1 AmazingRace Android Anwendung

Folgende Dokumentation dokumentiert die Entwicklung der mobilen Anwendung *AmazingRace*, die in Java für die Android-Plattform *API-Level 22, Android 5.1.1* entwickelt wurde.

1.1 Lösungsidee

Die Anwendung soll in auf drei Aktivitäten aufgeteilt werden, wobei die Login-Aktivität als Launcher-Aktivität definiert werden soll. Über diese Aktivität wird die Anwendung gestartet. Nach einem erfolgreichem Login soll diese Aktivität beendet werden, da es keinen Grund gibt warum eine Benutzer wieder auf diese Aktivität zurückkehren sollte.

Nach dem Login wird der Benutzer zu einer Übersicht aller Routen weitergeleitet, wo er eine auswählen kann. In dieser Ansicht sollen die einzelnen Routen mit einem Kontext-Menu versehen werden, die je nach Zustand der Route folgende Optionen bereitstellen soll.

1. *Play (Es gibt einen offenen Checkpoint)*
   Wechselt zur Checkpoint-Aktivität.
2. *Open (Route bereits beendet)*
   Wechselt zur Checkpoint-Aktivität
3. *Reset (Wenn zumindest ein Checkpoint besucht wurde)*
   Setzt diese Route zurück und verbleibt auf dieser Aktivität

Ebenfalls soll eine Action-Bar implementiert werden, die folgende Optionen bereitstellten soll.

1. *Reset*
   Setzt alle Routen zurück und verbleibt auf dieser Aktivität
2. *Reload*
   Lädt alle Routen neu.
3. *Close*
   Öffnet einen Dialog, den der Benutzer bestätigen muss und beendet die Anwendung

Für alle Ladeoperationen soll ein ProgressDialog angezeigt werden, der eine dementsprechende Meldung beinhaltet (Z.b.: Load routes ...). Der verwendete Service soll über ein Interface abgebildet werden, somit beinhaltet die benutzenden Klasse keine Referenzen auf die konkrete Implementierung. Die Implementierung soll über eine *Factory* bereitgestellt werden, die die Service Instanz als *Singleton* verwaltet, da hier nur eine Instanz benötigt wird. Des Weiteren sollen *ViewModels* implementiert werden, die die gesamte View-Logik beinhalteten und keine Referenzen auf Android-Ressourcen beinhalten sollen.

Die Aktivitäten sollen die gesamte Interaktionslogik beinhalten und die möglichen *ServiceExceptions* der Serviceaufrufe, die über die View-Model erfolgen und von diesen weitergereicht werden, behandeln, da hierbei Referenzen auf Android-Ressourcen benötigt werden. Diese *ServiceException* soll in einer abstrakten Basisklasse behandelt werden, von der alle Aktivitäten ableiten. Somit ist diese Fehlerbehandlung (Anzeige für den Benutzer) in einer Klasse gekapselt und trotzdem können die Aktivitäten ihrerseits Fehlerbehandlungen durchführen.

1.1.1 Architektur

Folgend ist die Architektur der Anwedung *AmazingRace* dokumentiert.

Folgende Abbildung zeigt die Hierarchie der Klasse *AbstractActivity* die als Basisklasse aller Aktivitäten fungiert und gemeinsam genutzte Ressourcen und Konstanten bereitstellt.

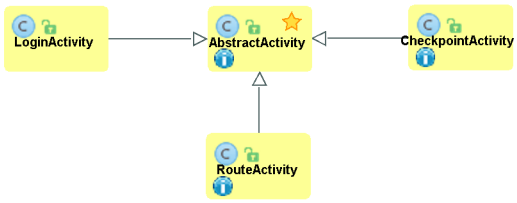

Abbildung 1: Klassenhierarchie der Klasse *AbstractActivity*

Folgende Abbildung zeigt die Paketstruktur der Anwendung *AmazingRace*. Als Wurzelnamensraum wurde *at.fh.ooe.moc5.amazingrace* gewählt unter dem sich alle Ressourcen befinden.

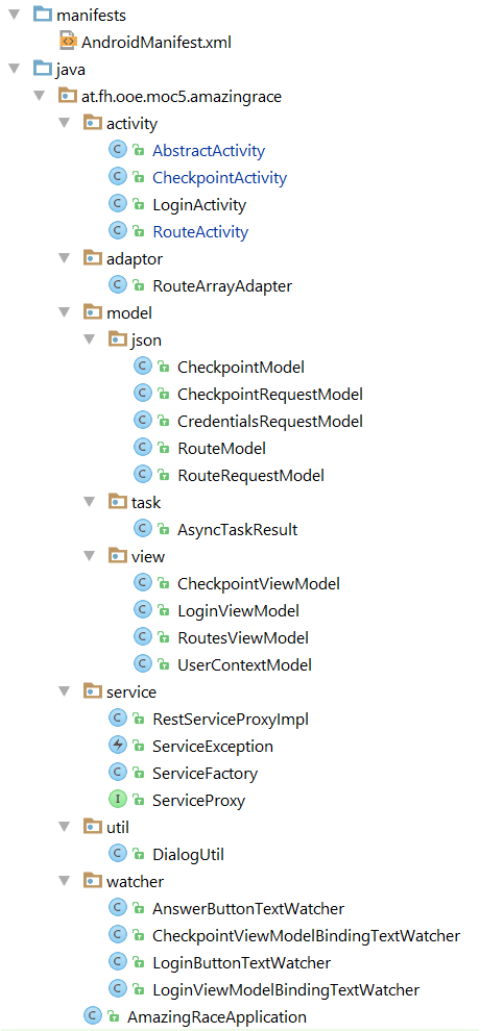

Abbildung 2: Paketstruktur der Anwendung

Folgende Abbildung zeigt die Ressourcenverzeichnisse in denen alle Ressourcen wie *Layout* usw. liegen. Es wurde hierbei darauf verzichtet, die Ressourcen für mehrere Endgeräte und Sprachen zu definieren.

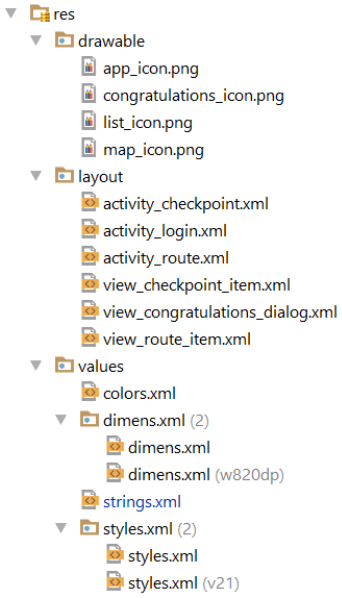

Abbildung 3: Ressourcen der Anwendung

Des Weiteren wurde eine Klasse *AmazingRaceApplication* eingeführt, die die unbehandelten Exceptions behandelt, den Benutzerkontext hält und gemeinsam verwendete Konstanten definiert.

1.1.2 Setup

Da ein Google Android API-Key verwendet wird wurde ein eigenes Zertifikat erstellt welches bei Google-Developer-Console registriert wurde. Dieses Zertifikat muss im Android-Studio bei der debug und Release Konfiguration eingestellt werden ansonsten wird Google Maps nicht funktionieren. Es sollte zwar ein Zertifikat für Debug und Release erstellt werden, aber es wurde hierbei darauf verzichtet und es wird nur ein Zertifikat für beide Konfigurationen verwendet.

1. *Alias:* amazingrace
2. *Key-Password:* amazingrace
3. *Keystore-Password:* amazingrace

Der Keystore befindet sich im Android-Studio-Projekt Wurzelverzeichnis und trägt den Name *amazingrace.jks*

Des Weiteren wurde folgender Emulator verwendet

1. *Device:* Nexus 5
2. *API-Level:* 22
3. *Architektur:* x86_64 (schneller und *ARM* hatte falsche Play-Service Version)

Die Applikation wurde ebenfalls auf eine *Samsung S6-Edge* getestet.

1.2 Source

Im folgenden ist der gesamte Source der Anwendung aufgeführt.

1.2.1 Manifest

Listing 1: AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="at.fh.ooe.moc5.amazingrace">

    <!-- Permissions -->
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.LOCATION_HARDWARE" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

    <application
        android:name=".AmazingRaceApplication"
        android:allowBackup="true"
        android:icon="@drawable/app_icon"
        android:label="@string/application_title"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <!-- Main Activity -->
        <activity
            android:name=".activity.LoginActivity"
            android:label="@string/application_title">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <!-- explicit accessible activities -->
        <activity
            android:name=".activity.RouteActivity"
            android:label="@string/activity_title_route" />
        <activity
            android:name=".activity.CheckpointActivity"
            android:label="@string/activity_title_checkpoint" />

        <!-- Meta-Data for google maps -->
        <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="@string/google_maps_key" />
    </application>
</manifest>
```

1.2.2 Aktivitäten

Listing 2: AmazingRaceApplication.java

```java
package at.fh.ooe.moc5.amazingrace;

import android.app.Application;
import android.util.Log;
import android.widget.Toast;

import at.fh.ooe.moc5.amazingrace.model.view.UserContextModel;

/**
 * Created by Thomas on 12/24/2015.
 * This is the custom application implementation which handles uncaught exceptions.
 * The application will get killed on a an uncaught exception.
 */
public class AmazingRaceApplication extends Application {

    public UserContextModel loggedUser;

    public static final String INTENT_EXTRA_ROUTE = "INTENT_EXTRA_ROUTE";

    /**
     * Registers an exception handler for uncaught exceptions.
     */
    public void onCreate() {
        super.onCreate();
        // Setup handler for uncaught exceptions.
        Thread.setDefaultUncaughtExceptionHandler(new Thread.UncaughtExceptionHandler() {
            @Override
            public void uncaughtException(Thread thread, Throwable e) {
                handleUncaughtException(thread, e);
            }
        });
    }

    /**
     * Handles uncaught exceptions which are considered critical, therefore the application will
     ↪  exist with an error.
     *
     * @param thread the current thread the exception occurred
     * @param e      the occurred exception
     */
    public void handleUncaughtException(Thread thread, Throwable e) {
        e.printStackTrace();
        Log.e("error", "Uncaught exception catched", e);
        System.exit(1);
    }

    /**
     * gets the logged user context
     *
     * @return the user context representing the logged user
     */
    public UserContextModel getLoggedUser() {
        return loggedUser;
    }

    /**
     * Sets the logged user which can be accessed by all implemented activities.
     *
     * @param loggedUser the logged user
```

```
59          */
60          public void setLoggedUser(UserContextModel loggedUser) {
61              this.loggedUser = loggedUser;
62          }
63      }
```

Listing 3: AbstractActivity.java

```java
package at.fh.ooe.moc5.amazingrace.activity;

import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.widget.Toast;

import java.io.Serializable;
import java.util.Objects;

import at.fh.ooe.moc5.amazingrace.AmazingRaceApplication;
import at.fh.ooe.moc5.amazingrace.R;
import at.fh.ooe.moc5.amazingrace.service.ServiceException;
import at.fh.ooe.moc5.amazingrace.util.DialogUtil;

/**
 * Created by Thomas on 12/28/2015.
 * <p/>
 * This is the base class for all activities which forces the activity to define one view model it uses.
 * This class provides common functionality usable by all activities.
 */
public class AbstractActivity<T extends Serializable> extends AppCompatActivity {

    protected T viewModel;
    protected boolean restored = Boolean.FALSE;
    protected AmazingRaceApplication application;
    protected Boolean validViewModel = Boolean.TRUE;

    private ProgressDialog progress;

    private static final String VIEW_MODEL = "AbstractActivity#VIEW_MODEL";

    // region Constants

    /**
     * Enumeration for specifying the available menu group ids
     */
    public static enum MenuGroup {
        ROUTE_ITEM(0),
        OPTIONS(1);

        public final int value;

        private MenuGroup(int value) {
            this.value = value;
        }

        public static MenuGroup getMenuGroupForValue(int value) {
            for (MenuGroup group : MenuGroup.values()) {
                if (group.value == value) {
                    return group;
                }
            }
```

```
60
61              throw new IllegalArgumentException("MenuGroup with value '" + value + "' not found");
62          }
63      }
64
65      /**
66       * Enumeration for specifying the available menu ids
67       */
68      public static enum MenuId {
69          PLAY(0),
70          RESET(1),
71          RESET_ALL_ROUTES(2),
72          CLOSE(3),
73          RELOAD(4);
74
75          public final int value;
76
77          private MenuId(int value) {
78              this.value = value;
79          }
80
81          public static MenuId getMenuIdForValue(int value) {
82              for (MenuId id : MenuId.values()) {
83                  if (id.value == value) {
84                      return id;
85                  }
86              }
87
88              throw new IllegalArgumentException("MenuGroup with value '" + value + "' not found");
89          }
90      }
91      //endregion
92
93      //region Lifecycle Methods
94
95      /**
96       * Initializes the backed application so that it can be accessed by the concrete activity
    ↪  application.
97       * It also tries to get the saved view model from the savedInstanceState which can be null
    ↪  here if haven't been saved before.
98       *
99       * @param savedInstanceState
100      */
101     @Override
102     protected void onCreate(Bundle savedInstanceState) {
103         super.onCreate(savedInstanceState);
104         application = (AmazingRaceApplication) getApplication();
105         if (savedInstanceState != null) {
106             viewModel = (T) savedInstanceState.get(VIEW_MODEL);
107             restored = (viewModel != null);
108         }
109     }
110
111     /**
112      * Re-initializes the application on resume of the activity.
113      */
114     @Override
115     protected void onResume() {
116         super.onResume();
117         application = (AmazingRaceApplication) getApplication();
118         if (viewModel == null) {
119             validViewModel = Boolean.FALSE;
120             openInvalidUserAccountDialog();
```

```
121         }
122     }
123
124     /**
125      * Saves the backed view model.
126      *
127      * @param outState the out coming state
128      */
129     @Override
130     protected void onSaveInstanceState(Bundle outState) {
131         super.onSaveInstanceState(outState);
132         outState.putSerializable(VIEW_MODEL, viewModel);
133     }
134     //endregion
135
136     /**
137      * Handles a service exception by displaying a toast with a proper message
138      *
139      * @param exception the occurred ServiceException
140      * @return true if handled here, false otherwise, which indicates that no error code or an
    ↪  unknown error occurred was set.
141      */
142     protected boolean handleServiceException(ServiceException exception) {
143         Objects.requireNonNull(exception);
144         boolean handled = Boolean.FALSE;
145         ServiceException.ServiceErrorCode errorCode = exception.getErrorCode();
146         if (errorCode != null) {
147             switch (errorCode) {
148                 case INVALID_REQUEST:
149                     Toast.makeText(AbstractActivity.this, R.string.error_request_invalid,
                        ↪  Toast.LENGTH_LONG).show();
150                     handled = Boolean.TRUE;
151                     break;
152                 case TIMEOUT:
153                     Toast.makeText(AbstractActivity.this, R.string.error_request_timeout,
                        ↪  Toast.LENGTH_LONG).show();
154                     handled = Boolean.TRUE;
155                     break;
156                 case UNKNOWN:
157                     Toast.makeText(AbstractActivity.this, R.string.error_unknown,
                        ↪  Toast.LENGTH_LONG).show();
158                     handled = Boolean.TRUE;
159                     break;
160                 case INVALID_CREDENTIALS:
161                     openInvalidUserAccountDialog();
162                     handled = Boolean.TRUE;
163             }
164         } else {
165             Toast.makeText(AbstractActivity.this, R.string.error_unknown,
                ↪  Toast.LENGTH_LONG).show();
166         }
167         Log.e(this.getClass().getSimpleName(), "ServiceException occurred", exception);
168
169         return handled;
170     }
171
172     /**
173      * Opens the invalid user dialog which indicates that the user has become invalid during
    ↪  application invocation.
174      * After the user clicked ok the user will be logged out.
175      */
176     protected void openInvalidUserAccountDialog() {
177         DialogUtil.createErrorDialog(AbstractActivity.this, R.string.error_no_network, new
            ↪  DialogInterface.OnClickListener() {
```

```
178             @Override
179             public void onClick(DialogInterface dialog, int which) {
180                 dialog.dismiss();
181                 application.setLoggedUser(null);
182                 finishAffinity();
183                 startActivity(new Intent(AbstractActivity.this, LoginActivity.class));
184             }
185         }).show();
186     }
187 
188     /**
189      * This method checks if an internet connection is available for the application.
190      * If a network connection is not available then a dialog is shown to notify the user.
191      *
192      * @return true if network is available false otherwise
193      * @see AbstractActivity#isOnline()
194      */
195     protected boolean checkAndDisplayAvailableNetwork() {
196         final boolean online = isOnline();
197         if (!online) {
198             DialogUtil.createErrorDialog(AbstractActivity.this, R.string.error_no_network, new
                 ↪  DialogInterface.OnClickListener() {
199                 @Override
200                 public void onClick(DialogInterface dialog, int which) {
201                     dialog.dismiss();
202                 }
203             }).show();
204         }
205         return online;
206     }
207 
208     /**
209      * Opens an dialog to ask the user if he really wants to quit the application.
210      */
211     protected void openCloseApplicationDialog() {
212         DialogUtil.createYesNoAlertDialog(AbstractActivity.this, R.string.dialog_title_warning,
             ↪  R.string.warning_want_quit, new DialogInterface.OnClickListener() {
213             @Override
214             public void onClick(DialogInterface dialog, int which) {
215                 dialog.dismiss();
216                 switch (which) {
217                     case DialogInterface.BUTTON_POSITIVE:
218                         application.setLoggedUser(null);
219                         finishAffinity();
220                         break;
221                 }
222             }
223         }).show();
224     }
225 
226     /**
227      * Opens a progress dialog with an custom message.
228      *
229      * @param messageId the resource id of the message
230      */
231     protected void openProgressDialog(int messageId) {
232         progress = ProgressDialog.show(AbstractActivity.this, "", getString(messageId),
             ↪  Boolean.TRUE);
233     }
234 
235     /**
236      * Closes the progress dialog.
237      */
```

```
238     protected void closeProgressDialog() {
239         if (progress != null) {
240             progress.dismiss();
241             progress = null;
242         }
243     }
244 
245     /**
246      * Helper method for checking if a network connection is available.
247      *
248      * @return true if a network connection is available, false otherwise
249      */
250     protected boolean isOnline() {
251         ConnectivityManager cm =
252                 (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
253         NetworkInfo netInfo = cm.getActiveNetworkInfo();
254         return netInfo != null && netInfo.isConnected();
255     }
256 }
```

Listing 4: LoginActivity.java

```java
package at.fh.ooe.moc5.amazingrace.activity;

import android.content.Intent;
import android.os.*;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import at.fh.ooe.moc5.amazingrace.R;
import at.fh.ooe.moc5.amazingrace.model.task.AsyncTaskResult;
import at.fh.ooe.moc5.amazingrace.model.view.LoginViewModel;
import at.fh.ooe.moc5.amazingrace.model.view.UserContextModel;
import at.fh.ooe.moc5.amazingrace.service.ServiceException;
import at.fh.ooe.moc5.amazingrace.watcher.LoginButtonTextWatcher;
import at.fh.ooe.moc5.amazingrace.watcher.LoginViewModelBindingTextWatcher;

/**
 * This class represents the login activity where an user need to login.
 */
public class LoginActivity extends AbstractActivity<LoginViewModel> implements
 ↪  View.OnClickListener {

    // region Lifecycle Methods

    /**
     * Sets acitivites content layout and initializes the view model
     *
     * @param savedInstanceState
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        // Here we want always have an new view model
        viewModel = new LoginViewModel();
    }

    /**
     * Prepares the activity views if view model is valid.
     */
    @Override
    protected void onResume() {
        super.onResume();
        // View model will always be valid here
        prepareViews();
    }

    /**
     * Open dialog when back button is pressed on this activity.
     *
     * @see AbstractActivity#openCloseApplicationDialog();
     */
    @Override
    public void onBackPressed() {
        super.onBackPressed();
        openCloseApplicationDialog();
    }
    // endregion

```

```
60      //region Listeners
61
62      /**
63       * Onclick implementation which
64       *
65       * @param v
66       */
67      public void onClick(View v) {
68          switch (v.getId()) {
69              case R.id.loginBtn:
70                  loginUser();
71                  break;
72              case R.id.cancelBtn:
73                  openCloseApplicationDialog();
74                  break;
75          }
76      }
77      //endregion
78
79
80      //region Helper
81
82      /**
83       * Prepares the activity views.
84       */
85      private void prepareViews() {
86          // prepare username view
87          EditText username = (EditText) findViewById(R.id.usernameEdTxt);
88          username.addTextChangedListener(new LoginViewModelBindingTextWatcher(viewModel,
             ↪  username));
89          username.addTextChangedListener(new LoginButtonTextWatcher(this, viewModel));
90          username.setText(viewModel.getPassword());
91
92          // prepare password view
93          EditText password = (EditText) findViewById(R.id.passwordEdTxt);
94          password.addTextChangedListener(new LoginViewModelBindingTextWatcher(viewModel,
             ↪  password));
95          password.addTextChangedListener(new LoginButtonTextWatcher(this, viewModel));
96          username.setText(viewModel.getPassword());
97
98          // Prepare login button
99          Button loginButton = (Button) findViewById(R.id.loginBtn);
100         loginButton.setEnabled(Boolean.FALSE);
101         loginButton.setOnClickListener(this);
102
103         // prepare cancel button
104         Button cancelButton = (Button) findViewById(R.id.cancelBtn);
105         cancelButton.setOnClickListener(this);
106     }
107
108     /**
109      * Resets the views by setting string value to empty string.
110      */
111     private void resetView() {
112         EditText username = (EditText) findViewById(R.id.usernameEdTxt);
113         EditText password = (EditText) findViewById(R.id.passwordEdTxt);
114
115         // Could be called on non visible activity
116         if ((username != null) && (password != null)) {
117             username.setText("");
118             password.setText("");
119         }
120     }
```

```
121
122     /**
123      * Performs asynchronous login service call.
124      */
125     private void loginUser() {
126         if (checkAndDisplayAvailableNetwork()) {
127             new AsyncTask<Object, Object, AsyncTaskResult<UserContextModel>>() {
128                 @Override
129                 protected AsyncTaskResult<UserContextModel> doInBackground(Object... params) {
130                     UserContextModel model = null;
131                     Exception exception = null;
132                     try {
133                         model = viewModel.loginAction();
134                     } catch (Exception e) {
135                         exception = e;
136                     }
137
138                     return new AsyncTaskResult<UserContextModel>(model, exception);
139                 }
140
141                 @Override
142                 protected void onPreExecute() {
143                     super.onPreExecute();
144                     openProgressDialog(R.string.progress_login);
145                 }
146
147                 @Override
148                 protected void onPostExecute(AsyncTaskResult<UserContextModel> result) {
149                     super.onPostExecute(result);
150                     closeProgressDialog();
151                     // Exception occurred
152                     if (result.exception != null) {
153                         resetView();
154                         if (result.exception instanceof ServiceException) {
155                             handleServiceException(((ServiceException) result.exception));
156                         } else {
157                             Toast.makeText(LoginActivity.this, R.string.error_unknown,
                                 ↪  Toast.LENGTH_SHORT).show();
158                         }
159                     }
160                     // Login failed due to invalid credentials
161                     else if (result.result == null) {
162                         resetView();
163                         Toast.makeText(LoginActivity.this, R.string.error_login_failed,
                             ↪  Toast.LENGTH_SHORT).show();
164                     }
165                     // Login ok
166                     else {
167                         application.setLoggedUser(result.result);
168                         final Intent intent = new Intent(LoginActivity.this, RouteActivity.class);
169                         startActivity(intent);
170                         // Finish this activity because no need to get back to this one
171                         finish();
172                     }
173                 }
174             }.execute();
175         }
176     }
177     // endregion
178 }
```

Listing 5: RouteActivity.java

```java
package at.fh.ooe.moc5.amazingrace.activity;

import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.Toast;

import java.util.List;

import at.fh.ooe.moc5.amazingrace.AmazingRaceApplication;
import at.fh.ooe.moc5.amazingrace.R;
import at.fh.ooe.moc5.amazingrace.adapter.RouteArrayAdapter;
import at.fh.ooe.moc5.amazingrace.model.json.RouteModel;
import at.fh.ooe.moc5.amazingrace.model.task.AsyncTaskResult;
import at.fh.ooe.moc5.amazingrace.model.view.RoutesViewModel;
import at.fh.ooe.moc5.amazingrace.service.ServiceException;

/**
 * This class represents the route activity which lists all available routes to the user.
 */
public class RouteActivity extends AbstractActivity<RoutesViewModel> implements
    AdapterView.OnItemClickListener {

    //region Lifecycle Methods

    /**
     * Initializes this activity.
     *
     * @param savedInstanceState
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_route);
        // Only if user is logged and view model hasn't been restored from bundle
        if ((application.getLoggedUser() != null) && (!restored)) {
            viewModel = new RoutesViewModel(application.getLoggedUser());
        }
        // Register route list for context menu
        registerForContextMenu(findViewById(R.id.listRoute));
    }

    /**
     * Prepares the activity views if view model is valid
     */
    @Override
    protected void onResume() {
        super.onResume();
        if (validViewModel) {
            prepareView();
            loadRoutes();
        }
    }

```

```
60      /**
61       * Opens application close dialog if back button is pressed on this activity.
62       *
63       * @see AbstractActivity#openCloseApplicationDialog();
64       */
65      @Override
66      public void onBackPressed() {
67          super.onBackPressed();
68          openCloseApplicationDialog();
69      }
70
71      /**
72       * Creates the context menu items for the list view.
73       *
74       * @param menu     the menu to add options too.
75       * @param v        the view to add options for
76       * @param menuInfo the menuinfo related to the view
77       */
78      @Override
79      public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo
         ↪  menuInfo) {
80          super.onCreateContextMenu(menu, v, menuInfo);
81          // Creates options for list view
82          if (R.id.listRoute == v.getId()) {
83              onCreateRouteListContextMenu(menu, (ListView) v, (AdapterView.AdapterContextMenuInfo)
                 ↪  menuInfo);
84          }
85      }
86
87      /**
88       * Creates the menu items for the application action bar
89       *
90       * @param menu the menu to add options too
91       * @return true
92       */
93      @Override
94      public boolean onCreateOptionsMenu(Menu menu) {
95          menu.add(MenuGroup.OPTIONS.value, MenuId.RESET_ALL_ROUTES.value, 0,
             ↪  R.string.action_reset_all);
96          menu.add(MenuGroup.OPTIONS.value, MenuId.RELOAD.value, 0, R.string.action_reload);
97          menu.add(MenuGroup.OPTIONS.value, MenuId.CLOSE.value, 0, R.string.action_close);
98          return Boolean.TRUE;
99      }
100
101     /**
102      * Handles the menu options select and delegates to the proper method.
103      *
104      * @param item the menu item which has been selected
105      * @return
106      */
107     @Override
108     public boolean onOptionsItemSelected(MenuItem item) {
109         switch (MenuId.getMenuIdForValue(item.getItemId())) {
110             case RESET_ALL_ROUTES:
111                 resetRoute(null);
112                 return Boolean.TRUE;
113             case RELOAD:
114                 loadRoutes();
115                 return Boolean.TRUE;
116             case CLOSE:
117                 openCloseApplicationDialog();
118                 return Boolean.TRUE;
119             default:
```

```
120                 return super.onOptionsItemSelected(item);
121             }
122         }
123 
124         //endregion
125 
126         //region Utilities
127 
128         /**
129          * Creates the menu items for the list view
130          *
131          * @param menu     the context menu to add items too
132          * @param v        the list view to create items for
133          * @param menuInfo the list view related menu info
134          */
135         private void onCreateRouteListContextMenu(ContextMenu menu, ListView v,
136                                                   AdapterView.AdapterContextMenuInfo menuInfo) {
137             RouteModel route = (RouteModel) v.getItemAtPosition(menuInfo.position);
138             if (route.getNextCheckpoint() != null) {
139                 menu.add(MenuGroup.ROUTE_ITEM.value, MenuId.PLAY.value, 0, R.string.action_play);
140                 if (!route.getVisitedCheckpoints().isEmpty()) {
141                     menu.add(MenuGroup.ROUTE_ITEM.value, MenuId.RESET.value, 1,
                         ↪  R.string.action_reset);
142                 }
143             } else {
144                 menu.add(MenuGroup.ROUTE_ITEM.value, MenuId.PLAY.value, 0, R.string.action_open);
145                 menu.add(MenuGroup.ROUTE_ITEM.value, MenuId.RESET.value, 0, R.string.action_reset);
146             }
147         }
148 
149         /**
150          * Handles the selected context item
151          *
152          * @param item the selected menu item
153          * @return true if item has been handled
154          */
155         @Override
156         public boolean onContextItemSelected(MenuItem item) {
157             switch (MenuGroup.getMenuGroupForValue(item.getGroupId())) {
158                 case ROUTE_ITEM:
159                     return handleRouteItemSelected(item);
160                 default:
161                     return super.onContextItemSelected(item);
162             }
163         }
164 
165         /**
166          * Handles the selected route menu item.
167          *
168          * @param item the selected menu item
169          * @return true if item has been handled
170          */
171         private boolean handleRouteItemSelected(MenuItem item) {
172             final RouteModel model = (RouteModel) ((ListView)
                 ↪  findViewById(R.id.listRoute)).getItemAtPosition(((AdapterView.AdapterContextMenuInfo)
                 ↪  item.getMenuInfo()).position);
173             switch (MenuId.getMenuIdForValue(item.getItemId())) {
174                 case PLAY:
175                     goToCheckpointActivity(model);
176                     return Boolean.TRUE;
177                 case RESET:
178                     resetRoute(model);
179                     return Boolean.TRUE;
```

```
180             default:
181                 return Boolean.FALSE;
182         }
183     }
184 
185     /**
186      * Prepares this activity views.
187      */
188     private void prepareView() {
189         // prepare route list view
190         ListView listView = (ListView) findViewById(R.id.listRoute);
191         listView.removeAllViewsInLayout();
192         if (listView.getAdapter() == null) {
193             final RouteArrayAdapter adapter = new RouteArrayAdapter(RouteActivity.this);
194             listView.setAdapter(adapter);
195             listView.setOnItemClickListener(this);
196         }
197     }
198 
199     /**
200      * Loads the routes into the adapter
201      */
202     private void loadRoutes() {
203         if (checkAndDisplayAvailableNetwork()) {
204             new AsyncTask<Object, Object, AsyncTaskResult<List<RouteModel>>>() {
205                 @Override
206                 protected AsyncTaskResult<List<RouteModel>> doInBackground(Object... params) {
207                     List<RouteModel> routes = null;
208                     Exception exception = null;
209                     try {
210                         routes = viewModel.loadRoutes();
211                     } catch (Exception e) {
212                         exception = e;
213                     }
214 
215                     return new AsyncTaskResult<List<RouteModel>>(routes, exception);
216                 }
217 
218                 @Override
219                 protected void onPreExecute() {
220                     super.onPreExecute();
221                     openProgressDialog(R.string.progress_loading_routes);
222                 }
223 
224                 @Override
225                 protected void onPostExecute(AsyncTaskResult<List<RouteModel>> result) {
226                     super.onPostExecute(result);
227                     closeProgressDialog();
228                     // Error occurred
229                     if (result.exception != null) {
230                         // ServiceException occurred
231                         if (result.exception instanceof ServiceException) {
232                             handleServiceException(((ServiceException) result.exception));
233                         } else {
234                             Toast.makeText(RouteActivity.this, R.string.error_unknown,
                                 ↪  Toast.LENGTH_LONG).show();
235                         }
236                     }
237                     // Lists where loaded
238                     else {
239                         RouteArrayAdapter adapter = (RouteArrayAdapter) ((ListView)
                             ↪  findViewById(R.id.listRoute)).getAdapter();
240                         adapter.clear();
```

```
241                         adapter.addAll(result.result);
242                     }
243                 }
244             }.execute();
245         }
246     }
247
248     /**
249      * Resets the given route.
250      *
251      * @param model the route to reset
252      */
253     private void resetRoute(final RouteModel model) {
254         if (checkAndDisplayAvailableNetwork()) {
255             new AsyncTask<Object, Object, AsyncTaskResult<Boolean>>() {
256                 @Override
257                 protected AsyncTaskResult<Boolean> doInBackground(Object... params) {
258                     Boolean result = null;
259                     Exception exception = null;
260                     try {
261                         if (model != null) {
262                             result = viewModel.resetRoute(model);
263                         } else {
264                             result = viewModel.resetAllRoutes();
265                         }
266                     } catch (Exception e) {
267                         exception = e;
268                     }
269
270                     return new AsyncTaskResult<Boolean>(result, exception);
271                 }
272
273                 @Override
274                 protected void onPreExecute() {
275                     super.onPreExecute();
276                     openProgressDialog(R.string.progress_resetting_routes);
277                 }
278
279                 @Override
280                 protected void onPostExecute(AsyncTaskResult<Boolean> result) {
281                     super.onPostExecute(result);
282                     closeProgressDialog();
283                     // Error occurred
284                     if (result.exception != null) {
285                         // ServiceException occurred
286                         if (result.exception instanceof ServiceException) {
287                             handleServiceException(((ServiceException) result.exception));
288                         } else {
289                             Toast.makeText(RouteActivity.this, R.string.error_unknown,
                                 ↪  Toast.LENGTH_LONG).show();
290                         }
291                     }
292                     // Reset failed on rest method
293                     else if (!result.result) {
294                         Toast.makeText(RouteActivity.this, R.string.error_route_reset_failed,
                             ↪  Toast.LENGTH_LONG).show();
295                     }
296                     // Reset ok
297                     else {
298                         loadRoutes();
299                     }
300                 }
301             }.execute();
```

```
302         }
303     }
304
305     /**
306      * Fires an intent which redirects teh user to the checkpoint activity.
307      *
308      * @param model the route to serialize to the checkpoint activity
309      */
310     private void goToCheckpointActivity(RouteModel model) {
311         Intent intent = new Intent(RouteActivity.this, CheckpointActivity.class);
312         intent.putExtra(AmazingRaceApplication.INTENT_EXTRA_ROUTE, model);
313         startActivity(intent);
314     }
315     //endregion
316
317     //region Listener
318
319     /**
320      * Clcik listener which redirects to the checkpoint activity
321      *
322      * @param parent   the adapter
323      * @param view     the view
324      * @param position the item position
325      * @param id       the view id
326      */
327     @Override
328     public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
329         goToCheckpointActivity(((RouteModel) parent.getItemAtPosition(position)));
330     }
331     //endregion
332 }
```

Listing 6: CheckpointActivity.java

```java
package at.fh.ooe.moc5.amazingrace.activity;

import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Color;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapFragment;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.maps.model.PolylineOptions;

import at.fh.ooe.moc5.amazingrace.AmazingRaceApplication;
import at.fh.ooe.moc5.amazingrace.R;
import at.fh.ooe.moc5.amazingrace.model.json.CheckpointModel;
import at.fh.ooe.moc5.amazingrace.model.json.RouteModel;
import at.fh.ooe.moc5.amazingrace.model.task.AsyncTaskResult;
import at.fh.ooe.moc5.amazingrace.model.view.CheckpointViewModel;
import at.fh.ooe.moc5.amazingrace.service.ServiceException;
import at.fh.ooe.moc5.amazingrace.util.DialogUtil;
import at.fh.ooe.moc5.amazingrace.watcher.AnswerButtonTextWatcher;
import at.fh.ooe.moc5.amazingrace.watcher.CheckpointViewModelBindingTextWatcher;

public class CheckpointActivity extends AbstractActivity<CheckpointViewModel> implements
 ↪  View.OnClickListener, DialogInterface.OnClickListener {

    //region Lifecycle Methods

    /**
     * Prepares the view model depending on the intent serialized route model
     *
     * @param savedInstanceState the saved bundle instance
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_checkpoint);
        if (application.getLoggedUser() != null) {
            final Intent intent = getIntent();
            if (intent.getExtras().containsKey(AmazingRaceApplication.INTENT_EXTRA_ROUTE)) {
                RouteModel route = (RouteModel)
                 ↪  intent.getExtras().get(AmazingRaceApplication.INTENT_EXTRA_ROUTE);
                viewModel = new CheckpointViewModel(application.getLoggedUser(), route);
            }
        }
    }
```

```
59
60      /**
61       * Prepares this activity views if view mdoel is valid.
62       */
63      @Override
64      protected void onResume() {
65          super.onResume();
66          if (validViewModel) {
67              prepareView();
68          }
69      }
70
71      /**
72       * Creates the options menu items
73       *
74       * @param menu the menu to add items too
75       * @return true
76       */
77      @Override
78      public boolean onCreateOptionsMenu(Menu menu) {
79          if (!viewModel.getVisitedCheckpoints().isEmpty()) {
80              menu.add(MenuGroup.OPTIONS.value, MenuId.RESET.value, 0, R.string.action_reset);
81          }
82          menu.add(MenuGroup.OPTIONS.value, MenuId.CLOSE.value, 0, R.string.action_close);
83          return Boolean.TRUE;
84      }
85
86      /**
87       * Handles an options menu item select
88       *
89       * @param item the selected menu item
90       * @return true if items has been handled
91       */
92      @Override
93      public boolean onOptionsItemSelected(MenuItem item) {
94          switch (MenuId.getMenuIdForValue(item.getItemId())) {
95              case RESET:
96                  resetRoute();
97                  return Boolean.TRUE;
98              case CLOSE:
99                  openCloseApplicationDialog();
100                 return Boolean.TRUE;
101             default:
102                 return super.onOptionsItemSelected(item);
103         }
104     }
105     //endregion
106
107     //region Helper
108
109     /**
110      * Prepares this acitivy views.
111      */
112     public void prepareView() {
113         // set default container visibility
114         findViewById(R.id.routeCheckpointContainer).setVisibility(View.VISIBLE);
115         findViewById(R.id.routeCheckpointMapContainer).setVisibility(View.VISIBLE);
116         findViewById(R.id.routeCheckpointListContainer).setVisibility(View.GONE);
117
118         // prepare list view of visited checkpoints
119         // ListView is not used because of outer wrapping ScrollView
120         LinearLayout listViewReplacement = (LinearLayout)
            ↪  findViewById(R.id.routeCheckpointListView);
```

```
121         listViewReplacement.removeAllViews();
122         listViewReplacement.setVisibility(View.VISIBLE);
123         for (int i = 0; i < viewModel.getVisitedCheckpoints().size(); i++) {
124             CheckpointModel model = viewModel.getRoute().getVisitedCheckpoints().get(i);
125             View view = View.inflate(CheckpointActivity.this, R.layout.view_checkpoint_item,
                ↪  null);
126             ((TextView) view.findViewById(R.id.checkpointItemText)).setText(model.getName());
127             ((ImageView) view.findViewById(R.id.checkpointItemImage))
128                     .setImageResource(R.drawable.maps_marker_green_icon);
129             listViewReplacement.addView(view);
130         }
131         // Add next checkpoint to list with open marker
132         if (viewModel.getNextCheckpoint() != null) {
133             final CheckpointModel model = viewModel.getNextCheckpoint();
134             View view = View.inflate(CheckpointActivity.this, R.layout.view_checkpoint_item,
                ↪  null);
135             ((TextView) view.findViewById(R.id.checkpointItemText)).setText(
136                     model.getName()
137             );
138             ((ImageView) view.findViewById(R.id.checkpointItemImage))
139                     .setImageResource(R.drawable.maps_marker_red_icon);
140             listViewReplacement.addView(view);
141         }
142
143         // route already finished
144         if (viewModel.getNextCheckpoint() == null) {
145             findViewById(R.id.routeCheckpointContainer).setVisibility(View.GONE);
146             findViewById(R.id.routeCheckpointListContainer).setVisibility(View.VISIBLE);
147         }
148         // checkpoints still open
149         else {
150             Button answerButton = (Button) findViewById(R.id.answerBtn);
151             answerButton.setEnabled(Boolean.FALSE);
152             answerButton.setOnClickListener(this);
153
154             ((TextView) findViewById(R.id.nextCheckpointTxt))
155                     .setText(viewModel.getNextCheckpoint().getName());
156             ((TextView) findViewById(R.id.nextCheckpointHintTxt))
157                     .setText(viewModel.getNextCheckpoint().getHint());
158
159             EditText answer = (EditText) findViewById(R.id.hintAnswerEdTxt);
160             answer.setText("");
161             answer.addTextChangedListener(new CheckpointViewModelBindingTextWatcher(viewModel));
162             answer.addTextChangedListener(new AnswerButtonTextWatcher(this, viewModel));
163         }
164
165         // Prepare maps
166         MapFragment mapFragment = (MapFragment)
            ↪  getFragmentManager().findFragmentById(R.id.routeCheckpointMap);
167         mapFragment.getMapAsync(new OnMapReadyCallback() {
168             @Override
169             public void onMapReady(GoogleMap map) {
170                 // clear former set map properties
171                 map.clear();
172                 // enable ui functions
173                 map.getUiSettings().setZoomControlsEnabled(Boolean.TRUE);
174                 map.getUiSettings().setAllGesturesEnabled(Boolean.TRUE);
175                 map.getUiSettings().setCompassEnabled(Boolean.TRUE);
176                 LatLng zoomLocation = null;
177                 PolylineOptions lineOptions = null;
178                 // Get visited points
179                 if (!viewModel.getRoute().getVisitedCheckpoints().isEmpty()) {
180                     lineOptions = new PolylineOptions();
```

```
181                     for (int i = 0; i < viewModel.getRoute().getVisitedCheckpoints().size(); i++)
                         ↪  {
182                         CheckpointModel model =
                             ↪  viewModel.getRoute().getVisitedCheckpoints().get(i);
183                         LatLng location = new LatLng(model.getLatitude(), model.getLongitude());
184                         map.addMarker(new MarkerOptions().position(location)
185                                 .title((i + 1) + ". " + model.getName())
186                                 .icon(
187                                         BitmapDescriptorFactory.defaultMarker(
188                                                 BitmapDescriptorFactory.HUE_GREEN
189                                         )
190                                 )
191                         ).showInfoWindow();
192                         lineOptions.add(location);
193                         // Remember last for focusing on it
194                         if (i == (viewModel.getVisitedCheckpoints().size() - 1)) {
195                             zoomLocation = location;
196                         }
197                     }
198                     map.addPolyline(lineOptions.color(Color.GREEN));
199                 }
200                 // Add next checkpoint which is not part of list and gets red line from it to the
                     ↪  former marker
201                 if (viewModel.getRoute().getNextCheckpoint() != null) {
202                     zoomLocation = new
                         ↪  LatLng(viewModel.getRoute().getNextCheckpoint().getLatitude(),
                         ↪  viewModel.getRoute().getNextCheckpoint().getLongitude());
203                     map.addMarker(
204                             new MarkerOptions()
205                                     .position(zoomLocation)
206                                     .title(viewModel.getNextCheckpoint().getName())
207                                     .icon(BitmapDescriptorFactory
208                                             .defaultMarker(BitmapDescriptorFactory.HUE_RED))
209                     ).showInfoWindow();
210                     // If visited checkpoints exist
211                     if (!viewModel.getRoute().getVisitedCheckpoints().isEmpty()) {
212                         map.addPolyline(new PolylineOptions().add(
213                                 lineOptions.getPoints().get(lineOptions.getPoints().size()
                                     ↪  - 1), zoomLocation).color(Color.RED)
214                         );
215                     }
216                 }
217                 // move camera to last or next checkpoint
218                 map.moveCamera(CameraUpdateFactory.newLatLngZoom(zoomLocation, 17));
219             }
220         });
221     }
222     //endregion
223
224     //region Listeners
225
226     /**
227      * Toggles the visibility if one container header gets clicked
228      *
229      * @param view the clicked view
230      */
231     public void toggleVisibility(View view) {
232         View checkpointListContainer = findViewById(R.id.routeCheckpointListContainer);
233         View checkpointMapContainer = findViewById(R.id.routeCheckpointMapContainer);
234
235         // Always references to parent LinearLayout so not all clickable children need to be
             ↪  handled here
236         switch (((View) view.getParent()).getId()) {
```

```
237             case R.id.routeCheckpointListContainerLabel:
238                 checkpointListContainer.setVisibility((View.VISIBLE ==
                    ↪  checkpointListContainer.getVisibility()) ? View.GONE : View.VISIBLE);
239                 break;
240             case R.id.routeCheckpointMapContainerLabel:
241                 checkpointMapContainer.setVisibility((View.VISIBLE ==
                    ↪  checkpointMapContainer.getVisibility()) ? View.GONE : View.VISIBLE);
242                 break;
243         }
244     }
245
246     /**
247      * Handles the button clicks by redirecting to proper method.
248      *
249      * @param v the clicked view
250      */
251     @Override
252     public void onClick(View v) {
253         switch (v.getId()) {
254             case R.id.answerBtn:
255                 onAnswerButtonClick((Button) v);
256                 break;
257         }
258     }
259
260     @Override
261     public void onClick(DialogInterface dialog, int which) {
262         dialog.dismiss();
263     }
264     //endregion
265
266     /**
267      * Handles the answer button click
268      *
269      * @param button the clicked answer button
270      */
271     private void onAnswerButtonClick(Button button) {
272         if (checkAndDisplayAvailableNetwork()) {
273             new AsyncTask<Object, Object, AsyncTaskResult<Boolean>>() {
274                 @Override
275                 protected AsyncTaskResult<Boolean> doInBackground(Object... params) {
276                     Boolean result = null;
277                     Exception exception = null;
278                     try {
279                         result = viewModel.validateSecret();
280                     } catch (Exception e) {
281                         exception = e;
282                     }
283
284                     return new AsyncTaskResult<Boolean>(result, exception);
285                 }
286
287                 @Override
288                 protected void onPreExecute() {
289                     super.onPreExecute();
290                     openProgressDialog(R.string.progress_validating_secred);
291                 }
292
293                 @Override
294                 protected void onPostExecute(AsyncTaskResult<Boolean> result) {
295                     super.onPostExecute(result);
296                     closeProgressDialog();
297                     if (result.exception != null) {
```

```
298                         // ServiceException occurred
299                         if (result.exception instanceof ServiceException) {
300                             handleServiceException(((ServiceException) result.exception));
301                         } else {
302                             Toast.makeText(CheckpointActivity.this, R.string.error_unknown,
                                ↪  Toast.LENGTH_LONG).show();
303                         }
304                     } else if (result.result) {
305                         reloadRoute(Boolean.TRUE);
306                     } else {
307                         Toast.makeText(CheckpointActivity.this, R.string.info_secret_wrong,
                            ↪  Toast.LENGTH_LONG).show();
308                     }
309                 }
310             }.execute();
311         }
312     }
313
314     /**
315      * Reloads the route
316      */
317     private void reloadRoute(final boolean checkIfFinished) {
318         if (checkAndDisplayAvailableNetwork()) {
319             new AsyncTask<Object, Object, AsyncTaskResult<Boolean>>() {
320                 @Override
321                 protected AsyncTaskResult<Boolean> doInBackground(Object... params) {
322                     Boolean found = null;
323                     Exception exception = null;
324                     try {
325                         found = viewModel.reloadRoute();
326                     } catch (Exception e) {
327                         exception = e;
328                     }
329                     return new AsyncTaskResult<Boolean>(found, exception);
330                 }
331
332                 @Override
333                 protected void onPostExecute(AsyncTaskResult<Boolean> result) {
334                     super.onPostExecute(result);
335                     closeProgressDialog();
336                     // Exception occurred
337                     if (result.exception != null) {
338                         if (result.exception instanceof ServiceException) {
339                             handleServiceException(((ServiceException) result.exception));
340                         } else {
341                             Toast.makeText(CheckpointActivity.this, R.string.error_unknown,
                                ↪  Toast.LENGTH_SHORT).show();
342                         }
343                     }
344                     // Route found
345                     else if (result.result) {
346                         prepareView();
347                         if (checkIfFinished) {
348                             if (viewModel.getNextCheckpoint() != null) {
349                                 Toast.makeText(CheckpointActivity.this, R.string.info_secret_ok,
                                    ↪  Toast.LENGTH_LONG).show();
350                             } else {
351                                 DialogUtil.createCustomContentDialog(CheckpointActivity.this,
                                    ↪  R.layout.view_congratulations_dialog,
                                    ↪  R.string.dialog_title_info, CheckpointActivity.this).show();
352                             }
353                         }
354                     }
```

```
355                     // Route not found, go back to routes activity
356                     else {
357                         startActivity(new Intent(CheckpointActivity.this, RouteActivity.class));
358                         finish();
359                     }
360                 }
361 
362                 @Override
363                 protected void onPreExecute() {
364                     super.onPreExecute();
365                     openProgressDialog(R.string.progress_updating_route);
366                 }
367             }.execute();
368         }
369     }
370 
371     /**
372      * Resets the current route.
373      */
374     private void resetRoute() {
375         if (checkAndDisplayAvailableNetwork()) {
376             new AsyncTask<Object, Object, AsyncTaskResult<Boolean>>() {
377                 @Override
378                 protected AsyncTaskResult<Boolean> doInBackground(Object... params) {
379                     Boolean result = null;
380                     Exception exception = null;
381                     try {
382                         result = viewModel.resetRoute();
383                     } catch (Exception e) {
384                         exception = e;
385                     }
386 
387                     return new AsyncTaskResult<Boolean>(result, exception);
388                 }
389 
390                 @Override
391                 protected void onPreExecute() {
392                     super.onPreExecute();
393                     openProgressDialog(R.string.progress_resetting_routes);
394                 }
395 
396                 @Override
397                 protected void onPostExecute(AsyncTaskResult<Boolean> result) {
398                     super.onPostExecute(result);
399                     closeProgressDialog();
400                     // Error occurred
401                     if (result.exception != null) {
402                         // ServiceException occurred
403                         if (result.exception instanceof ServiceException) {
404                             handleServiceException(((ServiceException) result.exception));
405                         } else {
406                             Toast.makeText(CheckpointActivity.this, R.string.error_unknown,
                                 ↪  Toast.LENGTH_LONG).show();
407                         }
408                     }
409                     // Reset failed on rest method
410                     else if (!result.result) {
411                         Toast.makeText(CheckpointActivity.this, R.string.error_route_reset_failed,
                             ↪  Toast.LENGTH_LONG).show();
412                     }
413                     // Reset ok
414                     else {
415                         reloadRoute(Boolean.FALSE);
```

```
416                     }
417                 }
418             }.execute();
419         }
420     }
421     //endregion
422 }
```

1.2.3 Adapter

Listing 7: RouteArrayAdapter.java

```java
package at.fh.ooe.moc5.amazingrace.adapter;

import android.content.Context;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;

import at.fh.ooe.moc5.amazingrace.R;
import at.fh.ooe.moc5.amazingrace.model.json.RouteModel;

/**
 * Created by Thomas on 12/25/2015.
 * The adapter for the route items
 */
public class RouteArrayAdapter extends ArrayAdapter<RouteModel> {

    public RouteArrayAdapter(Context ctx) {
        super(ctx, 0);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        if (convertView == null) {
            convertView = View.inflate(getContext(), R.layout.view_route_item, null);
        }

        RouteModel route = getItem(position);
        ImageView icon = ((ImageView) convertView.findViewById(R.id.routeItemImage));
        if (route.getNextCheckpoint() == null) {
            icon.setImageResource(R.drawable.done_icon);
        } else {
            icon.setImageResource(R.drawable.app_icon);
        }
        ((TextView)
         ↪  convertView.findViewById(R.id.routeItemText)).setText(route.toItemString(getContext().getString(R.s

        return convertView;
    }
}
```

1.2.4 JSON-Models

Listing 8: CheckpointModel.java

```java
package at.fh.ooe.moc5.amazingrace.model.json;

import com.google.gson.annotations.SerializedName;

import java.io.Serializable;
import java.util.Objects;

/**
 * Created by Thomas on 12/25/2015.
 */
public class CheckpointModel implements Serializable {

    @SerializedName("Id")
    private String id;
    @SerializedName("Number")
    private int number;
    @SerializedName("Name")
    private String name;
    @SerializedName("Hint")
    private String hint;
    @SerializedName("Latitude")
    private double latitude;
    @SerializedName("Longitude")
    private double longitude;
    private transient boolean unvisited = Boolean.FALSE;

    public CheckpointModel() {
    }

    //region Getter and Setter
    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public int getNumber() {
        return number;
    }

    public void setNumber(int number) {
        this.number = number;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getHint() {
        return hint;
    }

    public void setHint(String hint) {
```

```
60          this.hint = hint;
61      }
62
63      public double getLatitude() {
64          return latitude;
65      }
66
67      public void setLatitude(double latitude) {
68          this.latitude = latitude;
69      }
70
71      public double getLongitude() {
72          return longitude;
73      }
74
75      public void setLongitude(double longitude) {
76          this.longitude = longitude;
77      }
78
79      public boolean isUnvisited() {
80          return unvisited;
81      }
82
83      public void setUnvisited(boolean unvisited) {
84          this.unvisited = unvisited;
85      }
86      //endregion
87
88      @Override
89      public boolean equals(Object o) {
90          if (this == o) return true;
91          if (o == null || getClass() != o.getClass()) return false;
92          CheckpointModel that = (CheckpointModel) o;
93          return Objects.equals(id, that.id);
94      }
95
96      @Override
97      public int hashCode() {
98          return Objects.hash(id);
99      }
100 }
```

Listing 9: CheckpointRequestModel.java

```java
package at.fh.ooe.moc5.amazingrace.model.json;

import com.google.gson.annotations.SerializedName;

import java.io.Serializable;
import java.util.Objects;

/**
 * Created by Thomas on 12/25/2015.
 */
public class CheckpointRequestModel extends CredentialsRequestModel implements Serializable {

    @SerializedName("CheckpointId")
    public final String checkpointId;
    @SerializedName("Secret")
    public final String secret;

    public CheckpointRequestModel(String username, String password, String checkpointId, String
     ↪  secret) {
        super(username, password);
        this.checkpointId = checkpointId;
        this.secret = secret;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        if (!super.equals(o)) return false;
        CheckpointRequestModel that = (CheckpointRequestModel) o;
        return Objects.equals(checkpointId, that.checkpointId);
    }

    @Override
    public int hashCode() {
        return Objects.hash(super.hashCode(), checkpointId);
    }
}
```

Listing 10: CredentialsRequestModel.java

```java
package at.fh.ooe.moc5.amazingrace.model.json;

import com.google.gson.annotations.SerializedName;

import java.io.Serializable;
import java.util.Objects;

/**
 * Created by Thomas on 12/25/2015.
 */
public class CredentialsRequestModel implements Serializable {

    @SerializedName("UserName")
    public final String userName;
    @SerializedName("Password")
    public final String password;

    public CredentialsRequestModel(String userName, String password) {
        this.userName = userName;
        this.password = password;
    }

    public String toQueryString() {
        return String.format("userName=%s&password=%s", userName, password);
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        CredentialsRequestModel that = (CredentialsRequestModel) o;
        return Objects.equals(userName, that.userName);
    }

    @Override
    public int hashCode() {
        return Objects.hash(userName);
    }
}
```

Listing 11: RouteModel.java

```java
package at.fh.ooe.moc5.amazingrace.model.json;

import com.google.gson.annotations.SerializedName;

import java.io.Serializable;
import java.util.Comparator;
import java.util.List;
import java.util.Objects;

/**
 * Created by Thomas on 12/25/2015.
 */
public class RouteModel implements Serializable {

    @SerializedName("Id")
    private String id;
    @SerializedName("Name")
    private String name;
    @SerializedName("VisitedCheckpoints")
    private List<CheckpointModel> visitedCheckpoints;
    @SerializedName("NextCheckpoint")
    private CheckpointModel nextCheckpoint;

    public RouteModel() {
    }

    //region Getter and Setter
    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public List<CheckpointModel> getVisitedCheckpoints() {
        return visitedCheckpoints;
    }

    public void setVisitedCheckpoints(List<CheckpointModel> visitedCheckpoints) {
        this.visitedCheckpoints = visitedCheckpoints;
    }

    public CheckpointModel getNextCheckpoint() {
        return nextCheckpoint;
    }

    public void setNextCheckpoint(CheckpointModel nextCheckpoint) {
        this.nextCheckpoint = nextCheckpoint;
    }
    //endregion

```

```
61      //region Utilities
62      public boolean isDone() {
63          return nextCheckpoint == null;
64      }
65      //endregion
66
67      public String toItemString(final String doneString) {
68          final StringBuilder sb = new StringBuilder(name);
69          if (!isDone()) {
70              Objects.requireNonNull(doneString);
71              sb.append(" (");
72              sb.append(visitedCheckpoints.size()).append("x").append(doneString);
73              sb.append(")");
74          }
75          return sb.toString();
76      }
77
78      @Override
79      public boolean equals(Object o) {
80          if (this == o) return true;
81          if (o == null || getClass() != o.getClass()) return false;
82          RouteModel that = (RouteModel) o;
83          return Objects.equals(id, that.id);
84      }
85
86      @Override
87      public int hashCode() {
88          return Objects.hash(id);
89      }
90  }
```

Listing 12: RouteRequestModel.java

```java
package at.fh.ooe.moc5.amazingrace.model.json;

import com.google.gson.annotations.SerializedName;

import java.io.Serializable;
import java.util.Objects;

/**
 * Created by Thomas on 12/25/2015.
 */
public class RouteRequestModel extends CredentialsRequestModel {

    @SerializedName("RouteId")
    public final String routeId;

    public RouteRequestModel(CredentialsRequestModel model, String routeId) {
        this(model.userName, model.password, routeId);
    }

    public RouteRequestModel(String username, String password, String routeId) {
        super(username, password);
        this.routeId = routeId;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        if (!super.equals(o)) return false;
        RouteRequestModel that = (RouteRequestModel) o;
        return Objects.equals(routeId, that.routeId);
    }

    @Override
    public int hashCode() {
        return Objects.hash(super.hashCode(), routeId);
    }
}
```

1.2.5 Task-Models

Listing 13: AsyncTaskResult.java

```java
package at.fh.ooe.moc5.amazingrace.model.task;

/**
 * Created by Thomas on 12/25/2015.
 * This class represents an async task result which allows handling of any occurred exception
 * in the postExecute method instead of the doInBackground where we are not in the UI Thread.
 */
public class AsyncTaskResult<T> {

    public final T result;
    public final Exception exception;

    public AsyncTaskResult(T result, Exception exception) {
        this.result = result;
        this.exception = exception;
    }
}
```

1.2.6 View-Models

Listing 14: LoginViewModel.java

```java
package at.fh.ooe.moc5.amazingrace.model.view;

import org.apache.commons.lang3.StringUtils;

import java.io.Serializable;

import at.fh.ooe.moc5.amazingrace.model.json.CredentialsRequestModel;
import at.fh.ooe.moc5.amazingrace.service.ServiceProxy;
import at.fh.ooe.moc5.amazingrace.service.ServiceException;
import at.fh.ooe.moc5.amazingrace.service.ServiceProxyFactory;

/**
 * Created by Thomas on 12/24/2015.
 */
public class LoginViewModel implements Serializable {

    private String username;
    private String password;

    private final ServiceProxy restProxy;

    public LoginViewModel() {
        restProxy = ServiceProxyFactory.createServiceProxy();
    }

    /**
     * Ansers the question if this model is valid whic its is if a username and password are set.
     *
     * @return true if valid, false otherwise
     */
    public boolean isValid() {
        return (((username != null) && (!StringUtils.isEmpty(username))) && ((password != null) &&
        ↪   (!StringUtils.isEmpty(password))));
    }


    //region Actions

    /**
     * Logs the user in by validating the user credentials via the proxy service.
     *
     * @return true if username password are valid, false otherwise
     * @throws ServiceException if an rpoxy service exception occurred
     */
    public UserContextModel loginAction() throws ServiceException {
        final CredentialsRequestModel model = new CredentialsRequestModel(username, password);
        final boolean isValidCredentials = restProxy.validateCredentials(model);
        if (isValidCredentials) {
            return new UserContextModel(model);
        }
        return null;
    }

    /**
     * Resets this model by resetting the username and password
     */
    public void reset() {
        username = null;
        password = null;
```

```
59      }
60
61      //endregion
62      //region Getter and Setter
63      public void setUsername(String username) {
64          this.username = username;
65      }
66
67      public String getUsername() {
68          return username;
69      }
70
71      public void setPassword(String password) {
72          this.password = password;
73      }
74
75      public String getPassword() {
76          return password;
77      }
78      //endregion
79  }
```

Listing 15: RoutesViewModel.java

```
1  package at.fh.ooe.moc5.amazingrace.model.view;
2
3  import java.io.Serializable;
4  import java.util.Collections;
5  import java.util.Comparator;
6  import java.util.List;
7  import java.util.Objects;
8
9  import at.fh.ooe.moc5.amazingrace.model.json.RouteModel;
10 import at.fh.ooe.moc5.amazingrace.model.json.RouteRequestModel;
11 import at.fh.ooe.moc5.amazingrace.service.ServiceProxy;
12 import at.fh.ooe.moc5.amazingrace.service.ServiceProxyFactory;
13 import at.fh.ooe.moc5.amazingrace.service.ServiceException;
14
15 /**
16  * Created by Thomas on 12/25/2015.
17  */
18 public class RoutesViewModel implements Serializable {
19
20     private UserContextModel userContext;
21     private RouteModel selectedRoute;
22     private List<RouteModel> routes;
23
24     private ServiceProxy proxy;
25
26     public RoutesViewModel(UserContextModel userContext) {
27         Objects.requireNonNull(userContext, "UserContext must not be null");
28         this.userContext = userContext;
29         proxy = ServiceProxyFactory.createServiceProxy();
30     }
31
32     //region Actions
33
34     /**
35      * Loads the routes
36      *
37      * @return the found routes
38      * @throws ServiceException if the proxy service throw an exception
39      */
40     public List<RouteModel> loadRoutes() throws ServiceException {
41         routes = proxy.loadRoutes(userContext.getCredentialsModel());
42         Collections.sort(routes, new Comparator<RouteModel>() {
43             @Override
44             public int compare(RouteModel lhs, RouteModel rhs) {
45                 if (lhs.isDone() == rhs.isDone()) {
46                     if (lhs.getVisitedCheckpoints().size() == rhs.getVisitedCheckpoints().size())
                        ↪  {
47                         return lhs.getName().toUpperCase().compareTo(rhs.getName().toUpperCase());
48                     } else {
49                         return
                            ↪  Integer.valueOf(rhs.getVisitedCheckpoints().size()).compareTo(lhs.getVisitedCheckpo
50                     }
51                 } else {
52                     return Boolean.valueOf(lhs.isDone()).compareTo(rhs.isDone());
53                 }
54             }
55         });
56         return routes;
57     }
58
```

```
59      /**
60       * Resets the given route
61       *
62       * @param model the route to reset
63       * @return true if reset, false otherwise
64       * @throws ServiceException if the proxy service throw an exception
65       */
66      public boolean resetRoute(RouteModel model) throws ServiceException {
67          return proxy.resetRoute(new RouteRequestModel(userContext.getCredentialsModel(),
            ↪  model.getId()));
68      }
69
70      /**
71       * Resets all routes
72       *
73       * @return true if reset, false otherwise
74       * @throws ServiceException if the proxy service throw an exception
75       */
76      public boolean resetAllRoutes() throws ServiceException {
77          return proxy.resetAllRoutes(userContext.getCredentialsModel());
78      }
79      //endregion
80
81      //region Getter and Setter
82      public RouteModel getSelectedRoute() {
83          return selectedRoute;
84      }
85
86      public void setSelectedRoute(RouteModel selectedRoute) {
87          this.selectedRoute = selectedRoute;
88      }
89
90      public List<RouteModel> getRoutes() {
91          return routes;
92      }
93      //endregion
94  }
```

Listing 16: UserContextModel.java

```java
package at.fh.ooe.moc5.amazingrace.model.view;

import java.io.Serializable;
import java.util.Calendar;
import java.util.Objects;

import at.fh.ooe.moc5.amazingrace.model.json.CredentialsRequestModel;

/**
 * Created by Thomas on 12/24/2015.
 * This model represents an logged user for the application.
 */
public class UserContextModel implements Serializable {

    private final CredentialsRequestModel model;
    private final Calendar loginDate;

    public UserContextModel(CredentialsRequestModel model) {
        Objects.requireNonNull(model);
        this.model = model;
        this.loginDate = Calendar.getInstance();
    }

    //region Getter and Setter
    public CredentialsRequestModel getCredentialsModel() {
        return model;
    }

    public Calendar getLoginDate() {
        return loginDate;
    }
    //endregion
}
```

Listing 17: CheckpointViewModel.java

```java
package at.fh.ooe.moc5.amazingrace.model.view;

import java.io.Serializable;
import java.util.List;
import java.util.Objects;

import at.fh.ooe.moc5.amazingrace.model.json.CheckpointModel;
import at.fh.ooe.moc5.amazingrace.model.json.RouteModel;
import at.fh.ooe.moc5.amazingrace.model.json.RouteRequestModel;
import at.fh.ooe.moc5.amazingrace.model.json.CheckpointRequestModel;
import at.fh.ooe.moc5.amazingrace.service.ServiceProxy;
import at.fh.ooe.moc5.amazingrace.service.ServiceException;
import at.fh.ooe.moc5.amazingrace.service.ServiceProxyFactory;

/**
 * Created by Thomas on 12/25/2015.
 */
public class CheckpointViewModel implements Serializable {

    private String answer;
    private final UserContextModel userContext;
    private RouteModel route;

    private ServiceProxy proxy;

    public CheckpointViewModel(UserContextModel userContext, RouteModel route) {
        Objects.requireNonNull(userContext, "View model needs user context set");
        Objects.requireNonNull(route, "View model needs route set");

        this.userContext = userContext;
        this.route = route;
        proxy = ServiceProxyFactory.createServiceProxy();
    }

    //region Actions
    public boolean validateSecret() throws ServiceException {
        if (!isValid()) {
            return Boolean.FALSE;
        }

        return proxy.validateCheckpointSecret(new
         ↪  CheckpointRequestModel(userContext.getCredentialsModel().userName,
         ↪  userContext.getCredentialsModel().password, getNextCheckpoint().getId(), answer));
    }

    public boolean reloadRoute() throws ServiceException {
        final List<RouteModel> routes = proxy.loadRoutes(userContext.getCredentialsModel());
        int idx = -1;
        if ((idx = routes.indexOf(route)) != -1) {
            route = routes.get(idx);
        }

        return (route != null);
    }

    public boolean resetRoute() throws ServiceException {
        return proxy.resetRoute(new RouteRequestModel(userContext.getCredentialsModel(),
         ↪  route.getId()));
    }

```

```
58      public boolean isValid() {
59          return (answer != null) && (!answer.trim().isEmpty());
60      }
61      //endregion
62
63      //region Getter and Setter
64      public String getAnswer() {
65          return answer;
66      }
67
68      public void setAnswer(String answer) {
69          this.answer = answer;
70      }
71
72      public UserContextModel getUserContext() {
73          return userContext;
74      }
75
76      public RouteModel getRoute() {
77          return route;
78      }
79
80      public CheckpointModel getNextCheckpoint() {
81          return route.getNextCheckpoint();
82      }
83
84      public List<CheckpointModel> getVisitedCheckpoints() {
85          return route.getVisitedCheckpoints();
86      }
87      //endregion
88  }
```

1.2.7 Service

Listing 18: ServiceProxy.java

```
1  package at.fh.ooe.moc5.amazingrace.service;
2
3  import java.io.Serializable;
4  import java.util.List;
5
6  import at.fh.ooe.moc5.amazingrace.model.json.CredentialsRequestModel;
7  import at.fh.ooe.moc5.amazingrace.model.json.RouteModel;
8  import at.fh.ooe.moc5.amazingrace.model.json.RouteRequestModel;
9  import at.fh.ooe.moc5.amazingrace.model.json.CheckpointRequestModel;
10
11 /**
12  * Created by Thomas on 12/25/2015.
13  * The specification of the service proxy
14  */
15 public interface ServiceProxy extends Serializable {
16
17     /**
18      * Validates the given user credentials.
19      *
20      * @param model the model containing authentication data
21      * @return true if the credentials are valid, false otherwise
22      * @throws ServiceException if an service error occurs. See contained error code for details
23      */
24     boolean validateCredentials(CredentialsRequestModel model) throws ServiceException;
25
26     /**
27      * Validates the checkpoint secret.
28      *
29      * @param model the model containing the checkpoint data and authentication data
30      * @return true if valid, false otherwise
31      * @throws ServiceException if an service error occurs. See contained error code for details
32      */
33     boolean validateCheckpointSecret(CheckpointRequestModel model) throws ServiceException;
34
35     /**
36      * Loads all routes.
37      *
38      * @param model the credentials model for authentication
39      * @return the loaded routes
40      * @throws ServiceException if an service error occurs. See contained error code for details
41      */
42     List<RouteModel> loadRoutes(CredentialsRequestModel model) throws ServiceException;
43
44     /**
45      * Resets the given route.
46      *
47      * @param model the model containing the route data and authentication data
48      * @return true if reset, false otherwise
49      * @throws ServiceException if an service error occurs. See contained error code for details
50      */
51     boolean resetRoute(RouteRequestModel model) throws ServiceException;
52
53     /**
54      * Resets all routes.
55      *
56      * @param model the model containing the authentication data
57      * @return true if reset, false otherwise
58      * @throws ServiceException if an service error occurs. See contained error code for details
59      */
```

```
60      boolean resetAllRoutes(CredentialsRequestModel model) throws ServiceException;
61  }
```

Listing 19: RestServiceProxyImpl.java

```java
package at.fh.ooe.moc5.amazingrace.service;

import com.google.gson.GsonBuilder;
import com.google.gson.JsonSyntaxException;
import com.google.gson.reflect.TypeToken;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.SocketTimeoutException;
import java.net.URL;
import java.util.List;
import java.util.Objects;

import javax.net.ssl.HttpsURLConnection;

import at.fh.ooe.moc5.amazingrace.model.json.CredentialsRequestModel;
import at.fh.ooe.moc5.amazingrace.model.json.RouteModel;
import at.fh.ooe.moc5.amazingrace.model.json.RouteRequestModel;
import at.fh.ooe.moc5.amazingrace.model.json.CheckpointRequestModel;
import at.fh.ooe.moc5.amazingrace.service.ServiceException.ServiceErrorCode;

/**
 * Created by Thomas on 12/24/2015.
 * <p/>
 * Class for communicating with the rest backend.
 */
public class RestServiceProxyImpl implements ServiceProxy {

    public static final String REST_URL =
     ↪  "https://demo.nexperts.com/MOC5/AmazingRaceService/AmazingRaceService.svc";
    public static final String CHECK_CREDENTIALS = "/CheckCredentials";
    public static final String GET_ROUTES_METHOD = "/GetRoutes";
    public static final String METHOD_VISIT_CHECKPOINT = "/InformAboutVisitedCheckpoint";
    public static final String METHOD_RESET_ALL_ROUTES = "/ResetAllRoutes";
    public static final String METHOD_RESET_ROUTE = "/ResetRoute";

    public static final int DEFAULT_TIME_OUT = 3000;

    @Override
    public boolean validateCredentials(CredentialsRequestModel model) throws ServiceException {
        Objects.requireNonNull(model, "Cannot check credentials for null model");

        try {
            URL url = new URL(new
             ↪  StringBuilder(REST_URL).append(CHECK_CREDENTIALS).append("?").append(model.toQueryString()).toS
            HttpsURLConnection connection = (HttpsURLConnection) url.openConnection();
            connection.setRequestMethod("GET");
            connection.setConnectTimeout(DEFAULT_TIME_OUT);
            connection.setUseCaches(Boolean.FALSE);

            return invokeBooleanResultMethod(connection, null, Boolean.FALSE);
        } catch (ServiceException e) {
            throw e;
        } catch (Exception e) {
            throw new ServiceException(ServiceErrorCode.UNKNOWN, e);
        }
    }

```

```
59      @Override
60      public boolean validateCheckpointSecret(CheckpointRequestModel model) throws ServiceException
        ↪  {
61          Objects.requireNonNull(model, "Cannot check checkpoint for null model");
62
63          try {
64              URL url = new URL(new
                ↪  StringBuilder(REST_URL).append(METHOD_VISIT_CHECKPOINT).toString());
65              HttpsURLConnection connection = (HttpsURLConnection) url.openConnection();
66              connection.setRequestMethod("POST");
67              connection.setRequestProperty("Content-Type", "application/json");
68              connection.setConnectTimeout(DEFAULT_TIME_OUT);
69              connection.setUseCaches(Boolean.FALSE);
70              connection.setDoInput(Boolean.TRUE);
71              connection.setDoOutput(Boolean.TRUE);
72
73              return invokeBooleanResultMethod(connection, model, Boolean.TRUE);
74          } catch (ServiceException e) {
75              throw e;
76          } catch (Exception e) {
77              throw new ServiceException(ServiceErrorCode.UNKNOWN, e);
78          }
79      }
80
81      @Override
82      public List<RouteModel> loadRoutes(CredentialsRequestModel model) throws ServiceException {
83          Objects.requireNonNull(model, "Cannot get routes with missing credentials");
84
85          if (!validateCredentials(model)) {
86              throw new ServiceException(ServiceErrorCode.INVALID_CREDENTIALS);
87          }
88
89          try {
90              URL url = new URL(new
                ↪  StringBuilder(REST_URL).append(GET_ROUTES_METHOD).append("?").append(model.toQueryString()).toS
91              HttpsURLConnection connection = (HttpsURLConnection) url.openConnection();
92              connection.setRequestMethod("GET");
93              connection.setConnectTimeout(DEFAULT_TIME_OUT);
94              connection.setUseCaches(Boolean.FALSE);
95
96              final String response = readResponse(connection);
97              GsonBuilder builder = new GsonBuilder();
98              return builder.create().fromJson(response, new TypeToken<List<RouteModel>>() {
99              }.getType());
100         } catch (JsonSyntaxException e) {
101             throw new ServiceException(ServiceErrorCode.INVALID_REQUEST, e);
102         } catch (Exception e) {
103             throw new ServiceException(ServiceErrorCode.UNKNOWN, e);
104         }
105     }
106
107     @Override
108     public boolean resetRoute(RouteRequestModel model) throws ServiceException {
109         Objects.requireNonNull(model, "Cannot reset route for null model");
110
111         try {
112             URL url = new URL(new StringBuilder(REST_URL).append(METHOD_RESET_ROUTE).toString());
113             HttpsURLConnection connection = (HttpsURLConnection) url.openConnection();
114             connection.setRequestMethod("POST");
115             connection.setRequestProperty("Content-Type", "application/json");
116             connection.setConnectTimeout(DEFAULT_TIME_OUT);
117             connection.setUseCaches(Boolean.FALSE);
118             connection.setDoInput(Boolean.TRUE);
```

```
119             connection.setDoOutput(Boolean.TRUE);
120
121             return invokeBooleanResultMethod(connection, model, Boolean.TRUE);
122         } catch (ServiceException e) {
123             throw e;
124         } catch (Exception e) {
125             throw new ServiceException(ServiceErrorCode.UNKNOWN, e);
126         }
127     }
128
129     @Override
130     public boolean resetAllRoutes(CredentialsRequestModel model) throws ServiceException {
131         Objects.requireNonNull(model, "Cannot reset all routes for null model");
132
133         try {
134             URL url = new URL(new
                ↪  StringBuilder(REST_URL).append(METHOD_RESET_ALL_ROUTES).toString());
135             HttpsURLConnection connection = (HttpsURLConnection) url.openConnection();
136             connection.setRequestMethod("POST");
137             connection.setRequestProperty("Content-Type", "application/json");
138             connection.setConnectTimeout(DEFAULT_TIME_OUT);
139             connection.setUseCaches(Boolean.FALSE);
140             connection.setDoInput(Boolean.TRUE);
141             connection.setDoOutput(Boolean.TRUE);
142
143             return invokeBooleanResultMethod(connection, model, Boolean.TRUE);
144         } catch (ServiceException e) {
145             throw e;
146         } catch (Exception e) {
147             throw new ServiceException(ServiceErrorCode.UNKNOWN, e);
148         }
149     }
150
151     // region Helper
152
153     /**
154      * Invokes a remote service method with an boolean result.
155      *
156      * @param connection       the connection to the service method
157      * @param jsonModel        the jsonModel to write into the connection, maybe null.
158      * @param checkCredentials true if the credentials shall be checked befor calling the actual
    ↪   service
159      * @param <T>              the type of the json model which must be at least the
    ↪   CredentialsModel
160      * @return true or false
161      * @throws ServiceException if the request failed for any reason, see contained error code for
    ↪   details
162      */
163     private <T extends CredentialsRequestModel> boolean
         ↪  invokeBooleanResultMethod(HttpsURLConnection connection, T jsonModel, boolean
         ↪  checkCredentials) throws ServiceException {
164         Objects.requireNonNull(connection, "Cannot invoke method on null connection");
165         if (checkCredentials) {
166             Objects.requireNonNull(jsonModel, "Cannot check credentials for null model");
167         }
168         if ((checkCredentials) && (!validateCredentials(new
             ↪  CredentialsRequestModel(jsonModel.userName, jsonModel.password)))) {
169             throw new ServiceException(ServiceErrorCode.INVALID_CREDENTIALS);
170         }
171
172         try {
173             // Write Data for Post requests
174             if ((connection.getRequestMethod().equals("POST")) && (jsonModel != null)) {
```

```
175                 writeData(connection, jsonModel);
176             }
177             // Read response
178             final String response = readResponse(connection);
179             if ((!"true".equals(response)) && (!"false".equals(response))) {
180                 throw new ServiceException(ServiceErrorCode.INVALID_REQUEST);
181             }
182             return Boolean.parseBoolean(response);
183         } catch (ServiceException e) {
184             throw e;
185         } catch (Exception e) {
186             throw new ServiceException(ServiceErrorCode.UNKNOWN, e);
187         }
188     }
189 
190     /**
191      * Reads the response from the connection.
192      *
193      * @param connection the connection to read response from
194      * @return the response string
195      * @throws ServiceException if an error occurs
196      * @see ServiceErrorCode for the ServiceException contained error code
197      */
198     private String readResponse(HttpsURLConnection connection) throws ServiceException {
199         Objects.requireNonNull(connection, "Cannot read from null connection");
200         try {
201             if (connection.getResponseCode() != 200) {
202                 throw new ServiceException(ServiceErrorCode.REQUEST_NOT_OK);
203             }
204             try (final BufferedReader reader = new BufferedReader(new
                 ↪  InputStreamReader(connection.getInputStream()))) {
205                 final StringBuilder builder = new StringBuilder();
206                 String line;
207                 while ((line = reader.readLine()) != null) {
208                     builder.append(line);
209                 }
210                 return builder.toString();
211             }
212         } catch (SocketTimeoutException ste) {
213             throw new ServiceException(ServiceErrorCode.TIMEOUT, ste);
214         } catch (Exception e) {
215             throw new ServiceException(ServiceErrorCode.UNKNOWN, e);
216         }
217     }
218 
219     /**
220      * Writes data to the connection.
221      *
222      * @param connection        the connection to write to
223      * @param jsonModelInstance the json model instance
224      * @throws ServiceException if the write fails, see contained error code for details
225      */
226     private void writeData(HttpsURLConnection connection, Object jsonModelInstance) throws
         ↪  ServiceException {
227         Objects.requireNonNull(connection, "Cannot write to null connection");
228         try {
229             try (final BufferedWriter writer = new BufferedWriter(new
                 ↪  OutputStreamWriter(connection.getOutputStream()))) {
230                 final String data = new GsonBuilder().create().toJson(jsonModelInstance);
231                 writer.write(data, 0, data.length());
232                 writer.flush();
233             }
234         } catch (JsonSyntaxException jse) {
```

```
235             throw new ServiceException(ServiceErrorCode.INVALID_REQUEST, jse);
236         } catch (SocketTimeoutException ste) {
237             throw new ServiceException(ServiceErrorCode.TIMEOUT, ste);
238         } catch (Exception e) {
239             throw new ServiceException(ServiceErrorCode.UNKNOWN, e);
240         }
241     }
242     // endregion
243 }
```

Listing 20: ServiceException.java

```java
package at.fh.ooe.moc5.amazingrace.service;

/**
 * Created by Thomas on 12/25/2015.
 */
public class ServiceException extends Exception {
    /**
     * Enumeration which defines the common service errors
     */
    public static enum ServiceErrorCode {
        INVALID_REQUEST,
        TIMEOUT,
        UNKNOWN,
        REQUEST_NOT_OK,
        INVALID_CREDENTIALS
    }

    private ServiceErrorCode errorCode;

    public ServiceException() {
        this(null, null, null);
    }

    public ServiceException(String detailMessage) {
        this(null, detailMessage, null);
    }

    public ServiceException(String detailMessage, Throwable throwable) {
        this(null, detailMessage, throwable);
    }

    public ServiceException(Throwable throwable) {
        this(null, null, throwable);
    }

    public ServiceException(ServiceErrorCode errorCode) {
        this(errorCode, null, null);
    }

    public ServiceException(ServiceErrorCode errorCode, String detailMessage) {
        this(errorCode, detailMessage, null);
    }


    public ServiceException(ServiceErrorCode errorCode, Throwable throwable) {
        this(errorCode, null, throwable);
    }

    public ServiceException(ServiceErrorCode errorCode, String detailMessage, Throwable throwable)
    ↪  {
        super(detailMessage, throwable);
        this.errorCode = errorCode;
    }

    public ServiceErrorCode getErrorCode() {
        return errorCode;
    }
}
```

Listing 21: ServiceProxyFactory.java

```java
package at.fh.ooe.moc5.amazingrace.service;

/**
 * Created by Thomas on 12/24/2015.
 */
public class ServiceProxyFactory {

    private static ServiceProxy proxy;

    private ServiceProxyFactory() {
    }

    public static ServiceProxy createServiceProxy() {
        if (proxy == null) {
            proxy = new RestServiceProxyImpl();
        }
        return proxy;
    }
}
```

1.2.8 Utilities

Listing 22: DialogUtil.java

```java
package at.fh.ooe.moc5.amazingrace.util;

import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;

import java.util.Objects;

import at.fh.ooe.moc5.amazingrace.R;

/**
 * Created by Thomas on 12/24/2015.
 * Utility for creating dialogs.
 */
public class DialogUtil {

    /**
     * Creates a error alert dialog.
     *
     * @param ctx      the context to create the alert dialog for
     * @param message  the resource id of the alert dialog error message
     * @param listener the button listener
     * @return the created alert dialog
     */
    public static AlertDialog createErrorDialog(final Context ctx, final int message, final
     ↪  DialogInterface.OnClickListener listener) {
        Objects.requireNonNull(ctx, "Cannot create dialog for null context");
        Objects.requireNonNull(listener, "Listener for buttons mut be given");

        return new AlertDialog.Builder(ctx)
                .setTitle(R.string.dialog_title_error)
                .setMessage(message)
                .setPositiveButton(R.string.action_ok, listener)
                .create();
    }

    /**
     * Creates a yes, no alert dialog
     *
     * @param ctx      the context to create the alert dialog for
     * @param title    the resource id of the alert dialog title
     * @param message  the resource id of the alert dialog message
     * @param listener the yes, no listener
     * @return the created alert dialog
     */
    public static AlertDialog createYesNoAlertDialog(final Context ctx, final int title, final int
     ↪  message, final DialogInterface.OnClickListener listener) {
        Objects.requireNonNull(ctx, "Cannot create dialog for null context");
        Objects.requireNonNull(listener, "Listener for buttons mut be given");

        return new AlertDialog.Builder(ctx)
                .setTitle(title)
                .setMessage(message)
                .setPositiveButton(R.string.action_yes, listener)
                .setNegativeButton(R.string.action_no, listener)
                .create();
    }

    /**
```

```
58      * Creates a alert dialog with an custom view as content.
59      *
60      * @param ctx      the context to create dialog for
61      * @param viewId   the resource id of the view
62      * @param title    the resource id of the alert dialog title
63      * @param listener the button listener
64      * @return the created alert dialog
65      */
66     public static AlertDialog createCustomContentDialog(final Context ctx, final int viewId, final
        ↪  int title, final DialogInterface.OnClickListener listener) {
67         Objects.requireNonNull(ctx, "Cannot create dialog for null context");
68         Objects.requireNonNull(listener, "Listener for buttons mut be given");
69
70         return new AlertDialog.Builder(ctx)
71                 .setTitle(title)
72                 .setView(viewId)
73                 .setPositiveButton(R.string.action_ok, listener)
74                 .create();
75     }
76 }
```

1.2.9 Watcher

Listing 23: AnswerButtonTextWatcher.java

```java
package at.fh.ooe.moc5.amazingrace.watcher;

import android.text.Editable;
import android.text.TextWatcher;
import android.widget.Button;

import at.fh.ooe.moc5.amazingrace.R;
import at.fh.ooe.moc5.amazingrace.activity.CheckpointActivity;
import at.fh.ooe.moc5.amazingrace.model.view.CheckpointViewModel;

/**
 * Created by Thomas on 12/24/2015.
 * A watcher which enables the answer button if the backed view model is valid, disables it
 ↪  otherwise
 */
public class AnswerButtonTextWatcher implements TextWatcher {

    private final CheckpointViewModel viewModel;
    private final CheckpointActivity activity;

    public AnswerButtonTextWatcher(CheckpointActivity activity, CheckpointViewModel viewModel) {
        this.activity = activity;
        this.viewModel = viewModel;
    }

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

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

    @Override
    public void afterTextChanged(Editable s) {
        ((Button) activity.findViewById(R.id.answerBtn)).setEnabled(viewModel.isValid());
    }
}
```

Listing 24: CheckpointViewModelBindingTextWatcher.java

```java
package at.fh.ooe.moc5.amazingrace.watcher;

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

import java.util.Objects;

import at.fh.ooe.moc5.amazingrace.model.view.CheckpointViewModel;

/**
 * Created by Thomas on 12/24/2015.
 * Performs a binding to the given view model
 */
public class CheckpointViewModelBindingTextWatcher implements TextWatcher {

    private final CheckpointViewModel view;

    public CheckpointViewModelBindingTextWatcher(CheckpointViewModel view) {
        Objects.requireNonNull(view);
        this.view = view;
    }

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

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

    @Override
    public void afterTextChanged(Editable s) {
        view.setAnswer(s.toString());
    }
}
```

Listing 25: LoginButtonTextWatcher.java

```java
package at.fh.ooe.moc5.amazingrace.watcher;

import android.text.Editable;
import android.text.TextWatcher;
import android.widget.Button;

import at.fh.ooe.moc5.amazingrace.R;
import at.fh.ooe.moc5.amazingrace.activity.LoginActivity;
import at.fh.ooe.moc5.amazingrace.model.view.LoginViewModel;

/**
 * Created by Thomas on 12/24/2015.
 * A watcher which enables the login button if the backed view model is valid, disables it
 ↪  otherwise
 */
public class LoginButtonTextWatcher implements TextWatcher {

    private LoginViewModel viewModel;
    private final LoginActivity activity;

    public LoginButtonTextWatcher(LoginActivity activity, LoginViewModel viewModel) {
        this.activity = activity;
        this.viewModel = viewModel;
    }

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

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

    @Override
    public void afterTextChanged(Editable s) {
        Button loginButton = (Button) activity.findViewById(R.id.loginBtn);
        loginButton.setEnabled(viewModel.isValid());
    }
}
```

Listing 26: LoginViewModelBindingTextWatcher.java

```java
package at.fh.ooe.moc5.amazingrace.watcher;

import android.text.Editable;
import android.text.TextWatcher;
import android.widget.EditText;

import java.util.Objects;

import at.fh.ooe.moc5.amazingrace.R;
import at.fh.ooe.moc5.amazingrace.model.view.LoginViewModel;

/**
 * Created by Thomas on 12/24/2015.
 * Performs a binding to the given view model
 */
public class LoginViewModelBindingTextWatcher implements TextWatcher {

    private final LoginViewModel view;
    private final EditText text;

    public LoginViewModelBindingTextWatcher(LoginViewModel view, EditText text) {
        Objects.requireNonNull(view);
        Objects.requireNonNull(text);
        this.view = view;
        this.text = text;
    }

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

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

    @Override
    public void afterTextChanged(Editable s) {
        switch (text.getId()) {
            case R.id.usernameEdTxt:
                view.setUsername(s.toString());
                break;
            case R.id.passwordEdTxt:
                view.setPassword(s.toString());
                break;
        }
    }
}
```

1.3 Layout

Listing 27: activity_login.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="at.fh.ooe.moc5.amazingrace.activity.LoginActivity">


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:orientation="vertical"
        android:weightSum="1">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:orientation="horizontal">

            <TextView
                android:id="@+id/usernameLabel"
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:layout_gravity="center"
                android:layout_weight="1"
                android:gravity="left|center"
                android:text="@string/label_username"
                android:textAppearance="?android:attr/textAppearanceMedium" />

            <EditText
                android:id="@+id/usernameEdTxt"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center|center"
                android:layout_weight="8" />
        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:orientation="horizontal">

            <TextView
                android:id="@+id/passwordLabel"
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:layout_gravity="left|center"
                android:layout_weight="1"
                android:gravity="left|center"
                android:text="@string/label_password"
                android:textAppearance="?android:attr/textAppearanceMedium" />

```

```
60              <EditText
61                  android:id="@+id/passwordEdTxt"
62                  android:layout_width="wrap_content"
63                  android:layout_height="wrap_content"
64                  android:layout_gravity="center|center"
65                  android:layout_weight="8"
66                  android:inputType="textPassword" />
67          </LinearLayout>
68
69          <RelativeLayout
70              android:layout_width="match_parent"
71              android:layout_height="wrap_content"
72              android:layout_weight="0.07"
73              android:orientation="horizontal"
74              android:weightSum="1">
75
76              <LinearLayout
77                  android:layout_width="wrap_content"
78                  android:layout_height="wrap_content"
79                  android:layout_alignParentLeft="false"
80                  android:layout_alignParentTop="false"
81                  android:layout_centerHorizontal="true"
82                  android:layout_centerInParent="true"
83                  android:layout_centerVertical="true"
84                  android:orientation="horizontal">
85
86                  <Button
87                      android:id="@+id/loginBtn"
88                      android:layout_width="wrap_content"
89                      android:layout_height="wrap_content"
90                      android:layout_margin="5pt"
91                      android:gravity="center|center"
92                      android:text="@string/action_login" />
93
94                  <Button
95                      android:id="@+id/cancelBtn"
96                      android:layout_width="wrap_content"
97                      android:layout_height="wrap_content"
98                      android:layout_margin="5pt"
99                      android:gravity="center|center"
100                     android:text="@string/action_cancel" />
101
102             </LinearLayout>
103
104         </RelativeLayout>
105     </LinearLayout>
106 </RelativeLayout>
```

Listing 28: activity_route.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="at.fh.ooe.moc5.amazingrace.activity.RouteActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:weightSum="1">

        <ListView
            android:id="@+id/listRoute"
            android:layout_width="match_parent"
            android:layout_height="match_parent"/>
    </LinearLayout>
</RelativeLayout>
```

Listing 29: activity_checkpoint.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".activity.CheckpointActivity">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:fillViewport="true">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_alignParentStart="true"
            android:layout_alignParentTop="true"
            android:orientation="vertical">

            <LinearLayout
                android:id="@+id/routeCheckpointContainer"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_alignParentStart="true"
                android:layout_alignParentTop="true"
                android:layout_margin="5pt"
                android:orientation="vertical">

                <TextView
                    android:id="@+id/nextCheckpointTxt"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:layout_marginBottom="5pt"
                    android:textAppearance="?android:attr/textAppearanceMedium" />

                <TextView
                    android:id="@+id/nextCheckpointHintTxt"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:layout_marginBottom="5pt"
                    android:textAppearance="?android:attr/textAppearanceMedium" />

                <EditText
                    android:id="@+id/hintAnswerEdTxt"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:layout_marginBottom="5pt" />

                <Button
                    android:id="@+id/answerBtn"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:gravity="center|center"
                    android:text="@string/action_answer" />
            </LinearLayout>

```

```
61              <LinearLayout
62                  android:id="@+id/routeCheckpointMapContainerLabel"
63                  android:layout_width="match_parent"
64                  android:layout_height="wrap_content"
65                  android:layout_marginBottom="5pt"
66                  android:orientation="horizontal">
67
68                  <ImageView
69                      android:layout_width="20pt"
70                      android:layout_height="20pt"
71                      android:clickable="true"
72                      android:onClick="toggleVisibility"
73                      android:src="@drawable/app_icon" />
74
75                  <TextView
76                      android:layout_width="match_parent"
77                      android:layout_height="match_parent"
78                      android:layout_marginLeft="5pt"
79                      android:clickable="true"
80                      android:gravity="left|center"
81                      android:onClick="toggleVisibility"
82                      android:text="@string/checkpoint_group_map"
83                      android:textAppearance="?android:attr/textAppearanceLarge"
84                      android:textStyle="bold" />
85              </LinearLayout>
86
87              <LinearLayout
88                  android:id="@+id/routeCheckpointMapContainer"
89                  android:layout_width="match_parent"
90                  android:layout_height="200dp"
91                  android:layout_marginBottom="10pt"
92                  android:layout_alignParentStart="true"
93                  android:layout_alignParentTop="true"
94                  android:orientation="vertical">
95
96                  <fragment
97                      android:id="@+id/routeCheckpointMap"
98                      android:name="com.google.android.gms.maps.MapFragment"
99                      android:layout_width="match_parent"
100                     android:layout_height="match_parent" />
101             </LinearLayout>
102
103             <LinearLayout
104                 android:id="@+id/routeCheckpointListContainerLabel"
105                 android:layout_width="match_parent"
106                 android:layout_height="wrap_content"
107                 android:layout_marginBottom="5pt"
108                 android:orientation="horizontal">
109
110                 <ImageView
111                     android:layout_width="20pt"
112                     android:layout_height="20pt"
113                     android:clickable="true"
114                     android:onClick="toggleVisibility"
115                     android:src="@drawable/list_icon" />
116
117                 <TextView
118                     android:layout_width="match_parent"
119                     android:layout_height="match_parent"
120                     android:layout_marginLeft="5pt"
121                     android:clickable="true"
122                     android:gravity="left|center"
123                     android:onClick="toggleVisibility"
```

```
                    android:text="@string/checkpoint_group_list"
                    android:textAppearance="?android:attr/textAppearanceLarge"
                    android:textStyle="bold" />
            </LinearLayout>

            <LinearLayout
                android:id="@+id/routeCheckpointListContainer"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_alignParentStart="true"
                android:layout_alignParentTop="true"
                android:orientation="vertical">

                <LinearLayout
                    android:id="@+id/routeCheckpointListView"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:layout_below="@+id/routeListTitleTxt"
                    android:orientation="vertical" />
            </LinearLayout>
        </LinearLayout>
    </ScrollView>
</RelativeLayout>
```

Listing 30: view_checkpoint_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_centerVertical="true"
    android:orientation="horizontal">

    <ImageView
        android:id="@+id/checkpointItemImage"
        android:layout_width="15pt"
        android:layout_height="15pt"
        android:layout_margin="3pt" />

    <TextView
        android:id="@+id/checkpointItemText"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="3pt"
        android:gravity="left|center" />
</LinearLayout>
```

Listing 31: view_route_item.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_centerVertical="true"
    android:orientation="horizontal">

    <ImageView
        android:id="@+id/routeItemImage"
        android:layout_width="15pt"
        android:layout_height="15pt"
        android:layout_margin="3pt" />

    <TextView
        android:id="@+id/routeItemText"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="3pt"
        android:gravity="left|center" />
</LinearLayout>
```

Listing 32: view_congratulations_dialog.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_centerVertical="true"
    android:orientation="vertical"
    android:weightSum="1">

    <ImageView
        android:layout_width="50pt"
        android:layout_height="50pt"
        android:src="@drawable/congratulations_icon"
        android:layout_gravity="center|center"/>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="5pt"
        android:layout_weight="1"
        android:gravity="center|center"
        android:text="@string/info_congratulations" />
</LinearLayout>
```

1.4 Values

Listing 33: strings.xml

```
<resources>
    <string name="application_title">AmazingRace</string>
    <string name="activity_title_login">Login AmazingRace</string>
    <string name="activity_title_route">AmazingRace Routes</string>
    <string name="activity_title_checkpoint">Route checkpoints</string>
    <string name="action_answer">Answer</string>
    <string name="action_cancel">Cancel</string>
    <string name="action_ok">Ok</string>
    <string name="action_yes">Yes</string>
    <string name="action_login">Login</string>
    <string name="action_settings">Settings</string>
    <string name="action_no">No</string>
    <string name="action_play">Play</string>
    <string name="action_reset">Reset</string>
    <string name="action_reset_all">Reset All</string>
    <string name="action_close">Close</string>
    <string name="action_open">Open</string>
    <string name="action_reload">Reload</string>
    <string name="label_username">Username</string>
    <string name="label_password">Password</string>
    <string name="progress_login">Logging in ...</string>
    <string name="progress_loading_routes">Loading routes ...</string>
    <string name="progress_updating_route">Updating route ...</string>
    <string name="progress_validating_secred">Validating secret</string>
    <string name="progress_title">Processing</string>
    <string name="progress_resetting_routes">Resetting route(s) ...</string>
    <string name="dialog_title_error">Error</string>
    <string name="dialog_title_warning">Warning</string>
    <string name="dialog_title_info">Info</string>
    <string name="error_request_invalid">Sorry. Request failed because it was invalid</string>
    <string name="error_request_timeout">Sorry. Request failed because of an timeout</string>
    <string name="error_unknown">Sorry. An unknown error occurred</string>
    <string name="error_login_failed">Username and/or password not correct</string>
    <string name="error_user_became_invalid">Your user is not valid anymore. You will be logged
    ↪ out.</string>
    <string name="error_user_not_logged">Your are no longer logged in. You will be returned to the
    ↪ login page</string>
    <string name="error_route_reset_failed">The route(s) could not be reset. Please try
    ↪ again</string>
    <string name="error_no_network">This application needs an active internet connection to
    ↪ work</string>
    <string name="warning_want_quit">Do you really want to quit ?</string>
    <string name="info_secret_ok">Secret ok. Next checkpoint has been enabled</string>
    <string name="info_secret_wrong">Sorry wrong secret. Guess again</string>
    <string name="info_congratulations">Congratulations, you have finished this route</string>
    <string name="finished">finished</string>
    <string name="done">done</string>
    <string name="open">open</string>
    <string name="checkpoint_group_secret">Checkpoint Secret</string>
    <string name="checkpoint_group_list">Checkpoint List</string>
    <string name="checkpoint_group_map">Checkpoint Map</string>
    <string name="google_maps_key">AIzaSyCNV6I75G-wopgeM3st0HeUiTzgWcINjy8</string>
</resources>
```

1.5 Tests

Folgend sind die Tests der Anwendung *AmazingRace* angeführt.

1.5.1 Login Aktivität

Folgend sind die Tests des Login angeführt.

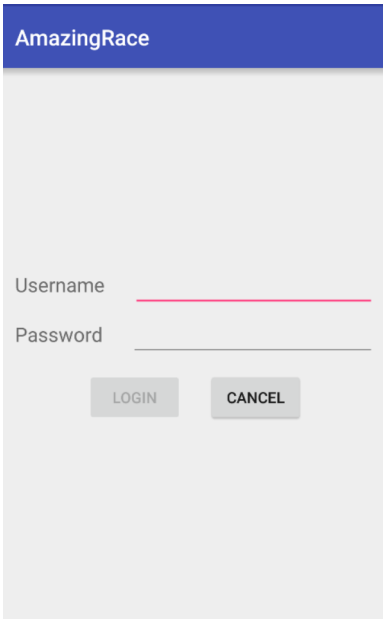

Abbildung 4: Login Aktivität

Der Button *Login* ist hierbei solange deaktiviert solange eine der beiden Eingabefelder keinen Text beinhaltet. Dies bedeutet aber auch, das leere Passwörter nicht unterstützt werden.

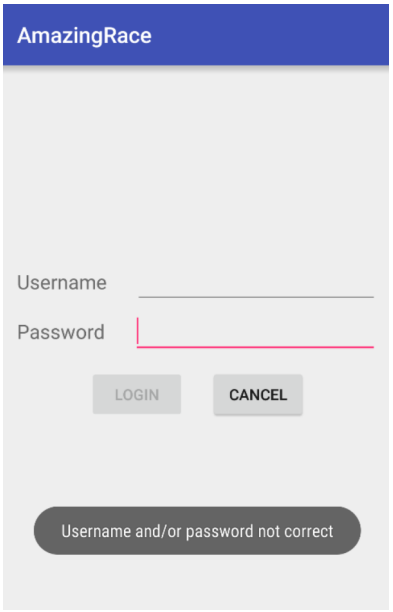

Abbildung 5: Login fehlgeschlagen

Sollte der Login fehlschlagen so werden beide Eingabefelder zurückgesetzt und eine Fehlermeldung in Form eines Toasts angezeigt.

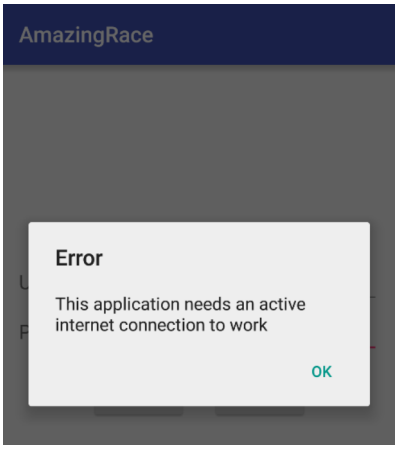

Abbildung 6: Keine Internetverbindung

Sollte keine Netzwerkverbindung vorhanden sein so wird ein entsprechender Dialog angezeigt.

1.5.2 Routen Aktivität

Folgend sind die Tests für die Routen Aktivität angeführt.

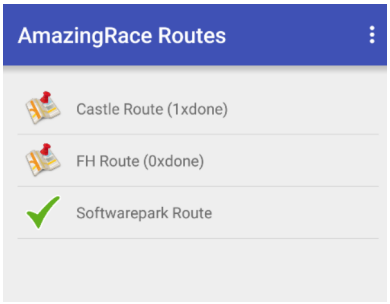

Abbildung 7: Routen Aktivität

Nach einem erfolgreichen Login oder durch drücken des *Back Button* in der Checkpoint Aktivität gelangt man zur Übersicht der verfügbaren Routen. Hierbei werden die Routen wie folgt sortiert.

1. Offene Routen
   (a) Anzahl der besuchten Checkpoints absteigend
   (b) Routenname aufsteigend
2. Abgeschlossene Routen
   (a) Routenname aufsteigend

Ist eine Route bereits abgeschlossen so wird eine entsprechender Icon angezeigt.

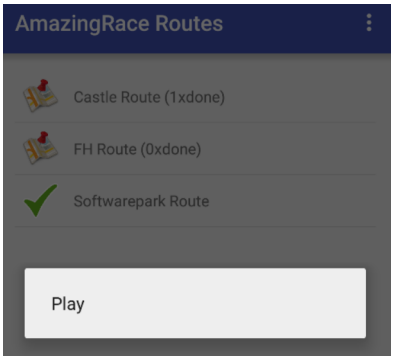

Abbildung 8: Kontextmenü einer offenen Route

Ist eine Route offen so wird beim Kontextmenü lediglich eine Option *Play* angezeigt.

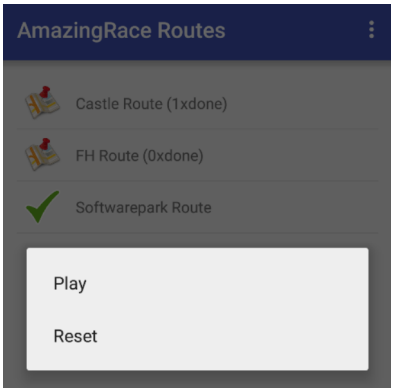

Abbildung 9: Kontextmenü einer gestarteten Route

Wurde eine Route bereits gestartet so werden beim Kontextmenü zwei Optionen *Play, Reset* angezeigt.

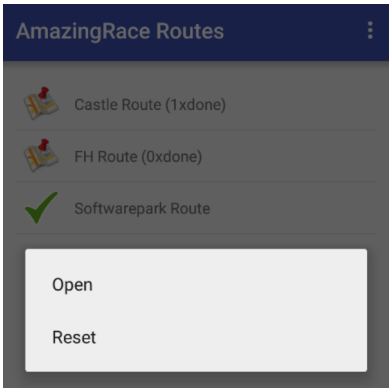

Abbildung 10: Kontextmenü einer abgeschlossenen Route

Ist eine Route bereits abgeschlossen so werden beim Kontextmenü zwei Optionen *Open, Reset* angezeigt.

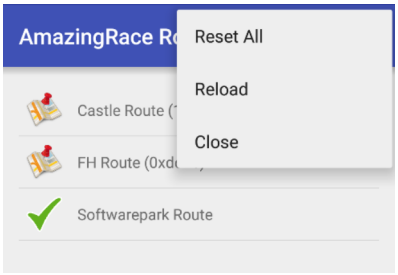

Abbildung 11: Routen Aktivität Menü

Die Aktivität stellt zwei Optionen *Reset All, Reload, Close* über ein Menü zur Verfügung.

1. *Reset All*
   Setzt alle zur Verfügung stehenden Routen zurück
2. *Reload*
   Lädt alle zur Verfügung stehenden Routen erneut
3. *Close*
   Öffnet einen Dialog, über den im Falle einer Bestätigung die Anwendung geschlossen wird.

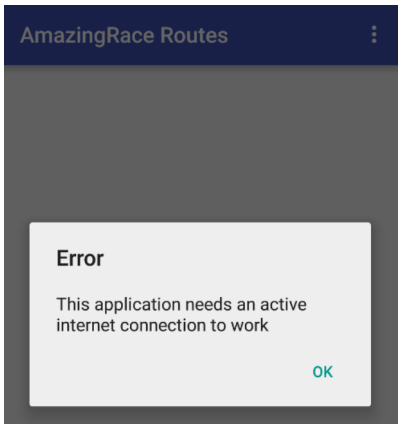

Abbildung 12: Keine Netzwerkverbindung

Sollte keine Netzwerkverbindung vorhanden sein so wird ein entsprechender Dialog angezeigt.

1.6 Checkpoint Aktivität

Folgend sind die Tests für die Checkpoint Aktivität angeführt.

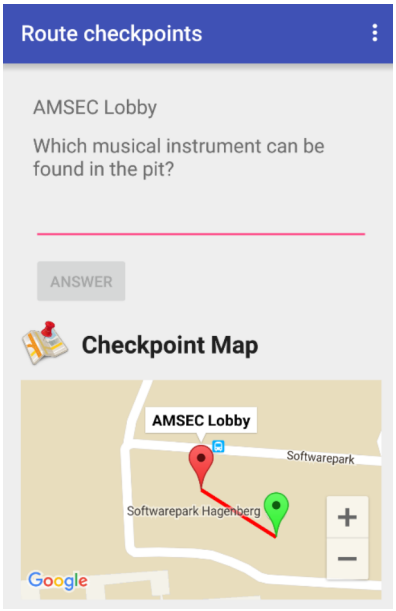

Abbildung 13: Offene Route Teil 1

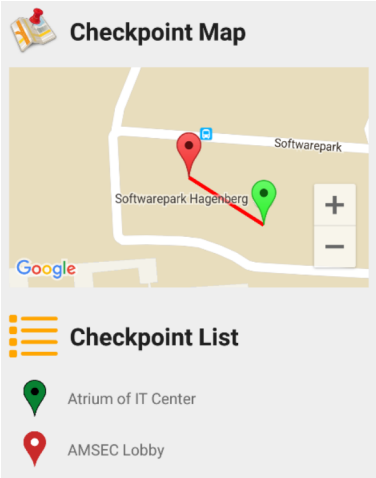

Abbildung 14: Offene Route Teil 2

Nachdem eine Route ausgewählt wurde wir auf die Checkpoint Aktivität gewechselt. Es gibt zwei Möglichkeiten die Checkpoints anzuzeigen.

1. *Google Map*
   Zeigt alle Checkpoints an wobei ein noch nicht besuchter Checkpoint rot markiert wird

2. *Checkpoint Liste*
   Zeigt alle besuchten Checkpoints in Listenform an wobei ein nicht besuchter Checkpoint ebenfalls rot markiert wird.

In diesem Fall ist die Route noch nicht abgeschlossen.

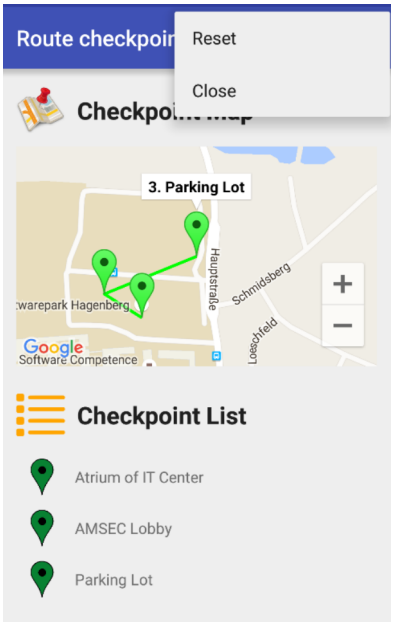

Abbildung 15: Optionen einer abgeschlossene oder gestarteten Route

Bei einer abgeschlossenen oder bereits gestarteten Route stehen zwei Optionen zur Verfügung *Reset, Close*.

1. *Reset*
   Setzt die aktuelle Route zurück

2. *Close*
   Öffnet einen Dialog wobei bei dessen Bestätigung die Anwendung geschlossen wird.

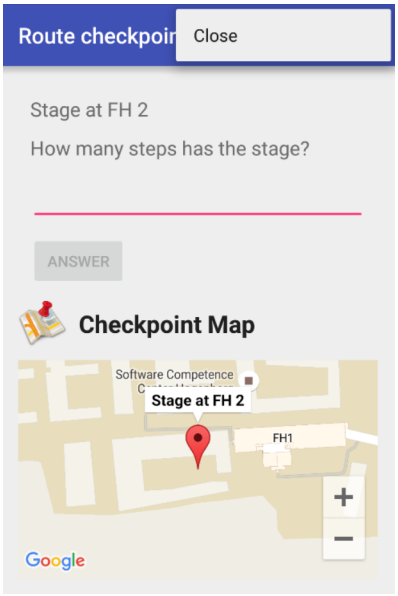

Abbildung 16: Optionen einer offenen Route

Bei einer offene Route steht nur eine Option zur Verfügung *Close*.

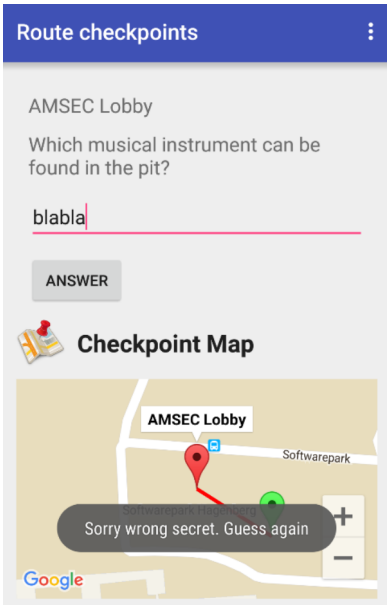

Abbildung 17: Frage falsch beantwortet

Sollte eine Frage falsch beantwortet worden sein so wird eine Meldung in Form eines Toast ausgegeben.

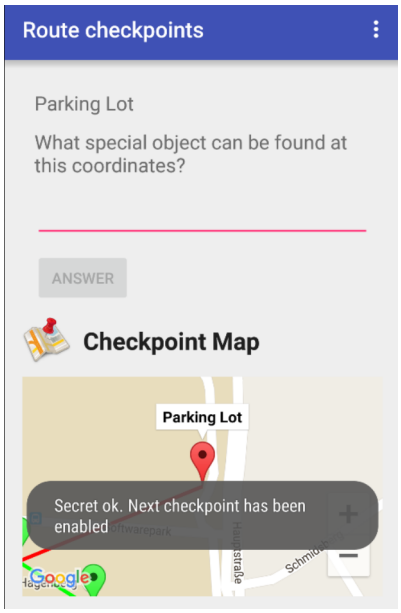

Abbildung 18: Frage korrekt beantwortet

Sollte eine Frage korrekt beantwortet worden sein so wird eine Meldung in Form eines Toast ausgegeben.

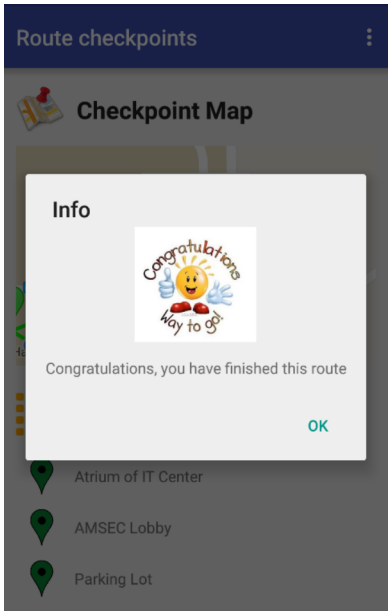

Abbildung 19: Route abgeschlossen

Nachdem die letzte Frage korrekt beantwortet wurde wird ein Dialog mit einer Erfolgsmeldung ausgegeben. Es wird auf der Checkpoint Aktivität verblieben wobei die Ansicht sich wie bei einer abgeschlossenen Route ändert.