|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |
| Институт кибербезопасности и цифровых технологий |
| Кафедра КБ-14 «Цифровые технологии обработки данных» |

**ОТЧЕТ**

**по практической работе №7**

|  |  |  |  |
| --- | --- | --- | --- |
| Выполнил |  |  | Бурмистров И.Г.  *фамилия, имя, отчество* |
| шифр | 22Б0616 | группа | БСБО-07-22 |
|  |  |  |  |
| Проверил |  | *ученая степень, должность* | Изергин Д.А.  *фамилия, имя, отчество* |

**Москва 2025г.**

В ходе данной работы были созданы модули «TimeService», «httpurlconnect» и «firebase» (см. Рисунок 1).

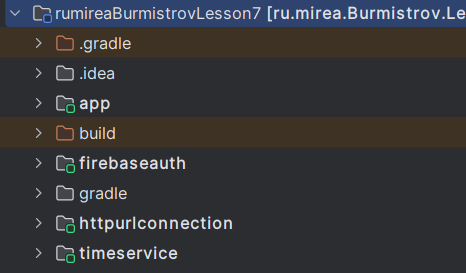
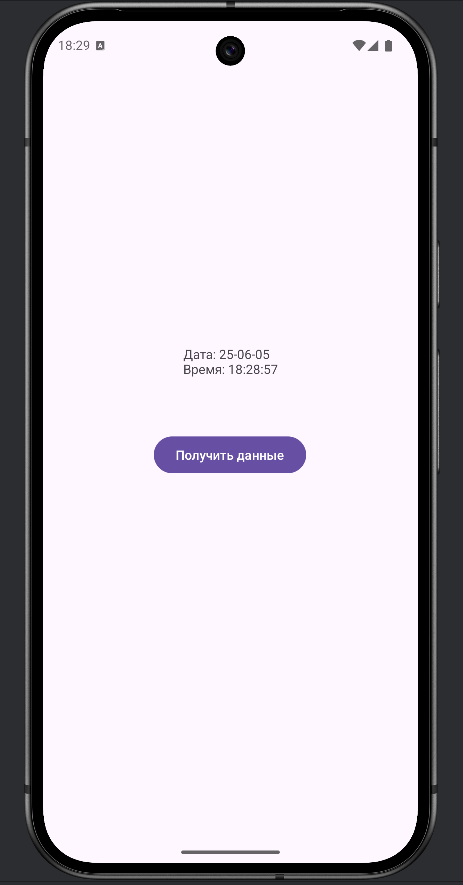


Рисунок 1. Модули проекта

В первом модуле «TimeService» было реализовано получение времени и даты из сервера (см. Рисунок 2 и Листинги 1-2).

  
Рисунок 2. Пример получения даты и времени

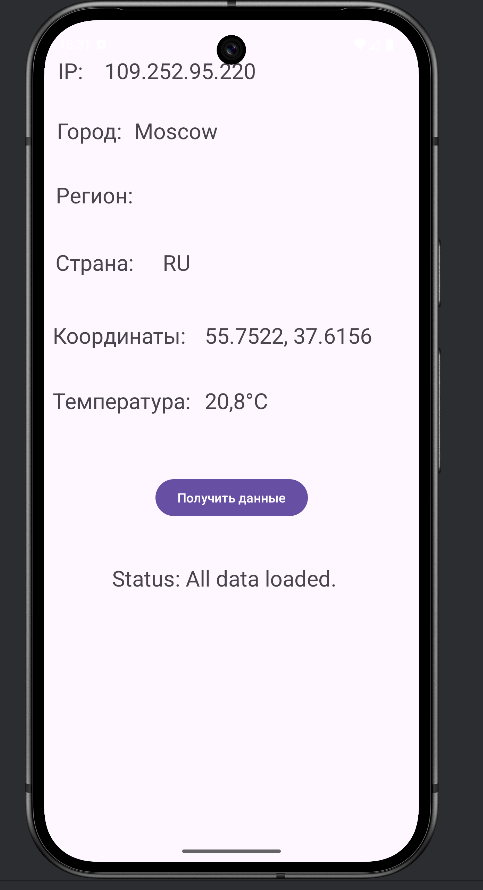
public class MainActivity extends AppCompatActivity {  
  
 private TextView text;  
 private Button button;  
 private final String HOST = "time.nist.gov";  
 private final int PORT = 13;  
 private static final String *TAG* = "TimeService";  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 EdgeToEdge.*enable*(this);  
 setContentView(R.layout.*activity\_main*);  
 ViewCompat.*setOnApplyWindowInsetsListener*(findViewById(R.id.*main*), (v, insets) -> {  
 Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.*systemBars*());  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);  
 return insets;  
 });  
  
 text = findViewById(R.id.*textView*);  
 button = findViewById(R.id.*button*);  
  
 button.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 GetTimeTask timeTask = new GetTimeTask();  
 timeTask.execute();  
 }  
 });  
 }  
  
 private class GetTimeTask extends AsyncTask<Void, Void, String> {  
 @Override  
 protected void onPreExecute() {  
 super.onPreExecute();  
 text.setText("Loading...");  
 button.setEnabled(false);  
 }  
  
 @Override  
 protected String doInBackground(Void... params) {  
 String timeResult = null;  
 Socket socket = null;  
 BufferedReader reader = null;  
 int retries = 0;  
 final int MAX\_RETRIES = 5;  
  
 try {  
 Log.*d*(*TAG*, "Attempting to connect to " + HOST + ":" + PORT);  
 socket = new Socket();  
 socket.connect(new java.net.InetSocketAddress(HOST, PORT), 5000);  
 socket.setSoTimeout(5000);  
  
 Log.*d*(*TAG*, "Connected. Getting reader.");  
 reader = SocketUtils.*getReader*(socket);  
  
 String line;  
 while ((line = reader.readLine()) != null && retries < MAX\_RETRIES) {  
 Log.*d*(*TAG*, "Read line: " + line);  
 if (!line.trim().isEmpty() && line.contains("-") && line.contains(":")) {  
 timeResult = line;  
 break;  
 }  
 retries++;  
 }  
  
 if (timeResult == null) {  
 Log.*w*(*TAG*, "Could not get a valid time string after " + retries + " attempts or stream ended.");  
 return "Error: No valid data received from server.";  
 }  
  
 Log.*d*(*TAG*, "Raw timeResult (final): " + timeResult);  
  
 } catch (java.net.SocketTimeoutException e) {  
 Log.*e*(*TAG*, "SocketTimeoutException: " + e.getMessage());  
 return "Error: Connection or read timed out.";  
 } catch (IOException e) {  
 Log.*e*(*TAG*, "IOException: " + e.getMessage());  
 e.printStackTrace();  
 return "Error: Could not connect or read from server.";  
 } finally {  
 try {  
 if (reader != null) {  
 reader.close();  
 }  
 if (socket != null) {  
 socket.close();  
 }  
 } catch (IOException e) {  
 Log.*e*(*TAG*, "Error closing resources: " + e.getMessage());  
 e.printStackTrace();  
 }  
 }  
 return timeResult;  
 }  
 @Override  
 protected void onPostExecute(String result) {  
 super.onPostExecute(result);  
 button.setEnabled(true);  
 if (result != null && !result.startsWith("Error:")) {  
 String[] parts = result.split(" ");  
 if (parts.length >= 3) {  
 String date = parts[1]; // YR-MM-DD  
 String time = parts[2]; // HH:MM:SS  
 text.setText("Дата: " + date + "\nВремя: " + time);  
 } else {  
 text.setText("Error: Malformed server response.\nRaw: " + result);  
 Log.*e*(*TAG*, "Malformed server response: " + result);  
 }  
 } else {  
 text.setText(result);  
 }  
 }  
 }  
}

Листинг 1. MainActivity

public class SocketUtils {  
 public static BufferedReader getReader(Socket s) throws IOException {  
 return (new BufferedReader(new InputStreamReader(s.getInputStream())));  
 }  
 public static PrintWriter getWriter(Socket s) throws IOException {  
 return (new PrintWriter(s.getOutputStream(), true));  
 }  
}

Листинг 2. SocketUtils

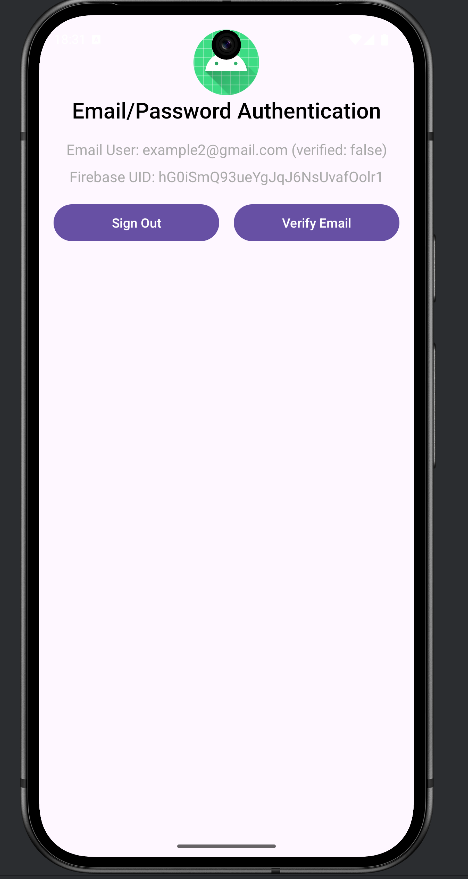
Далее был создан модуль «HttpURLConnection», в котором отправляется запрос серверу, а затем полученные данные записываются в приложение (см. Рисунок 2 и Листинг 3).

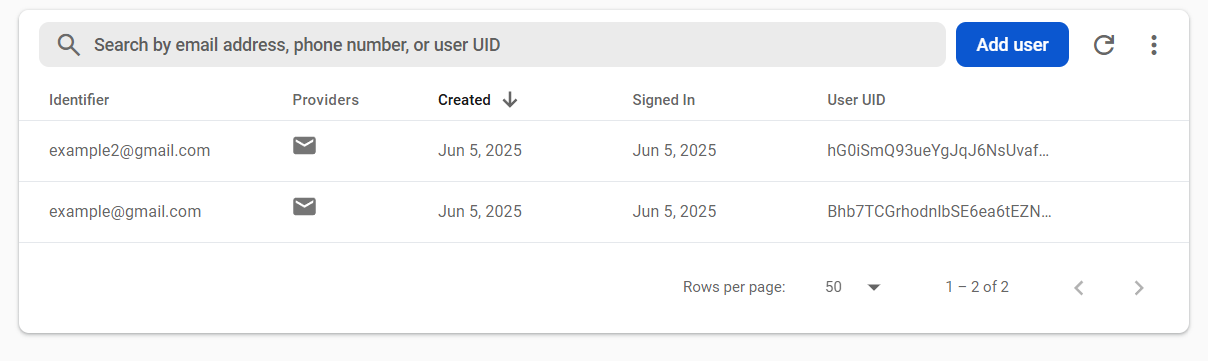
  
Рисунок 3. Записанная информация

public class MainActivity extends AppCompatActivity {  
  
 private static final String *TAG* = "HttpURLConnectionExample";  
 private TextView textViewIp, textViewCity, textViewRegion, textViewCountry, textViewCoordinates, textViewWeather, textViewStatus;  
 private Button buttonGetData;  
  
 // IP Info  
 private String ip;  
 private String city;  
 private String region;  
 private String country;  
 private String latitude;  
 private String longitude;  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 textViewIp = findViewById(R.id.*textViewIp*);  
 textViewCity = findViewById(R.id.*textViewCity*);  
 textViewRegion = findViewById(R.id.*textViewRegion*);  
 textViewCountry = findViewById(R.id.*textViewCountry*);  
 textViewCoordinates = findViewById(R.id.*textViewCoordinates*);  
 textViewWeather = findViewById(R.id.*textViewWeather*);  
 textViewStatus = findViewById(R.id.*textViewStatus*);  
 buttonGetData = findViewById(R.id.*buttonGetData*);  
  
 buttonGetData.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 ConnectivityManager connMgr =  
 (ConnectivityManager) getSystemService(Context.*CONNECTIVITY\_SERVICE*);  
 NetworkInfo networkInfo = null;  
 if (connMgr != null) {  
 networkInfo = connMgr.getActiveNetworkInfo();  
 }  
  
 if (networkInfo != null && networkInfo.isConnected()) {  
 textViewStatus.setText("Loading IP info...");  
 new DownloadIpInfoTask().execute("https://ipinfo.io/json");  
 } else {  
 Toast.*makeText*(MainActivity.this, "No internet connection", Toast.*LENGTH\_SHORT*).show();  
 textViewStatus.setText("Status: No internet");  
 }  
 }  
 });  
 textViewStatus.setText("Status: Idle");  
 }  
  
 private class DownloadIpInfoTask extends AsyncTask<String, Void, String> {  
 @Override  
 protected String doInBackground(String... urls) {  
 try {  
 return downloadContent(urls[0]);  
 } catch (IOException e) {  
 Log.*e*(*TAG*, "Unable to retrieve web page. URL may be invalid. " + e.getMessage());  
 return "Error: Unable to retrieve web page.";  
 }  
 }  
  
 @Override  
 protected void onPostExecute(String result) {  
 Log.*d*(*TAG*, "IPInfo JSON: " + result);  
 try {  
 JSONObject responseJson = new JSONObject(result);  
 ip = responseJson.optString("ip");  
 city = responseJson.optString("city");  
 region = responseJson.optString("region");  
 country = responseJson.optString("country");  
 String loc = responseJson.optString("loc"); // "latitude,longitude"  
 if (!loc.isEmpty() && loc.contains(",")) {  
 String[] coordinates = loc.split(",");  
 latitude = coordinates[0];  
 longitude = coordinates[1];  
 }  
  
 textViewIp.setText(ip);  
 textViewCity.setText(city);  
 textViewRegion.setText(region);  
 textViewCountry.setText(country);  
 if (latitude != null && longitude != null) {  
 textViewCoordinates.setText(latitude + ", " + longitude);  
 // Now fetch weather  
 textViewStatus.setText("Loading weather data...");  
 String weatherUrl = "https://api.open-meteo.com/v1/forecast?latitude=" + latitude +  
 "&longitude=" + longitude + "&current\_weather=true";  
 Log.*d*(*TAG*, "Weather URL: " + weatherUrl);  
 new DownloadWeatherTask().execute(weatherUrl);  
 } else {  
 textViewCoordinates.setText("N/A");  
 textViewWeather.setText("N/A (no coordinates)");  
 textViewStatus.setText("Status: IP info loaded, no coordinates for weather.");  
 }  
  
 } catch (JSONException e) {  
 Log.*e*(*TAG*, "JSONException in IPInfo: " + e.getMessage());  
 textViewStatus.setText("Error parsing IP info.");  
 textViewIp.setText("Error");  
 }  
 }  
 }  
  
 private class DownloadWeatherTask extends AsyncTask<String, Void, String> {  
 @Override  
 protected String doInBackground(String... urls) {  
 try {  
 return downloadContent(urls[0]);  
 } catch (IOException e) {  
 Log.*e*(*TAG*, "Unable to retrieve weather data. " + e.getMessage());  
 return "Error: Unable to retrieve weather data.";  
 }  
 }  
  
 @Override  
 protected void onPostExecute(String result) {  
 Log.*d*(*TAG*, "Weather JSON: " + result);  
 try {  
 JSONObject responseJson = new JSONObject(result);  
 if (responseJson.has("current\_weather")) {  
 JSONObject currentWeather = responseJson.getJSONObject("current\_weather");  
 double temperature = currentWeather.getDouble("temperature");  
 textViewWeather.setText(String.*format*("%.1f°C", temperature));  
 textViewStatus.setText("Status: All data loaded.");  
 } else {  
 textViewWeather.setText("N/A");  
 textViewStatus.setText("Status: Weather data not found in response.");  
 }  
  
 } catch (JSONException e) {  
 Log.*e*(*TAG*, "JSONException in Weather: " + e.getMessage());  
 textViewWeather.setText("Error");  
 textViewStatus.setText("Error parsing weather data.");  
 }  
 }  
 }  
  
  
 private String downloadContent(String myurl) throws IOException {  
 InputStream inputStream = null;  
 String data = "";  
 try {  
 URL url = new URL(myurl);  
 HttpURLConnection connection = (HttpURLConnection) url.openConnection();  
 connection.setReadTimeout(100000); // milliseconds  
 connection.setConnectTimeout(100000); // milliseconds  
 connection.setRequestMethod("GET");  
 connection.setInstanceFollowRedirects(true);  
 connection.setUseCaches(false);  
 connection.setDoInput(true);  
  
 int responseCode = connection.getResponseCode();  
 if (responseCode == HttpURLConnection.*HTTP\_OK*) { // 200 OK  
 inputStream = connection.getInputStream();  
 ByteArrayOutputStream bos = new ByteArrayOutputStream();  
 int read = 0;  
 while ((read = inputStream.read()) != -1) {  
 bos.write(read);  
 }  
 bos.close();  
 data = bos.toString();  
 } else {  
 data = "Error: " + responseCode + " " + connection.getResponseMessage();  
 Log.*e*(*TAG*, data);  
 }  
 connection.disconnect();  
 } finally {  
 if (inputStream != null) {  
 inputStream.close();  
 }  
 }  
 return data;  
 }  
}

Листинг 3. MainActivity

Далее был создан модуль «firebaseauth», в котором была реализована регистрация и авторизация через «firebase» (см. Рисунок 4 и Листинг 4).

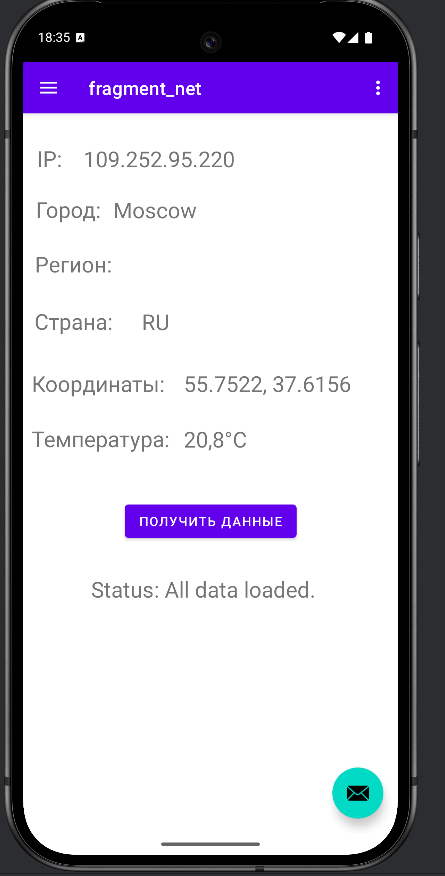
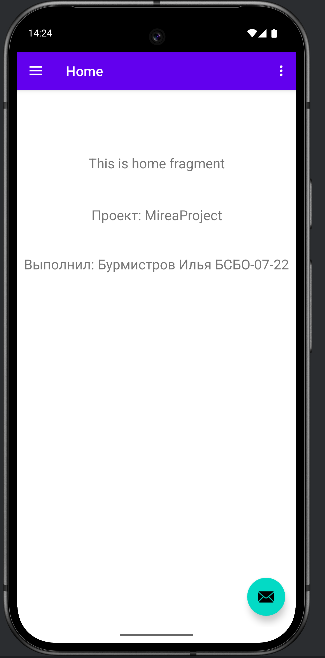
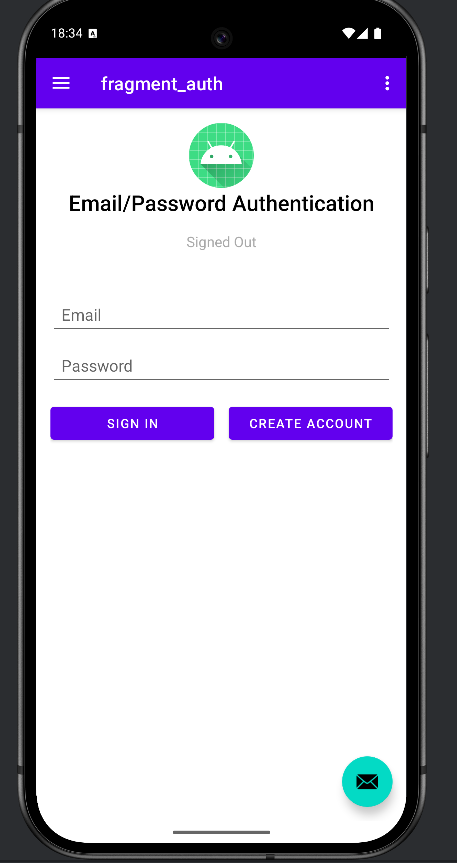


  
Рисунок 4. Пример авторизации

public class MainActivity extends AppCompatActivity implements View.OnClickListener {  
  
 private static final String *TAG* = "FirebaseAuthExample";  
  
 private TextView mStatusTextView;  
 private TextView mDetailTextView;  
 private EditText mEmailField;  
 private EditText mPasswordField;  
  
 private Button mEmailSignInButton;  
 private Button mEmailCreateAccountButton;  
 private Button mSignOutButton;  
 private Button mVerifyEmailButton;  
  
 private LinearLayout mEmailPasswordFields;  
 private LinearLayout mEmailPasswordButtons;  
 private LinearLayout mSignedInButtons;  
  
 private FirebaseAuth mAuth;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 // Views  
 mStatusTextView = findViewById(R.id.*statusTextView*);  
 mDetailTextView = findViewById(R.id.*detailTextView*);  
 mEmailField = findViewById(R.id.*fieldEmail*);  
 mPasswordField = findViewById(R.id.*fieldPassword*);  
  
 // Buttons  
 mEmailSignInButton = findViewById(R.id.*emailSignInButton*);  
 mEmailCreateAccountButton = findViewById(R.id.*emailCreateAccountButton*);  
 mSignOutButton = findViewById(R.id.*signOutButton*);  
 mVerifyEmailButton = findViewById(R.id.*verifyEmailButton*);  
  
 // Layouts  
 mEmailPasswordFields = findViewById(R.id.*emailPasswordFields*);  
 mEmailPasswordButtons = findViewById(R.id.*emailPasswordButtons*);  
 mSignedInButtons = findViewById(R.id.*signedInButtons*);  
  
 // Click listeners  
 mEmailSignInButton.setOnClickListener(this);  
 mEmailCreateAccountButton.setOnClickListener(this);  
 mSignOutButton.setOnClickListener(this);  
 mVerifyEmailButton.setOnClickListener(this);  
  
 // Initialize Firebase Auth  
 mAuth = FirebaseAuth.*getInstance*();  
 }  
  
 @Override  
 public void onStart() {  
 super.onStart();  
 // Check if user is signed in (non-null) and update UI accordingly.  
 FirebaseUser currentUser = mAuth.getCurrentUser();  
 updateUI(currentUser);  
 }  
  
 private void createAccount(String email, String password) {  
 Log.*d*(*TAG*, "createAccount:" + email);  
 if (!validateForm()) {  
 return;  
 }  
 mAuth.createUserWithEmailAndPassword(email, password)  
 .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {  
 @Override  
 public void onComplete(@NonNull Task<AuthResult> task) {  
 if (task.isSuccessful()) {  
 Log.*d*(*TAG*, "createUserWithEmail:success");  
 FirebaseUser user = mAuth.getCurrentUser();  
 updateUI(user);  
 // Optionally send verification email here  
 // sendEmailVerification();  
 } else {  
 Log.*w*(*TAG*, "createUserWithEmail:failure", task.getException());  
 try {  
 throw task.getException();  
 } catch (FirebaseAuthWeakPasswordException e) {  
 mPasswordField.setError(getString(R.string.*error\_weak\_password*));  
 mPasswordField.requestFocus();  
 } catch (FirebaseAuthInvalidCredentialsException e) {  
 mEmailField.setError(getString(R.string.*error\_invalid\_email*));  
 mEmailField.requestFocus();  
 } catch (FirebaseAuthUserCollisionException e) {  
 mEmailField.setError(getString(R.string.*error\_email\_exists*));  
 mEmailField.requestFocus();  
 } catch (Exception e) {  
 Toast.*makeText*(MainActivity.this, getString(R.string.*auth\_failed*) + ": " + e.getMessage(),  
 Toast.*LENGTH\_SHORT*).show();  
 }  
 updateUI(null);  
 }  
 }  
 });  
 }  
  
 private void signIn(String email, String password) {  
 Log.*d*(*TAG*, "signIn:" + email);  
 if (!validateForm()) {  
 return;  
 }  
 mAuth.signInWithEmailAndPassword(email, password)  
 .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {  
 @Override  
 public void onComplete(@NonNull Task<AuthResult> task) {  
 if (task.isSuccessful()) {  
 Log.*d*(*TAG*, "signInWithEmail:success");  
 FirebaseUser user = mAuth.getCurrentUser();  
 updateUI(user);  
 } else {  
 Log.*w*(*TAG*, "signInWithEmail:failure", task.getException());  
 try {  
 throw task.getException();  
 } catch (FirebaseAuthInvalidCredentialsException e) { // Covers user not found & wrong password  
 Toast.*makeText*(MainActivity.this, getString(R.string.*error\_sign\_in\_failed*), Toast.*LENGTH\_SHORT*).show();  
 } catch (Exception e) {  
 Toast.*makeText*(MainActivity.this, getString(R.string.*auth\_failed*) + ": " + e.getMessage(),  
 Toast.*LENGTH\_SHORT*).show();  
 }  
 updateUI(null);  
 }  
 }  
 });  
 }  
  
 private void signOut() {  
 mAuth.signOut();  
 updateUI(null);  
 }  
  
 private void sendEmailVerification() {  
 final FirebaseUser user = mAuth.getCurrentUser();  
 if (user != null && !user.isEmailVerified()) {  
 user.sendEmailVerification()  
 .addOnCompleteListener(this, new OnCompleteListener<Void>() {  
 @Override  
 public void onComplete(@NonNull Task<Void> task) {  
 if (task.isSuccessful()) {  
 Toast.*makeText*(MainActivity.this,  
 getString(R.string.*verification\_email\_sent*) + " " + user.getEmail(),  
 Toast.*LENGTH\_SHORT*).show();  
 } else {  
 Log.*e*(*TAG*, "sendEmailVerification", task.getException());  
 Toast.*makeText*(MainActivity.this,  
 getString(R.string.*error\_sending\_verification\_email*),  
 Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
 } else {  
 Toast.*makeText*(this, "User already verified or not signed in.", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
  
 private boolean validateForm() {  
 boolean valid = true;  
  
 String email = mEmailField.getText().toString();  
 if (TextUtils.*isEmpty*(email)) {  
 mEmailField.setError("Required.");  
 valid = false;  
 } else {  
 mEmailField.setError(null);  
 }  
  
 String password = mPasswordField.getText().toString();  
 if (TextUtils.*isEmpty*(password)) {  
 mPasswordField.setError("Required.");  
 valid = false;  
 } else {  
 mPasswordField.setError(null);  
 }  
 return valid;  
 }  
  
 private void updateUI(FirebaseUser user) {  
 if (user != null) {  
 mStatusTextView.setText(getString(R.string.*emailpassword\_status\_fmt*,  
 user.getEmail(), user.isEmailVerified()));  
 mDetailTextView.setText(getString(R.string.*firebase\_status\_fmt*, user.getUid()));  
  
 mEmailPasswordFields.setVisibility(View.*GONE*);  
 mEmailPasswordButtons.setVisibility(View.*GONE*);  
 mSignedInButtons.setVisibility(View.*VISIBLE*);  
  
 mVerifyEmailButton.setEnabled(!user.isEmailVerified());  
 } else {  
 mStatusTextView.setText(R.string.*signed\_out*);  
 mDetailTextView.setText(null);  
  
 mEmailPasswordFields.setVisibility(View.*VISIBLE*);  
 mEmailPasswordButtons.setVisibility(View.*VISIBLE*);  
 mSignedInButtons.setVisibility(View.*GONE*);  
 }  
 }  
  
 @Override  
 public void onClick(View v) {  
 int i = v.getId();  
 String email = mEmailField.getText().toString();  
 String password = mPasswordField.getText().toString();  
  
 if (i == R.id.*emailCreateAccountButton*) {  
 createAccount(email, password);  
 } else if (i == R.id.*emailSignInButton*) {  
 signIn(email, password);  
 } else if (i == R.id.*signOutButton*) {  
 signOut();  
 } else if (i == R.id.*verifyEmailButton*) {  
 sendEmailVerification();  
 }  
 }  
}

Листинг 4. MainActivity

Далее в проекте «mireaproject» **(который находится в Lesson3)** было изменено стартовое окно на окно авторизации. Пока пользователь не войдёт в систему навигация работать не будет. После авторизации или регистрации открывается домашняя страница и разблокируется навигация. Далее был добавлен отдельный фрагмент, показывающий сетевые данные (см. Рисунок 5).

  
Рисунок 5. Новые окна в проекте