Содержание

Аннотация

Сапер - игра-головоломка, в которой представлена сетка кликабельных клеток с скрытыми минами, разбросанными по игровому полю. Цель игры состоит в том, чтобы пометить все клетки с минами и открыть все клетки, не содержащие мин. Игрок постепенно открывает клетки (либо помечает их заминироватнными), используя подсказки о количестве соседних мин в открытых клетках.

1 Ключевые слова

Сапер (англ. Minesweeper) - игра-головоломка, главной задачей которой является найти все «заминированные» клетки.

Android — операционная система для мобильных устройств, разработка приложений под которую ведется преимущественно на языках Java и Kotlin.

Kotlin — объектно-ориентированный язык программирования, с статической типизацией, совместимый с Java. Часто используется для мобильной разработки под различные операционные системы, в том числе Android.

Android Credential Manager API - библиотека Android Jetpack, которая объединяет поддержку API для большинства основных методов аутентификации, включая пароли, ключи доступа и решения для интегрированного входа (например, вход с помощью аккаунта Google).

Yandex Mobile Ads SDK — набор библиотек для показа объявлений из Рекламной сети Яндекса в мобильных приложениях на Android, iOS, Unity и Flutter. С помощью SDK можено размещать в приложениях рекламу разных форматов и управлять ей.

Android SDK для API ВКонтакте - набор библиотек, позволяющий быстро интегрировать API ВКонтакте в приложение для Android. SDK упрощает использование API ВКонтакте в Android-приложениях. Позволяет реализовать возможность пройти авторизацию без ввода логина и пароля.

Firebase - платформа для разработки приложений, которая помогает создавать и развивать приложения и игры. Основной сервис платформы — облачная СУБД класса NoSQL, позволяющая разработчикам приложений хранить данные и синхронизировать их между несколькими клиентами. Поддержаны особенности интеграции с приложениями под операционные системы Android и iOS, реализован API для приложений на JavaScript, Java, Objective-C и Node.js.

2 Введение

Цель данного проекта: разработать мобильную игру Сапер для Android на языке Kotlin.

Задачи:

- Изучить язык Kotlin;
- Научиться работать с Android Credential Manager API;
- Научиться работать с Yandex Mobile Ads SDK для Android;
- Научиться работать с Android SDK для API ВКонтакте;
- Разработать прототип приложения
- Разработать и внедрить дизайн приложения
- Доработать приложение и интегрировать рекламу;
- Протестировать приложение;
- Опубликовать приложение в RuStore.

Сама игра является реализацией классического Сапера с выбором уровня сложности и возможностью воспользоваться подсказками при прохождении, которые можно получить за просмотр рекламных роликов.

Для сохранения лучших результатов игроков в приложении реализована авторизация с помощью e-mail и пароля. Также реализованы альтернативные способы входа: с помощью Google-аккаунта и с помощью аккаунта ВКонтакте. Данные пользователей хранятся в базе данных Firebase.

3 Обзор литературы

Существуют различные подходы к реализации раскрытия клеток поля. Некоторые из них описаны в статье [7]. Самый простой устроен следующим образом: открывается только та клетка, на которую нажал игрок. При этом если в клетке нет мины и в ней не написано число заминированных соседей (то есть это такая незаминированная клетка, все соседние клетки с которой можно безопасно открыть - будем называть такую клетку пустой), игрок должен вручную нажать на все еще нераскрытые соседние клетки.

Чтобы исключить эту бесполезную работу, я реализовал второй вариант раскрытия клеток. Если игрок открыл пустую клетку, все соседние с ней раскрываются рекурсивно (то есть если какая-то соседняя клетка также оказалась пустой, то еще не раскрытые ее соседи

тоже раскрываются, и так далее). Таким образом, раскрывается вся компонента пустых клеток, достижимых из открытой игроком клетки, а также граница этой компоненты, которая состоит из незаминированных клеток, каждая из которых обязана иметь заминированного соседа.

Есть и более продвинутые методы раскрытия клеток. Например, можно усовершенствовать программу, чтобы она выставляла флажки на тех клетках, в которых однозначно находится мина (например, если есть раскрытая клетка с числом «1» в ней и все соседи этой клетки, кроме одного, раскрыты, можно утверждать, что в этот сосед заминирован и его необходимо пометить флажком). Можно пойти дальше: реализовать алгоритм, который генерирует множество возможных расстановок мин в зависимости от текущего состояния раскрытых клеток, находит клетки, которые во всех случаях определяются одинаково, и открывает/помечает их флажками.

Я остановился на втором варианте, поскольку он позволяет сделать игру не скучной, открывая самые очевидные клетки, и при этом не делает слишком много действий за игрока. Также я учитывал размеры поля в своей реализации игры - я не стал реализовывать уровни сложности с огромными полями, как в некоторых десктопных версиях «Сапера», в которых могут потребоваться алгоритмы, делающие за игрока больше работы, так как иначе не получится проходить уровни за разумное время. Также выбранный мной метод оптимален по количеству потребляемой памяти и времени работы - он не потребляет дополнительной пямяти (поскольку хранится только одно игровое поле) и работает за линейное относительно числа клеток время.

4 Структура приложения

В приложении реализовано три основных класса: `MainActivity` и `GameActivity`, наследующие класс `AppCompatActivity`, и класс `GameFieldView`, наследующий класс `View`.

Описание класса MainActivity

Класс MainActivity реализует главный экран приложения. В нём реализована авторизация пользователя через Google и VK, а также регистрация новых пользователей. Данный класс также отвечает за навигацию пользователя по основным функциональным элементам приложения, таким как начало игры, просмотр информации о приложении и выход из приложения.

Основные методы

- `signIn`: Метод для запуска процесса аутентификации через Google.

- `onActivityResult`: Метод, вызываемый при получении результата от другой активности или фрагмента. Обрабатывает результаты аутентификации через VK и Google.
- `firebaseAuthWithGoogle`: Метод для аутентификации пользователя в Firebase с использованием токена Google.
- `showSignInWindow()`: Метод для отображения диалогового окна входа в систему (по e-mail и паролю).
- `showRegisterWindow()`: Метод для отображения диалогового окна регистрации нового пользователя.

Описание класса GameActivity

В классе GameActivity реализовано отображение информации об игре: число очков, лучший результат, время, прошедшее с начала игры и число доступных подсказок. Также в этом классе реализовано взаимодействие с рекламным контентом - переключатель типа SwitchCompat становится доступным после просмотра рекламных роликов.

Основные методы:

- `loadRewardedAd`: Метод для загрузки рекламы с возможностью получения награды.
- `showAd`: Метод для отображения рекламы с возможностью получения награды.
- `onClick`: Метод, обрабатывающий нажатия на кнопки.
- `updateHintSwitchAvailability`: Метод для обновления доступности переключателя режима подсказок.

Описание класса GameFieldView

`GameFieldView` - пользовательский вид, представляющий игровое поле игры «Сапёр». Этот класс отвечает за отображение игрового поля, обработку пользовательских взаимодействий и логику игры. Взаимодействие с GameActivity (для передачи текущего числа подсказок и числа очков игрока, а также состояния игры) происходит посредством листенеров.

Внутренний класс Cell:

`Cell` представляет отдельную ячейку на игровом поле и хранит информацию о её состоянии.

Основные методы:

- `onSizeChanged`: Вызывается при изменении размера вида.
- `onDraw`: Отрисовывает игровое поле на холсте.

- `onTouchEvent`: Обрабатывает события касания и делегирует их обработку `GestureDetector`.
- `generateBoard`: Генерирует игровое поле с заданными размерами и расставляет мины случайным образом.
- `revealCell`: Раскрывает ячейку по координатам и обновляет состояние игры в зависимости от результата.
- `resetGame`: Сбрасывает текущую игру и начинает новую.

Листенеры:

- `OnScoreChangeListener`: Слушатель изменений счёта игрока.
- `OnHintsCountChangeListener`: Слушатель изменений количества оставшихся подсказок.
- `OnGameEndListener`: Слушатель события окончания игры.

Прочее

Вспомогательный класс `User` представляет модель пользователя в приложении. Он содержит информацию о пользователе, необходимую для аутентификации и хранения данных в базе данных.

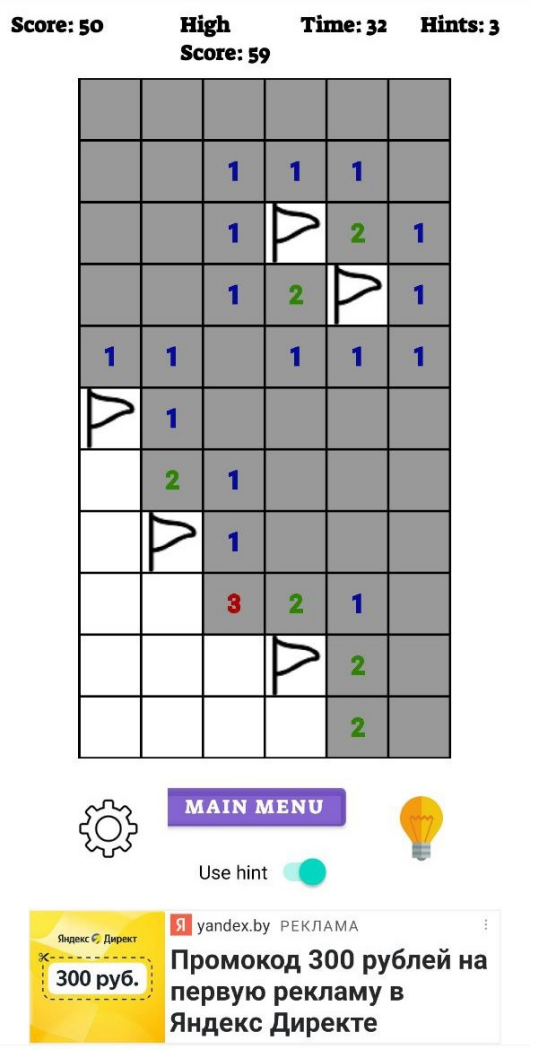
Класс `DifficultyDialog` представляет диалоговое окно для выбора уровня сложности игры.

5 Игровой процесс

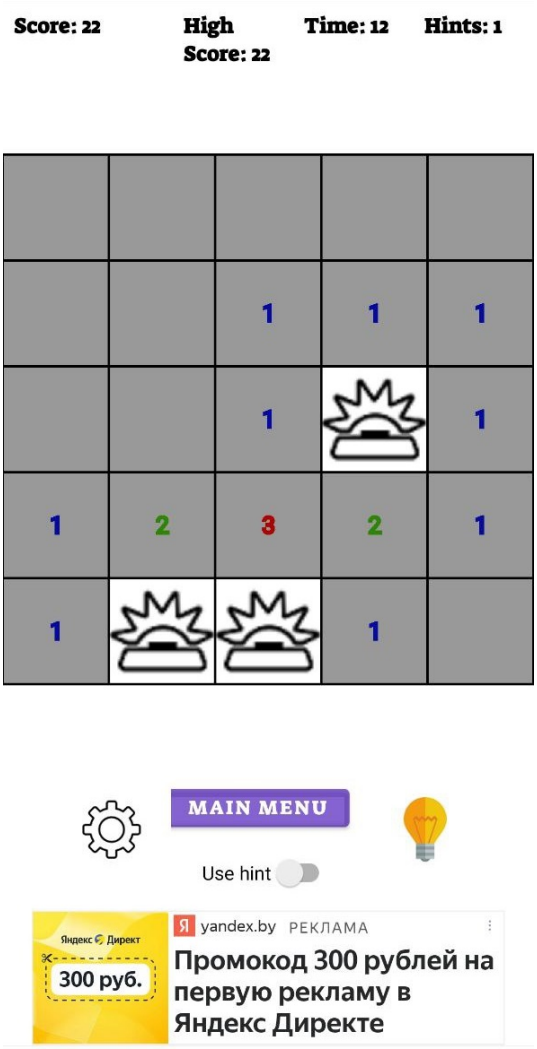
У игрока есть возможность выбрать один из пяти уровней сложности (чем выше уровень сложности, тем больше размеры поля и количество мин на нем).

Генерация расстановки мин происходит после первого хода игрока, чтобы исключить возможность первым ходом открыть клетку с миной. Раскрытие клеток поля происходит рекурсивно (подробнее этот процесс описан в разделе 3).

Процесс прохождения игры - см. Рисунок 5.1a и Рисунок 5.1b.



(a) 5.1a - Игровой процесс



(b) Рисунок 5.1b - Победа

Пометить клетку как заминированную можно долгим нажатаем на нее. При этом на ней появится флажок. При коротком нажатии на клетку, помеченную флажком, ничего не происходит. Убрать флажок можно также долгим нажатием на клетку.

При включенном режиме «Use hint» можно открыть любую клетку поля, которая еще не открыта и не помечена флажком, не опасаясь проиграть. Если в клетке оказалась мина, она открывается в точности как клетка без мины. После открытия клетки число доступных

подсказок уменьшается на 1.

Игра заканчивается поражением, если игрок открывает заминированную клетку.

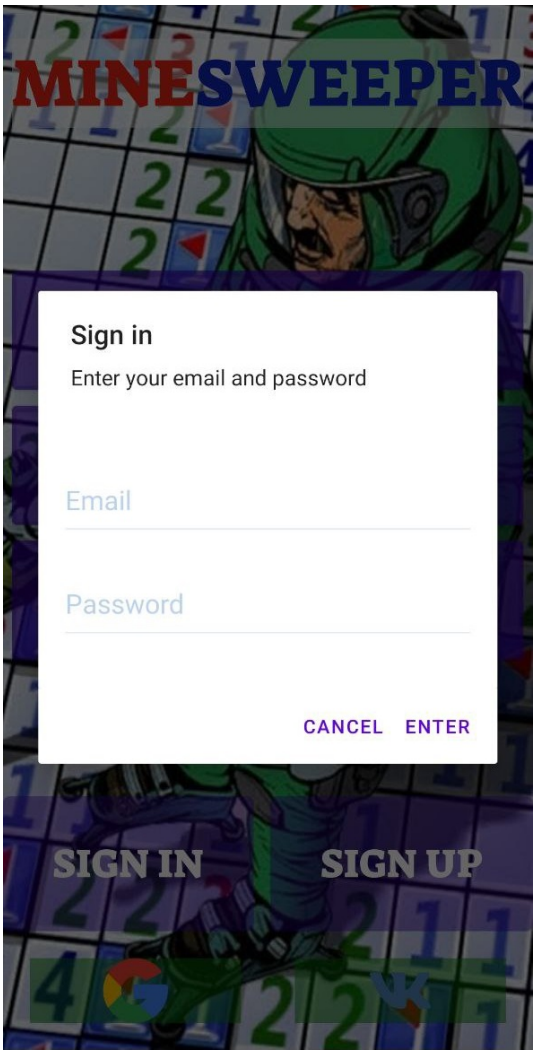
Когда все клетки, свободные от мин, открыты, а все заминированные клетки помечены флажками, игра успешно пройдена. В этом случае открываются заминированные клетки и останавливается таймер.

6 Пользовательский интерфейс

Про реализацию навигации и уведомлений в приложении, а также тестирование отдельных функций я прочитал в книге Alex Forrester "How to Build Android Apps with Kotlin-[1].

Дизайн окон регистрации и авторизации с использованием e-mail и пароля реализован с использованием виджета CardView [2]. Данный виджет позволяет отображать окна регистрации и авторизации поверх главного меню. Это показано на Рисунок 6.1a.

Для отображения всплывающих окон с оповещениями об успешной регистрации и успешной/неуспешной авторизации использован класс Snackbar [3]. Всплывающее уведомление об успешном входе показано на Рисунок 6.1b.



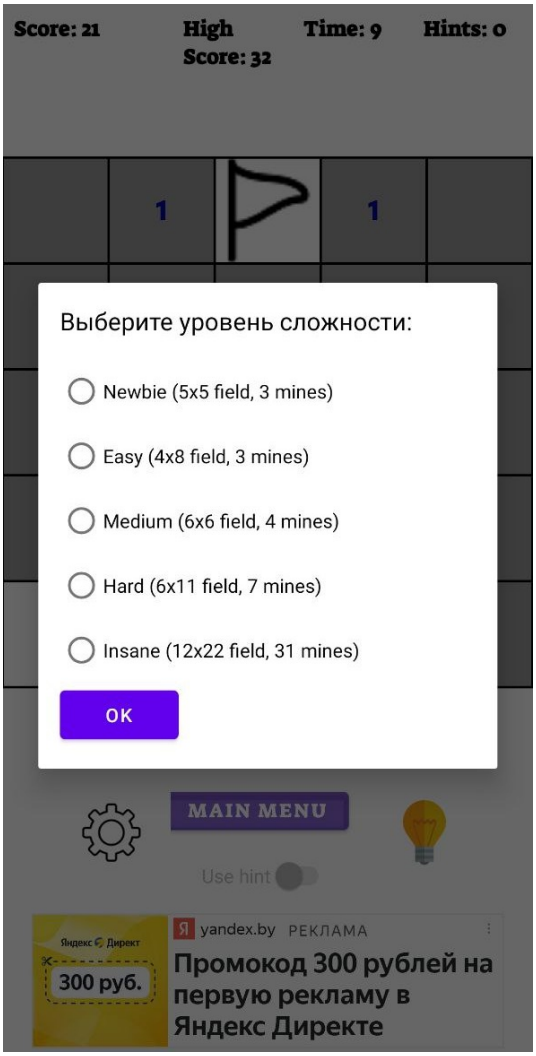
(a) Рисунок 6.1a - Окно входа с помощью логина и пароля

(b) Рисунок 6.1b - Уведомление об успешной авторизации

Текстовые поля для ввода информации пользователем реализованы с помощью библиотеки MaterialEditText [10], которая позволяет отображать различные эффекты, например подсвечивать поле с паролем, если введенный пароль не соотвествует требованиям или отображать подсказку, что требуется ввести в конкретном поле.

Для удобства пользователя игровое поле отображается с помощью SwipeRefreshView - чтобы начать новую игру достаточно провести по экрану сверху вниз.

Пользователю доступен выбор уровня сложности: Рисунок 6.2a.



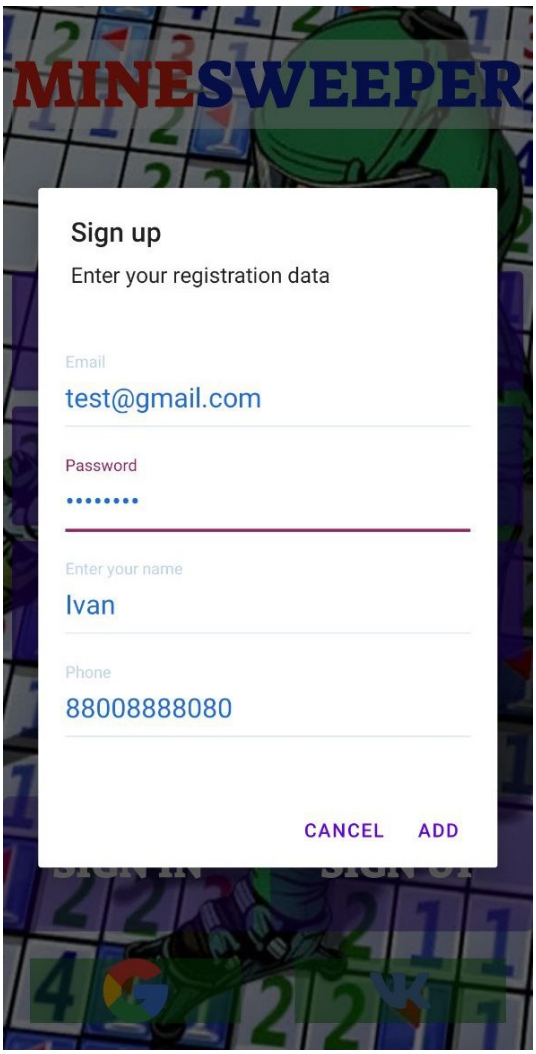
(a) Рисунок 6.2a - Выбор уровня сложности

7 Авторизация в приложении

Реализована регистрация пользователя с помощью e-mail и номер телефона: Рисунок 7.1a. Для проверки формата введенного пользователем e-mail и номера телефона, а также надежности пароля используются регулярные выражения из библиотеки java.util.regex. В случае несоответствия введенных данных установленному формату пользователь получает соответствующее уведомление: Рисунок 7.1b.

Авторизация в приложении реализована по e-mail и паролю, которые пользователь вводит при регистрации.

Релизована авторизация с помощью Google-аккаунта с использованием Android Credential

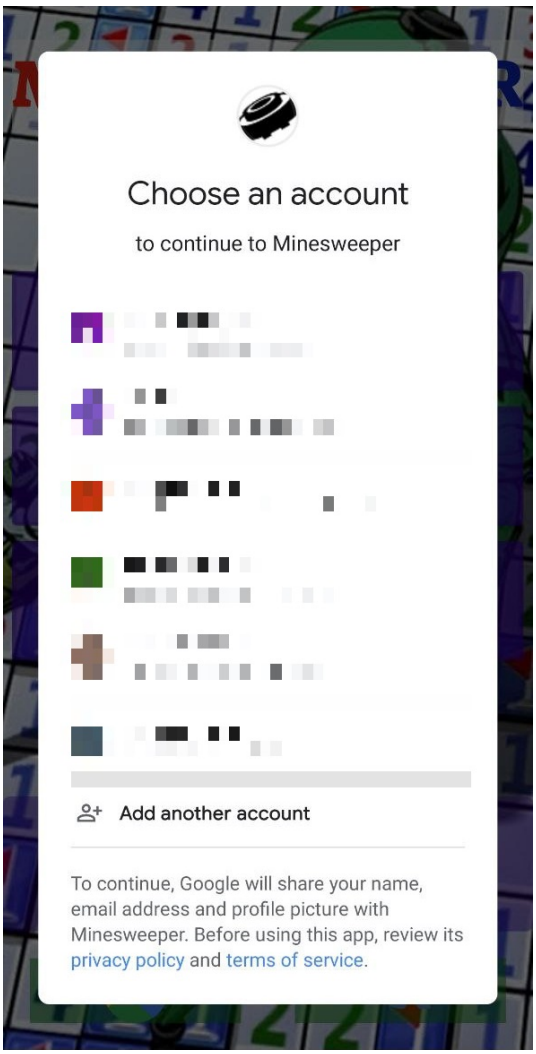


(a) Рисунок 7.1a - Окно регистрации с помощью e-mail и номер телефона

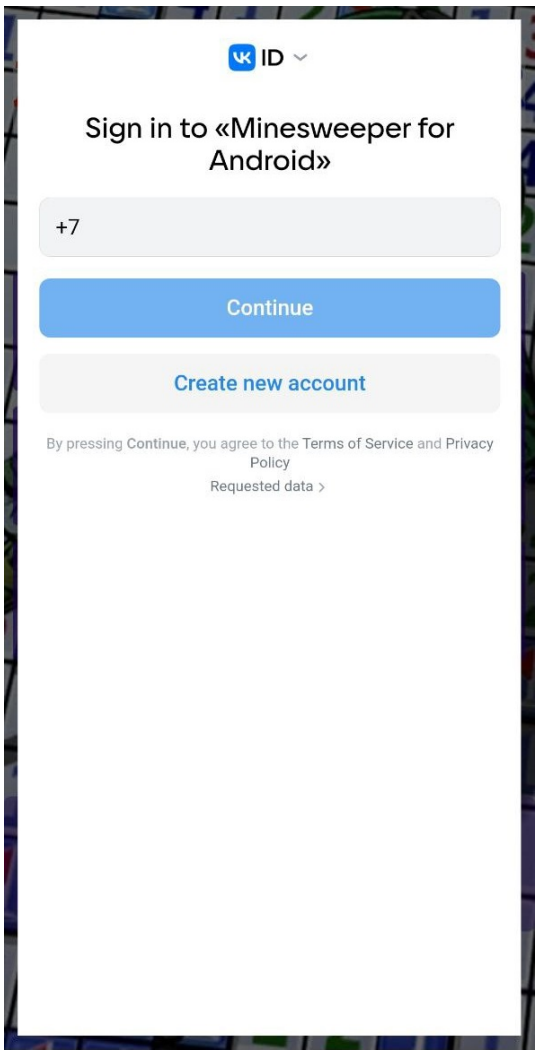
(b) Рисунок 7.1b - Ошибка регистрации из-за некорректного e-mail

Manager API [6] (Рисунок 7.2a) и с помощью аккаунта ВКонтакте с использованием VK Sign-In API [8] (Рисунок 7.2b).

Данные пользователей: e-mail, номер телефона, имя и пароль при регистрации вручную и индетификатор акканута Google\ВКонтакте при регистрации черех них, а также лучший результат хранятся в базе данных Google Firebase Database [4].



(a) Рисунок 7.2a - Окно регистрации с помощью Google аккаунт

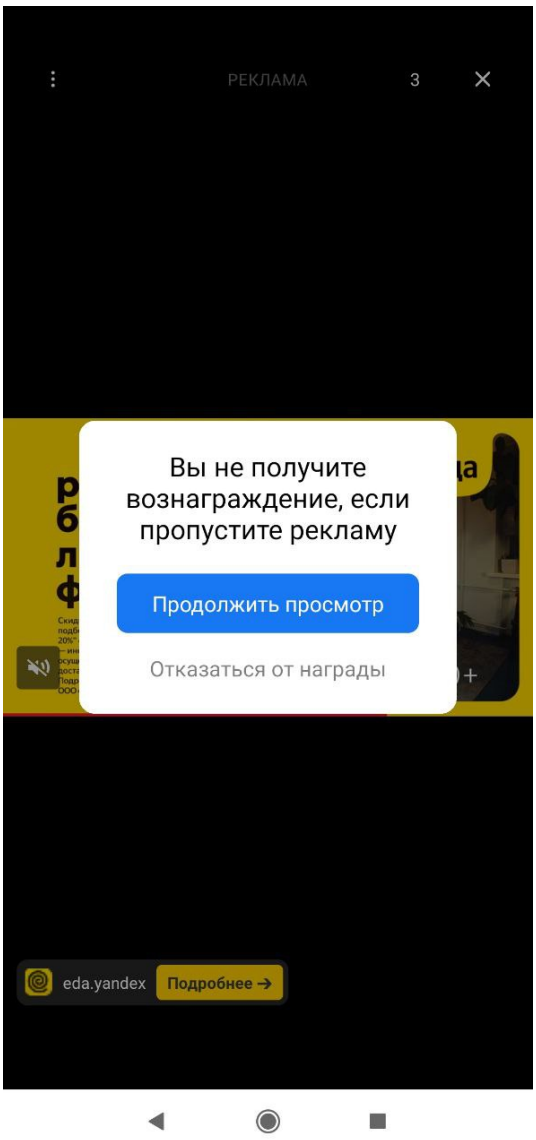


(b) Рисунок 7.2b - Окно регистрации с помощью аккаунт ВКонтакте

8 Реклама в приложении

Изначально я планировал использовать Google Ads SDK [5] для реализации рекламы в виде баннера (Banner Ads) и просмотра рекламных роликов для получения подсказок в раскрытии клеток поля (Rewarded Ads). Данный сервис Google позволяет достаточно удобно реализовать эти функции. Также он связан с Google Firebase, что упрощает работу. Но, как оказалось, из-за прекращения работы части сервисов Google в России, использовать Google Ads SDK не представляется возможным.

Поэтому я использовал Yandex Mobile Ads SDK [9] для отображения рекламы в приложении. Во время игры отображается реклама в виде баннера (Рисунок 5.1b), а также у пользователя есть возможность получать подсказки в прохождении игры за просмотр коротких видео : Рисунок 8.1a.



(а) Рисунок 8.1a - Реклама, за просмотр которой игрок получает вознаграждение

9 Заключение

В рамках курсового проекта я:

- Изучил язык Kotlin;

- Научился работать с Android Credential Manager API;

- Научился работать с Yandex Mobile Ads SDK для Android;

- Научился работать с Android SDK для API ВКонтакте;

- Разработал прототип приложения

- Разработал и внедрил дизайн приложения

- Доработал приложение и интегрировал рекламу;

- Протестировал приложение;

- Опубликовал приложение в RuStore.

В ходе реализации проекта возникли проблемы, связанные с прекращением работы части сервисов Google в России, но их удалось решить с использованием отечественных аналогов от компаний Yandex и ВКонтакте.

Для дальнейшего развития проекта запланировано:

1). Приспособить приложение под гаджеты с различной диагональю и разрешением дисплея.

2). Добавить больше уровней сложности с возможностью увеличения и прокрутки игрового поля для удобства пользователя.

3). Оптимизировать код - некоторые функции можно реализовать более эффективно.

10 Материалы

Github-репозиторий с кодом проекта: https://github.com/ivanz851/minesweeper_remastered.

Приложение доступно для скачивания в RuStore: ссылка для скачивания.

Список литературы

[1] Alexandru Dumbravan Alex Forrester Eran Boudjnah. *How to Build Android Apps with Kotlin*. Packt Publishing, 2023.

[2] android.com. *Create a card-based layout*. URL: https://developer.android.com/develop/ui/views/layout/cardview (дата обр. 20.04.2024).

[3] android.com. *Snackbar*. URL: https://developer.android.com/reference/com/google/android/material/snackbar/Snackbar (дата обр. 20.04.2024).

[4] google.com. *Firebase documentation*. URL: https://firebase.google.com/docs (дата обр. 20.04.2024).

[5] google.com. *Google Mobile Ads SDK documentation*. URL: https://developers.google.com/admob/android/sdk (дата обр. 20.04.2024).

[6] google.com. *Integrating Google Sign-In into Your Android App*. URL: https://developers.google.com/identity/sign-in/android/sign-in (дата обр. 20.04.2024).

[7] Magnus Hovland Hoff. *Solving Minesweeper*. URL: https://magnushoff.com/articles/minesweeper/ (дата обр. 20.04.2024).

[8] vk.com. *Руководство по использованию Android SDK для API ВКонтакте*. URL: https://dev.vk.com/ru/sdk/android (дата обр. 20.04.2024).

[9] yandex.ru. *Yandex Mobile Ads SDK для Android*. URL: https://yandex.ru/support2/mobile-ads/ru/dev/platforms (дата обр. 20.04.2024).

[10] Kai Zhu. *MaterialEditText*. URL: https://github.com/rengwuxian/MaterialEditText (дата обр. 20.04.2024).