Министерство науки и высшего образования Российской Федерации
Муромский институт (филиал)
Федерального государственного бюджетного образовательного учреждения высшего образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»

Факультет ИТР

Кафедра ПИн

КУРСОВАЯ РАБОТА

По Разработке приложений для мобильных операционных систем

Тема «Социальная сеть»

Руководитель

Колпаков А.А.
(фамилия, инициалы)

(подпись) (дата)

Студент ПИН - 121
(группа)

Ермилов М.В.
(фамилия, инициалы)

(подпись) (дата)

Муром 2024

В курсовой работе разработано мобильное приложение для платформы Android с использованием Kotlin и Android Studio. Реализованы функции авторизации, управления постами и взаимодействия с новостной лентой. Особое внимание уделено работе с сервером через API и обеспечению безопасности с помощью сессий.

The term paper presents a mobile application for the Android platform using Kotlin and Android Studio. It implements features such as user authentication, post management, and interaction with the news feed. Special attention is given to server communication via API and session security.

Содержание

Введение

В современном мире социальные сети играют важную роль в коммуникации между людьми, предоставляя возможности для мгновенного обмена сообщениями, публикации контента, взаимодействия с сообществами и поддержания социальных связей. Успех таких платформ во многом зависит от удобства использования, доступности и функциональности интерфейса для пользователя.

Одним из ключевых элементов любой социальной сети является её серверная часть — API, которая предоставляет доступ к данным пользователей, их публикациям, комментариям и другим элементам взаимодействия. API служит посредником между клиентскими приложениями и сервером, обеспечивая обмен данными в стандартизированных форматах, таких как JSON или XML. Это позволяет создавать различные клиентские приложения, которые могут визуализировать и обрабатывать данные, полученные от сервера, а также отправлять новые данные.

Целью данного курсового проекта является разработка мобильного приложения для социальной сети, которое будет взаимодействовать с существующим сервером API. Приложение должно предоставить пользователю удобный интерфейс для взаимодействия с социальной сетью, включая возможности просмотра профилей, публикаций, отправки сообщений и уведомлений в режиме реального времени.

Для достижения этой цели разработка мобильного приложения включает в себя несколько ключевых этапов: анализ требований, создание алгоритмов для работы с API, реализация пользовательского интерфейса, а также тестирование и оптимизация функциональности. Важно также обеспечить безопасность передачи данных между клиентом и сервером, а также реализацию механизмов управления сессиями и взаимодействия в реальном времени.

1. Анализ технического задания

1.1. Общие требования

Мобильное приложение должно взаимодействовать с существующим сервером API социальной сети, обеспечивая пользователям удобный и функциональный интерфейс для выполнения следующих действий:

- Регистрация и авторизация пользователя с использованием сессий (cookie-файлов).
- Отправка и приём сообщений между пользователями в реальном времени.
- Получение и отображение данных пользователей, их профилей и публикаций.
- Поддержка уведомлений о новых сообщениях и действиях других пользователей.
- Обеспечение безопасной передачи данных между клиентом и сервером через HTTPS.

1.2. Функциональные требования

1. **Авторизация и аутентификация**:
    - Приложение должно реализовать механизм авторизации пользователя с использованием API. После успешной аутентификации сервер отправляет сессионные данные (COOKIE), которые клиентское приложение должно сохранять и использовать для дальнейших запросов к API.
    - Реализовать механизм автоматической проверки актуальности сессии, а также её продление при необходимости.
2. **Работа с профилем пользователя**:
    - Приложение должно обеспечивать возможность просмотра профиля пользователя, включая его фото, личные данные, подписчиков и публикации.

- Возможность редактирования профиля, включая изменение аватара и личных данных.

3. **Обмен сообщениями**:

   - Реализовать механизм отправки и приёма сообщений с использованием WebSocket-соединения. Сообщения должны передаваться в реальном времени.
   - В случае временной потери соединения приложение должно сохранять непрочитанные сообщения и отображать их пользователю при восстановлении связи.

4. **Публикации и лента**:

   - Возможность получения и отображения ленты новостей, содержащей публикации пользователей. Лента должна обновляться автоматически при появлении новых постов.
   - Пользователи могут создавать новые публикации, добавлять текст, изображения и комментарии.

5. **Уведомления**:

   - Реализовать систему уведомлений для оповещения пользователей о новых сообщениях, комментариях или упоминаниях в публикациях.
   - Уведомления должны приходить как в фоновом режиме, так и в активной сессии пользователя.

1.3. Нефункциональные требования

1. **Безопасность**:

   - Приложение должно использовать защищённый протокол HTTPS для передачи данных, что гарантирует безопасность данных пользователей.
   - Сервер использует SSL-сертификат для шифрования данных. Все запросы должны передаваться с использованием HTTPS, обеспечивая конфиденциальность информации.

- Для предотвращения утечек данных должны быть реализованы механизмы проверки подлинности сессий и шифрования паролей на стороне клиента перед отправкой.

2. **Производительность**:
    - Приложение должно поддерживать быструю загрузку данных и их обновление при взаимодействии с API.
    - Взаимодействие через WebSocket-соединение должно быть оптимизировано для минимизации задержек при передаче сообщений.
3. **Интерфейс и пользовательский опыт**:
    - Приложение должно быть интуитивно понятным, с продуманной навигацией и дизайном, что обеспечит пользователю положительный опыт взаимодействия.
    - Должна быть реализована поддержка различных экранов мобильных устройств, что потребует адаптации дизайна под различные размеры и разрешения дисплеев.

1.4. Описание работы API

API сервера социальной сети предоставляет доступ к данным пользователей, включая их профили, публикации, сообщения и другие взаимодействия. Основные особенности работы API:

- **Формат данных**: Все данные передаются в формате JSON, что позволяет легко их обрабатывать на стороне клиента.
- **Методы взаимодействия**: Большая часть запросов реализована через HTTPS с использованием методов GET для получения данных и POST для отправки данных на сервер.
- **Авторизация через COOKIE**: После успешной авторизации сервер передаёт клиенту cookie-файл, который используется для последующих запросов для проверки сессии пользователя.

- **WebSocket для обмена сообщениями**: API поддерживает двустороннюю передачу данных в реальном времени через WebSocket-соединение, что используется для отправки и получения мгновенных сообщений.

1.5. Интеграция технологий

Для реализации требований, указанных в техническом задании, будут использованы следующие технологии:

- **HTTPS и SSL**: Для обеспечения безопасной передачи данных между клиентом и сервером.
- **WebSocket**: Для реализации двустороннего обмена данными в реальном времени.
- **JSON**: Для стандартизированной передачи данных между клиентом и сервером.
- **Кроссплатформенные фреймворки**: Такие как Flutter или React Native, для создания единого кода, работающего на разных мобильных платформах.

2. Разработка алгоритмов

Для реализации мобильного приложения, в котором требуется работа с интернет-ресурсами, были разработаны несколько ключевых алгоритмов. Одним из них является создание POJO (Plain Old Java Object) классов для представления данных, получаемых от сервера, и их дальнейшая обработка в приложении.

2.1. Создание POJO классов

Для представления данных, получаемых от сервера, были созданы следующие POJO классы:

- **Photo**: Представляет изображение, которое может быть связано с постом.

```
data class Photo (
    val response: Int?,  // Поле ответа от сервера
    val id: Int?,
    val album: Int?,
    val file: String?,
    val key: String?,
    val rating: Rating?
): Serializable
```

- **Post**: Представляет пост, который может включать в себя текст, фотографии и информацию о пользователе.

```
data class Post (
    val response: Int,
    val text: String?,
    val photo: List<Photo>?,
    var user: User?,
    val date: Int?,
    val rating: Rating?,
    val userId: Int?,
    val date_edit: Int?,
    val id: Int?
): Serializable
```

- **PostList**: Объект, представляющий список постов, возвращаемых с сервера.

```
data class PostList(
    val response: Int,
    val list: List<Post>?
): Serializable
```

- **Rating**: Содержит информацию о рейтинге поста.

```
data class Rating (
    val like: Int?,
    val dislike: Int?,
    val type: Int?,
    val comment: Int?
): Serializable
```

- **User**: Класс, который описывает пользователя, который может быть автором поста или чьей-то страницей.

```
data class User(
    val response: Int?,
    val id: Int?,
    val name: String?,
    val tag: String?,
    val status: String?,
    val dateBirth: String?,
    val icon: Photo?,
    val cover: Photo?,
    val verified: Boolean?,
    val group: Boolean?,
    val musician: Boolean?,
    val popular: Boolean?,
    val countFriend: Int?,
    val countSubscribers: Int?,
    val countSubscriptions: Int?,
    val online: Int?
): Serializable
```

- **UserList**: Содержит список пользователей.

```
data class UserList(
    val response: Int,
    val list: Array<User>?
): Serializable
```

- **Verification**: Класс для обработки ответа на запросы подтверждения пользователя.

```
data class Verification(
    val response: Int,
    val key: String?
)
```

2.2. Интерфейс для работы с API

Для взаимодействия с сервером был создан интерфейс ApiService, содержащий методы для выполнения различных запросов, таких как авторизация, получение данных о пользователях и постах. Все методы аннотированы с использованием библиотеки Retrofit и соответствуют REST API.

Пример интерфейса для авторизации и работы с постами:

```
interface ApiService {

    @POST("api/login")

    @FormUrlEncoded

    fun login(

        @Field("email") email: String,

        @Field("password") password: String

    ): Call<Verification>


    @POST("api/login-confirm")

    @FormUrlEncoded

    fun verifyCode(

        @Field("key") key: String,

        @Field("code") code: String

    ): Call<User>


    @POST("api/session")

    fun session(): Call<User>


    @POST("api/get_user")
```

```
@FormUrlEncoded
fun getUser(
    @Field("user") user: Int
): Call<User>

@POST("api/get_post")
@FormUrlEncoded
fun news(
    @Field("list") list: Int,
    @Field("all") all: Boolean
): Call<PostList>

@POST("api/get_post")
@FormUrlEncoded
fun listPost(
    @Field("list") list: Int,
    @Field("user") user: Int
): Call<PostList>

@POST("api/get_post")
@FormUrlEncoded
fun post(
    @Field("id") id: Int,
    @Field("user") user: Int
```

```
    ): Call<Post>


    @POST("api/delete_post")
    @FormUrlEncoded
    fun deletePost(
        @Field("post") post: Int
    ): Call<Post>
}
```

2.3. Обработка сессий и куки

Для обеспечения устойчивой работы с сессиями был создан класс **AddCookiesInterceptor**, который отслеживает сохраненные сессии и добавляет куки в заголовки всех запросов.

Пример класса, который добавляет куки:

```
class AddCookiesInterceptor(private val context: Context) : Interceptor {
    override fun intercept(chain: Interceptor.Chain): Response {
        val sharedPreferences = context.getSharedPreferences("user_prefs",
Context.MODE_PRIVATE)
        val sessionId = sharedPreferences.getString("sessionId", null)

        val originalRequest = chain.request()
        val builder = originalRequest.newBuilder()

        if (sessionId != null) {
            // Добавляем куки в заголовок
            builder.addHeader("Cookie", "sessionId=${sessionId}")
```

```
        }


        return chain.proceed(builder.build())
    }
}
```

Этот алгоритм позволяет автоматически передавать информацию о текущей сессии при каждом запросе, что обеспечивало стабильную работу приложения и улучшало пользовательский опыт.

3. Руководство программиста

Для успешной работы над мобильным приложением, разработанным с использованием Android Studio и языка программирования Kotlin, программист должен обладать рядом навыков и ресурсов.

3.1. Требования к знаниям

1. **Знание Kotlin или Java**: Для разработки мобильного приложения на платформе Android требуется знание хотя бы одного из двух основных языков программирования: Kotlin или Java. В данном проекте использовался язык Kotlin, который является официально поддерживаемым языком для разработки Android-приложений. Однако знание Java также будет полезным, так как большая часть Android API написана на этом языке.
2. **Опыт работы с Android SDK и Android Studio**: Важным элементом является опыт работы с Android SDK (Software Development Kit), который содержит необходимые инструменты для разработки, тестирования и отладки приложений для Android. Android Studio, в свою очередь, предоставляет интегрированную среду разработки, оптимизированную для работы с Android SDK. Программисту необходимо знать основные принципы работы с этой средой, такие как создание проектов, настройка зависимостей, использование инструментов отладки и профилирования.

3.2. Требования к оборудованию

1. **Мощное оборудование**: Разработка с использованием Android Studio требует достаточно производительных машин. Это связано с тем, что Android Studio является ресурсозатратным приложением, особенно когда используется эмулятор для тестирования приложений. Эмулятор Android создает виртуальное устройство, что требует значительных вычислительных ресурсов для правильной работы и быстрой загрузки. Поэтому важно, чтобы рабочая станция имела:
    - Многоядерный процессор (желательно i7 или выше);
    - Не менее 8 ГБ оперативной памяти, но для комфортной работы рекомендуется 16 ГБ;
    - SSD для быстрого доступа к данным;
    - Видеокарта с поддержкой OpenGL для работы с эмуляторами.
2. **Реальное устройство для тестирования**: Несмотря на наличие эмуляторов в Android Studio, рекомендуется проводить тестирование на реальных устройствах для более точной оценки производительности приложения, а также для проверки его работы в реальных условиях.

Эмулятор, хотя и удобен, может не всегда точно отображать поведение на реальном устройстве.

3.3. Рекомендации для разработчиков

1. **Версии Android Studio и SDK**: Для стабильной работы рекомендуется использовать последнюю стабильную версию Android Studio. Также важно следить за обновлениями Android SDK, чтобы иметь доступ к новым инструментам и улучшениям, особенно с учетом быстрых изменений в экосистеме Android.
2. **Ресурсы для обучения**: Рекомендуется использовать официальные материалы Google, такие как документация Android Developer, а также онлайн-курсы и форумы для решения возникающих вопросов и обмена опытом с другими разработчиками.
3. **Версионный контроль**: Для обеспечения кооперации в команде разработки и безопасной работы с кодом важно использовать системы контроля версий, такие как Git. Хранение и управление кодом через репозитории на GitHub или GitLab позволяет разработчикам отслеживать изменения, работать с ветками и легко интегрировать код с другими частями проекта.

4. Руководство пользователя

Мобильное приложение предоставляет пользователю удобный интерфейс для взаимодействия с его аккаунтом и социальными сетями. В приложении доступны следующие основные функции:

4.1. Вход в аккаунт

Для того чтобы начать использовать приложение, пользователю необходимо авторизоваться. Для этого нужно войти в аккаунт, созданный на официальном сайте. После ввода логина и пароля, пользователь получает доступ ко всем функциям приложения.

4.2. Просмотр и удаление постов

После успешной авторизации пользователь может:

- Просматривать свои посты, которые были опубликованы в социальной сети.
- Удалять посты, которые больше не интересуют или не актуальны. Для этого достаточно выбрать нужный пост и нажать кнопку "Удалить".

4.3. Просмотр новостей

В приложении пользователю доступны два типа новостей:

1. **Новости друзей**: Это лента новостей, содержащая посты и активности людей, добавленных в друзья. Пользователь может отслеживать обновления своих знакомых и взаимодействовать с их постами.
2. **Общие новости**: Лента новостей, содержащая посты от других пользователей, которые не являются друзьями, но могут быть интересны в зависимости от темы.

4.4. Просмотр страниц пользователей

Пользователь может просматривать страницы других пользователей, даже если они не находятся в списке друзей. На странице пользователя отображается его информация, фото, статус и посты. Это позволяет быть в курсе событий других людей и, при необходимости, начать общение.

Заключение

В процессе разработки мобильного приложения на платформе Android с использованием языка Kotlin и Android Studio был решен ряд ключевых задач, направленных на создание функционального и удобного интерфейса для пользователей. Приложение предоставляет возможности для авторизации, управления постами, просмотра новостей и взаимодействия с пользователями, что соответствует современным стандартам мобильной разработки.

Особое внимание было уделено работе с API и интеграции с сервером для получения и отправки данных. В ходе разработки использовались такие инструменты, как Retrofit для взаимодействия с REST API, а также создание POJO классов для представления получаемых данных. Для удобства работы с сессиями был реализован механизм добавления cookies в запросы, что обеспечивало стабильность и безопасность взаимодействия с сервером.

Кроме того, в проекте была учтена важность высокой производительности и надежности. Для тестирования приложения использовался как эмулятор, так и реальные устройства, что позволило убедиться в корректности работы функционала в различных условиях.

В целом, проект продемонстрировал успешную реализацию ключевых принципов мобильной разработки, таких как взаимодействие с интернет-ресурсами, обработка данных, работа с пользовательским интерфейсом и оптимизация производительности. Данное приложение может служить основой для создания более сложных и масштабируемых мобильных решений.

Список литературы:

Введение в разработку приложений для смартфонов на ОС Android / А.Семакова – М.: Национальный открытый Университет «ИНТУИТ», 2016

Колисниченко Д.Н. Программирование для Android 5. Самоучитель. — СПб.: БХВ-Петербург, 2015. — 303 с.

Дейтел П., Дейтел Х., Уолд А. Android для разработчиков. 3-е изд. — СПб.: Питер, 2016.

Гриффитс Дэвид, Гриффитс Дон Head First. Программирование для Android. 2-е изд. — СПб.: Питер, 2018. — 912 с.

Приложение 1 — Основные ссылки

Код программы [Электронный ресурс]: https://github.com/m9agrest/mobile (дата обращения: 21 декабря 2024).

Основной сайт [Электронный ресурс]: https://adapt-key.com (дата обращения: 21 декабря 2024).

Скачать приложение [Электронный ресурс]: https://github.com/m9agrest/mobile/blob/main/mobile.apk (дата обращения: 21 декабря 2024).

Приложение 2 — Скриншоты программы

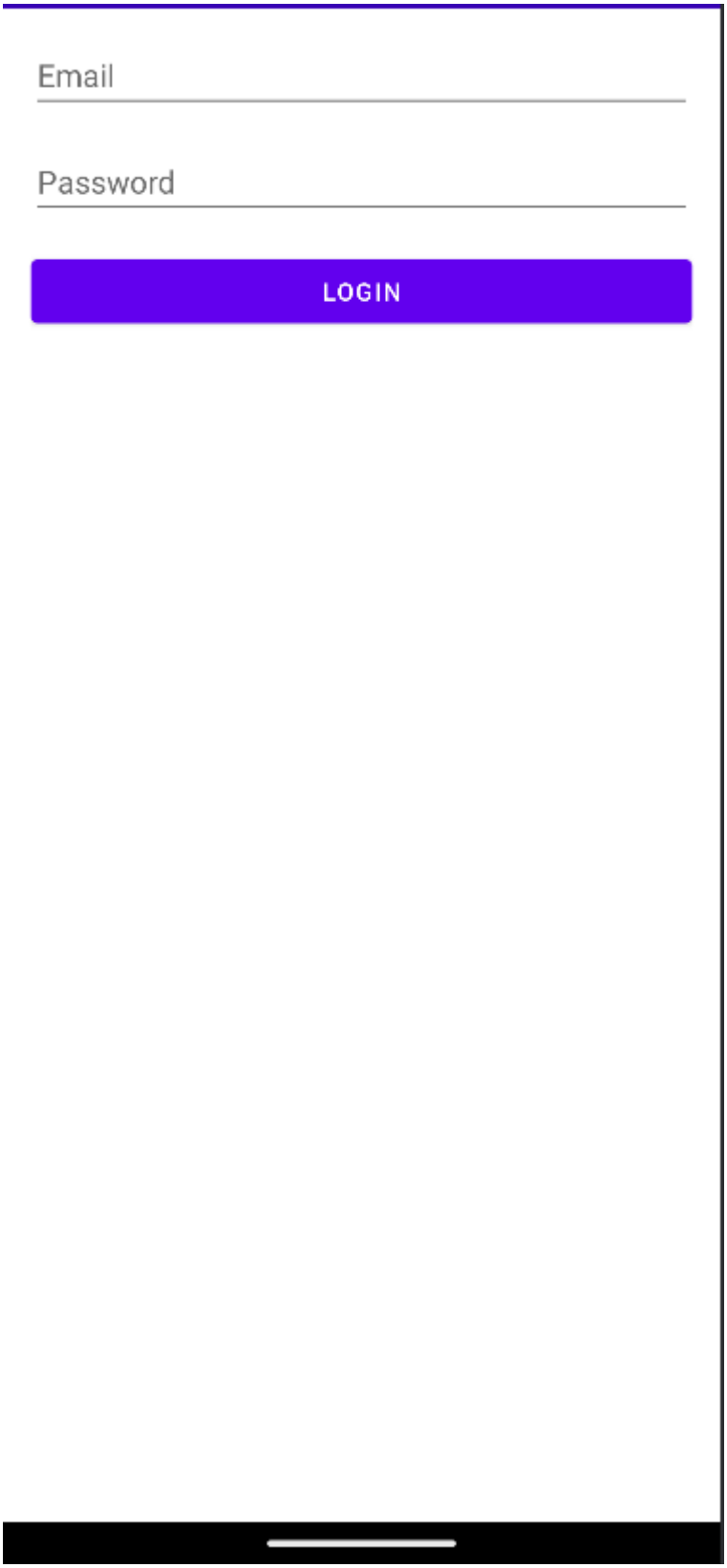

Рисунок 1 — Окно входа в приложение

Рисунок 2 — Профиль пользователя

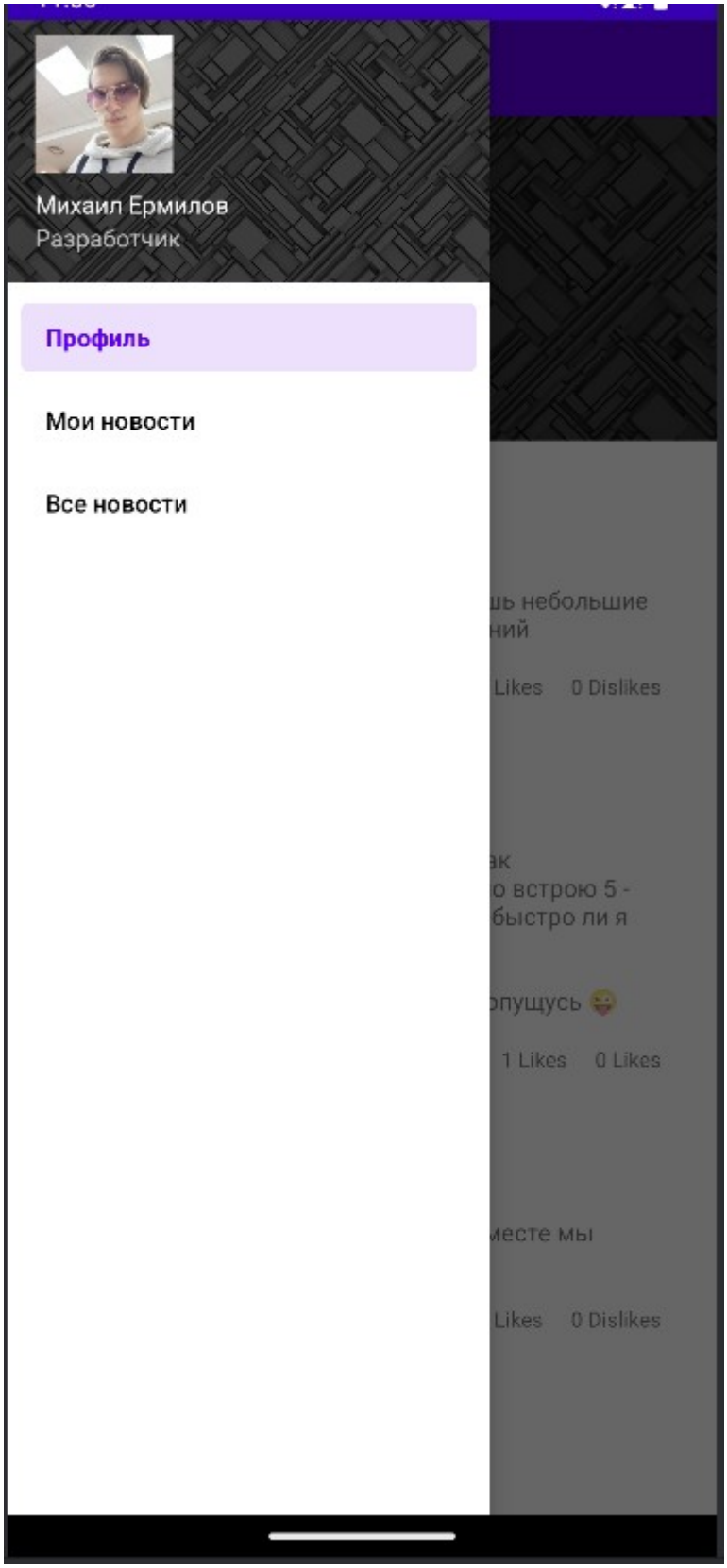

Рисунок 3 — Меню

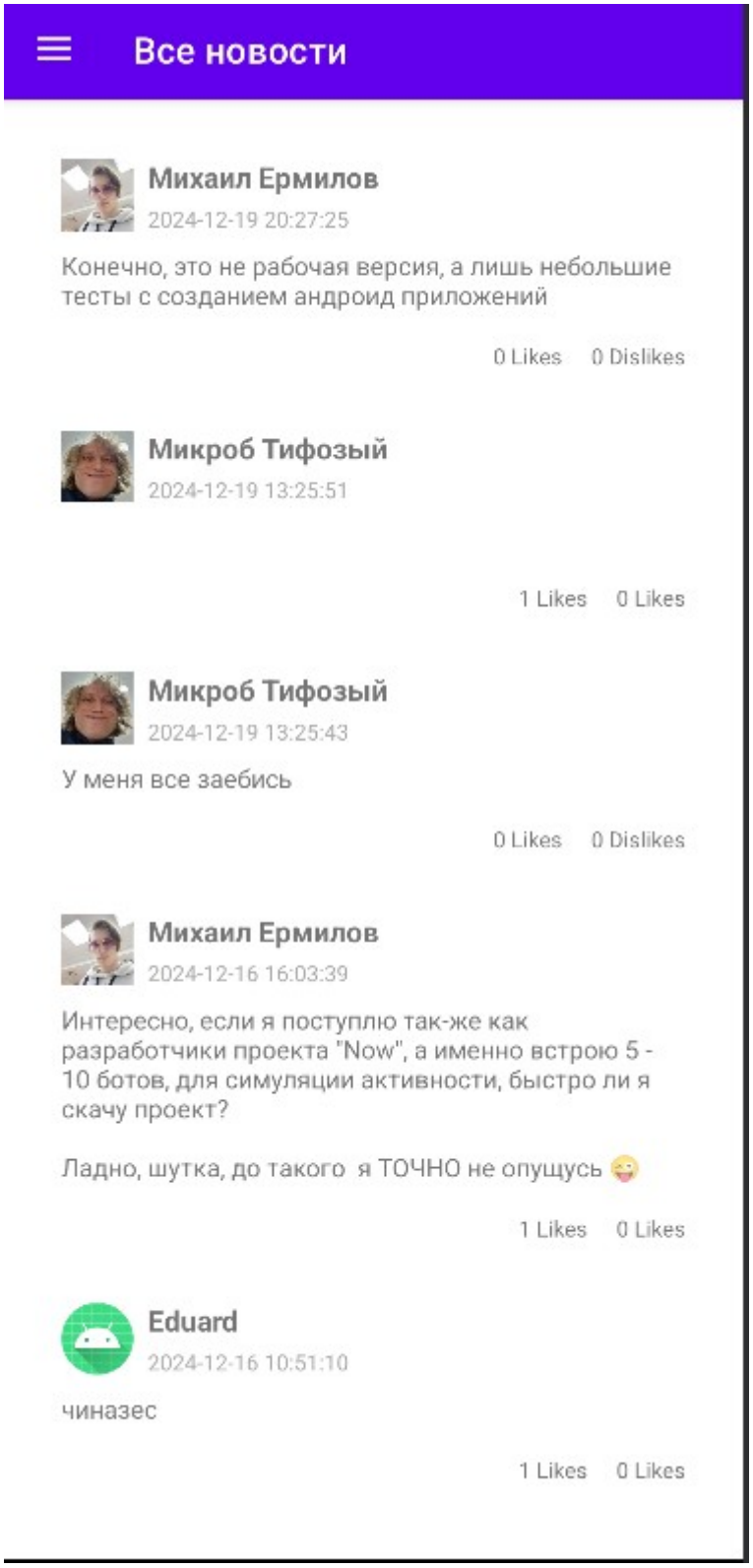

Рисунок 4 — Лента новостей

Рисунок 5 — страница другого пользователя