09.02.07 ПР-312

**ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

**ПП.01.01 Разработка мобильных приложений**

|  |  |  |
| --- | --- | --- |
| **Практикант** |  | **Филатов Н.А.** |
| **Руководители практики** |  | **Домбровский Н.С. Смирнова Е.Е..** |

Оглавление

[**Задание на разработку** 3](file:///C:\Users\nikit\AppData\Local\Temp\7d611b1e-e5d4-4a5f-990a-3fc9e448ac9c_Отчет%20ПП01.01%20Мазур.zip.c9c\Отчет%20ПП01.01%20Мазур.docx#_Toc201078210)

[**Анализ индивидуального задания** 4](file:///C:\Users\nikit\AppData\Local\Temp\7d611b1e-e5d4-4a5f-990a-3fc9e448ac9c_Отчет%20ПП01.01%20Мазур.zip.c9c\Отчет%20ПП01.01%20Мазур.docx#_Toc201078211)

[**Проектирование дизайна приложения** 6](file:///C:\Users\nikit\AppData\Local\Temp\7d611b1e-e5d4-4a5f-990a-3fc9e448ac9c_Отчет%20ПП01.01%20Мазур.zip.c9c\Отчет%20ПП01.01%20Мазур.docx#_Toc201078212)

[**Описание данных приложения** 7](file:///C:\Users\nikit\AppData\Local\Temp\7d611b1e-e5d4-4a5f-990a-3fc9e448ac9c_Отчет%20ПП01.01%20Мазур.zip.c9c\Отчет%20ПП01.01%20Мазур.docx#_Toc201078213)

[**Описание дизайна приложения** 9](file:///C:\Users\nikit\AppData\Local\Temp\7d611b1e-e5d4-4a5f-990a-3fc9e448ac9c_Отчет%20ПП01.01%20Мазур.zip.c9c\Отчет%20ПП01.01%20Мазур.docx#_Toc201078214)

[**Реализация функций приложения** 14](file:///C:\Users\nikit\AppData\Local\Temp\7d611b1e-e5d4-4a5f-990a-3fc9e448ac9c_Отчет%20ПП01.01%20Мазур.zip.c9c\Отчет%20ПП01.01%20Мазур.docx#_Toc201078215)

[**Заключение** 20](file:///C:\Users\nikit\AppData\Local\Temp\7d611b1e-e5d4-4a5f-990a-3fc9e448ac9c_Отчет%20ПП01.01%20Мазур.zip.c9c\Отчет%20ПП01.01%20Мазур.docx#_Toc201078216)

[**Список литературы** 21](file:///C:\Users\nikit\AppData\Local\Temp\7d611b1e-e5d4-4a5f-990a-3fc9e448ac9c_Отчет%20ПП01.01%20Мазур.zip.c9c\Отчет%20ПП01.01%20Мазур.docx#_Toc201078217)

**Задание на разработку**

**Цель:**

Разработать мобильное приложение для поиска и бронирования отелей с функциями управления профилем, избранным и чатом с менеджерами «***Путешествия: Travenor***».

**Основные задачи:**

1. Разработка технического задания, включая описание функциональных требований и проектирование структуры данных.
2. Создание серверной части с использованием Supabase.
3. Верстка мобильного приложения на основе макета в Figma.
4. Реализация функционала приложения, включая аутентификацию, бронирование, чат с менеджерами и другие ключевые функции.
5. Тестирование приложения по методологии TDD.
6. Документирование результатов разработки.

**Анализ индивидуального задания**

Разработка мобильного приложения для бронирования отелей представляет собой комплексную задачу, требующую интеграции нескольких ключевых функциональных модулей, включая управление пользователями, бронированиями, чатами и фильтрацию отелей.

**Модуль 1. Разработка технического задания**

На начальном этапе определяется концепция приложения, формулируются цели и задачи, разрабатывается структура данных и создается детализированный дизайн-макет в Figma.

Модуль 2. Разработка серверной части

В качестве серверной части приложения используется Supabase, обеспечивающий:

* Хранение данных (отели, бронирования, пользователи).
* Аутентификацию (email + пароль, восстановление через OTP).
* Политики безопасности (ограничение доступа к данным на основе ролей).
* Хранение медиафайлов (фотографии отелей, аватары пользователей).

**Модуль 3. Верстка мобильного приложения**

Интерфейс приложения разрабатывается в соответствии с макетом в Figma, с адаптацией под различные размеры экранов. Основные элементы:

**Навигация:**

* + Нижнее меню для быстрого перехода между разделами (главная, поиск, бронирования, чаты, профиль).

**Экраны:**

* + Onboarding, авторизация, восстановление пароля.
  + Поиск отелей с фильтрами.
  + Детальная страница отеля.
  + Чат с менеджером.

**Модуль 4. Разработка функционала**

Финальный этап включает реализацию всех запланированных функций:

Аутентификация и безопасность:

* + Вход по email и паролю.
  + Восстановление пароля через OTP.
  + Локальное хранение сессии.

Работа с отелями:

* + Поиск и фильтрация (Название, количество звезд, поиск по контексту опимания).
  + Добавление в избранное.
  + Бронирование с выбором дат.

Чат с менеджером:

* + Отправка и получение сообщений.

Локализация:

* Поддержка русского и английского языков.

**Проектирование дизайна приложения**

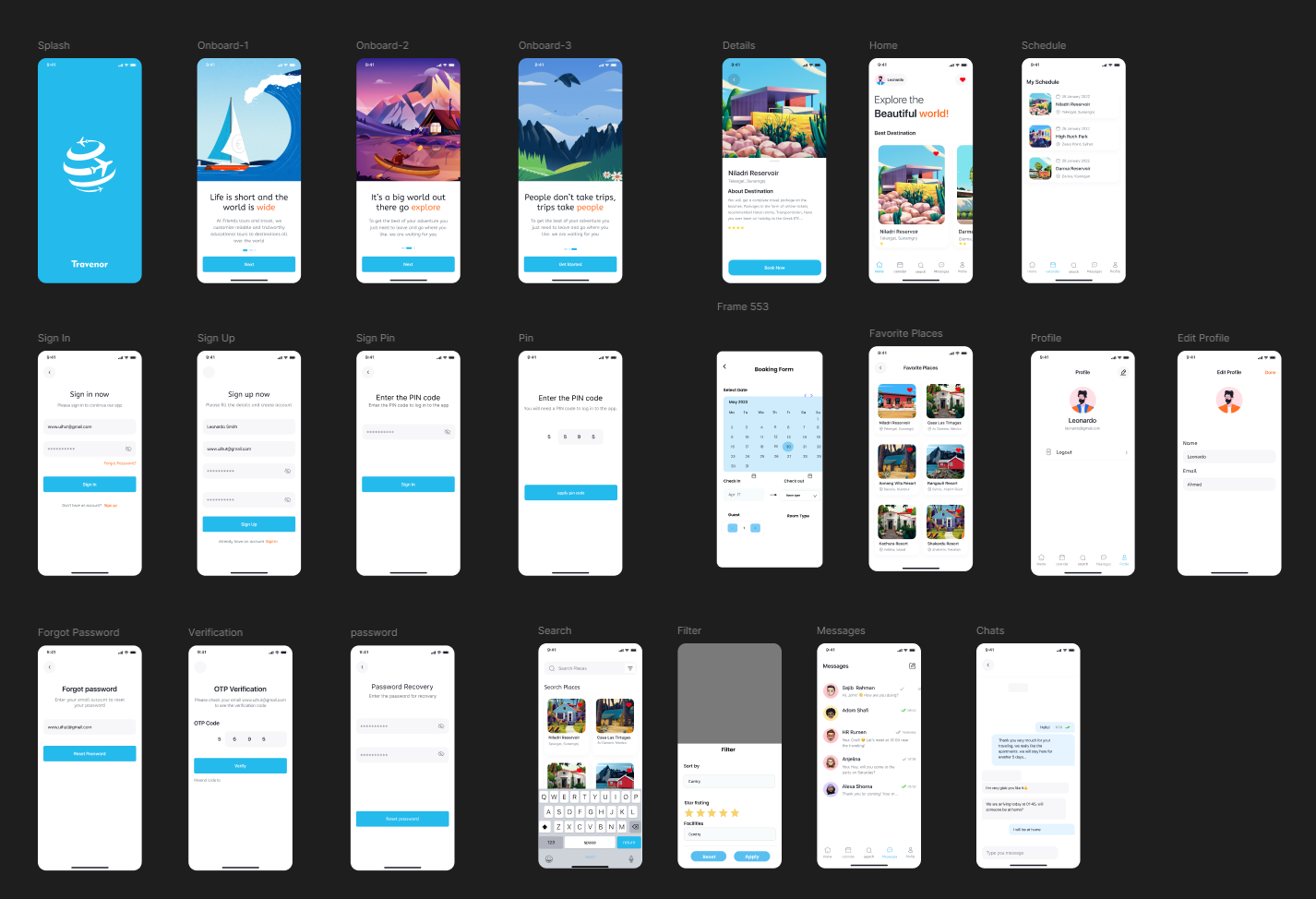
****

Рисунок 1 Макет Figma

**Описание важных элементов:**

* Экран приветствия: Простой и понятный интерфейс с кнопками для входа и регистрации.
* Экран поиска отелей: Фильтры по контексту, количеству звезд и местоположению.
* Экран чата: Удобный интерфейс для общения с менеджерами отелей.
* Экран избранного: отображает элементы которые выбрал клиент

**Описание данных приложения**

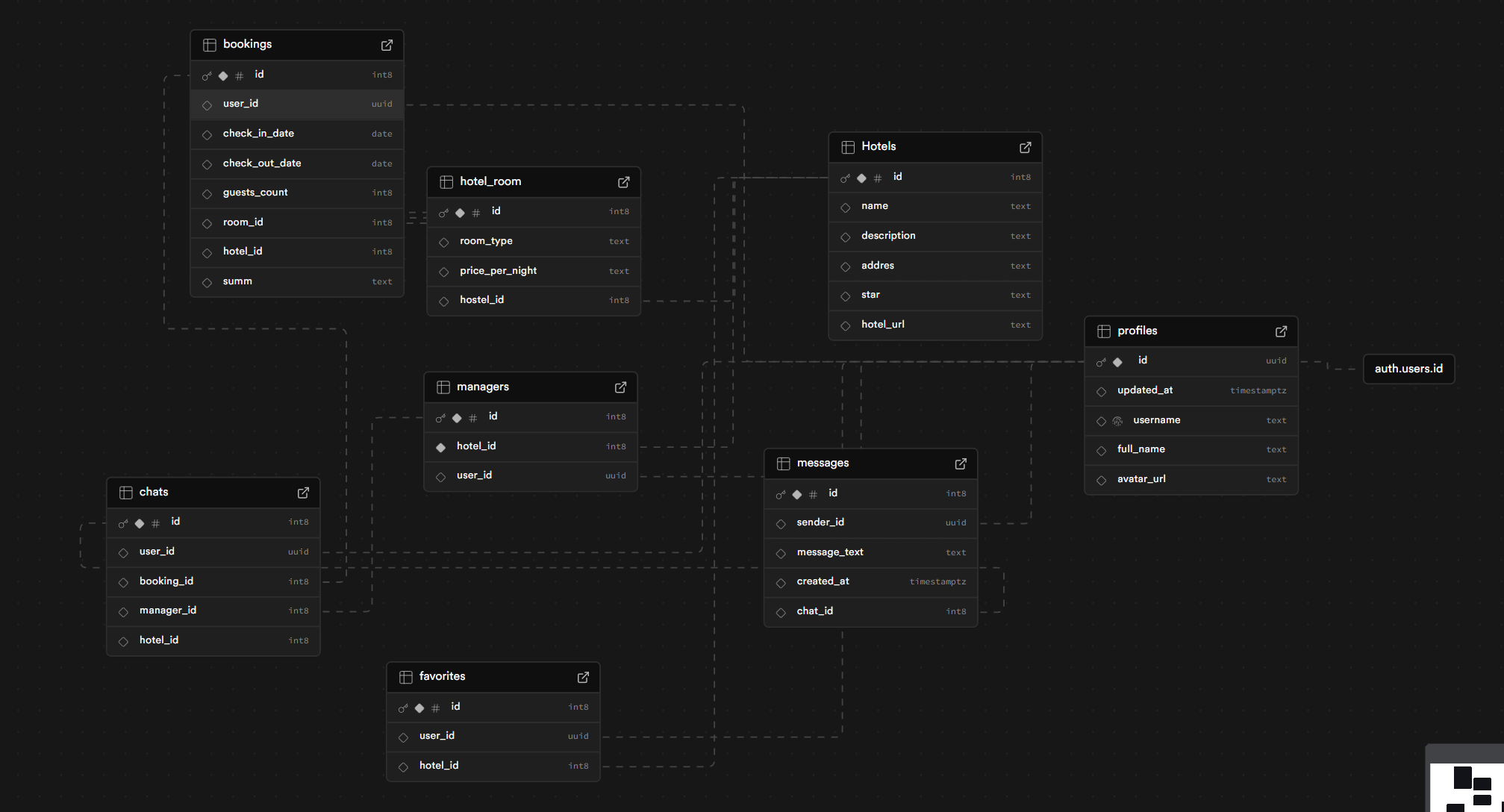


Рисунок 2 Схема БД для приложения

Users (id, full\_name, email, avatar\_url)

Chats (id, user\_id, manager\_id, hotel\_id,booking\_id)

Messages (id, chat\_id, sender\_id, text, created\_at)

Favorites (id, user\_id, hotel\_id)

Hotels (id, name, address, description, stars, hotel\_url)

Bookings (id, user\_id, hotel\_id, check\_in\_date, check\_out\_date, guests\_count, hotel\_id, ssum)

Hotel\_room (id, room\_type,prise\_per\_night,hostel\_id)

**Политики безопасности:**

Была использована политик предложенная самим сервисом которая предоставляет политику доступа по которой:

* Все могут просматривать профили
* Пользователи могут создавать/обновлять только свои профили

**Функции:**

* Аутентификация через email и пароль.
* Восстановление пароля через OTP.

**Описание дизайна приложения**

**1. Фрагмент главного экрана**

Главный экран приложения (рис. 3) на нем реализовано отображение данных пользователя переход на экран избранного и отображении списка отелей:

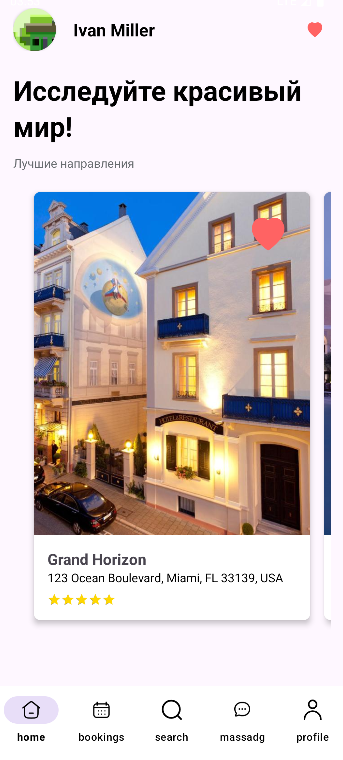


Рисунок 3 Главный экран приложения

2. Фрагмент отображения бронирований.

Экран отображения боронований изображён на (рис. 4) на этом экране отображаются апартаменты и итоговая цена за проживании на определенный период времени.



Рисунок 4 Экран мои бронирования

3. фрагмент поиска и фильтрации отелей

Экран поиска изображен на (рис. 5) на нем реализован поиск по названию отеля для быстрого нахождения нужного отеля, а также фильтрация по нескольким критериям таким как: количество звёзд, место положение , и по контексту описания.

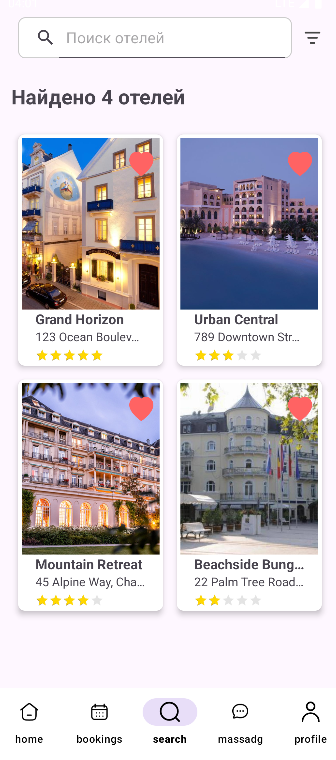


Рисунок 5 Экран поиска

4. Фрагмент для отображения чатов

Экран чатов пользователя изображен на (рис. 6) На нем отображаются все чаты в которых учувствует пользователь, чаты создаются после бронирования нового отеля с его менеджером для уточнения каких либо вопросов.



Рисунок 6 Экран чатов пользователя

5. Фрагмент профиля пользователя

Экран профиля изображен на (рис.7) этот экран отображает данные пользователя позволяет выйти из аккаунта и редактировать данные.

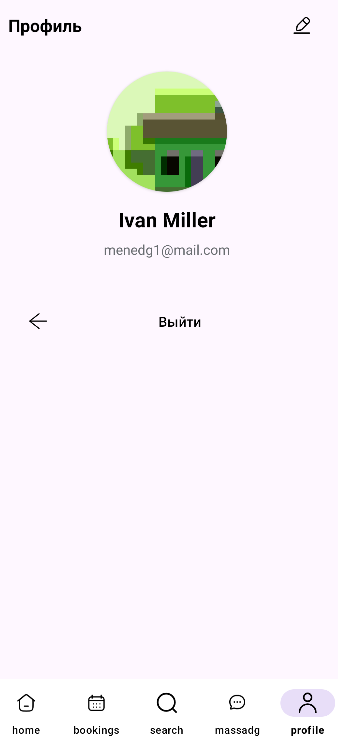


Рисунок 7 Экран профиля

**Реализация функций приложения**

**Функция добавления в избранное (addFavorite)**

**Назначение:**

Функция addFavorite предназначена для добавления отеля в список избранного пользователя. Она отправляет запрос на сервер, связывая идентификатор пользователя userId с идентификатором отеля hotelId в таблице избранного.

**Логика работы:**

1. Создается JSON-объект с параметрами:

* user\_id – ID пользователя, добавляющего отель в избранное.
* hotel\_id – ID отеля, который нужно добавить.

1. Отправляет POST-запрос на серверный на эндпоинт favorites

Литсинг 1:

void addFavorite(String userId, String hotelId, final SimpleCallback callback) {  
 String json = "{ \"user\_id\": \"" + userId + "\", \"hotel\_id\": \"" + hotelId + "\" }";  
 MediaType JSON = MediaType.get("application/json; charset=utf-8");  
  
 RequestBody body = RequestBody.create(JSON, json);  
  
 String url = DOMAIN\_NAME + REST\_PATH + "favorites";  
  
 Request request = new Request.Builder()  
 .url(url)  
 .post(body)  
 .addHeader("apikey", API\_KEY)  
 .addHeader("Authorization", DataBinding.getBearerToken())  
 .addHeader("Prefer", "return=minimal")  
 .build();  
  
 client.newCall(request).enqueue(new Callback() {  
 @Override  
 public void onFailure(@NonNull Call call, @NonNull IOException e) {  
 callback.onError(e);  
 }  
  
 @Override  
 public void onResponse(@NonNull Call call, @NonNull Response response) throws IOException {  
 if (response.isSuccessful()) {  
 callback.onSuccess();  
 } else {  
 callback.onError(new IOException("Не удалось добавить в избранное"));  
 }  
 }  
 });  
}

**Функция получения списка отелей (fetchHotels)**

**Назначение:**

Функция fetchHotels выполняет запрос к серверу для получения полного списка отелей, доступных в системе. Она предназначена для отображения отелей в мобильном приложении.

**Логика работы:**

1. **Формирование запроса:**
   * Создается GET-запрос к эндпоинту Hotels с параметром select=\* (получение всех полей)
   * URL формируется как: {DOMAIN\_NAME}/rest/v1/Hotels?select=\*
2. **Настройка заголовков:**
   * Добавляется обязательный заголовок apikey для доступа к API
   * Включается токен авторизации через Authorization: Bearer {token}
3. **Выполнение запроса:**
   * Запрос выполняется асинхронно через OkHttp
   * Используется callback-интерфейс для обработки результата

**Листинг 2**

public void fetchHotels(final SBC\_Callback callback) {  
 String url = DOMAIN\_NAME + REST\_PATH + "Hotels?select=\*";  
  
 Request request = new Request.Builder()  
 .url(url)  
 .get()  
 .addHeader("apikey", API\_KEY)  
 .addHeader("Authorization", DataBinding.getBearerToken())  
 .build();  
  
 client.newCall(request).enqueue(new Callback() {  
 @Override  
 public void onFailure(@NonNull Call call, @NonNull IOException e) {  
 callback.onFailure(e);  
 }  
  
 @Override  
 public void onResponse(@NonNull Call call, @NonNull Response response) throws IOException {  
 if (response.isSuccessful() && response.body() != null) {  
 String responseBody = response.body().string();  
 callback.onResponse(responseBody);  
 } else {  
 callback.onFailure(new IOException("Failed to fetch hotels"));  
 }  
 }  
 });  
}

**Функция обновления профиля (updateProfile)**

**Назначение:**

Функция updateProfile выполняет частичное обновление данных пользовательского профиля в базе данных через REST API. Она отправляет измененные данные на сервер и обрабатывает результат операции.

1. **Подготовка данных:**
   * Конвертирует объект ProfileUpdate в JSON-строку с помощью Gson
   * Создает тело запроса (RequestBody) с типом содержимого application/json
2. **Формирование запроса:**
   * Использует PATCH-метод для частичного обновления
   * URL включает фильтр по ID пользователя: profiles?id=eq.{userId}
   * Добавляет обязательные заголовки:
     + apikey - ключ доступа к API
     + Authorization - Bearer Token для аутентификации
3. **Выполнение запроса:**
   * Запрос выполняется асинхронно через OkHttp
   * Результат возвращается через callback-интерфейс

Листинг 3

public void updateProfile(String userId, ProfileUpdate profileUpdate, SBC\_Callback callback) {

MediaType JSON = MediaType.get("application/json; charset=utf-8");

Gson gson = new Gson();

String json = gson.toJson(profileUpdate);

RequestBody body = RequestBody.create(JSON, json);

Request request = new Request.Builder()

.url(DOMAIN\_NAME + REST\_PATH + "profiles?id=eq." + userId)

.patch(body)

.addHeader("apikey", API\_KEY)

.addHeader("Authorization", DataBinding.getBearerToken())

.build();

client.newCall(request).enqueue(new Callback() {

@Override

public void onFailure(@NonNull Call call, @NonNull IOException e) {

callback.onFailure(e);

}

@Override

public void onResponse(@NonNull Call call, @NonNull Response response) throws IOException {

if (response.isSuccessful()) {

callback.onResponse("OK");

} else {

callback.onFailure(new IOException("Ошибка сервера"));

}

}

});

}

**Заключение**

В рамках данной работы было разработано мобильное приложение "Trevanor", интегрирующее следующие функции бронирование отелей, общение с менеджером отелей и добавление в избранное. Проект успешно реализован в соответствии с поставленными задачами и современными стандартами мобильной разработки.

На начальном этапе было разработано детализированное техническое задание, включающее оптимизированную структуру данных и продуманный дизайн-макет в Figma, который стал основой для создания адаптивного интерфейса.

Серверная часть приложения построена на базе Supabase. Интерфейс приложения выполнен в единой стилистике. Особое внимание было уделено адаптивности – приложение корректно отображается на устройствах с различными размерами экранов.

Данная практика помогла нам закрепить приобретённые навыки в ходе обучения и узнать новые способы реализации решений.

**Список литературы**

1. Харди, Б. Разработка приложений для Android. Полное руководство для начинающих. – 2-е изд. – СПб.: Питер, 2022.
2. Дейтел, П. Android 13 для разработчиков на Java. – М.: Эксмо, 2023.
3. Берк, К. Android-программирование для профессионалов. – 4-е изд. – М.: ДМК Пресс, 2023.
4. Гриффитс, Д. Программирование под Android. – 3-е изд. – СПб.: БХВ-Петербург, 2022.
5. Колисниченко, Д. Программирование для Android. – М.: БХВ-Петербург, 2011.
6. Коматинени, С. Android 3 для профессионалов. – М.: Вильямс, 2012.
7. Голощапов, А. Google Android. Программирование для мобильных устройств. – М.: БХВ-Петербург, 2012.
8. Хорстманн, К. Java. Библиотека профессионала. Том 2. – 12-е изд. – М.: Диалектика, 2023.
9. Блох, Дж. Эффективное программирование на Java для Android. – 3-е изд. – М.: ДМК Пресс, 2022.
10. Маклин, Г. Android Security: защита приложений на Java. – М.: Эксмо, 2023.