09.02.07 ПР-312

**ОТЧЕТ ПО ПРОИЗВОДСВЕННОЙ ПРАКТИКЕ**

**ПП.01.01 Разработка мобильных приложений**

**ПМ.01 Разработка модулей программного обеспечения**

|  |  |  |
| --- | --- | --- |
| **Практикант** |  | **Мазур И.Д.** |
| **Руководители практики** |  | **Домбровский Н.С.** |
|  |  |  |

Оглавление

[**Задание на разработку** 3](#_Toc201078210)

[**Анализ индивидуального задания** 4](#_Toc201078211)

[**Проектирование дизайна приложения** 6](#_Toc201078212)

[**Описание данных приложения** 9](#_Toc201078213)

[**Описание дизайна приложения** 13](#_Toc201078214)

[**Реализация функций приложения** 22](#_Toc201078215)

[**Заключение** 32](#_Toc201078216)

[**Список литературы** 34](#_Toc201078217)

**Задание на разработку**

Цель практики:

Разработать мобильное приложение «Электронная почта: Brees»

Структура задания:

1. Модуль 1 Разработка технического задания:

- описание функциональных требований;

- проектирование структуры данных;

- создание макета приложения.

2. Модуль 2 Разработка серверной части;

3. Модуль 3 Верстка мобильного приложения;

4. Модуль 4 Разработка функционала мобильного приложения;

5. Модуль 5 Тестирование приложения;

6. Модуль 6 Документирование результатов разработки.

Общие требования к приложению:

1. в работе необходимо использовать систему контроля версий Git, и ежедневно сохранять разработанные элементы проекта;

2. проект приложения должен быть структурирован, то есть исходные файлы должны быть в соответствующей папке, общие классы необходимо поместить в папку classes.

3. в приложении должны корректно обрабатываться запросы к серверу. В случае получения ошибки от сервера или отсутствия соединения с сетью Интернет необходимо отобразить соответствующий текст ошибки в диалоговом окне, которое должно закрываться только пользователем.

**Анализ индивидуального задания**

Разработка мобильного приложения для управления персональными финансами и электронной почтой представляет собой комплексную задачу, требующую интеграции нескольких ключевых функциональных модулей.

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. Ключевой акцент делается на создании удобного интерфейса, который будет включать все необходимые экраны для работы с почтой и финансовыми операциями.

Модуль 2. Разработка серверной части

В качестве backend-решения используется Supabase, обеспечивающий хранение данных, аутентификацию пользователей и безопасное управление контентом. Особое внимание уделяется реализации механизмов восстановления пароля через OTP-код, а также созданию политик безопасности для защиты пользовательской информации.

Модуль 3. Верстка мобильного приложения

Интерфейс приложения разрабатывается строго в соответствии с утвержденным макетом, с обязательной адаптацией под различные размеры экранов. В процессе верстки особое внимание уделяется созданию интуитивно понятной навигации между экранами и разработке диалоговых окон для обработки различных сценариев взаимодействия.

Модуль 4. Разработка функционала

Финальный этап включает реализацию всех запланированных функций: от системы аутентификации до сложных механизмов работы с почтой и финансовыми операциями. Приложение получает возможность фильтрации данных, управления транзакциями и письмами, а также поддержку двух языков интерфейса.

**Проектирование дизайна приложения**

**Основные принципы дизайна**

При создании интерфейса учитывались следующие принципы:

1. Минимализм и удобство — лаконичный дизайн без лишних элементов, упор на удобство навигации.
2. Адаптивность — макеты корректно отображаются на экранах разных размеров.
3. Консистентность — единый стиль цветов, шрифтов и иконок для всех экранов.
4. Доступность — соблюдение контрастности, читаемости текста и удобных размеров интерактивных элементов.

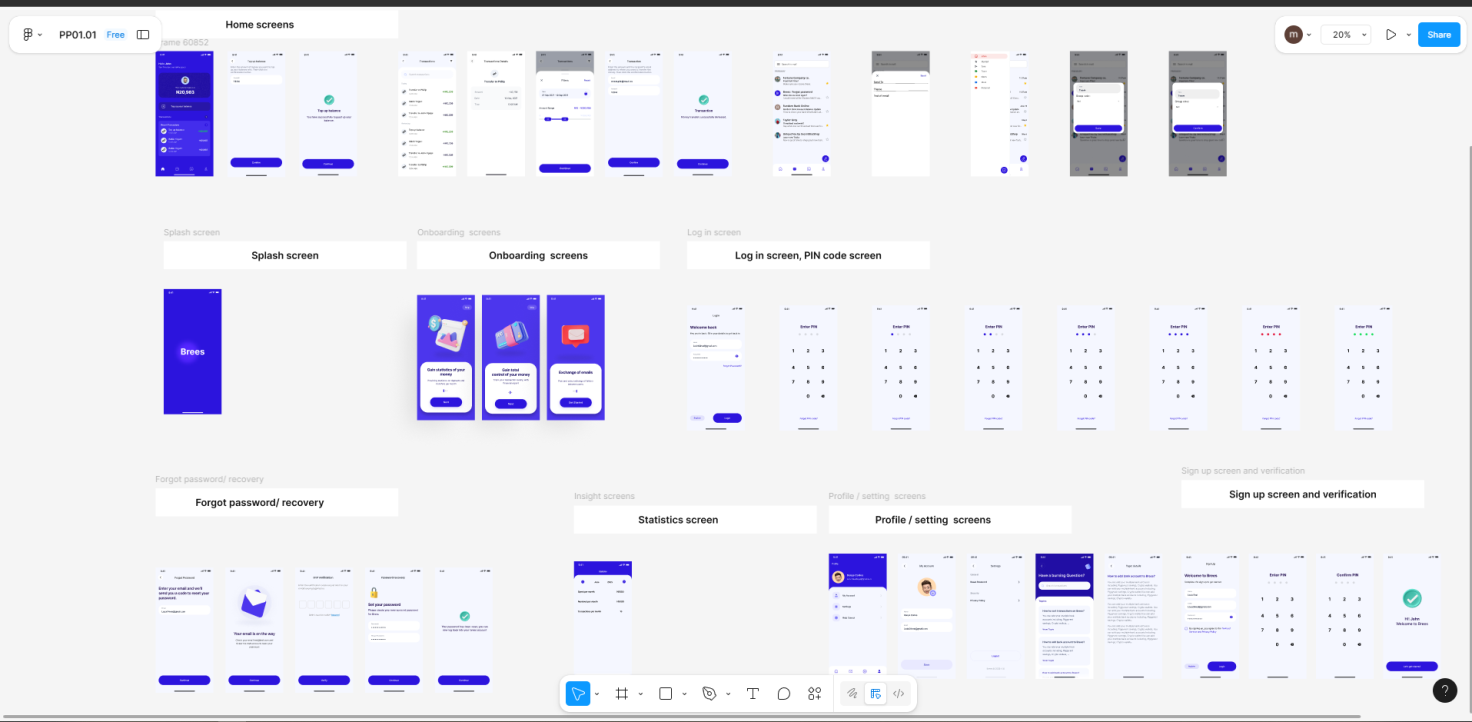


Рисунок 1 Макет Figma

**Основные экраны интерфейса**

**1.1. Финансовая панель (главный экран)**

* Отображает приветствие пользователя и текущий баланс
* Включает кнопку быстрого пополнения счета
* Содержит раздел последних транзакций с детализацией:
  + Временные метки операций
  + Названия получателей/отправителей
  + Суммы переводов с цветовой индикацией
* Дизайн выполнен в минималистичном стиле с акцентом на важные финансовые показатели

**1.2. Почтовый клиент**

* Представляет список входящих сообщений с:
  + Именами отправителей
  + Краткими превью текста
  + Темой письма
  + Визуальным разделением по категориям (PRIMARY)
* Реализована строка поиска для быстрой навигации
* Использованы компактные карточки писем с четкой иерархией информации

**1.3. Экран авторизации**

* Чистый минималистичный дизайн с формой входа
* Поля для email и пароля с визуальными разделителями
* Кнопки дополнительных действий:
  + Восстановление пароля
  + Переход к регистрации
* Сохраняет последний использованный email для удобства

**1.4. Процесс регистрации**

* Многоэтапная форма с:
  + Вводом персональных данных
  + Созданием PIN-кода
  + Чекбоксом принятия условий
* Цифровая клавиатура для ввода PIN
* Визуальное разделение на логические блоки

**1.5. Профиль пользователя**

* Отображает:
  + Аватар и основные данные
  + Настройки аккаунта
  + Ссылки на службу поддержки
* Использованы переключатели для быстрого изменения параметров
* Четкое визуальное разделение секций

**Описание данных приложения**

Структура данных приложения разработана для обеспечения работы двух ключевых модулей:

* **Системы электронной почты** (хранение и обработка писем)
* **Финансовых операций** (учет транзакций и балансов пользователей)

База данных реализована на **Supabase (PostgreSQL)** и включает 5 взаимосвязанных таблиц (рис. 2), оптимизированных для быстрого доступа и целостности данных.

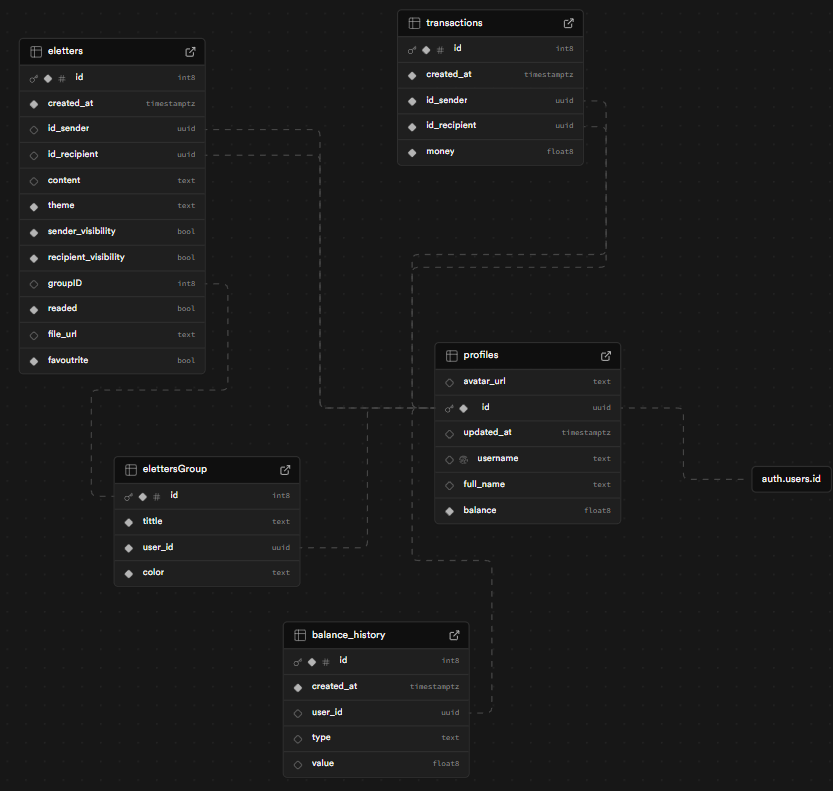


Рисунок 2 Схема БД

**Основные таблицы и их назначение**

**1. Таблица**deltiers**(Письма)**

| **Поле** | **Тип** | **Описание** |
| --- | --- | --- |
| content | text | Текст письма |
| theme | text | Тема письма |
| file\_url | text | Ссылка на вложение |
| groupID | int8 | Идентификатор папки |
| readed | bool | Флаг прочтения |
| favourite | bool | Избранное |

* Поддержка видимости для отправителя/получателя (sender\_visibility, recipient\_visibility)
* Возможность группировки писем через groupID

**2 Таблица**transactions**(Транзакции)**

| **Поле** | **Тип** | **Описание** |
| --- | --- | --- |
| money | float8 | Сумма перевода |
| Id\_sender | uuid | ID отправителя |
| Id\_recipient | uuid | ID получателя |

**3. Таблица**profiles**(Профили)**

| **Поле** | **Тип** | **Описание** |
| --- | --- | --- |
| avatar\_url | text | Ссылка на аватар |
| balance | float8 | Текущий баланс |
| full\_name | text | Полное имя |

**4. Таблица**deltiersGroup**(Группы писем)**

| **Поле** | **Тип** | **Описание** |
| --- | --- | --- |
| title | text | Название папки |
| color | text | Цвет метки |

**5. Таблица**balance\_history**(История баланса)**

| **Поле** | **Тип** | **Описание** |
| --- | --- | --- |
| type | text | Тип операции (пополнение/списание) |
| value | float8 | Сумма изменения |

**Связи между таблицами**

1. **Письма ↔ Профили**:
   * deltiers.Id\_sender → profiles.id
   * deltiers.Id\_recipient → profiles.id
2. **Транзакции ↔ Профили**:
   * transactions.Id\_sender → profiles.id
   * transactions.Id\_recipient → profiles.id
3. **Группы ↔ Пользователи**:
   * deltiersGroup.user\_id → profiles.id
4. **История баланса ↔ Профили**:
   * balance\_history.user\_id → profiles.id

**Описание дизайна приложения**

**1. Фрагмент главного экрана (fragment\_home.xml)**

Главный экран приложения (рис. 3) представляет собой информационную панель с ключевыми финансовыми показателями:

* Приветственное сообщение с именем пользователя
* Карточка баланса с круглым аватаром и отображением текущего состояния счета
* Кнопка быстрого пополнения баланса с иконкой "+"
* Секция последних транзакций с:
  + Заголовком и кнопкой перехода
  + Прокручиваемым списком операций
  + Возможностью обновления по жесту

Дизайн выполнен в фиолетово-синей цветовой гамме с использованием:

* Шрифтов семейства Inter различных начертаний
* Закругленных карточек с тенями
* Четкой визуальной иерархии элементов

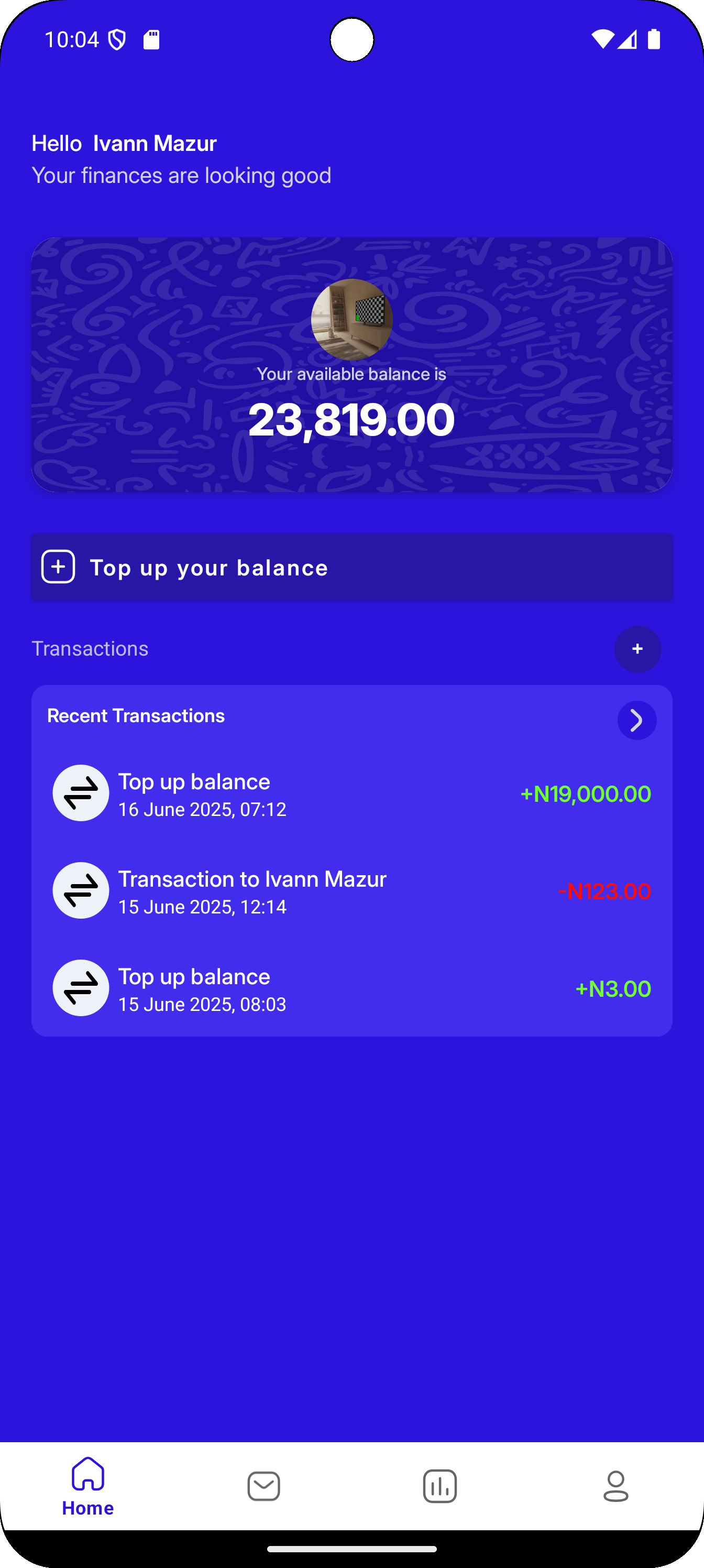


Рисунок 3 Фрагмент главного экрана

**2. Фрагмент почты (fragment\_notifications.xml)**

Экран работы с электронной почтой (рис. 4) включает:

* Поисковую строку с возможностью фильтрации писем
* Выдвижное боковое меню для навигации по папкам:
  + Основной список папок
  + Пользовательские группы писем
  + Кнопка создания новых групп
* Основную область с:
  + Заголовком текущей папки (PRIMARY)
  + Прокручиваемым списком писем с поддержкой обновления
  + Плавающей кнопкой создания нового письма

Особенности интерфейса:

* Светло-серый фон для основного контента
* Белые панели для элементов управления
* Интерактивные иконки для действий

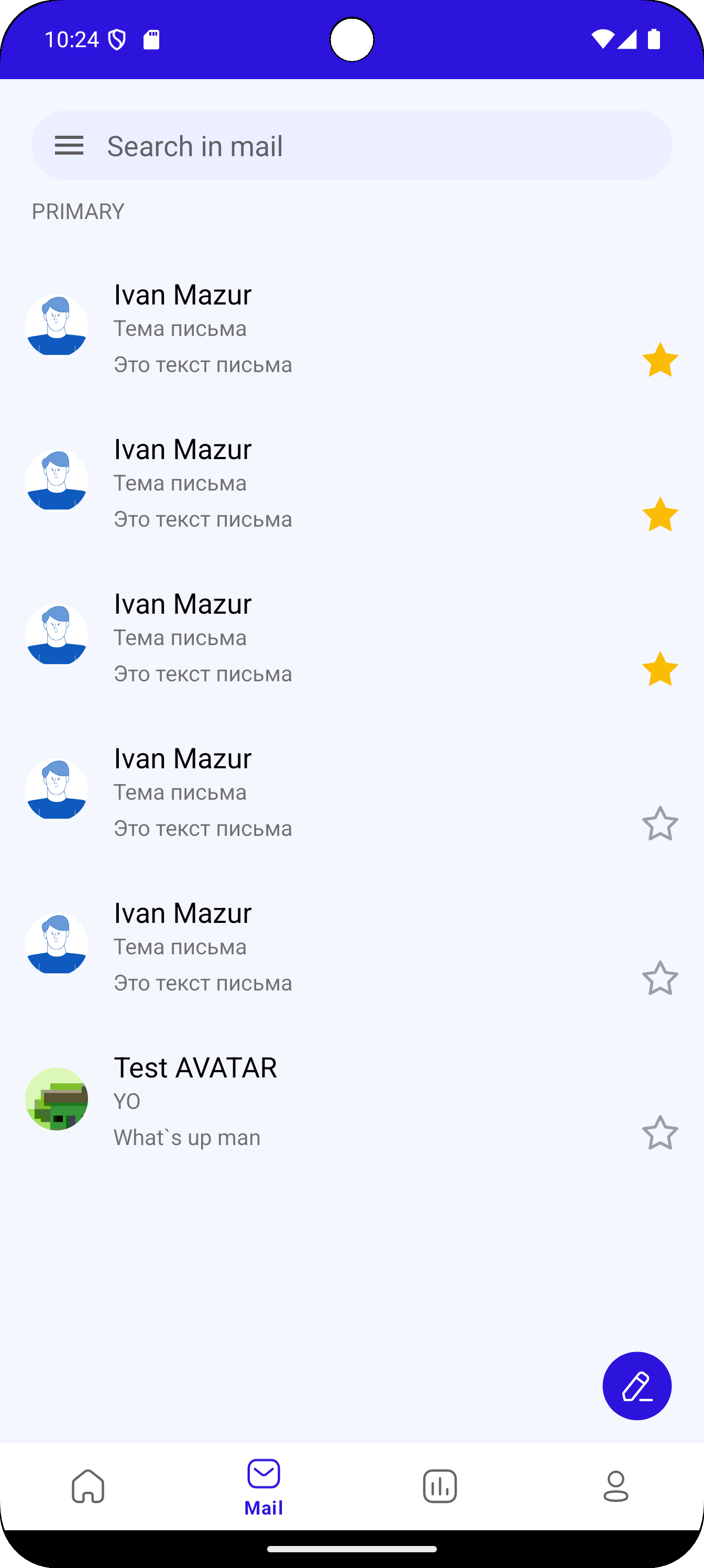


Рисунок 4 Фрагмент почты

**3. Экран истории транзакций (activity\_transactions\_history.xml)**

Страница просмотра финансовых операций содержит:

* Шапку с заголовком и кнопками:
  + Назад
  + Фильтры
* Поисковую строку для быстрого нахождения операций
* Основной список транзакций с:
  + Возможностью прокрутки
  + Поддержкой поиска и фильтрации

Дизайн особенности:

* Чистый белый фон для списка
* Темно-серые тексты для лучшей читаемости
* Минималистичные разделители между элементами

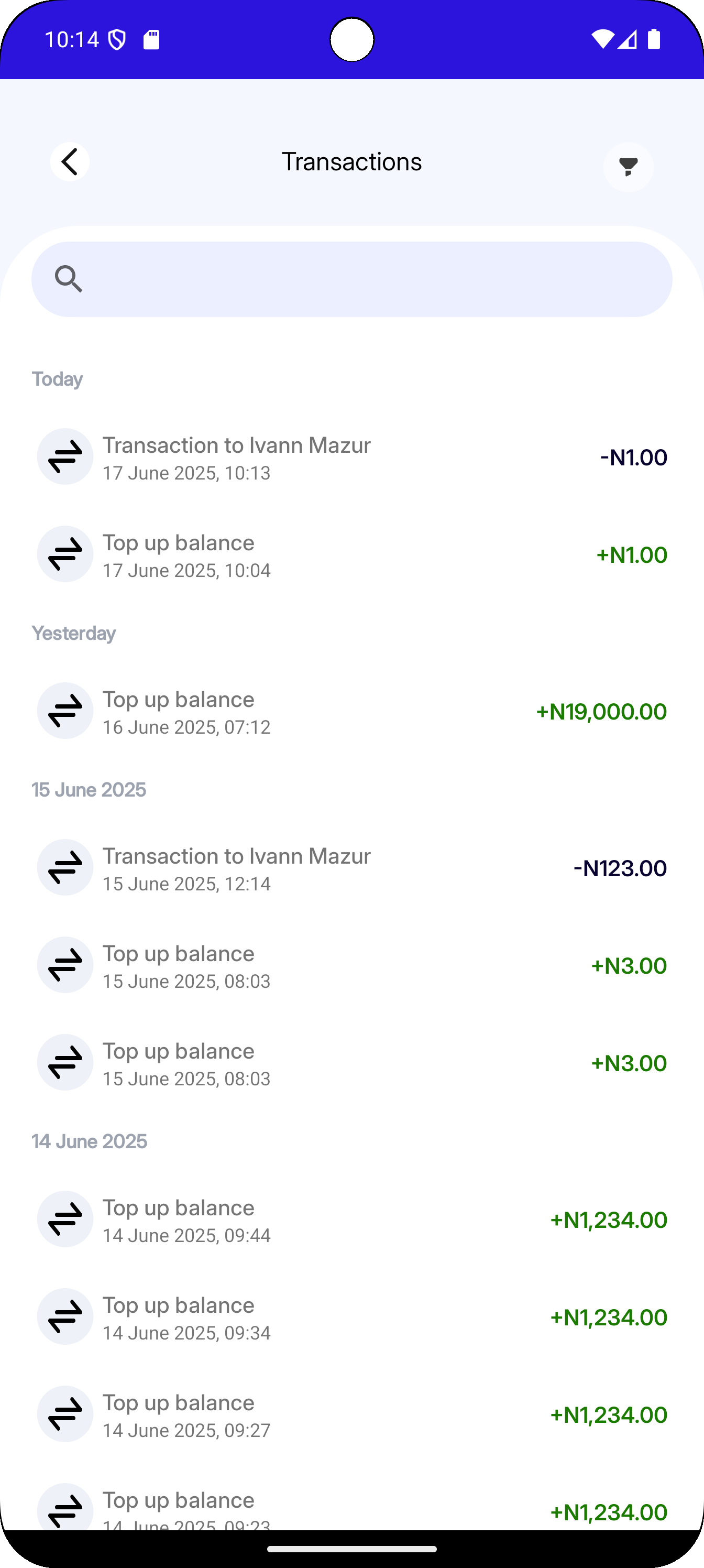


Рисунок 5 Экран истории транзакций

**4. Экран профиля (fragment\_profile.xml)**

Личный кабинет пользователя включает:

* Шапку с заголовком "Profile"
* Блок персональной информации:
  + Аватар (круглое изображение)
  + Имя пользователя
  + Электронная почта
* Нижнюю панель с кнопками:
  + Мой аккаунт
  + Настройки
  + Центр помощи

Стилевые решения:

* Темный фон для верхней части
* Светлая панель для функциональных кнопок
* Иконки сопровождения для каждой кнопки
* Плавные переходы между секциями

**Общие принципы дизайна**

Для всех экранов характерны:

1. Единая стилистика:
   * Шрифтовая гарнитура Inter
   * Согласованная цветовая палитра
   * Похожие радиусы скруглений
2. Продуманная эргономика:
   * Важные элементы в зоне легкой досягаемости
   * Достаточные размеры интерактивных элементов
   * Логичная группировка связанных функций
3. Адаптивность:
   * Корректное отображение на разных размерах экранов
   * Оптимальное использование пространства
   * Сохранение удобства навигации
4. Интерактивность:
   * Визуальная обратная связь при действиях
   * Плавные анимации переходов
   * Интуитивно понятные жесты

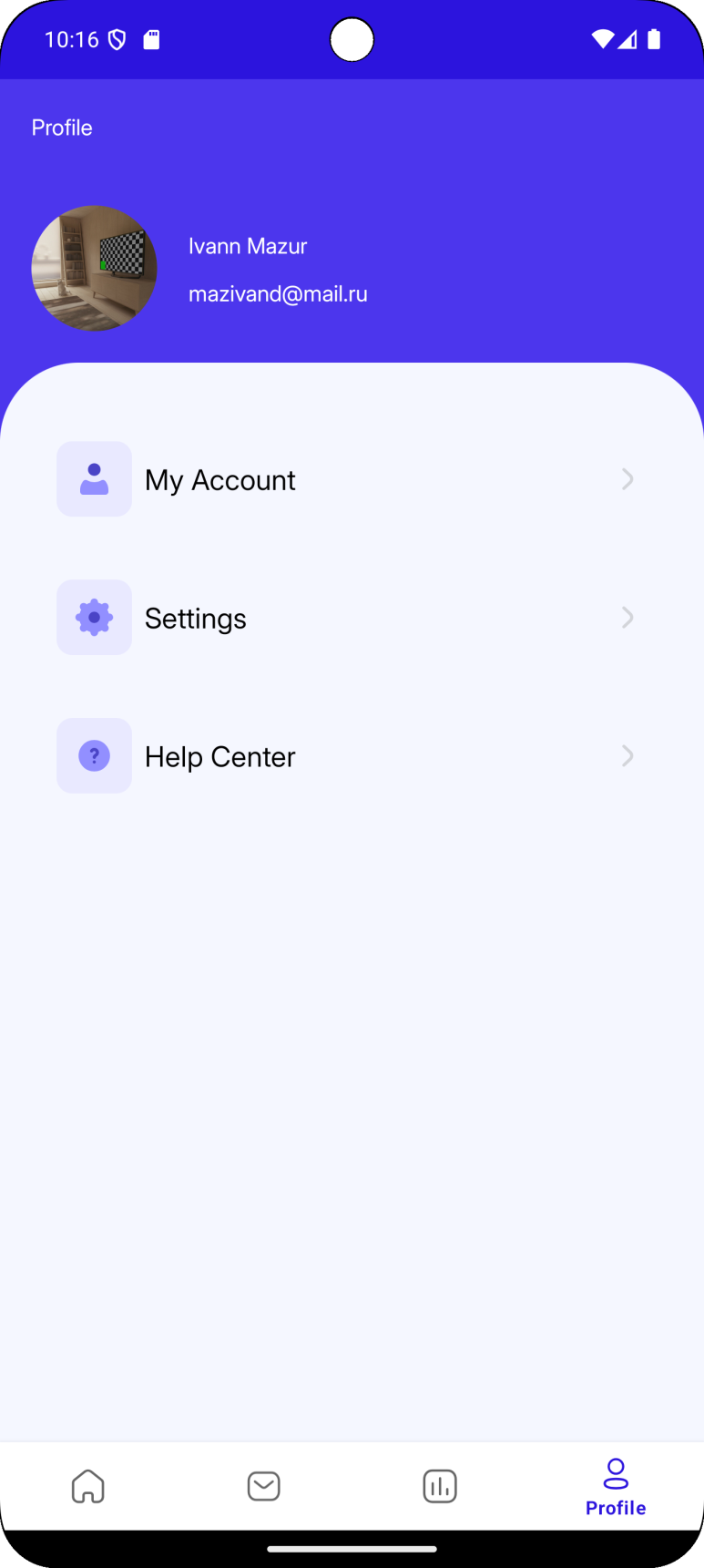


Рисунок 6 Экран профиля

Разработанный дизайн обеспечивает удобство работы как с финансовыми операциями, так и с электронной почтой, сохраняя при этом единый стиль и высокую степень удобства использования.

**Реализация функций приложения**

**Функция перевода денежных средств (transferMoney)**

**Назначение:**  
Осуществляет перевод указанной суммы денег от текущего пользователя к получателю, идентифицируемому по email.

**Логика работы:**

1. Формирует JSON-запрос, содержащий:
   * Сумму перевода (amount)
   * Email получателя (recipient\_email)
   * ID отправителя (sender\_id), получаемый из SessionManager
2. Отправляет POST-запрос на серверный endpoint rpc/transfer\_money
3. Обрабатывает ответ:
   * При успехе передает результат в callback
   * При ошибке возвращает исключение с кодом ошибки

**Особенности реализации:**

* Использует асинхронный запрос через OkHttp
* Добавляет заголовки авторизации (Bearer Token)
* Обрабатывает возможные исключения при формировании запроса

Литсинг 1 Метода транзакций:

public void transferMoney(Context context, String Email, Float amount, SBC\_Callback callback) {

JSONObject jsonBody = new JSONObject();

SessionManager sessionManager = new SessionManager(context);

try {

jsonBody.put("amount", amount);

jsonBody.put("recipient\_email", Email);

jsonBody.put("sender\_id", sessionManager.getUserId());

} catch (Exception e) {

e.printStackTrace();

return;

}

MediaType mediaType = MediaType.parse("application/json");

RequestBody body = RequestBody.create(mediaType, jsonBody.toString());

Request request = new Request.Builder()

.url(DOMAIN\_NAME + REST\_PATH + "rpc/transfer\_money")

.method("POST", body)

.addHeader("Content-Type", "application/json")

.addHeader("apikey", API\_KEY)

.addHeader("Authorization", "Bearer " + sessionManager.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(response.body().string());

} else {

callback.onFailure(new IOException("Ошибка сервера: " + response.code()));

}

}

});

}

**Функция получения профиля пользователя (fetchUserProfile)**

**Назначение:**  
Получает полную информацию о профиле текущего авторизованного пользователя.

**Логика работы:**

1. Проверяет авторизацию пользователя через SessionManager
2. Формирует GET-запрос к endpoint profiles с фильтром по ID пользователя
3. Обрабатывает ответ:
   * При успехе возвращает данные профиля
   * При ошибке возвращает соответствующее исключение

**Особенности реализации:**

* Проверяет наличие авторизации перед запросом
* Использует параметризованный URL для точного запроса
* Поддерживает обработку различных форматов ошибок сервера

Листинг 2 получение данных пользователя:

public void fetchUserProfile(Context context, SBC\_Callback callback) {

SessionManager sessionManager = new SessionManager(context);

String userId = sessionManager.getUserId();

if (userId == null || userId.isEmpty()) {

callback.onFailure(new IOException("Пользователь не авторизован"));

return;

}

String url = DOMAIN\_NAME + REST\_PATH + "profiles?id=eq." + userId;

Request request = new Request.Builder()

.url(url)

.get()

.addHeader("apikey", API\_KEY)

.addHeader("Authorization", "Bearer " +

sessionManager.getBearerToken())

.addHeader("Content-Type", "application/json")

.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 body = response.body().string();

callback.onResponse(body);

} else {

String errorBody = response.body() != null ? response.body().string() : "Неизвестная ошибка";

callback.onFailure(new IOException("Ошибка сервера: " + response.code() + ", " + errorBody));

}

}

});

}

**Функция загрузки аватара (uploadAvatar)**

**Назначение:**  
Загружает изображение аватара пользователя на сервер в хранилище.

**Логика работы:**

1. Преобразует URI изображения в реальный путь к файлу
2. Создает multipart/form-data запрос с содержимым файла
3. Отправляет PUT-запрос в хранилище avatars
4. Обрабатывает ответ сервера

**Особенности реализации:**

* Использует специальный утилитарный класс RealPathUtil для получения пути
* Поддерживает загрузку файлов различных изображений (image/\*)
* Обеспечивает детализированную обработку ошибок загрузки

Листинг 3 метод загрузки аватара пользователя:

public void uploadAvatar(Context context, Uri uri, String fileName, SBC\_Callback callback) {

String realPath = RealPathUtil.getRealPath(context, uri);

SessionManager sessionManager = new SessionManager(context);

if (realPath == null) {

callback.onFailure(new IOException("Не удалось получить путь файла"));

return;

}

File file = new File(realPath);

RequestBody requestBody = RequestBody.create(MediaType.parse("image/\*"), file);

MultipartBody body = new MultipartBody.Builder()

.setType(MultipartBody.FORM)

.addFormDataPart("file", fileName, requestBody)

.build();

String url = DOMAIN\_NAME + "/storage/v1/object/avatars/" + fileName;

Request request = new Request.Builder()

.url(url)

.put(body)

.addHeader("apikey", API\_KEY)

.addHeader("Authorization", "Bearer " + sessionManager.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(response.body().string());

} else {

String errorBody = response.body() != null ? response.body().string() : "Empty response";

callback.onFailure(new IOException("Upload failed: " + response.code() + ", Body: " + errorBody));

}

}

});  
 }

**Функция обновления пароля (updateUserPassword)**

**Назначение:**  
Изменяет пароль текущего авторизованного пользователя.

**Логика работы:**

1. Формирует JSON-запрос с новым паролем
2. Отправляет PUT-запрос на endpoint обновления пользователя
3. Обрабатывает ответ сервера

**Особенности реализации:**

* Использует PUT-метод для обновления данных
* Обрабатывает возможные исключения при формировании запроса
* Сохраняет целостность процесса изменения пароля

Листинг 4 метод обновления пароля пользователя:

public void updateUserPassword(Context context, String newPassword, final SBC\_Callback callback) {

try {

JSONObject jsonObject = new JSONObject();

jsonObject.put("password", newPassword);

RequestBody body = RequestBody.create(

jsonObject.toString(),

MediaType.parse("application/json")

);

SessionManager sessionManager = new SessionManager(context);

Request request = new Request.Builder()

.url(DOMAIN\_NAME + AUTH\_PATH + "user")

.method("PUT", body)

.addHeader("apikey", API\_KEY)

.addHeader("Authorization", "Bearer " +

sessionManager.getBearerToken())

.addHeader("Content-Type", "application/json")

.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(response.body().string());

} else {

callback.onFailure(new IOException("Ошибка сервера: " + response.code()));

}

}

});

} catch (Exception e) {

callback.onFailure(new IOException("Ошибка создания запроса: " + e.getMessage()));

}

}

**Заключение**

В ходе выполнения практического задания была успешно разработана мобильное приложение «Brees», объединяющее функционал электронной почты и управления персональными финансами. Реализация проекта велась в соответствии с поставленными задачами и модульной структурой, что позволило создать целостное и удобное для пользователя решение.

**Ключевые достижения:**

1. Разработано техническое задание, включающее:
   * Оптимизированную структуру данных
   * Полноценный дизайн-макет в Figma
2. Реализована серверная часть на базе Supabase, обеспечивающая:
   * Надежную аутентификацию пользователей
   * Эффективное взаимодействие между клиентом и сервером
3. Создан адаптивный интерфейс, отличающийся:
   * Единой стилистикой и продуманной эргономикой
   * Удобной навигацией между разделами
   * Поддержкой различных размеров экранов
4. Реализован основной функционал:
   * Работа с электронной почтой (отправка, получение, фильтрация)
   * Финансовые операции (переводы, история транзакций)
   * Персонализация профиля пользователя

Проведенная работа позволила закрепить теоретические знания и приобрести практические навыки в области мобильной разработки, проектирования интерфейсов и работы с базами данных.

**Список литературы**

1. Харди, Б. Разработка приложений для Android. Полное руководство для начинающих : [пер. с англ.] / Б. Харди, Б. Филлипс. – 2-е изд. – СПб. : Питер, 2022. – 720 с. – ISBN 978-5-4461-1589-3.
2. Дейтел, П. Android 13 для разработчиков на Java : учеб. пособие / П. Дейтел, Х. Дейтел. – М. : Эксмо, 2023. – 896 с. – ISBN 978-5-04-123456-7.
3. Берк, К. Android-программирование для профессионалов : [пер. с англ.] / К. Берк, Б. Стюарт. – 4-е изд. – М. : ДМК Пресс, 2023. – 584 с. – ISBN 978-5-97060-987-6.
4. Гриффитс, Д. Программирование под Android : [пер. с англ.] / Д. Гриффитс, Д. Хелд. – 3-е изд. – СПб. : БХВ-Петербург, 2022. – 512 с. – ISBN 978-5-9775-6543-2.
5. Негус, К. Разработка Android-приложений на Java. Углубленный курс / К. Негус. – М. : Вильямс, 2023. – 432 с. – ISBN 978-5-8459-2345-6.
6. Фрай, Б. Современная разработка Android-приложений на Java / Б. Фрай, Д. Марсикано. – М. : Эксмо, 2022. – 368 с. – ISBN 978-5-04-119876-0.
7. Скиена, С. Алгоритмы для Android-разработчиков на Java / С. Скиена. – 2-е изд. – СПб. : Питер, 2023. – 288 с. – ISBN 978-5-4461-1890-0.
8. Хорстманн, К. Java. Библиотека профессионала. Том 2. Расширенные средства программирования для Android / К. Хорстманн. – 12-е изд. – М. : Диалектика, 2023. – 864 с. – ISBN 978-5-907584-32-1.
9. Блох, Дж. Эффективное программирование на Java для Android / Дж. Блох. – 3-е изд. – М. : ДМК Пресс, 2022. – 416 с. – ISBN 978-5-97060-876-3.
10. Маклин, Г. Android Security: защита приложений на Java / Г. Маклин. – М. : Эксмо, 2023. – 256 с. – ISBN 978-5-04-123789-6.