Санкт-Петербургское государственное бюджетное

профессиональное образовательное учреждение

«Ижорский колледж»

КУРСОВАЯ РАБОТА

ТЕМА: «Разработка мобильного приложения «Календарь событий»

по ПМ.01 МДК.01.03 Разработка мобильных приложений

Выполнил

обучающийся группы 331с

Храпенко Павел Андреевич

Проверил

преподаватель спец. дисциплин

специальности 09.02.07

Информационные системы и программирование

Венедиктов Д.В.

Санкт-Петербург, 2025ЗАДАНИЕ ДЛЯ ВЫПОЛНЕНИЯ КУРСОВОЙ РАБОТЫ

Цель данной курсовой работы заключается в разработке мобильного приложения, отвечающего современным требованиям пользователей и особенностям платформы. На первом этапе необходимо провести исследование целевой аудитории и определить её потребности, что послужит основой для дальнейшей разработки.

Важным аспектом является создание прототипа приложения, в котором будут учтены все основные функции и пользовательский интерфейс. Рекомендуется использовать методологии Agile или Scrum для гибкой разработки, что позволит вовремя вносить изменения по мере необходимости.

При реализации приложения следует учитывать принципы юзабилити и дизайна, чтобы обеспечить максимальный комфорт и удовлетворение пользователей. Неотъемлемой частью работы станет тестирование: как функциональное, так и пользовательское, что позволит выявить и исправить возможные ошибки.

Завершением курсовой работы станет представление итогового проекта, включающего в себя документацию и описание процесса разработки. Ожидается, что разработанное приложение будет иметь потенциал для дальнейшего улучшения и внедрения в реальную среду.

ПЛАН-ГРАФИК ВЫПОЛНЕНИЯ КУРСОВОЙ РАБОТЫ

|  |  |  |
| --- | --- | --- |
| Дата | Содержание работ | Отметка о выполнении |
| 09.09 | Выбор темы курсовой работы |  |
| 16.09 | Знакомство с методическими рекомендациями по выполнению курсовой работы |  |
| 23.09 | Составление развёрнутого плана выполнения курсовой работы |  |
| 07.10 | Работа с нормативно-правовыми документами, учебной литературой |  |
| 21.10 | Анализ предметной области. Постановка задачи |  |
| 28.10 | Составление введения к курсовой работе в соответствии с методическими рекомендациями |  |
| 04.11 | Описание предметной области и функции решаемых задач |  |
| 11.11 | Выбор средств для выполнения курсовой работы. Выбор среды разработки |  |
| 25.11 | Разработка интерфейса (frontend-разработка) |  |
| 13.01 | Разработка логики работы приложения (backend-разработка) |  |
| 17.02 | Разработка подключаемой базы данных (backend -разработка) |  |
| 24.03 | Составление заключения к курсовой работе |  |
| 31.03 | Составление списка источников информации, используемых при выполнении курсовой работы |  |
| 07.04 | Подготовка курсовой работы к защите |  |

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 2](#_if3tck2ngil2)

[ГЛАВА 1 ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ РАЗРАБАТЫВАЕМОГО МОБИЛЬНОГО ПРИЛОЖЕНИЯ 2](#_gc1dw3cjo0fi)

[1.1 Анализ предметной области 2](#_t79mp5lftm87)

[1.1.1 Исследование рынка существующих мобильных приложений похожей тематики 2](#_8fhd4bcmkwo2)

[1.1.2 Выявление требований к разрабатываемому мобильному приложению 2](#_960qn9y83ou)

[1.1.3 Анализ технологий для разработки мобильного приложения 2](#_2ulgzw3vmwr)

[1.2 Постановка задачи курсовой работы 2](#_3loi5s7pwbda)

[1.2.1 Цели курсовой работы 2](#_gfup73x2bsj2)

[1.2.2 Задачи курсовой работы 2](#_v9b29oyluec4)

[1.2.3 Оценка рисков разработки мобильного приложения 2](#_apk70m8fek9j)

[1.2.4 Техническое задание на разработку мобильного приложения 2](#_vajbfr8b6pfy)

[1.3 Описание предметной области и функции решаемых задач 2](#_2mc3aj2r4j0g)

[1.4 Выбор программных средств для разработки мобильного приложения 2](#_h8pg3fohgutk)

[1.5 Выбор среды для разработки мобильного приложения 2](#_rofd9pdy1kqd)

[ГЛАВА 2 РЕАЛИЗАЦИЯ РАЗРАБАТЫВАЕМОГО МОБИЛЬНОГО ПРИЛОЖЕНИЯ 2](#_22kq7f3pssy8)

[2.1 Разработка интерфейса мобильного приложения (frontend-разработка) 2](#_b7l1s297k3pb)

[2.2 Разработка логики работы мобильного приложения (backend-разработка) 2](#_vxvwjj74f0w2)

[2.3 Разработка подключаемой базы данных (backend-разработка) 2](#_cybnvzb1p66k)

[ЗАКЛЮЧЕНИЕ 2](#_hmk6bdja8anx)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 2](#_txou2hlbod56)

# ВВЕДЕНИЕ

В последние десятилетия мобильные технологии стали неотъемлемой частью нашей повседневной жизни. Смартфоны и планшеты являются основными инструментами для получения информации, общения и развлечения. В связи с этим, разработка мобильных приложений приобретает всё большее значение, как для бизнеса, так и для пользователей.

Мобильные приложения позволяют организациям улучшать взаимодействие с клиентами, оптимизировать процессы и повышать уровень сервиса. Пользователи, в свою очередь, ценят удобство, доступность и функциональность мобильных решений. Стремительное развитие технологий, таких как 5G, искусственный интеллект и облачные вычисления, открывает новые горизонты для создания инновационных приложений.

Цель данной курсовой работы – исследовать ключевые аспекты процесса разработки мобильных приложений, включая выбор платформы, языков программирования, а также принципов проектирования и тестирования. Также будет рассмотрен анализ существующих приложений и выделены основные тенденции, формирующие рынок мобильного ПО.

Актуальность выбранной темы обусловлена стремительным развитием мобильных технологий, а также растущей конкуренцией на рынке приложений. Успешная реализация мобильного приложения требует не только технических знаний, но и умения учитывать потребности целевой аудитории и трендов рынка.

# ГЛАВА 1 ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ РАЗРАБАТЫВАЕМОГО МОБИЛЬНОГО ПРИЛОЖЕНИЯ

## 1.1 Анализ предметной области

Разработка мобильных приложений – это динамично развивающаяся область, охватывающая множество аспектов, включая проектирование, программирование и тестирование приложений для мобильных устройств. В этом разделе представим ключевые элементы анализа предметной области.

### 1.1.1 Исследование рынка существующих мобильных приложений похожей тематики

Изучение уже существующих мобильных приложений, которые решают схожие задачи, является важным шагом в процессе разработки. Это позволяет не только понять, как другие разработчики справляются с подобными вызовами, но и выявить сильные и слабые стороны конкурентов.

Таблица 1 – Анализ рынка существующих мобильных приложений похожей тематики

|  |  |  |  |
| --- | --- | --- | --- |
| Целевая аудитория | Молодёжь | Молодежь и среднее поколние | Молодежь |
| Слабые стороны | Премиум функции | Бронирование на стороннем сайте | Бронирование мест на сайте организатора мероприятий |
| Сильные стороны | «Выбор редакции» на главном экране, аудиогид, создание своих поездок | «Любимое направления туристов», мероприятия в городах на главном экране, выбор отелей | «Рекомендации», События в твоем городе, покупка билетов по программе «Пушкинская карта», единый аккаунт. |
| Основные функции | Поиск мест для гида в разные города | Поиск мест для гида в разные города | Выбор мест для помещений |
| Название приложения | izi.travel. | Tripadvisor | Госуслуги. Культура |

### 1.1.2 Выявление требований к разрабатываемому мобильному приложению

Приложение "Календарь событий" должно помогать пользователям эффективно анализировать и ориентироваться по событиям в городе.

Вот его ключевые функции:

* Оповещения о близлежащих событиях.
* Офлайн-доступ к ключевой информации.

Таблица 2 – Системные требования для разрабатываемого мобильного приложения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Операционная система | Процессор | Оперативная память | Разрешение экрана | Сетевые требования |
| Android 10.0 или выше | Минимальные требования: двухъядерный процессор с частотой 1.8 ГГц  Рекомендуемые требования: четырехъядерный процессор с частотой 2.0 ГГц или выше | 2 ГБ ОЗУ | Минимальные требования: 1280 x 720 пикселей (HD)  Рекомендуемые требования: 1920 x 1080 пикселей (Full HD) или выше | Wi-Fi или мобильная сеть |

### 1.1.3 Анализ технологий для разработки мобильного приложения

Целевой платформой разрабатываемого мобильного приложения является операционная система Android от 10.0

Используемые языки программирования: Kotlin, Java.

Используемые фреймворки: Android SDK, Jetpack Compose, React Native, Ionic.

Используемые библиотеки: Dagger, Redux, Retrofit, Room, RxJava.

Инструменты для разработки баз данных: SQLite.

## 1.2 Постановка задачи курсовой работы

Постановка задачи является важным шагом в процессе разработки мобильного приложения, так как она определяет цели, объем работы и основные требования к продукту.

### 1.2.1 Цели курсовой работы

### Приложение "Календарь событий" решает проблему просмотра событий в городе как для местных жителей, так и для туристов. В нём сосредоточены функции, упрощающие поиск информации о том когда какое событие города должно произойти и экономящие время пользователя.

### Ценность для пользователя:

### Удобство: Вся информация о событиях города собрана в одном месте.

### Офлайн-доступ: Информация доступна без интернета, что особенно полезно для туристов.

### Интерактивность: Возможность получения уведомлений о событиях которые должны произойти в скором времени.

### 1.2.2 Задачи курсовой работы

1. Регистрация и авторизация: Пользователи могут создавать учетные записи и входить в приложение, используя электронную почту, социальные сети или другие методы.
2. Личный кабинет: Пользователи могут управлять своими данными, добавлять интересы и настроения, чтобы получать персонализированные рекомендации.
3. Офлайн-доступ: Возможность загрузки информации для использования без интернет-соединения.
4. События и мероприятия: Оповещения о текущих и предстоящих городских событиях и мероприятиях.
5. Интеграция с социальными сетями: Возможность делиться событиями через социальные сети.

### 1.2.3 Оценка рисков разработки мобильного приложения

1. Технические сложности:
   1. Интеграция различных API и сервисов
   2. Обеспечение совместимости с различными устройствами и операционными системами
   3. Оптимизация производительности и скорости работы приложения
   4. Обеспечение безопасности данных пользователей
2. Недостаток времени:
   1. Невозможность завершить проект в установленные сроки
   2. Проблемы с управлением проектом и расстановкой приоритетов задач
3. Финансовые ресурсы:
   1. Ограниченный бюджет на разработку и тестирование
4. Проблемы с тестированием:
   1. Невозможность покрыть все возможные сценарии использования приложения
   2. Ошибки и баги, которые могут проявиться только при использовании в реальных условиях

Чтобы уменьшить возможные риски, можно предпринять следующие шаги:

* Планирование и управление проектом: Создание четкого плана с установленными сроками и этапами разработки.
* Адаптивное планирование: Быстрая адаптация плана в случае возникновения непредвиденных обстоятельств.
* Постоянное тестирование: Регулярное тестирование приложения на каждом этапе разработки.

### 1.2.4 Техническое задание на разработку мобильного приложения

Техническое задание (ТЗ) является основанием для разработки мобильного приложения, определяет его основные требования и функции. Это приложение будет разрабатываться для платформы Android.

Создание мобильного приложения, которое предоставляет пользователю удобный доступ к календарю событий, оффлайн-доступ, мероприятия и события, интеграция с социальными сетями.

**Основные функции приложения:**

* регистрация и авторизация пользователей:
  + возможность регистрации через электронную почту, социальные сети;
  + восстановление пароля.
* личный кабинет:
  + отображение информации о пользователе: имя, контактные данные, история посещений.
* основные функции приложения:
* Оффлайн-доступ: Доступ к данным без интернета.
* События: Оповещения о ближайших мероприятиях.
* Интеграция с социальными сетями: делиться маршрутами и местами, возможность оставлять отзывы и оценки (при необходимости).
* система уведомлений (пуш-уведомления о событиях или об изменениях в приложении)

**Технические требования:**

* приложение должно работать на устройстве под управлением операционной системы Android (10.0 или выше);
* время загрузки приложения не должно превышать 2-3 секунды;
* поддержка русского и английского языков на первом этапе разработки.

**Дизайн:**

* оформление должно быть современным, простым и интуитивно понятным для пользователя;
* элементы интерфейса должны быть адаптивными для различных размеров экранов.

Данное техническое задание описывает ключевые аспекты разработки мобильного приложения. Оно служит основой для взаимодействия между заказчиком и командой разработчиков.

## 1.3 Описание предметной области и функции решаемых задач

Разработка мобильных приложений охватывает создание программного обеспечения, которое предназначено для работы на мобильных устройствах, таких как смартфоны и планшеты. Эта область включает в себя проектирование, программирование, тестирование и поддержку программных приложений, которые могут использоваться в различных отраслях, включая, но не ограничиваясь, здравоохранением, образованием, финансами, электронной коммерцией и развлечениями.

Я выбрал разработку мобильного приложения, которое дает отслеживание мероприятий в вашем городе.

## 1.4 Выбор программных средств для разработки мобильного приложения

Основной целью разрабатываемого мобильного приложения является предоставление пользователям удобного и эффективного инструмента для исследования предстоящих событий города. Приложение поможет как местным жителям, так и туристам находить и получать актуальную информацию о мероприятиях и других событиях, улучшая их городской опыт.

В данной курсовой работе я использовал следующие программные средства для разработки мобильного приложения:

* язык программирования – Kotlin;
* инструменты для управления проектами – Git;
* тестирование приложения – фреймворк Espresso;
* создание документации к разрабатываемого мобильному приложению – язык разметки Markdown.

## 1.5 Выбор среды для разработки мобильного приложения

Средой разработки для мобильного приложения к курсовой работе я выбрал Android Studio.

# ГЛАВА 2 РЕАЛИЗАЦИЯ РАЗРАБАТЫВАЕМОГО МОБИЛЬНОГО ПРИЛОЖЕНИЯ

## 2.1 Разработка интерфейса мобильного приложения (frontend-разработка)

1. Возраст:

* Молодёжь (18-30 лет): активные пользователи смартфонов, стремящиеся интересно провести время.
* Взрослые (30-50 лет): пользователи, интересующиеся культурными событиями.

1. Социальные группы:
   * Туристы: путешественники, желающие узнать город и стать к нему ближе.
   * Местные жители: люди, стремящиеся узнать больше о праздниках своего города.
   * Студенты: молодёжь, ищущая интересные мероприятия.
   * Семьи: пользователи, ищущие информацию о семейных мероприятиях.

Аналогичные приложения выглядят следующим образом:

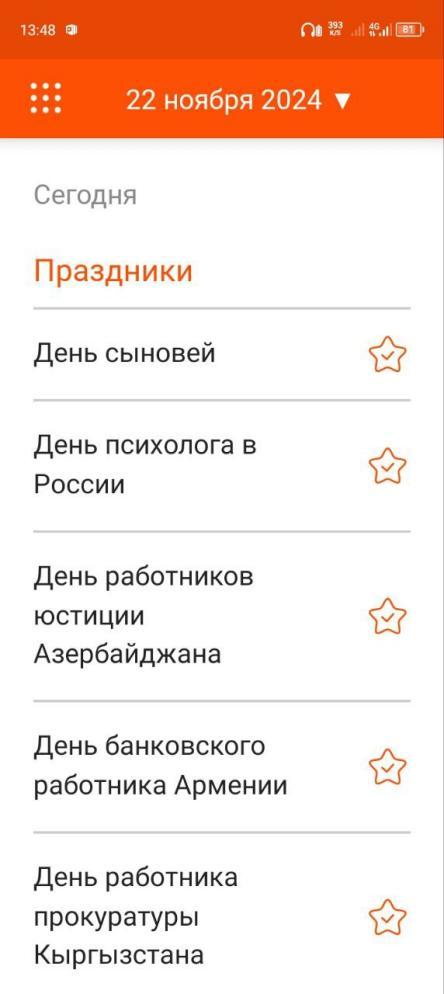


Рисунок 1 – главное окно приложения «calend.ru.»

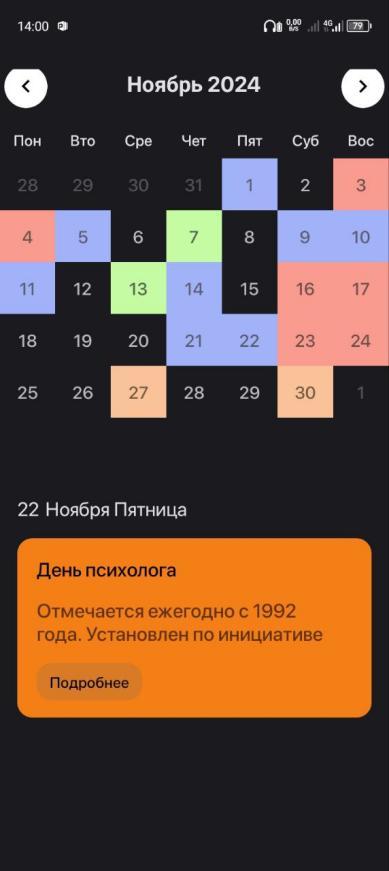


Рисунок 2 – главное окно приложения «Календарь - праздники России»



Рисунок 3 – главное окно приложения «Праздники России»



Рисунок 4 – главное окно разрабатываемого приложения

«Календарь событий»



Рисунок 5 - Личный кабинет пользователя

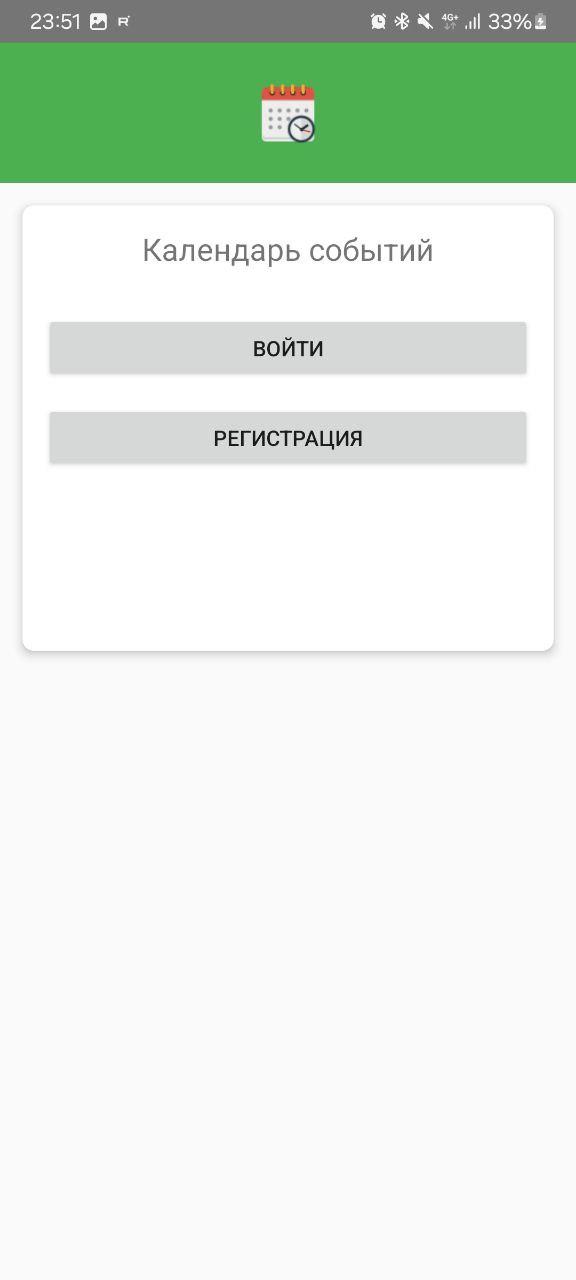


Рисунок 6 - Регистрация пользователя



Рисунок 8 - Услуги

Таблица 3 – Необходимые экраны разрабатываемого приложения

|  |  |
| --- | --- |
| Название экрана | Скриншот |
| Регистрация пользователя | 1 скриншот |
| Восстановление пароля | 1 скриншот |
| Главный экран приложения | 1 скриншот |
| Личный кабинет пользователя | 1 скриншот |
| даты | 1 скриншот |
| Услуги | 1 скриншот |
| Информация |  |
| Экран отзыва (или оценки) |  |
| Экран пуш-уведомления |  |

**Описание регистрации пользователя в приложении**

1. Скачайте приложение: Найдите "Календарь событий" на личном хранилище и установите его на своё устройство.
2. Откройте приложение: после установки, откройте приложение.
3. Нажмите "Регистрация": на главном экране выберите опцию "Регистрация".
4. Введите данные: Введите свою электронную почту, создайте пароль, подтвердите его и пройдите капчу.
5. Примите условия использования: ознакомьтесь с условиями использования и политикой конфиденциальности, затем подтвердите своё согласие.
6. Подтвердите регистрацию: вам на почту придёт письмо с подтверждением. Перейдите по ссылке в письме для завершения регистрации.
7. Вход в приложение: после подтверждения регистрации, вернитесь в приложение и войдите, используя свою электронную почту и пароль.

**Описание восстановления пароля пользователя в приложении**

* шаг 1 – Откройте приложение и на экране входа нажмите "Забыли пароль?".
* шаг 2 – Введите свою электронную почту, указанную при регистрации, и нажмите "Отправить".
* шаг 3 – Перейдите в свою почту, откройте письмо с инструкциями по восстановлению пароля и следуйте шагам, чтобы создать новый пароль.

**Описание главного экрана приложения**

* Панель навигации с разделами «Вход с учетной записью»
* Кнопки Регистрации и Авторизации
* Логотип компании

**Описание «Войти»**

Позволяет войти в приложение, если у пользователя есть аккаунт в приложении

**Описание «Регистрация»**

Позволяет пользователю зарегистрировать аккаунт, если его нет

**Описание «Войти через социальные сети»**

Позволяет пользователю войти в приложение через социальные сети (Google, VK и т.д.)

**Описание личного кабинета пользователя в приложении**

* Панель навигации с разделами: «Обзор», «Избранное», «Мои места», «Геолокация», «Профиль»
* Кнопка Выхода, Настройки аккаунта, Социальные сети, История, Настройки
* Логотип компании

**Описание «Настройки аккаунта»**

Позволяет пользователю настраивать свой аккаунт

**Описание «Социальные сети»**

Показывает пользователю, какие социальные сети привязаны и делиться своими местами, где он/а побывал/а

**Описание «История»**

Хранит в себе места, где побывал пользователь (если делалось все через приложение)

**Описание «Настройки»**

Позволяет настраивать приложение, давать/убирать разрешения

**Описание «Выйти»**

Позволяет пользователю выйти из учетной записи приложения

В разрабатываемом мобильном приложении я планирую использовать преимущественно следующие цвета:

Таблица 4 – Используемые цвета в приложении

|  |  |
| --- | --- |
| Название цвета | HEX |
| Черный | 000000 |
| Серый | 808080 |
| Зеленый | 4CAF50 |
| Белый | FFFFFF |

Преимуществами разрабатываемого мобильного приложения с визуальной точки зрения являются:

* Интуитивный дизайн интерфейса: Простота использования, позволяющая пользователям легко находить нужную информацию и пользоваться всеми функциями приложения без необходимости в обучении.
* Эстетичность и привлекательность: Приложение будет приятно глазу благодаря использованию современных дизайнов, ярких цветов и качественной графики, что улучшит общий опыт пользователя.
* Адаптивный дизайн: Оптимизация отображения на различных устройствах и экранах, включая смартфоны и планшеты, обеспечивающая комфортное использование независимо от размера экрана.

## 2.2 Разработка логики работы мобильного приложения (backend-разработка)

**MainActivity**

Это главный экран приложения, который содержит два кнопки:

1. **Кнопка "Вход" (**loginButton**)** — при нажатии запускает LoginActivity, где пользователь может войти в аккаунт.
2. **Кнопка "Регистрация" (**registerButton**)** — при нажатии открывает RegisterActivity для регистрации нового пользователя.

private fun registerUser() {

val username = usernameField.text.toString()

val email = emailField.text.toString()

val password = passwordField.text.toString()

if (username.isEmpty() || email.isEmpty() || password.isEmpty()) {

Toast.makeText(this, "Заполните все поля!", Toast.LENGTH\_SHORT).show()

return

}

## 

## 2.3 Разработка подключаемой базы данных (backend-разработка)

* **registerUser()**

Этот метод обрабатывает регистрацию нового пользователя:

1. **Получает** данные из полей ввода usernameField, emailField, passwordField.
2. **Проверяет**, чтобы поля не были пустыми. Если что-то не заполнено — отображает всплывающее уведомление (Toast).
3. **Формирует** JSON-объект с данными пользователя.
4. **Создает HTTP-запрос** с использованием OkHttpClient для отправки данных на сервер (http://calendar/api/register.php).
5. **Отправляет запрос** на сервер:
   * Если сервер **не доступен** (например, проблемы с интернетом) — показывается сообщение об ошибке.
   * Если сервер **ответил**, данные преобразуются в JSONObject, и выводится ответ сервера.

* **Основные моменты**
* Используется OkHttpClient для сетевого взаимодействия.
* **JSON** передается через POST-запрос.
* Реализована **асинхронная** обработка сети (enqueue), чтобы приложение не зависало при ожидании ответа.
* runOnUiThread гарантирует, что Toast вызывается в главном потоке UI.

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

val loginButton = findViewById<Button>(R.id.login\_button)

val registerButton = findViewById<Button>(R.id.register\_button)

loginButton.setOnClickListener {

startActivity(Intent(this, LoginActivity::class.java))

}

registerButton.setOnClickListener {

startActivity(Intent(this, RegisterActivity::class.java))

}

}

1. Создание базы данных (DBHelper)

Используется SQLiteOpenHelper, который создаёт Calendar.db.

В методе onCreate() создаётся таблица Users, содержащая:

id – Уникальный идентификатор пользователя.

name – Имя пользователя.

email – Уникальный email.

password – Пароль.

Пример кода:

class DBHelper(context: Context) : SQLiteOpenHelper(context, "Calendar.db", null, 1) {

override fun onCreate(db: SQLiteDatabase) {

db.execSQL("CREATE TABLE Users (id INTEGER PRIMARY KEY, name TEXT, email TEXT, password TEXT)")

}

}

1. Открытие соединения

Перед выполнением операций с базой создаётся экземпляр DBHelper:

val dbHelper = DBHelper(context)

val db = dbHelper.writableDatabase

writableDatabase используется для записи.

readableDatabase для чтения данных.

1. Добавление данных

Данные пользователя вставляются через ContentValues:

val values = ContentValues().apply {

put("name", name)

put("email", email)

put("password", password)

}

db.insert("Users", null, values)

insert() записывает данные в таблицу Users.

password можно шифровать, чтобы повысить безопасность.

1. Чтение данных

Запрашиваем пользователей из базы данных:

val cursor = db.rawQuery("SELECT \* FROM Users", null)

rawQuery() выполняет SQL-запрос.

Можно использовать cursor.moveToNext() для перебора записей.

1. Закрытие соединения

После работы с базой данных обязательно закрываем соединение:

db.close()

Это освобождает ресурсы, предотвращая утечки памяти.

# ЗАКЛЮЧЕНИЕ

В результате выполнения проекта по разработке мобильного приложения я достиг поставленных целей и задач. Приложение было спроектировано с учетом потребностей пользователей, обеспечивая интуитивно понятный интерфейс и высокую производительность.

Я успешно справился с внедрением ключевых функций, таких как возможность регистрации через электронную почту, социальные сети; отображение в личном кабинете информации о пользователе: имени, контактных данных, истории посещений, возможностью оставлять отзывы и оценки, системы уведомлений.

Также стоит отметить, что тестирование приложения выявило и устранило множество потенциальных ошибок, что значительно повысило его надежность.

В будущем я планирую продолжать развитие приложения, добавляя новые функции и улучшая его функционал на основе обратной связи от пользователей.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Венгроу, Д. Прикладные структуры данных и алгоритмы. Прокачиваем навыки. / Д. Венгроу. – Санкт-Петербург : Издательство «Питер», 2024. – 512 с.
2. Вроблевски, Л. Сначала мобильные! пер. с англ. П. Миронова. /  
   Л. Вроблевски. – Москва : Издательство «Манн, Иванов и Фербер», 2022. – 176 с.
3. Гаятри, М. Фулстек-тестирование. / М. Гаятри. – Москва : Издательство «Спринт Бук», 2024. – 416 с.
4. Данн, М., Льюис, Ш. Нативная разработка мобильных приложений. Перекрестный справочник для iOS и Android. / М. Данн,  
   Ш. Льюис. – Москва : Издательство «ДМК Пресс», 2020. – 376 с.
5. Петросян, Л.Э., Гусев, К.В. Разработка мобильных приложений на языке Kotlin. / Л.Э. Петросян, К.В. Гусев. – Москва : Издательство «Лань», 2024. – 104 с.
6. Семенчук, В. Мобильное приложение как инструмент бизнеса. /  
   В. Семенчук – Москва : Издательство «Альпина Паблишер», 2017. – 240 с.
7. Скин, Д., Гринхол, Д. Kotlin. Программирование для профессионалов. 2-е изд. / Д. Скин, Д. Гринхол. – Санкт-Петербург : Издательство «Питер», 2023. – 560 с.
8. Соколова, В.В. Разработка мобильных приложений : учебное пособие для среднего профессионального образования / В.В. Соколова. – Москва : Издательство «Юрайт», 2024. – 160 с.
9. Тидвелл, Д., Брюэр, Ч., Валенсия, Э. Разработка интерфейсов. Паттерны проектирования. 3-е изд. / Д. Тидвелл, Ч. Брюэр, Э. Валенсия. – Санкт-Петербург : Издательство «Питер», 2022. – 560 с.
10. Фишерман, Л.В. Git. Практическое руководство. Управление и контроль версий в разработке программного обеспечения. / Л.В. Фишерман. – Санкт-Петербург : Издательство «Наука и техника», 2021. – 304 с.
11. Хортон, Дж. Разработка Android-приложений с нуля. 3-е изд. /  
    Дж. Хортон. – Санкт-Петербург : Издательство «БХВ», 2023. – 576 с.
12. https://developer.android.com/?hl=ru (дата обращения: 22.03.2025).
13. https://www.figma.com/ (дата обращения: 15.03.2025).
14. https://www.github.com/ (дата обращения: 16.03.2025).
15. https://www.reddit.com/r/learnprogramming/?rdt=60398 (дата обращения: 16.03.2025).
16. https://www.stackoverflow.com/ (дата обращения: 15.03.2025).
17. «Mobile App Development for Dummies» – *AppDevelopersIndia,* 13.06.2024. – https://developersappindia.com/blog/mobile-app-development-for-dummies-a-beginners-guide-to-getting-started (дата обращения: 15.10.2024).

ПРИЛОЖЕНИЯ

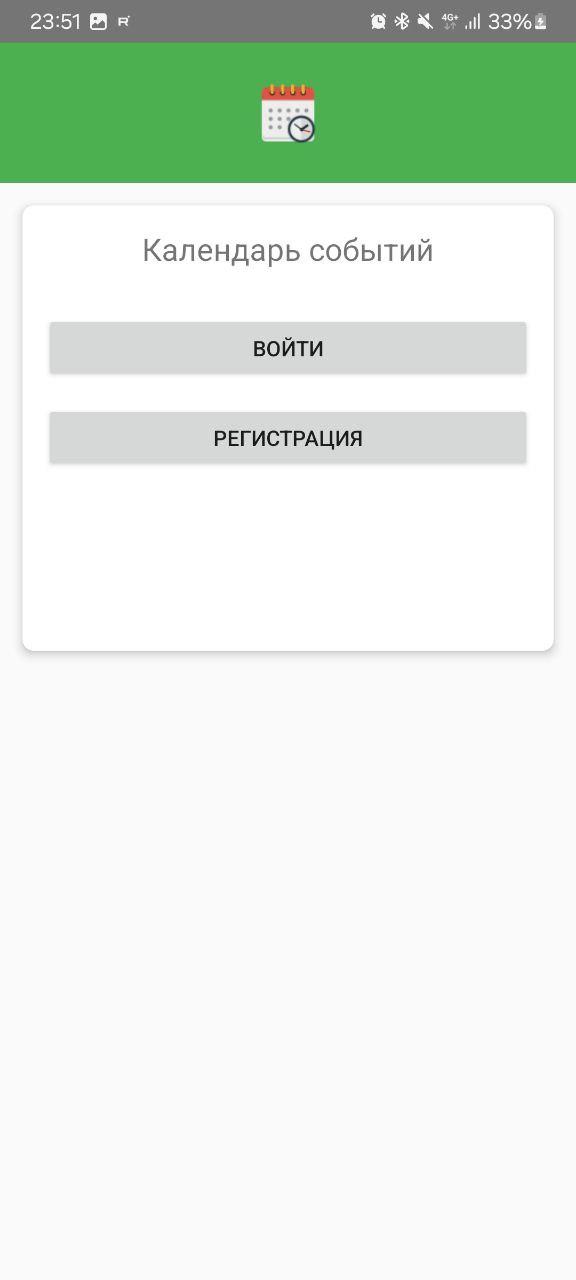


Рис. 1 – главный экран



Рис. 2 – экран регистрации



Рис. 3 – экран входа



Рисунок 4 – главное окно разрабатываемого приложения

«Календарь событий»



Рисунок 5 - Личный кабинет пользователя



Рисунок 6 - Услуги

* 1. MainActivity – главный экран при первом заходе в приложение

class MainActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main) // 🔹 Должен быть перед `findViewById`

val loginButton = findViewById<Button>(R.id.login\_button)

val registerButton = findViewById<Button>(R.id.register\_button)

loginButton.setOnClickListener {

startActivity(Intent(this, LoginActivity::class.java))

}

registerButton.setOnClickListener {

startActivity(Intent(this, RegisterActivity::class.java))

}

}

}

* 1. BaseActivity – начальная форма приложения (после логина или регистрации)

open class BaseActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_base)

val bottomNavigationView = findViewById<BottomNavigationView>(R.id.bottom\_navigation)

bottomNavigationView.setOnItemSelectedListener { item ->

when (item.itemId) {

R.id.nav\_home -> {

if (this !is HomeActivity) startActivity(Intent(this, HomeActivity::class.java))

true

}

R.id.nav\_places -> {

if (this !is PlacesActivity) startActivity(Intent(this, PlacesActivity::class.java))

true

}

R.id.nav\_profile -> {

if (this !is ProfileActivity) startActivity(Intent(this, ProfileActivity::class.java))

true

}

else -> false

}

}

}

}

* 1. ProfileActivity – экран профиля пользователя

class ProfileActivity : BaseActivity() {

private lateinit var logoutButton: Button

private lateinit var sharedPreferences: SharedPreferences

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_profile)

val bottomNavigationView = findViewById<BottomNavigationView>(R.id.bottom\_navigation)

bottomNavigationView.setOnItemSelectedListener { item ->

when (item.itemId) {

R.id.nav\_home -> {

startActivity(Intent(this, HomeActivity::class.java))

true

}

R.id.nav\_places -> {

startActivity(Intent(this, PlacesActivity::class.java))

true

}

R.id.nav\_profile -> {

startActivity(Intent(this, ProfileActivity::class.java))

true

}

else -> false

}

}

logoutButton = findViewById(R.id.logout\_button)

sharedPreferences = getSharedPreferences("UserSession", MODE\_PRIVATE)

logoutButton.setOnClickListener {

logoutUser()

}

}

private fun logoutUser() {

// Очищаем сохранённые данные и выходим из аккаунта

sharedPreferences.edit().apply {

putBoolean("loggedIn", false)

remove("email")

apply()

}

startActivity(Intent(this, MainActivity::class.java))

finish()

}

}

* 1. PlacesActivity – страница мест, где побывал пользователь (история)

class PlacesActivity : BaseActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_places)

val bottomNavigationView = findViewById<BottomNavigationView>(R.id.bottom\_navigation)

bottomNavigationView.setOnItemSelectedListener { item ->

when (item.itemId) {

R.id.nav\_home -> {

startActivity(Intent(this, HomeActivity::class.java))

true

}

R.id.nav\_places -> {

startActivity(Intent(this, PlacesActivity::class.java))

true

}

R.id.nav\_profile -> {

startActivity(Intent(this, ProfileActivity::class.java))

true

}

else -> false

}

}

}

}

* 1. HomeActivity – главный экран приложения, после успешного логина или регистрации

class HomeActivity : BaseActivity() {

private lateinit var foundSlider: ViewPager2

private lateinit var eventSlider: ViewPager2

private lateinit var nearbySlider: ViewPager2

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_home)

// Инициализация ViewPager2

foundSlider = findViewById(R.id.found\_for\_you\_slider)

eventSlider = findViewById(R.id.events\_slider)

nearbySlider = findViewById(R.id.nearby\_places\_slider)

val foundImages = listOf(R.drawable.image1, R.drawable.image2, R.drawable.image3)

val eventImages = listOf(R.drawable.event1, R.drawable.event2, R.drawable.event3)

val nearbyImages = listOf(R.drawable.place1, R.drawable.place2, R.drawable.place3)

foundSlider.adapter = ImageSliderAdapter(this, foundImages)

eventSlider.adapter = ImageSliderAdapter(this, eventImages)

nearbySlider.adapter = ImageSliderAdapter(this, nearbyImages)

startAutoScroll(foundSlider, foundImages)

startAutoScroll(eventSlider, eventImages)

startAutoScroll(nearbySlider, nearbyImages)

val bottomNavigationView = findViewById<BottomNavigationView>(R.id.bottom\_navigation)

bottomNavigationView.setOnItemSelectedListener { item ->

when (item.itemId) {

R.id.nav\_home -> {

startActivity(Intent(this, HomeActivity::class.java))

true

}

R.id.nav\_places -> {

startActivity(Intent(this, PlacesActivity::class.java))

true

}

R.id.nav\_profile -> {

startActivity(Intent(this, ProfileActivity::class.java))

true

}

else -> false

}

}

}

private fun startAutoScroll(viewPager: ViewPager2, imageList: List<Int>) {

val timer = Timer()

timer.schedule(object : TimerTask() {

override fun run() {

runOnUiThread {

val nextItem = (viewPager.currentItem + 1) % imageList.size

viewPager.setCurrentItem(nextItem, true)

}

}

}, 6000, 6000)

}

}

* 1. RegisterActivity – главный файл регистрации, включающую капчу и создания пользователя в бд

class RegisterActivity : AppCompatActivity() {

private lateinit var usernameField: EditText

private lateinit var emailField: EditText

private lateinit var passwordField: EditText

private lateinit var confirmPasswordField: EditText

private lateinit var registerButton: Button

private lateinit var captchaButton: Button

private var captchaPassed = false

private lateinit var dbHelper: DBHelper

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_register)

dbHelper = DBHelper(this)

usernameField = findViewById(R.id.username)

emailField = findViewById(R.id.email)

passwordField = findViewById(R.id.password)

confirmPasswordField = findViewById(R.id.confirm\_password)

registerButton = findViewById(R.id.register\_button)

captchaButton = findViewById(R.id.captcha\_button)

// Капча: подтверждаем, что пользователь не робот

captchaButton.setOnClickListener {

captchaPassed = true

Toast.makeText(this, "Капча пройдена!", Toast.LENGTH\_SHORT).show()

}

// Обработчик кнопки регистрации

registerButton.setOnClickListener {

registerUser()

}

}

private fun registerUser() {

val username = usernameField.text.toString()

val email = emailField.text.toString()

val password = passwordField.text.toString()

val confirmPassword = confirmPasswordField.text.toString()

if (username.isEmpty() || email.isEmpty() || password.isEmpty() || confirmPassword.isEmpty()) {

Toast.makeText(this, "Заполните все поля!", Toast.LENGTH\_SHORT).show()

return

}

if (password != confirmPassword) {

Toast.makeText(this, "Пароли не совпадают!", Toast.LENGTH\_SHORT).show()

return

}

if (!captchaPassed) {

Toast.makeText(this, "Пройдите капчу!", Toast.LENGTH\_SHORT).show()

return

}

dbHelper.insertUser(username, email, password)

Toast.makeText(this, "Регистрация успешна!", Toast.LENGTH\_SHORT).show()

startActivity(Intent(this, HomeActivity::class.java))

finish()

}

}

* 1. LoginActivity – главный файл по логину пользователя

class LoginActivity : AppCompatActivity() {

private lateinit var emailField: EditText

private lateinit var passwordField: EditText

private lateinit var loginButton: Button

private lateinit var sharedPreferences: SharedPreferences

private lateinit var dbHelper: DBHelper

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_login)

dbHelper = DBHelper(this)

emailField = findViewById(R.id.email)

passwordField = findViewById(R.id.password)

loginButton = findViewById(R.id.login\_button)

sharedPreferences = getSharedPreferences("UserSession", MODE\_PRIVATE)

// Проверяем, вошёл ли пользователь ранее

if (sharedPreferences.getBoolean("loggedIn", false)) {

startActivity(Intent(this, HomeActivity::class.java))

finish()

}

loginButton.setOnClickListener {

loginUser()

}

}

private fun loginUser() {

val email = emailField.text.toString()

val password = passwordField.text.toString()

if (email.isEmpty() || password.isEmpty()) {

Toast.makeText(this, "Заполните все поля!", Toast.LENGTH\_SHORT).show()

} else {

val loginResult = checkUserLogin(email, password)

if (loginResult == "Авторизация успешна") {

Toast.makeText(this, loginResult, Toast.LENGTH\_SHORT).show()

// Сохранение статуса пользователя

sharedPreferences.edit().apply {

putBoolean("loggedIn", true)

putString("email", email)

apply()

}

startActivity(Intent(this, HomeActivity::class.java))

finish()

} else {

Toast.makeText(this, loginResult, Toast.LENGTH\_SHORT).show()

}

}

}

private fun checkUserLogin(email: String, password: String): String {

val db = dbHelper.readableDatabase

var result = "Пользователь не найден"

val cursor = db.rawQuery("SELECT password FROM Users WHERE email = ?", arrayOf(email))

if (cursor.moveToFirst()) {

val storedPassword = cursor.getString(0)

result = if (storedPassword == password) "Авторизация успешна" else "Неправильный пароль"

}

cursor.close()

db.close()

return result

}

}

* 1. DBHelper – файл создания БД и внесения пользователя в таблицу User

class DBHelper(context: Context) : SQLiteOpenHelper(context, DATABASE\_NAME, null, DATABASE\_VERSION) {

companion object {

private const val DATABASE\_NAME = "Calendar.db"

private const val DATABASE\_VERSION = 1

}

override fun onCreate(db: SQLiteDatabase) {

val createTableQuery = """

CREATE TABLE Users (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

email TEXT NOT NULL UNIQUE,

password TEXT NOT NULL

)

"""

db.execSQL(createTableQuery)

}

override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {

db.execSQL("DROP TABLE IF EXISTS Users")

onCreate(db)

}

fun insertUser(name: String, email: String, password: String) {

val db = writableDatabase

val values = ContentValues().apply {

put("name", name)

put("email", email)

put("password", password)

}

db.insert("Users", null, values)

db.close()

}

}

* 1. ImageSliderAdapter – функция для прокрутки изображения

class ImageSliderAdapter(private val context: Context, private val imageList: List<Int>) :

RecyclerView.Adapter<ImageSliderAdapter.ImageViewHolder>() {

class ImageViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {

val imageView: ImageView = itemView.findViewById(R.id.image\_slider\_item)

}

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ImageViewHolder {

val view = LayoutInflater.from(context).inflate(R.layout.image\_slider\_item, parent, false)

return ImageViewHolder(view)

}

override fun onBindViewHolder(holder: ImageViewHolder, position: Int) {

holder.imageView.setImageResource(imageList[position])

}

override fun getItemCount(): Int = imageList.size

}