

private SwipeRefreshLayout mRefresher;
private Loader mLoader;

static {
 parseThread.start();
}

private static final Handler mParseHandler = new
Handler(parseThread.getLooper());
private Repository.AccessLevel mAccessLevel;
private final ProjectActivity.NavigationDragListener mNavListener;

CardAdapter(ColumnFragment parent,
 ProjectActivity.NavigationDragListener navListener,
 Repository.AccessLevel accessLevel,
 SwipeRefreshLayout refresher) {
 mParent = parent;
 mEditor = Editor.getEditor(mParent.getContext());
 mLoader = Loader.getLoader(parent.getContext());
 mAccessLevel = accessLevel;
 mNavListener = navListener;
 mRefresher = refresher;
 mRefresher.setRefreshing(true);
}

void setColumn(int columnId) {
 mColumn = columnId;
 mCards.clear();
 notifyDataSetChanged();
 loadCards(true);
}

public void notifyBottomReached() {
 if(!mIsLoading && !mMaxPageReached) {
 mPage++;
 loadCards(false);
 }
}

private void loadCards(boolean resetPage) {
 mIsLoading = true;
 mRefresher.setRefreshing(true);
 if(resetPage) {
 mPage = 1;
 mMaxPageReached = false;
 }
}
```

```
 final int oldSize = mCards.size();
 mCards.clear();
 notifyItemRangeRemoved(0, oldSize);
 }
 mLoader.loadCards(this, mParent.mColumn.getId(), mPage);
}

@Override
public void listLoadComplete(List<Card> cards) {
 if(!mParent.isAdded()) return;
 mRefresher.setRefreshing(false);
 mIsLoading = false;
 if(cards.size() > 0) {
 int oldLength = mCards.size();
 if(mPage == 1) {
 mParent.mParent.notifyFragmentLoaded();
 }
 for(Card c : cards) {
 mCards.add(Pair.create(c, null));
 }
 mParent.mCardCount.setText(String.valueOf(mCards.size()));
 notifyItemRangeInserted(oldLength, mCards.size());
 } else {
 mMaxPageReached = true;
 }
}

@Override
public void listLoadError(APIHandler.APIError error) {
}

void setAccessLevel(Repository.AccessLevel accessLevel) {
 mAccessLevel = accessLevel;
 notifyDataSetChanged();
}

void addCard(Card card) {
 mCards.add(0, Pair.create(card, null));
 notifyItemInserted(0);
}

void addCardFromDrag(Card card) {
 mCards.add(Pair.create(card, null));
 notifyItemInserted(mCards.size());
 mEditor.moveCard(null, mColumn, card.getId(), -1);
}

void addCardFromDrag(int pos, Card card) {
 mCards.add(pos, Pair.create(card, null));
 notifyItemInserted(pos);
 final int id = pos == 0 ? -1 : mCards.get(pos - 1).first.getId();
 mEditor.moveCard(null, mParent.mColumn.getId(), card.getId(), id);
}

void updateCard(Card card) {
 final int index = indexOf(card.getId());
 if(index != -1) {
 mCards.set(index, Pair.create(card, null));
 }
}
```

```

 notifyItemChanged(index);
 }

}

void updateCard(Card card, int oldId) {
 final int index = indexOf(oldId);
 if(index != -1) {
 mCards.set(index, Pair.create(card, null));
 notifyItemChanged(index);
 }
}

void moveCardFromDrag(int oldPos, int newPos) {
 final Pair<Card, SpannableString> card = mCards.get(oldPos);
 mCards.remove(oldPos);
 mCards.add(newPos, card);
 notifyItemMoved(oldPos, newPos);
 final int id = newPos == 0 ? -1 : mCards.get(newPos -
1).first.getId();
 mEditor.moveCard(null, mParent.mColumn.getId(), card.first.getId(),
id);
}

void removeCard(Card card) {
 final int index = indexOf(card.getId());
 if(index != -1) {
 mCards.remove(index);
 notifyItemRemoved(index);
 }
 //API call is handled in adapter to which card is added
}

int indexOf(int cardId) {
 for(int i = 0; i < mCards.size(); i++) {
 if(mCards.get(i).first.getId() == cardId) return i;
 }
 return -1;
}

List<Card> getCards() {
 final List<Card> cards = new ArrayList<>();
 for(Pair<Card, SpannableString> p : mCards) cards.add(p.first);
 return cards;
}

private void openMenu(View view, int position) {
 mParent.openMenu(view, mCards.get(position).first);
}

private void cardClick(CardHolder holder) {
 mParent.cardClick(holder.mText,
mCards.get(holder.getAdapterPosition()).first);
}

@Override
public CardHolder onCreateViewHolder(ViewGroup parent, int viewType) {
 return new CardHolder(LayoutInflater.from(parent.getContext())
.inflate(R.layout.viewholder_card,
parent, false));
}

```

```
}

@Override
public void onBindViewHolder(CardHolder holder, int position) {
 final int pos = holder.getAdapterPosition();
 final Card card = mCards.get(pos).first;
 if(mAccessLevel == Repository.AccessLevel.ADMIN || mAccessLevel ==
Repository.AccessLevel.WRITE) {
 holder.mCardView.setTag(card.getId());
 holder.mCardView.setOnLongClickListener(view -> {
 final ClipData data = ClipData.newPlainText("", "");
 final View.DragShadowBuilder shadowBuilder = new
View.DragShadowBuilder(view);
 if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
 view.startDragAndDrop(data, shadowBuilder, view, 0);
 } else {
 view.startDrag(data, shadowBuilder, view, 0);
 }
 view.setVisibility(View.INVISIBLE);
 return true;
 });
 holder.mCardView.setOnDragListener(
 new CardDragListener(mParent.getContext(), mNavListener)
);
 } else {
 holder.mMenuButton.setVisibility(View.GONE);
 }

 if(card.requiresLoadingFromIssue()) {
 holder.mSpinner.setVisibility(View.VISIBLE);
 mParent.loadIssue(new Loader.ItemLoader<Issue>() {
 int loadCount = 0;

 @Override
 public void loadComplete(Issue data) {
 if(mParent.isAdded() && !mParent.isRemoving()) {
 mCards.get(pos).first.setFromIssue(data);
 notifyItemChanged(pos);
 }
 final Bundle bundle = new Bundle();
 bundle.putString(Analytics.KEY_LOAD_STATUS,
Analytics.VALUE_SUCCESS);
 mAnalytics.logEvent(Analytics.TAG_ISSUE_LOADED, bundle);
 }

 @Override
 public void loadError(APIHandler.APIError error) {
 if(error != APIHandler.APIError.NO_CONNECTION) {
 final Bundle bundle = new Bundle();
 bundle.putString(Analytics.KEY_LOAD_STATUS,
Analytics.VALUE_FAILURE);
 mAnalytics.logEvent(Analytics.TAG_ISSUE_LOADED,
bundle);
 loadCount++;
 if(loadCount < 5) {
 mParent.loadIssue(this, card.getIssueId());
 }
 }
 }
 });
 }
}
```

```
 }
 }, card.getIssueId());
} else if(card.hasIssue()) {
 bindIssueCard(holder, pos);
} else {
 bindStandardCard(holder, pos);
}

}

private void bindStandardCard(CardHolder holder, int pos) {
 holder.mTitleLayout.setVisibility(View.GONE);
 if(mCards.get(pos).second == null) {
 final Card card = mCards.get(pos).first;
 holder.mText.setMarkdown(
 Markdown.formatMD(
 card.getNote(),
 mParent.mParent.mProject.getRepoPath()
),
 new HttpImageGetter(holder.mText),
 text -> mCards.set(pos, Pair.create(card, text))
);
 } else {
 holder.mText.setText(mCards.get(pos).second);
 }
 IntentHandler.addOnClickHandler(mParent.getActivity(), holder.mText);
}

private void bindIssueCard(CardHolder holder, int pos) {
 holder.mIssueIcon.setVisibility(View.VISIBLE);
 holder.mUserAvatar.setVisibility(View.VISIBLE);
 final Card card = mCards.get(pos).first;
 holder.mIssueIcon.setImageResource(
 card.getIssue().isClosed() ? R.drawable.ic_state_closed :
R.drawable.ic_state_open);

 holder.mUserAvatar.setImageUrl(card.getIssue().getOpenedBy().getAvatarUrl());
 IntentHandler.addOnClickHandler(mParent.getActivity(),
holder.mUserAvatar,
 card.getIssue().getOpenedBy().getLogin()
);
 IntentHandler.addOnClickHandler(mParent.getActivity(), holder.mText,
holder.mUserAvatar,
 holder.mCardView, card.getIssue()
);
 IntentHandler.addOnClickHandler(mParent.getActivity(),
holder.mIssueIcon, card.getIssue()
);
 IntentHandler.addOnClickHandler(mParent.getActivity(), holder.mTitle,
card.getIssue());
 holder.mTitleLayout.setVisibility(View.VISIBLE);

 holder.mTitle.setMarkdown(Formatter.bold(card.getIssue().getTitle()));
 if(mCards.get(pos).second == null) {
 holder.mText.setMarkdown(
 Formatter.buildIssueSpan(
 holder.itemView.getContext(),
 card.getIssue(),
 false,
```

```

 true,
 true,
 true,
 true
).toString(),
 new HttpImageGetter(holder.mText),
 text -> mCards.set(pos, Pair.create(card, text))
);
} else {
 holder.mText.setText(mCards.get(pos).second);
}
holder.mSpinner.setVisibility(View.GONE);
}

@Override
public int getItemCount() {
 return mCards.size();
}

class CardHolder extends RecyclerView.ViewHolder {

 @BindView(R.id.card_markdown) MarkdownTextView mText;
 @BindView(R.id.card_title) MarkdownTextView mTitle;
 @BindView(R.id.card_issue_progress) ProgressBar mSpinner;
 @BindView(R.id.viewholder_card) CardView mCardView;
 @BindView(R.id.card_menu_button) View mMenuButton;
 @BindView(R.id.card_drawable_wrapper) View mTitleLayout;
 @BindView(R.id.card_issue_drawable) ImageView mIssueIcon;
 @BindView(R.id.card_user_avatar) NetworkImageView mUserAvatar;

 @OnClick(R.id.card_menu_button)
 void onMenuClick(View v) {
 openMenu(v, getAdapterPosition());
 }

 CardHolder(View view) {
 super(view);
 ButterKnife.bind(this, view);
 view.setOnClickListener(v -> cardClick(this));
 mText.setOnClickListener(v -> cardClick(this));
 //mText.setParseHandler(mParseHandler);
 }
}
}

```

`listLoadComplete` performs the usual checks for the page number, and notifies the `ColumnFragment` parent that the data has been loaded.

`addcard` is used for inserting a new card at the start of the adapter.

The two `addCardFromDrag` methods are used to add a card, either to the start of the adapter or to a specific position in the adapter.

The `addCardFromDrag` method which takes a position parameter calls `Editor.movecard` with the id of the card being dropped onto, as this is required for inserting the card at a non-zero position.

`moveCardFromDrag` manages moving a Card from one position to the other when it is dragged. It moves the Pair of the Card and its SpannableString cache to the new position before calling `Editor.moveCard`.

In `onBindviewHolder` the `OnLongClickListener` is added to the itemView to create the Viewshadow and begin a drag and drop which will call the `CardDragListener` which is attached to the itemView with a reference to the `NavigationDragListener` from the parent `ColumnFragment`

### Loading issue cards

Card objects which are linked to an issue do not contain the issue, only its id. As such the `CardHolder` layout contains a `ProgressBar` spinner which is shown while the request is made to load the `Issue`.

The callback once the `Issue` has been loaded sets the card data at the position and calls `notifyItemChanged` which will call `bindIssueCard` as the `Issue` now exists.

### CardDragListener and the ACTION\_DRAG\_ENTERED event

Prelude:

- The `ProjectActivity` contains a single `NavigationDragListener` which is attached to:
  - The column header cards
  - The `RecyclerView` in each `ColumnFragment`
  - Each card in each `ColumnFragment`
  - The background layout of each `ColumnFragment`
- The `NavigationDragListener` handles:
  - Dragging the column header cards to the screen edge to trigger page scrolls
  - Dragging the `CardHolder` cards to the screen edge to trigger page scrolls
  - Dragging the `CardHolder` cards up and down across other cards to trigger `RecyclerViewscrolls`
- Each of the `CardDragListeners` feed their events to the `NavigationDragListener`

### ACTION\_DRAG\_ENTERED

As was stated before this event is triggered when the view shadow enters the bounds of another view with an `OnDragListener`.

The `NavigationDragListener` does the following:

- Checks that the view being entered has the `viewholder_card` id
- Casts the parent `RecyclerView` of the view being entered

- Casts the CardAdapter of the RecyclerView

It then finds the hit rectangle of the parent of the parent of the RecyclerView, which is a NestedScrollView.

Next, it initialises two indices, first and last.

It iterates through each item in the adapter, using getLocalVisibleRect with the extracted hit rectangle to check if the CardHolder is one screen.

If the CardHolder is not in the hit rectangle, and the first position has been found last is set.

The index of the target view is then found by searching the CardAdapter for the tag.

The relative position in the target View is then found from the event y position which is relative to the view and the View y position.

If the target position in the adapter is the first or last visible item, a scroll may happen.

In either case the actual position on screen is found, and the actual position plus the relative position is checked to see if it is in the outer 10% of the screen, in which case the dragUp or dragDown methods are called.

## **CardDragListener**

The CardDragListener manages the drop events when cards are shadowed and dragged around.

### **CardDragListener.java**

```
package com(tpb.projects.project;

import android.content.Context;
import android.graphics.drawable.Drawable;
import android.support.v7.widget.RecyclerView;
import android.view.DragEvent;
import android.view.View;

import com(tpb.github.data.models.Card;
import com(tpb.projects.R;
import com(tpb.projects.util.Logger;

/**
 * Created by theo on 22/12/16.
 */

class CardDragListener implements View.OnDragListener {
 private static final String TAG = CardDragListener.class.getSimpleName();

 private boolean isDropped = false;
 private Drawable selectedBG;
 private final int accent;
 private final View.OnDragListener mParent;

 CardDragListener(Context context, View.OnDragListener parent) {
 accent = context.getResources().getColor(R.color.colorAccent);
```

```

 mParent = parent;
 }

@Override
public boolean onDrag(View view, DragEvent event) {
 if(mParent != null) {
 mParent.onDrag(view, event);
 }
 final int action = event.getAction();
 final View sourceView = (View) event.getLocalState();
 if(sourceView.getId() == R.id.column_card || view.getTag() ==
sourceView.getTag()) {
 return true;
 }

 switch(action) {
 case DragEvent.ACTION_DROP:
 isDropped = true;
 int sourcePosition, targetPosition;

 view.setVisibility(View.VISIBLE);

 final RecyclerView target;
 final RecyclerView source = (RecyclerView)
sourceView.getParent();

 final CardAdapter sourceAdapter = (CardAdapter)
source.getAdapter();
 sourcePosition = sourceAdapter.indexOf((int)
sourceView.getTag());

 final Card card =
sourceAdapter.getCards().get(sourcePosition);
 if(view.getId() == R.id.viewholder_card) {
 target = (RecyclerView) view.getParent();
 } else {
 target = (RecyclerView) view;
 }
 final CardAdapter targetAdapter = (CardAdapter)
target.getAdapter();

 if(view.getId() == R.id.viewholder_card) {
 targetPosition = targetAdapter.indexOf((int)
view.getTag());
 Logger.i(TAG, "onDrag: Hovering over position " +
targetPosition);
 if(event.getY() < view.getHeight() / 2) {
 targetPosition = Math.max(0, targetPosition - 1);
 }
 if(source != target) {
 if(targetPosition >= 0) {
 targetAdapter.addCardFromDrag(targetPosition,
card);
 } else {
 targetAdapter.addCardFromDrag(card);
 }
 sourceAdapter.removeCard(card);
 } else if(sourcePosition != targetPosition) { //We are
moving a card

```

```

 sourceAdapter.moveCardFromDrag(sourcePosition,
targetPosition);
 }

} else if(view.getId() == R.id.column_recycler &&
((RecyclerView) view).getAdapter()

.getItemCount() == 0) {
 sourceAdapter.removeCard(card);
 targetAdapter.addCardFromDrag(card);
}
view.setBackground(selectedBG);

break;
case DragEvent.ACTION_DRAG_ENTERED:
// Log.i(TAG, "onDrag: Drag entered");
if(view.getId() == R.id.viewholder_card
|| (view.getId() == R.id.column_recycler &&
(RecyclerView) view)
.getAdapter().getItemCount() == 0)) {
selectedBG = view.getBackground();
view.setBackgroundColor(accent);
}
//This is when we have entered another view

break;
case DragEvent.ACTION_DRAG_EXITED:
Logger.i(TAG, "onDrag: Drag exited");
view.setBackground(selectedBG);
//This is when we have exited another view
break;
default:
break;
}

if(!isDropped) {
View vw = (View) event.getLocalState();
vw.setVisibility(View.VISIBLE);
}

return true;
}
}

```

It first passes any events to the NavigationDragListener.

Next, it checks that the view being dragged over is not a column header tag and is not the source of the shadow View.

### **ACTION\_DRAG\_ENTERED and ACTION\_DRAG\_EXITED**

When a view is entered, if it is a CardHolder or an empty RecyclerView, its background colour is saved and the view background is then set to the accent colour.

When the `View` is exited, the `View` background colour is reset to its `Drawable` value when it was first entered.

## **ACTION\_DROP**

When the shadowed `View` is dropped, the card is moved.

The source `RecyclerView` is cast from the parent `View` of the target.

The source `CardAdapter` is cast from the source `RecyclerView` adapter.

The position of the source is found, and the card is extracted from the source adapter.

If the `View` id is `viewholder_card` the target `RecyclerView` is the parent of the `View`.

Otherwise it is the `View` itself.

The target `CardAdapter` is then cast.

If the `View` is an empty `RecyclerView`, the movement is easy as the card is removed from the source adapter and `addCardFromDrag` is called on the target adapter.

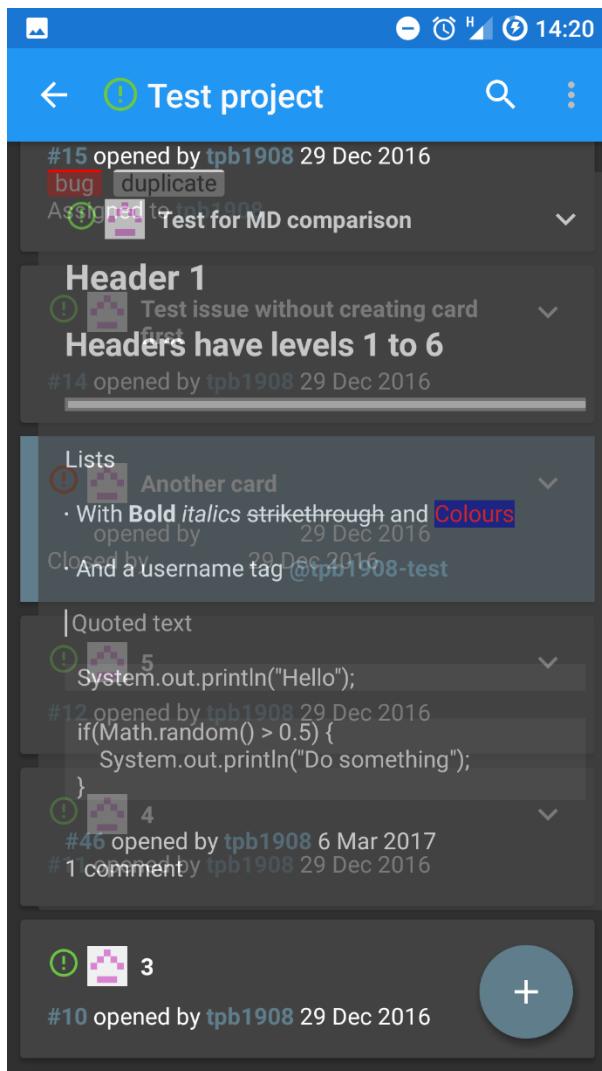
Otherwise, the logic is more complex as cards may need to be moved.

The target position is found from the target adapter.

If the event has occurred in the bottom half of the target `View`, the target position is the position below. Otherwise, the target position is the position of the target `View`.

If the source `RecyclerView` is not the target `RecyclerView`, `addCardFromDrag` is called on the target, and `removeCard` is called on the source. Otherwise, the positions are checked. If they are not the same `moveCardFromDrag` is called on the source adapter to move the `CardHolder` within its current adapter.

When in its drag state the `View` will appear as shown in the screenshot below:



This screenshot shows that the card being dragged is being held over the card highlighted blue, and will take its position if dropped.

## ProjectSearchAdapter

### ProjectSearchAdapter.java

```
package com(tpb.projects.project;

import android.content.Context;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.Filter;
import android.widget.TextView;

import com(tpb.github.data.models.Card;
import com(tpb.github.data.models.Label;
```

```
import com(tpb.projects.R;
import com(tpb.projects.util.search.ArrayFilter;
import com(tpb.projects.util.search.FuzzyStringSearcher;

import java.util.ArrayList;
import java.util.List;

import butterknife.ButterKnife;

/**
 * Created by theo on 02/02/17.
 */

class ProjectSearchAdapter extends ArrayAdapter<Card> {
 private static final String TAG =
ProjectSearchAdapter.class.getSimpleName();

 private final List<Card> data;
 private ArrayFilter<Card> mFilter;
 private final FuzzyStringSearcher mSearcher;

 public ProjectSearchAdapter(Context context, @NonNull List<Card> data) {
 super(context, R.layout.viewholder_search_suggestion, data);
 this.data = data;
 final ArrayList<String> strings = new ArrayList<>();
 String s;
 for(Card c : data) {
 if(c.hasIssue()) {
 s = "#" + c.getIssue().getNumber();
 for(Label l : c.getIssue().getLabels()) s += "\n" +
l.getName();
 strings.add(s + "\n" + c.getIssue().getBody());
 } else {
 strings.add(c.getNote());
 }
 }
 mSearcher = FuzzyStringSearcher.getInstance(strings);
 }

 @Override
 public long getItemId(int position) {
 return mFilter.getFiltered().get(position).getId();
 }

 @Nullable
 @Override
 public Card getItem(int position) {
 return mFilter.getFiltered().get(position);
 }

 @NonNull
 @Override
 public Filter getFilter() {
 if(mFilter == null) {
 mFilter = new ArrayFilter<>(this, mSearcher, data);
 }
 return mFilter;
 }
}
```

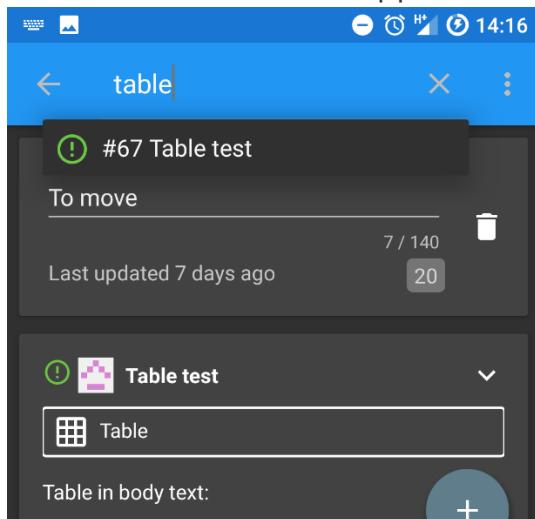
```
@NonNull
@Override
public View getView(int position, View convertView, @NonNull ViewGroup
parent) {
 if(convertView == null) {
 convertView = LayoutInflater.from(parent.getContext())
 .inflate(R.layout.viewholder_search_suggestion, parent,
 false
);
 }
 bindView(position, convertView);
 return convertView;
}

private void bindView(int pos, View view) {
 final int dp = data.indexOf(mFilter.getFiltered().get(pos));
 final String text;
 final Card c = data.get(dp);
 if(c.hasIssue()) {
 text = "#" + c.getIssue().getNumber() + " " + c.getIssue()

 .getTitle();
 } else {
 text = c.getNote();
 }
 final TextView tv = ButterKnife.findById(view, R.id.suggestion_text);
 tv.setText(text);
 if(c.hasIssue()) {
 tv.setCompoundDrawablesRelativeWithIntrinsicBounds(
 c.getIssue().isClosed() ?
 R.drawable.ic_state_closed :
 R.drawable.ic_state_open,
 0, 0, 0
);
 } else {
 tv.setCompoundDrawablesWithIntrinsicBounds(0, 0, 0, 0);
 }
}

@Override
public int getCount() {
 return mFilter.getFiltered().size();
}
```

When in use the results appear as shown below:



# Notifications

In order to display notifications, the application needs to register a service to run in the background and poll the GitHub API for notifications.

There are two ways which the service may be started.

First, it may be started on boot.

## NotificationServiceStartBroadcastReceiver

This is done by declaring the receiver with a BOOT\_COMPLETED action intent filter in the manifest.

```
<receiver
 android:name=".notifications.receivers.NotificationServiceStartBroadcastReceiver">
 <intent-filter>
 <action android:name="android.intent.action.BOOT_COMPLETED"/>
 <action android:name="android.intent.action.TIME_SET"/>
 </intent-filter>
</receiver>
```

The NotificationServiceStartBroadcastReceiver extends BroadcastReceiver and true to its name, starts the notification service when it receives a broadcast that the device has started.

### NotificationServiceStartBroadcastReceiver.java

```
package com(tpb.projects.notifications.receivers;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

/**
 * Created by theo on 04/04/17.
 */

public final class NotificationServiceStartBroadcastReceiver extends
BroadcastReceiver {

 @Override
 public void onReceive(Context context, Intent intent) {
 NotificationEventReceiver.setupAlarm(context);
 }
}
```

Registering the intent filter for the boot action means that the user will be able to begin receiving notifications immediately.

## NotificationEventReceiver

The NotificationEventReceiver extends WakefulBroadcastReceiver as the device must be awake to make the network request.

### NotificationEventReceiver.java

```
package com(tpb.projects.notifications.receivers;

import android.app.AlarmManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.support.annotation.IntRange;
import android.support.v4.content.WakefulBroadcastReceiver;

import com(tpb.projects.notifications.NotificationIntentService;
import com(tpb.projects.util.Logger;

import java.util.Date;

/*
 * Created by theo on 04/04/17.
 */

public class NotificationEventReceiver extends WakefulBroadcastReceiver {

 private static final String ACTION_START_NOTIFICATION_SERVICE =
"ACTION_START_NOTIFICATION_SERVICE";

 private static int NOTIFICATIONS_INTERVAL_IN_MINUTES = 1;

 public static void setupAlarm(Context context) {
 final AlarmManager alarmManager = (AlarmManager) context
 .getSystemService(Context.ALARM_SERVICE);
 final PendingIntent alarmIntent = getStartPendingIntent(context);
 alarmManager.setInexactRepeating(AlarmManager.RTC_WAKEUP,
 new Date().getTime(),
 NOTIFICATIONS_INTERVAL_IN_MINUTES * 60000,
 alarmIntent
);
 }

 private static PendingIntent getStartPendingIntent(Context context) {
 final Intent intent = new Intent(context,
NotificationEventReceiver.class);
 intent.setAction(ACTION_START_NOTIFICATION_SERVICE);
 return PendingIntent.getBroadcast(context, 0, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
 }

 public static void setUpdateInterval(@IntRange(from = 1, to = 60) int
minutes) {
 NOTIFICATIONS_INTERVAL_IN_MINUTES = minutes;
 }

 @Override
 public void onReceive(Context context, Intent intent) {
 final String action = intent.getAction();
 if(ACTION_START_NOTIFICATION_SERVICE.equals(action)) {
```

```

 Logger.i(getClass().getSimpleName(),
 "onReceive from alarm, starting notification service"
);
 // Start the service, keeping the device awake while it is
 launching.
 startWakefulService(context,
 NotificationIntentService.createIntentStartNotificationService(context)
);
 }
}

```

The `NotificationEventReceiver` contains the private string `ACTION_START_NOTIFICATION_SERVICE` which is used to ensure that the `Intent` received is from an `Intent` generated within the class.

In order to start the service, a repeating alarm is created.

First the system `AlarmManager` service is collected from the `Context`, and then a `PendingIntent` is created with `PendingIntent`.

The particularly perceptive may have already realised that a *Pending Intent* is not to be launched immediately and is instead used to trigger an action in the future. The `PendingIntent.getBroadcast` method returns a `PendingIntent` to trigger a broadcast with the `Intent` passed to it, which in this case has the `ACTION_START_NOTIFICATION_SERVICE` action, and should be sent to the `NotificationEventReceiver`.

The `PendingIntent.FLAG_UPDATE_CURRENT` flag indicates that if a `PendingIntent` with the same parameters already exists, it should be updated with the new `Intent` data.

The `PendingIntent` is then used to set up an alarm with the following parameters:

- `AlarmManager.RTC_WAKEUP` specifies that the alarm should be triggered according to the clock time, rather than time since boot
- `new Date().getTime()` is the current time in milliseconds, and should be used as the start time for the alarm
- `NOTIFICATIONS_INTERVAL_IN_MINUTES * 60000` is the duration between wakeups in milliseconds
- `alarmIntent` is the `PendingIntent` which was created with `getStartPendingIntent`

The `setInexactRepeating` has exactly the same parameter signature as `setRepeating`, but it does not ensure that the alarm will be triggered at exactly at the time specified.

This allows the system to bundle multiple alarms together, minimising the number of wakeups and any potential wakelocks while having negligible effect on the app as its content does not require highly accurate timing, such as an actual alarm clock app.

When the Intent is received in `onReceive` the action is checked, and if it is `ACTION_START_NOTIFICATION_SERVICE` a call is made to `startWakefulService` with the context and an Intent from the `NotificationIntentService` which adds the `ACTION_CHECK` action to an Intent.

## NotificationIntentService

The `NotificationIntentService` is where notifications are loaded and device notifications are displayed.

The class extends `IntentReceiver` and implements its constructor by passing `BuildConfig.APPLICATION_ID` as the service name. This ensure that only one instance of the service is ever running.

Intents are received through `onHandleIntent`.

These intents are of two types, the first are sent from the `NotificationEventReceiver` and have the action `ACTION_CHECK`, to check notifications.

The second type are sent from the `NotificationEventReceiver` itself when a notification has been dismissed.

### NotificationIntentService.java

```
package com(tpb.projects.notifications;

import android.app.IntentService;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.net.Uri;
import android.support.annotation.Nullable;
import android.support.v4.app.TaskStackBuilder;
import android.support.v4.content.WakefulBroadcastReceiver;
import android.support.v7.app.NotificationCompat;

import com(tpb.github.data.APIHandler;
import com(tpb.github.data.Editor;
import com(tpb.github.data.Loader;
import com(tpb.github.data.Util;
import com(tpb.github.data.models.Notification;
import com(tpb.mdtext.TextUtils;
import com(tpb.projects.BuildConfig;
import com(tpb.projects.R;
import com(tpb.projects.flow.Interceptor;
import com(tpb.projects.util.Logger;

import java.util.List;

/**
 * Created by theo on 04/04/17.
 */
```

```
public class NotificationIntentService extends IntentService implements
Loader.ListLoader<Notification> {
 private static final String TAG =
NotificationIntentService.class.getSimpleName();

 private static final String ACTION_CHECK = "ACTION_CHECK";
 private static final String ACTION_DELETE = "ACTION_DELETE";

 private long mLastLoadedSuccessfully = 0;

 public NotificationIntentService() {
 super(BuildConfig.APPLICATION_ID);
 }

 public static Intent createIntentStartNotificationService(Context context)
{
 final Intent intent = new Intent(context,
NotificationIntentService.class);
 intent.setAction(ACTION_CHECK);
 return intent;
}

@Override
protected void onHandleIntent(@Nullable Intent intent) {
 if(intent == null) return;
 try {
 final String action = intent.getAction();
 if(ACTION_CHECK.equals(action)) {
 loadNotifications();
 } else if(ACTION_DELETE.equals(action) &&
intent.getStringExtra("notification") != null) {
 Editor.getEditor(this).markNotificationThreadRead(
 (Notification)
intent.getParcelableExtra("notification")).getId()
 }
 }
 finally {
 Logger.i(TAG, "onHandleIntent: " + intent.toString());
 WakefulBroadcastReceiver.completeWakefulIntent(intent);
 }
}

private void loadNotifications() {
 Logger.i(TAG, "loadNotifications: Timestamp " + Util
 .toISO8601FromMilliseconds(mLastLoadedSuccessfully));
 Loader.getLoader(getApplicationContext()).loadNotifications(this,
mLastLoadedSuccessfully);
}

private android.app.Notification buildNotification(Notification notif) {
 final NotificationCompat.Builder builder = new
NotificationCompat.Builder(this);
 String title;
 switch(notif.getReason()) {
 case AUTHOR:
 title =
String.format(getString(R.string.text_notification_author),
 notif.getRepository().getFullName()
);

```

```
 builder.setSmallIcon(R.drawable.ic_person_white);
 break;
 case COMMENT:
 title =
String.format(getString(R.string.text_notification_comment),
 notif.getRepository().getName()
);
 builder.setSmallIcon(R.drawable.ic_comment_white);
 break;
 case ASSIGN:
 title = String.format(
 getString(R.string.text_notification_assign),
 "Issue",
 notif.getRepository().getFullName()
);
 builder.setSmallIcon(R.drawable.ic_person_white);
 break;
 case INVITATION:
 title = getString(R.string.text_notification_invitation);
 builder.setSmallIcon(R.drawable.ic_group_add_white);
 break;
 case MANUAL:
 title = getString(R.string.text_notification_manual,
 notif.getRepository().getFullName()
);
 builder.setSmallIcon(R.drawable.ic_watchers_white);
 break;
 case MENTION:
 title = getString(R.string.text_notification_mention,
 notif.getRepository().getFullName()
);
 builder.setSmallIcon(R.drawable.ic_mention_white);
 break;
 case SUBSCRIBED:
 if("issue".equalsIgnoreCase(notif.getType())) {
 title =
String.format(getString(R.string.text_notification_issue),
 notif.getRepository().getName()
);
 builder.setSmallIcon(R.drawable.ic_issue_white);
 } else {
 title = getString(R.string.text_notification_subscribed,
 notif.getRepository().getFullName()
);
 builder.setSmallIcon(R.drawable.ic_watchers_white);
 }
 break;
 default:
 title =
TextUtils.capitalizeFirst(notif.getReason().toString());
 break;
}
final TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
stackBuilder.addParentStack(Interceptor.class);
final Intent launchIntent = new Intent(Intent.ACTION_VIEW,
Uri.parse(notif.getUrl()));
launchIntent.putExtra("notif", notif);
stackBuilder.addNextIntent(launchIntent);
Logger.i(TAG, "buildNotification: URL " + notif.getUrl());
```

```

 builder.setContentIntent(
 stackBuilder.getPendingIntent(0,
PendingIntent.FLAG_UPDATE_CURRENT));
 builder.setCategory(android.app.Notification.CATEGORY_MESSAGE);
 builder.setGroup("GITHUB_GROUP");
 builder.setContentTitle(title);
 builder.setContentText(notif.getTitle());
 builder.setAutoCancel(true);
 builder.setDeleteIntent(generateDismissIntent(notif));
 return builder.build();
 }

 private PendingIntent generateDismissIntent(Notification notif) {
 return PendingIntent.getService(this, 0,
generateBroadcastDismissIntent(this, notif), PendingIntent.FLAG_ONE_SHOT);
 }

 public static Intent generateBroadcastDismissIntent(Context context,
Notification notif) {
 final Intent i = new Intent(context,
 NotificationIntentService.class
);
 i.setAction(ACTION_DELETE);
 i.putExtra("notification", notif);
 return i;
 }

 @Override
 public void listLoadComplete(List<Notification> notifications) {
 Logger.i(TAG, "listLoadComplete: " + notifications.size());
 mLastLoadedSuccessfully = Util.getUTCTimeInMillis();
 final NotificationManager manager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
 for(Notification n : notifications) {
 manager.notify((int) n.getId(), buildNotification(n));
 }
 }

 @Override
 public void listLoadError(APIHandler.APIError error) {
 Logger.e(TAG, "listLoadError: " + error);
 }
}

```

## Notification loading and displaying

If the Intent action is ACTION\_CHECK, loadNotifications is called.

This uses the Loader to load notifications since the last time that notifications were successfully loaded.

When listLoadComplete is called, mLastLoadedSuccessfully is updated, and the NotificationManager is used to send a notification for each Notification loaded.

`buildNotification` creates an `android.app.Notification` (Not a `Notification` model).

A `NotificationCompat.Builder` instance is created to build the notification.

The `Notification` reason enum is switched over to format the title of the notification, and set an icon appropriate to its reason for existing.

Next, a `TaskStackBuilder` is created.

This is used to ensure that the Activity launched if the `Notification` is clicked returns to the application that the user was in when they clicked on the notification, rather than returning to the top Activity on the stack for this app.

The Intent for launching the notification is set with the `ACTION_VIEW` action, and the `Notification` URL. The `Notification` is then added as an extra, allowing it to be dismissed from `Interceptor`.

The content intent on the builder is then set to the `PendingIntent` generated from the `TaskStackBuilder`, the category is set to `CATEGORY_MESSAGE`, the group is set to "GITHUB\_GRUOP" which allows notifications to be grouped together, the title is set to the title string created earlier, and the content is set to the title returned by GitHub.

`Auto cancel` is set to true, meaning that the notification will be removed when it is launched.

The delete intent is then set on the builder, which is to be called if the notification is swiped away by the user.

`generateDismissIntent` creates a `PendingIntent` with the `FLAG_ONE_SHOT` flag, indicating that it can only be used once.

`generateBroadcastDismissIntent` is used to generate the actual Intent. It creates an Intent for the `NotificationIntentService` class, using `ACTION_DELETE`, and adding the `Notification` as an extra.

Finally, the builder is built and returned as an `android.app.Notification`.

## **Dismissing notifications**

When a notification is dismissed or opened, it should be dismissed so that it is not shown again.

The callback to perform the network request is easily achievable when the notification is swiped away.

Calling `setDeleteIntent` results in the Intent being launched when the notification is deleted (who would have thought?).

The delete notification is received in `onHandleIntent`, and as the action is `ACTION_DELETE`, the `Editor` method `markNotificationThreadRead` is called.

Marking notifications read when they are launched is slightly more complicated.

The Intent which is fired on click is the Intent to launch the `Interceptor`.

In order to call back to the `NotificationIntentService` the `Notification` is added to the Intent.

In Interceptor, if the Intent has a notification extra, startService is called with NotificationIntentService.generateBroadcastDismissIntent which will call onHandleIntent in NotificationIntentService to mark the notification as read.