

Государственное бюджетное профессиональное образовательное учреждение
«Нижегородский Губернский колледж»

ОТЧЁТ ПО РАЗРАБОТКЕ МОБИЛЬНОГО ПРОГРАММНОГО ПРОДУКТА “FlyV”

Руководитель _____ Мамшева Ю.С.

02.12.2024

Студент _____ Астапчик Д.А.

02.12.2024

Специальность, группа: 09.02.07, 41П

Нижний Новгород
2024 г

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ	6
1.1 Общие сведения о программном продукте	6
1.2 Функциональные требования	6
1.3 Нефункциональные требования	8
2. ПРАКТИЧЕСКАЯ ЧАСТЬ.....	9
2.1 Проектирование архитектуры и выбор ТР	9
2.1.1 Паттерн проектирования	9
2.1.2 Проектирование требований к серверной части приложения	10
2.1.3 Проектирование требований к мобильному приложению	11
2.1.4 ТРПО для реализации мобильного приложения	12
2.2 Разработка макета приложения	13
2.2.1 Цветовая палитра.....	13
2.2.2 Шрифты	14
2.2.3 Макеты экранов	14
2.3 Разработка серверной части.....	18
2.3.1 Проектирование серверной части.....	18
2.3.2 Разработка серверной части	30
2.4 Тестирование серверной части	34
2.4.1 Тест-план	34
2.4.2 Тестирование	39
2.5 Разработка мобильного приложения	42
2.5.1 Настройка проекта	42
2.5.2 Настройка визуального стиля проекта	44
2.5.3 Подключение к серверной части	46
2.5.4 Работа с серверной частью.....	48
2.5.5 Views и ViewModels.....	52
2.5.6 Дополнительный необходимый функционал	62
2.6 Тестирование мобильного приложения.....	64

2.6.1 Разработка и реализация тестов (Android Studio)	64
2.6.2 Тест-план	66
2.6.3 Тестирование	71
2.7 Оценка эффективности использования “FlyV”	73
ЗАКЛЮЧЕНИЕ	75
ПРИЛОЖЕНИЕ А	76
ПРИЛОЖЕНИЕ Б.....	77
ПРИЛОЖЕНИЕ В	80
ПРИЛОЖЕНИЕ Г	81
ПРИЛОЖЕНИЕ Д	87

ВВЕДЕНИЕ

Разработка “FlyV” является актуальной и востребованной, по причине того, что она решает определённый перечень проблем, связанных с управлением отпускными процессами, повышает уровень автоматизации, прозрачности и удобства для сотрудников и руководства. В условиях конкурентной бизнес-среды наличие данного инструмента становится конкурентным преимуществом, повышая общую эффективность предприятия.

Основной целью разработки СУОП (система управления отпускными периодами) “FlyV” является предоставить возможность сотрудникам предприятия уйти в отпуск, взять отгул, больничный посредством использования приложения на мобильном устройстве. Аналогично, данный программный продукт предоставит возможность, руководству предприятия повысить качество контроля за кадровыми ресурсами, актуальную информацию о графиках отсутствия сотрудников на предприятии в рабочее время по тем или иным причинам, что понизит возможность возникновения конфликтных ситуаций и незапланированных отсутствий.

Задачи создания системы:

1. Упростить процесс ухода в отпуск сотрудниками предприятия, посредством выбора необходимого отпускного периода и подачи специализированной заявки руководству, посредством использования мобильного приложения;
2. Реализовать возможность взятия отгула, сотрудниками предприятия, в случаях возникновения форс-мажорных обстоятельств. Аналогично посредством использования технологий мобильной разработки;
3. Реализовать возможность взять больничный сотрудникам предприятия, посредством ранее указанных технологий;

4. Упростить процесс отслеживания и контроля кадрового ресурса, посредством предоставления информации, о планируемых отпусках, взятых больничных, отгулах и назначенных командировках, различными сотрудниками предприятия.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Общие сведения о программном продукте

На данном этапе будет рассмотрена общая информация, о разрабатываемом программном продукте.

1. Наименование системы: система управления отпускными периодами сотрудников (СУОП) – “FlyV”.

2. Характеристика: основным предназначением разрабатываемого мобильного программного обеспечения “FlyV” является реализация автоматизации процесса, что заключается в планировании и контроле отпуска в предприятия. Аналогично данное ПО, будет предоставлять более автоматизированный процесс, взятия отгула, больничного.

1.2 Функциональные требования

1. Реализовать процесс, что заключается в осуществлении авторизации и верификации пользовательских данных, при входе в разрабатываемое мобильное приложение – “FlyV”;

2. Реализовать возможность резервирования отпускных дней различными сотрудниками предприятия, посредством интерфейса разрабатываемой системы;

3. Реализовать функционал расчёта банка отпускных дней сотрудника предприятия;

4. Реализовать функционал, что заключается в предоставлении информации, сотрудникам предприятия, о их количестве банка отпускных дней, посредством графического интерфейса системы;

5. Реализовать функционал, что заключается в предоставлении сотрудникам предприятия возможности взять отгул в связи с происшествием непредвиденных обстоятельств;

6. Реализовать функционал, что предоставит возможность пользователям системы, отслеживать, кто на данный момент находится в командировке;

7. Реализовать функционал, что предоставит возможность пользователям системы, отслеживать, кто на данный момент находится в отпуске;

8. Реализовать функционал, что предоставит возможность пользователям системы, отслеживать, кто на данный момент находится на больничном;

9. Реализовать функционал, что предоставит возможность пользователям системы, отслеживать, кто на сегодняшний день взял отгул;

10. Реализовать функционал проверки условий отпуска:

- Хватает ли банка отпускных дней;
- Не должен ли в этот период сотрудник подменять другого сотрудника.

11. Реализовать, функционал, что будет производить перерасчёт баланса дней отпуска за стаж работы сотрудником на предприятии (1 год = 1 день);

12. Реализовать, функционал, что будет производить перерасчёт баланса отгулов (1 месяц = 3 дня);

13. Реализовать функционал, целью которого является установка системного напоминания в календаре мобильного телефона об отпуске, в момент, когда он будет оформлен.

1.3 Нефункциональные требования

1. Программный продукт, должен являться производительным;
2. Программный продукт, должен предоставлять безопасность и защищённость конфиденциальных данных пользователей;
3. Разрабатываемая система должна быть совместима и поддерживаться на устройствах с операционной системой Android;
4. Графический интерфейс “FlyV” должен быть адаптивным к различным Android устройствам;
5. Цветовая гамма системы: синий, зелёный, чёрный, белый, серый;
6. Разрабатываемое программное обеспечение, должно иметь приятный дизайн;
7. Разрабатываемое программное обеспечение, должно быть удобным для пользователя.

2. ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Проектирование архитектуры и выбор ТР

На данном этапе выполнения поставленной работы, было необходимо произвести проектирование архитектуры и выбор технологических решений, что будут в последствии эксплуатироваться для создания мобильного программного продукта “FlyV”.

2.1.1 Паттерн проектирования

На данном этапе проектирования программного продукта, было необходимо реализовать процесс, целью которого являлся выбор паттерна проектирования.

В результате проведения всестороннего анализа было установлено, что в процессе разработки программного обеспечения будет применяться паттерн Model-View-ViewModel (MVVM). Этот паттерн, предназначенный для обеспечения эффективной разработки программных модулей и их интеграции, способствует четкому разделению ответственности между различными компонентами системы. Паттерн MVVM структурирует код на три основные части: модель (Model), отвечающую за управление данными и бизнес-логикой, представление (View), занимающееся визуализацией данных и взаимодействием с пользователем, и модель представления (ViewModel), которая служит связующим звеном между моделью и представлением, преобразуя данные для отображения и управляя состоянием представления. Применение MVVM позволяет значительно упростить процесс разработки, тестирования и сопровождения программного обеспечения за счет отделения логики представления от пользовательского интерфейса, что, в свою очередь, улучшает тестируемость и модульность кода.

2.1.2 Проектирование требований к серверной части приложения

На данном этапе реализации программного продукта, было необходимо реализовать процесс, целью которого являлся проектирование требований к серверной части программного обеспечения.

В результате проведения всестороннего анализа было выявлено, что в качестве серверной части программного продукта будет использоваться специализированный комплекс серверных услуг – Supabase. Supabase основан на базе данных PostgreSQL, что является мощной и масштабируемой реляционной БД. Данное решение предоставляет услуги по эффективному управлению данными и их безопасному хранению. Аналогично, данный сервис имеет интеграцию с мобильными приложениями. Данный комплекс предоставляет функционал по автоматическому обновлению данных без необходимости использования специализированного запроса. Ранее указанный функционал осуществляется посредством того, что Supabase включает в себя поддержку функционала реального времени (real-time). Аналогично Supabase обеспечивает высокий уровень качества безопасности хранения данных, посредством предоставления различных механизмов аутентификации, авторизации и функционала, что осуществляет хэширование паролей пользователей системы.

2.1.3 Проектирование требований к мобильному приложению

На данном этапе проектирования программного продукта, было необходимо реализовать процесс, целью которого являлся проектирование требований к мобильному приложению.

По причине того, что ранее были полноценно рассмотрены функциональные и нефункциональные требования к мобильному программному продукту FlyV, будет рассмотрена UML-диаграмма вариантов использования (рис 1), в качестве иллюстрирующего материала для того, чтобы улучшить восприятие информации и упростить восприятие информации.

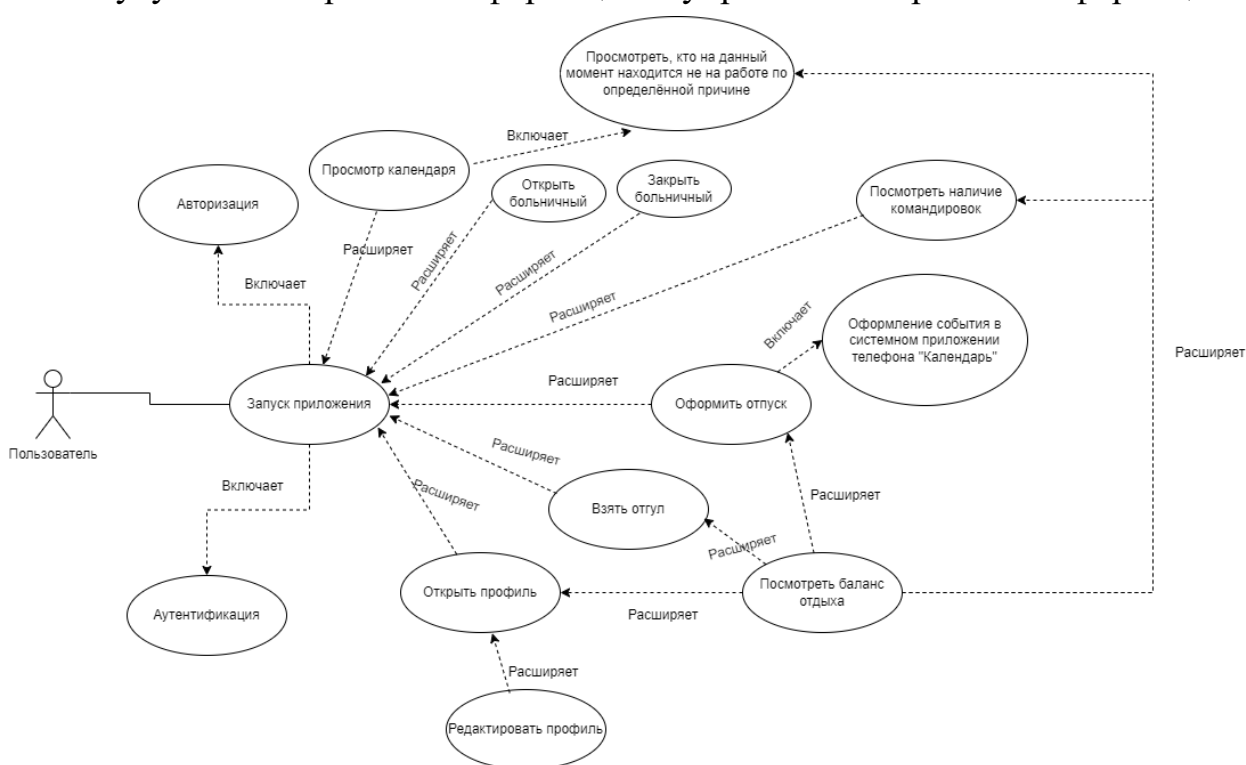


Рисунок 1. UML-диаграмма вариантов использования к системе “FlyV”

Для разработки данного программного продукта должен использоваться язык программирования Kotlin, что является официальным языком для разработки Android-приложений. Аналогично, данный ЯП обеспечивает безопасную и современную среду разработки.

2.1.4 ТРПО для реализации мобильного приложения

На данном этапе проектирования программного продукта, было необходимо реализовать процесс, целью которого являлся выбор ТРПО.

В результате проведения всестороннего анализа различных технологий разработки мобильного программного обеспечения, было принято решение использовать Figma для проектирования интерфейса, Android Studio (AS) в качестве интегрированной среды разработки (IDE) и язык программирования Kotlin.

Figma была выбрана в связи с следующими преимуществами: понятный интерфейс, наличие большого количества инструментов для создания различных прототипов, доступ к проектам с любого устройства, возможность экспорта различных элементов интерфейса – совместное использование всех возможностей Figma предоставит возможность произвести эффективное проектирование дизайна программного продукта “FlyV”.

Android Studio был выбран в качестве IDE, по причине того, что является официальной средой разработки для Android-приложений. Использование данной интегрированной среды разработки предоставит возможность использования большого количества технологий, что позволят создать необходимый дизайн и организовать логику работы программного продукта. Язык программирования Kotlin имеет высокую совместимость с IDE Android Studio и предназначен для написания программного кода для Android-приложений. Аналогично AS предоставляет мощные инструменты для отладки и профилирования приложений. Встроенные эмуляторы позволяют тестировать приложения на различных устройствах, что имеют операционную систему Android. Совместно, все ранее указанные пункты, предоставляют основание на решение использовать данное инструментальное программное обеспечение.

2.2 Разработка макета приложения

На данном этапе разработки мобильного программного продукта, было необходимо реализовать процесс, целью которого является разработка макета приложения. Рассмотрим результат произведённой деятельности.

2.2.1 Цветовая палитра

Рассмотрим, цветовую палитру, что была составлена, для разработки мобильного программного продукта “FlyV” (рис 2).

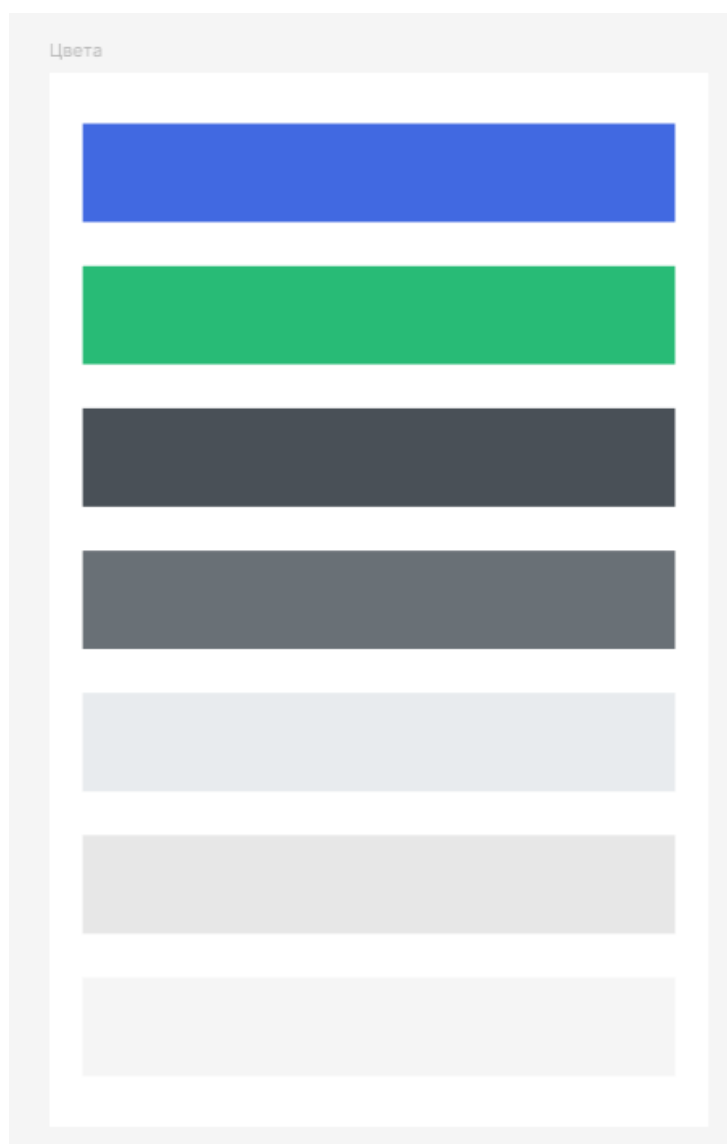


Рисунок 2 – Цветовая палитра мобильного приложения “FlyV”

2.2.2 Шрифты

В результате произведённого всестороннего анализа, было реализовано решение, что заключается в выборе специализированного шрифта “Inter” для эксплуатации в мобильном приложении в качестве заголовков и основного текста.

2.2.3 Макеты экранов

На данном этапе был произведён процесс, целью которого, являлась разработка макетов всех экранов мобильного программного продукта, с учётом ранее составленной цветовой палитры и выбранных шрифтов. Для того, чтобы избежать избыточности иллюстрирующего материала будут рассмотрены не все макеты окон. Чтобы рассмотреть все макеты необходимо перейти на проект в Figma по следующей ссылке: <https://www.figma.com/design/q6hvp1pT2QsWfcxqRGqnOh/FlyV?node-id=0-1&t=xfBbl48xgrvOwfDM-1> – или смотреть ссылку в репозитории проекта ПП.

Рассмотрим некоторый результат проектирования макетов окон мобильного приложения (рис 3-5).

Авторизация

FlyV

Рисунок 3 – Макет окна “Авторизация”

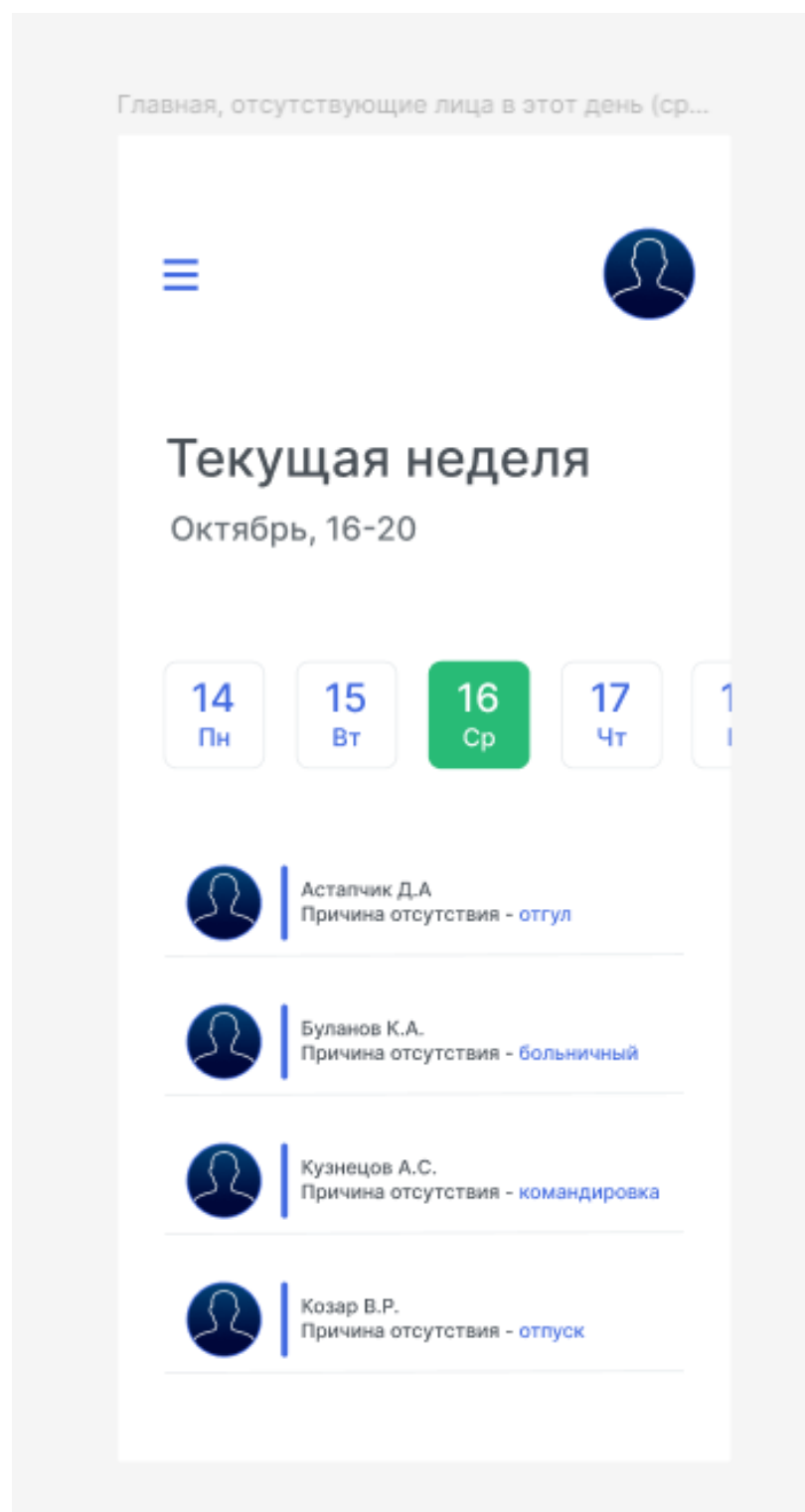


Рисунок 4 – Макет окна “Главная”

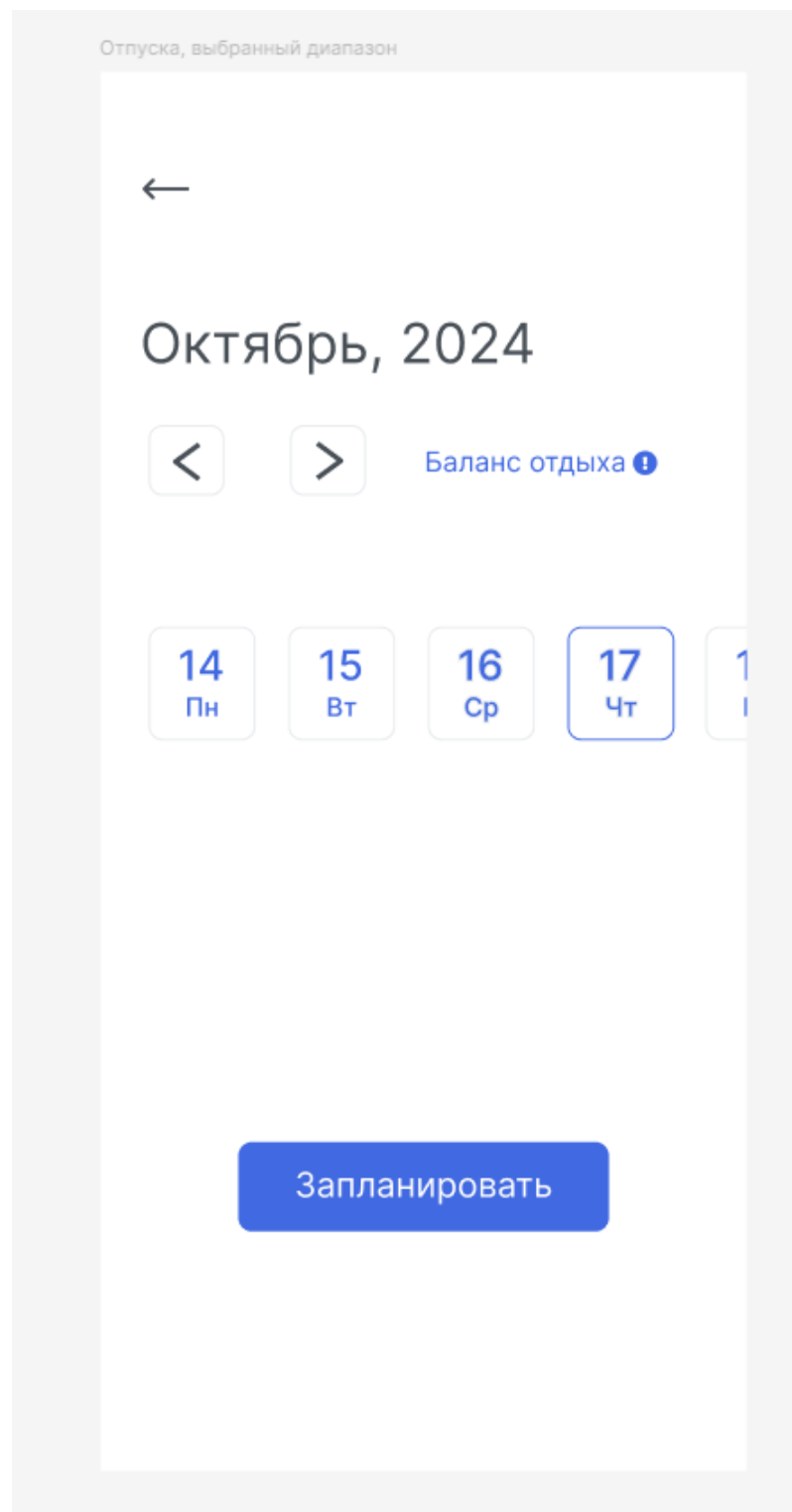


Рисунок 5 – Макет окна “Отпуска”

2.3 Разработка серверной части

На данном этапе разработки мобильного программного продукта “FlyV” было необходимо осуществить процессы, целью которых является проектирование и разработка серверной части. Рассмотрим результат осуществлённой деятельности.

2.3.1 Проектирование серверной части

На данном этапе было необходимо произвести процесс, целью которого является проектирование серверной части мобильного программного продукта. Для того, чтобы реализовать поставленную задачу, было необходимо выполнить следующий перечень задач:

1. Составить список сущностей необходимых для разработки программного обеспечения;
2. Составить описание атрибутов сущностей;
3. Составить список связей между сущностями;
4. Произвести создание ER-диаграммы “FlyV”;
5. Произвести создание даталогической модели программного продукта.

Список сущностей.

На основании произведённого анализа технического задания, был выявлен следующий перечень информации, необходимый для практического функционирования мобильного программного продукта “FlyV”:

1. Должна храниться информация о сотрудниках предприятия;
2. Должна храниться информация о различных официальных причинах отсутствия сотрудника на рабочем месте;
3. Должна храниться информация о городах, в которых работают сотрудники предприятия;
4. Должна храниться информация о заменах сотрудников другими сотрудниками предприятия на различных проектах, в период отсутствия определённых работников по определённой причине;
5. Должна храниться информация о сотрудниках предприятия и причине их отсутствия на определённый период времени;
6. Должна храниться некоторая информация о модулях, что должны будут выполнены за определённый период времени;
7. Должна храниться информация о сотруднике, что реализовывают совместно определённые модули.

Рассмотрим список необходимых сущностей (табл 1).

Таблица 1. Список сущностей

Наименование	Назначение
Сотрудники	Хранение информации о сотрудниках
Причины отсутствия	Хранение информации о всех официальных причинах отсутствия сотрудников на предприятии в рабочее время
Города	Хранение информации о городах, в которых производят работают сотрудники
Замены	Хранение информации о сотрудниках, что будет отсутствовать по определённой причине, и заменять отсутствующего сотрудника в определённый период времени
Отсутствия и сотрудники	Хранение информации о сотрудниках и их причинах отсутствия в рабочее время на предприятии
Модули	Хранение информации о модулях
Модули и сотрудники	Хранение информации о модулях и сотрудников, которые реализовывают данные модули

Описание атрибутов и сущностей.

На основании информации о сущностях были составлены атрибуты таблиц, выражающие определённое свойство. Рассмотрим результат произведённой деятельности (табл 2-8).

Таблица 2. Атрибуты сущности “Сотрудники”

Ключевое поле	Название	Назначение
ПК (Первичный ключ)	Идентификатор сотрудника	Ключевое поле, предназначенное для однозначной идентификации каждой записи в таблице. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждому сотруднику
ВК (Внешний ключ)	Идентификатор города	Внешний ключ – это атрибут отношения, ключ) который является первичным ключом другого отношения. В данном случае это атрибут сущности города. С помощью внешнего ключа будет определено какой город относится к конкретному пользователю
	Полное имя	
	Дата устройства	
	Количество отгулов	
	Баланс отпускных дней	
	Номер телефона	
	Работает/не работает	
	Персональная почта	
	Ссылка на фото профиля	

Таблица 3. Атрибуты сущности “Причины отсутствия”

Ключевое поле	Название	Назначение
ПК (Первичный ключ)	Идентификатор причины отсутствия	Ключевое поле, предназначенное для однозначной идентификации каждой записи в таблице. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждой причине отсутствия
	Причина	

Таблица 4. Атрибуты сущности “Города”

Ключевое поле	Название	Назначение
ПК (Первичный ключ)	Идентификатор города	Ключевое поле, предназначенное для однозначной идентификации каждой записи в таблице. Представляет собой первичный ключ. Это уникальное значение, соответствующее городу, в котором работают сотрудники компании
	Город	

Таблица 5. Атрибуты сущности “Замены”

Ключевое поле	Название	Назначение
ПК (Первичный ключ)	Идентификатор замены	Ключевое поле, предназначенное для однозначной идентификации каждой записи в таблице. Представляет собой первичный ключ. Это уникальное значение, соответствующее замене
ВК (Внешний ключ)	Идентификатор первого заменяемого сотрудника	Внешний ключ – это атрибут отношения, ключ) который является первичным ключом другого отношения. В данном случае это атрибут сущности сотрудники. С помощью внешнего ключа будет определено какой сотрудник является заменяющим
ВК (Внешний ключ)	Идентификатор второго заменяемого сотрудника	Внешний ключ – это атрибут отношения, ключ) который является первичным ключом другого отношения. В данном случае это атрибут сущности сотрудники. С помощью внешнего ключа будет определено какой сотрудник является заменяемым

Таблица 6. Атрибуты сущности “Отсутствия и сотрудники”

Ключевое поле	Название	Назначение
ПК (Первичный ключ)	Идентификатор отсутствия сотрудника	Ключевое поле, предназначенное для однозначной идентификации каждой записи в таблице. Представляет собой первичный ключ. Это уникальное значение, соответствующее отсутствию сотрудника
ВК (Внешний ключ)	Идентификатор причины	Внешний ключ – это атрибут отношения, ключ) который является первичным ключом другого отношения. В данном случае это атрибут сущности причины отсутствия. С помощью внешнего ключа будет определена причина отсутствия сотрудника
ВК (Внешний ключ)	Идентификатор сотрудника	Внешний ключ – это атрибут отношения, ключ) который является первичным ключом другого отношения. В данном случае это атрибут сущности сотрудники. С помощью внешнего ключа будет определен сотрудник, который отсутствует по определённой причине
	Дата начала отсутствия	
	Количество дней отсутствия	

Таблица 7. Атрибуты сущности “Модули”

Ключевое поле	Название	Назначение
ПК (Первичный ключ)	Идентификатор модуля	Ключевое поле, предназначенное для однозначной идентификации каждой записи в таблице. Представляет собой первичный ключ. Это уникальное значение, соответствующее модулю
	Модуль	
	Дата начала выполнения	
	Дата окончания выполнения	
	Нужное количество сотрудников	

Таблица 8. Атрибуты сущности “Модули и сотрудники”

Ключевое поле	Название	Назначение
ПК (Первичный ключ)	Идентификатор модуля и сотрудника	Ключевое поле, предназначенное для однозначной идентификации каждой записи в таблице. Представляет собой первичный ключ. Это уникальное значение, соответствующее модулю и сотруднику
ВК (Внешний ключ)	Идентификатор модуля	Внешний ключ – это атрибут отношения, ключ) который является первичным ключом другого отношения. В данном случае это атрибут сущности модули. С помощью внешнего ключа будет определен модуль, над которым работает сотрудник
ВК (Внешний ключ)	Идентификатор сотрудника	Внешний ключ – это атрибут отношения, ключ) который является первичным ключом другого отношения. В данном случае это атрибут сущности сотрудники. С помощью внешнего ключа будет определен сотрудник, который работает над определённым модулем

Список связей между сущностями.

На данном этапе был составлен перечень, информации, отражающий все связи между сущностями, необходимыми, для эффективной разработки программного обеспечения “FlyV” (табл 9).

Таблица 9. Список связей

№	Название связи	Сущности, участвующие в связи	Назначение
1	1: M	Города – Сотрудники	В одном городе могут проживать различные сотрудники
2	1: M	Причины отсутствия – Отсутствия и сотрудники	Различные сотрудники могут использовать одну причину отсутствия
3	1: M	Сотрудники – Отсутствия и сотрудники	Один сотрудник может использовать различные причины отсутствия
4	1:1	Сотрудники – Замены	Один сотрудник может заменить другого сотрудника
5	1:1	Сотрудники – Замены	Один сотрудник может быть заменён одним сотрудником
6	1: M	Модули – Модули и сотрудники	Над одним модулем может работать множество сотрудников
7	1: M	Сотрудники – Модули и сотрудники	Один сотрудник может работать над множеством модулей

ER-диаграмма.

На данном этапе, исходя из произведённого анализа всех необходимых данных, были реализованы процессы, целью которых является разработка сущностей и установка специализированных связей (рис 6).

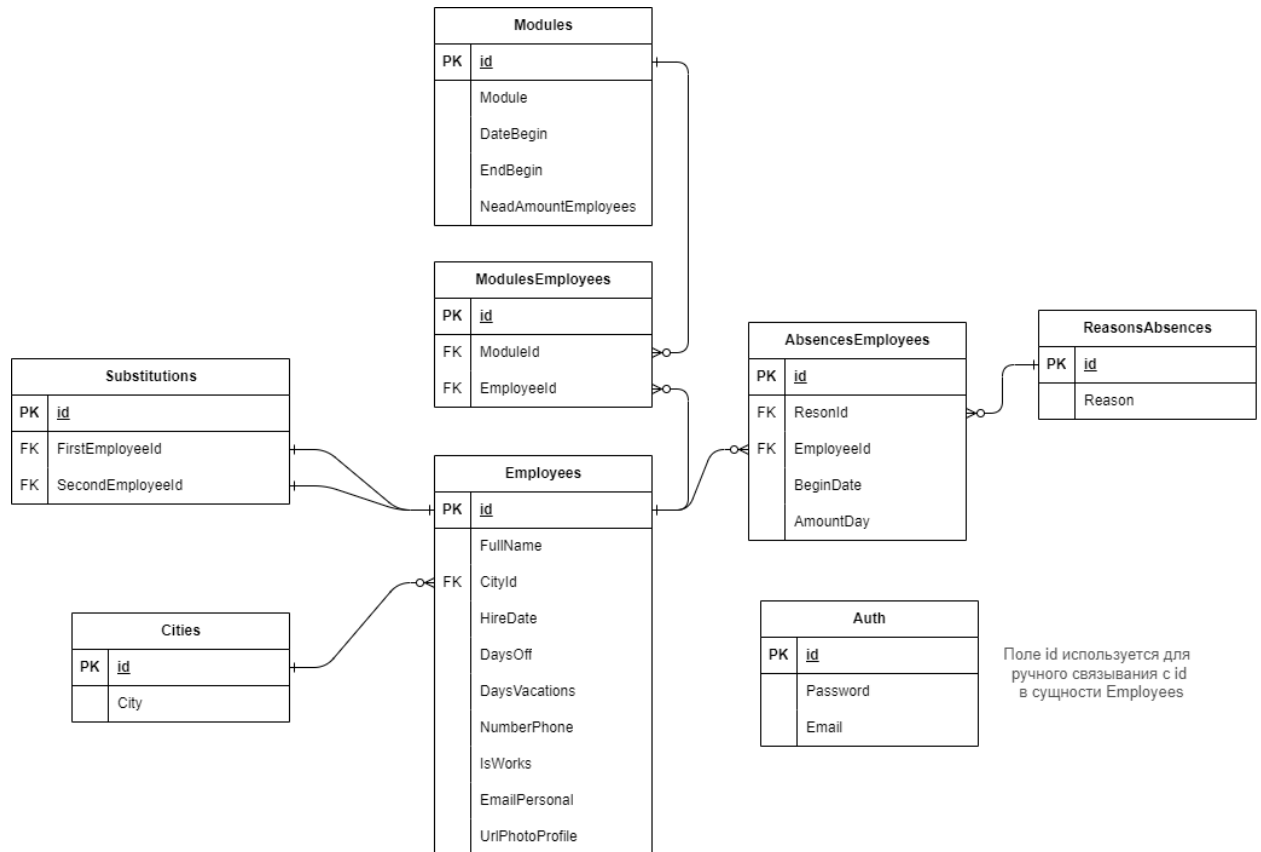


Рисунок 6 – ER-диаграмма для ИП "FlyV"

Аналогично для того, чтобы повысить уровень понимания взаимосвязей сущностей на разных уровнях, было необходимо представить сущность Auth, что является индивидуальной системой, предоставляющей различный функционал. Данная система может в себе содержать почту и захешированный пароль пользователя, а также предоставляет функционал авторизации пользователя. Специализированный id из сущности Auth, вручную устанавливается конкретному пользователю системы, в сущности, Employees в поле id.

Даталогическая модель базы данных “FlyV”

В последствии был произведён процесс, целью которого являлась разработка даталогической модели базы данных посредством эксплуатации средств платформы Supabase. Реализованную даталогическую модель базы данных смотреть в Приложение А.

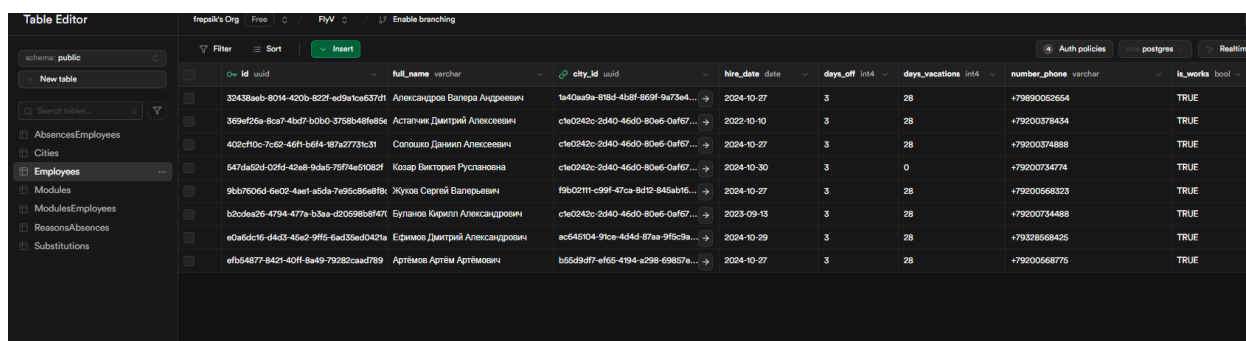
2.3.2 Разработка серверной части

На данном этапе было необходимо произвести процесс разработки серверной части, посредством использования всех необходимых средств специализированной платформы Supabase. Для того, чтобы реализовать поставленную задачу, было необходимо произвести следующий перечень заданий:

1. Разработка и создание всех сущностей, отношений между ними и заполнение данными;
2. Разработка и создание специализированных процедур и планировщиков заданий.

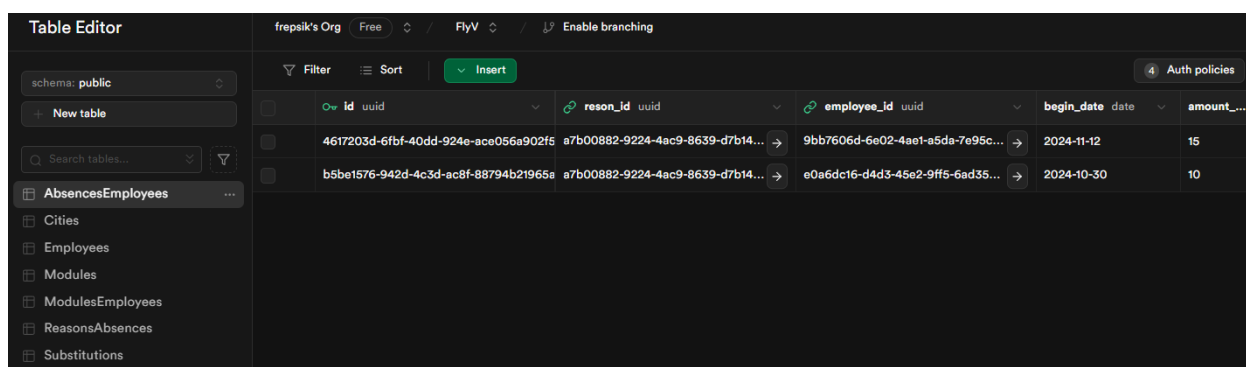
Создание сущностей.

Для того, чтобы избежать избыточности материала, будут рассмотрены сущности, что чаще использовались в ходе разработки мобильного приложения. Для того, чтобы провести анализ созданных сущностей, смотреть в Приложение Б. Рассмотрим результат произведённой деятельности (рис 7,8).



id	full_name	city_id	hire_date	days_off	days_vacations	number_phone	is_works
32438aeb-8014-420b-822f-ed9a1ce637d1	Александров Валера Андреевич	1e40aa9a-818d-4b8f-869f-9a73e4...	2024-10-27	3	28	+79890052654	TRUE
369ef26a-8ca7-4bd7-b0b0-3768b48f85e4	Астапкин Дмитрий Алексеевич	c1e0242c-2d40-46d0-80e6-0af67...	2022-10-10	3	28	+79200378434	TRUE
402cf0c0-7c62-46f8-b6f4-187a2773c31	Солошко Даниил Алексеевич	c1e0242c-2d40-46d0-80e6-0af67...	2024-10-27	3	28	+79200374888	TRUE
547da30f-02f6-42e8-9da5-7574a5f082f	Козар Викторья Руслановна	c1e0242c-2d40-46d0-80e6-0af67...	2024-10-30	3	0	+79200734774	TRUE
9bb7606d-6e02-4ae1-a5da-7e95c8e8f8f8	Жуков Сергей Валерьевич	f9b0211f-c99f-47ca-8d12-845ab16...	2024-10-27	3	28	+79200568323	TRUE
b2cdea25-4794-477a-53aa-d20598e8f47f	Бутанов Кирилл Александрович	c1e0242c-2d40-46d0-80e6-0af67...	2023-09-13	3	28	+79200734488	TRUE
e0a6dc16-d4d3-45e2-9ff5-6ad35e0421a	Ефимов Дмитрий Александрович	ac648f04-91ce-4d4d-87aa-9f5c9a...	2024-10-29	3	28	+79328568425	TRUE
efb54877-8421-40ff-8a49-79282caad789	Артемьев Артем Артемьевич	b56d9df7-ef65-4194-a298-69857e...	2024-10-27	3	28	+79200568775	TRUE

Рисунок 7 – Результат создания сущности “Сотрудники”



id	reson_id	employee_id	begin_date	amount
4617203d-6fbf-40dd-924e-ace056a902f6	a7b00882-9224-4ac9-8639-d7b14...	9bb7606d-6e02-4ae1-a5da-7e95c...	2024-11-12	15
b5be1576-942d-4c3d-ac8f-88794b21965a	a7b00882-9224-4ac9-8639-d7b14...	e0a6dc16-d4d3-45e2-9ff5-6ad35...	2024-10-30	10

Рисунок 8 – Результат создания сущности “Отсутствия и сотрудники”

В последствии был реализован процесс, что заключался в добавлении специализированных политик, что предоставят возможность вести определённую работу с данными на клиенте мобильного программного продукта. Аналогично для того, чтобы избежать избыточности материала будут рассмотрены политики, для часто используемых сущностей. Для того, чтобы провести анализ созданных политик смотреть в Приложение Б.

Рассмотрим результат произведённой деятельности (рис 9-10).

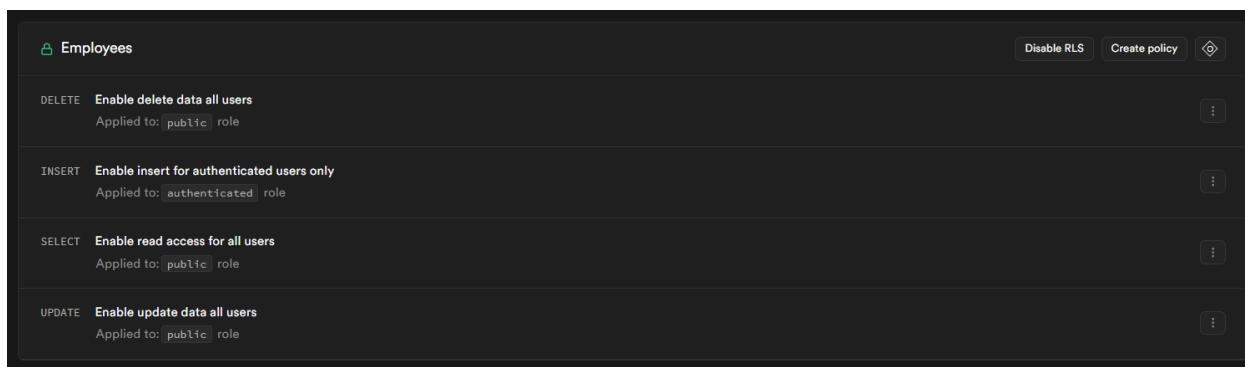


Рисунок 9 – Результат добавления политик для сущности “Сотрудники”

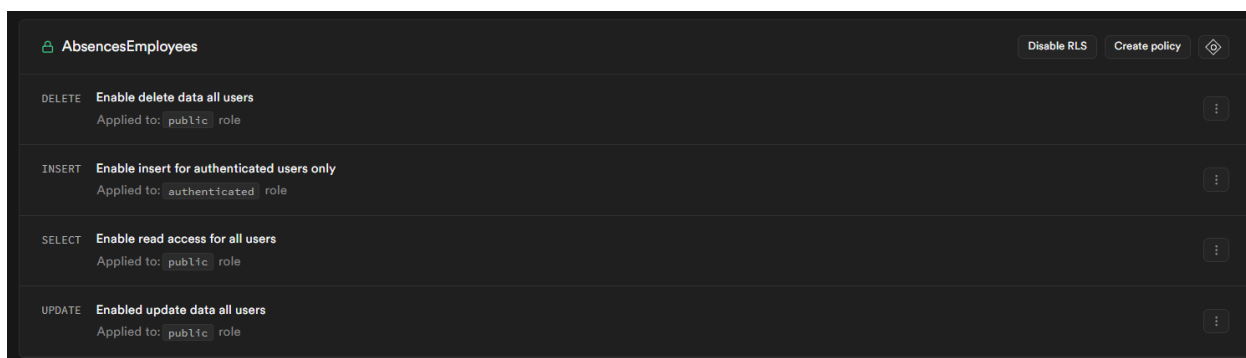


Рисунок 10 – Результат добавления политик для сущности
“Отсутствия и сотрудники”

Создание процедур и планировщиков заданий.

На данном этапе было необходимо реализовать задачу, что заключалась в необходимости производить автоматизированный перерасчёт баланса отпускных дней каждого сотрудника предприятия, спустя 1 рабочий год. Аналогично, было необходимо произвести алгоритм, цель которого заключалась в автоматизированном перерасчёте количества отгулов в начале рабочего месяца.

Данная задача была реализована посредством создания специализированных процедур и планировщиков заданий. Рассмотрим результат произведённой деятельности (рис 11-15).

```
1  create
2  or replace procedure UpdateDaysOffOnceAMonth()
3  language plpgsql as $$
4  begin
5      update "Employees"
6      set days_off = 3
7      where days_off < 3;
8  end;
9  $$;
```

Рисунок 11 – Процедура обновления отгулов, в начале рабочего месяца

```
11  create
12  or replace procedure UpdateDaysVacationsUserAfterOneYearWorks()
13  language plpgsql as $$
14  begin
15      update "Employees"
16      set days_vacations = 28
17      where extract(month from hire_date) = extract(month from now())
18      and extract(day from hire_date) = extract(day from now() at time zone 'Europe/Moscow')
19      and days_vacations < 28;
20  end;
21  $$;
```

Рисунок 12 – Процедура обновления баланса отпускных дней, спустя 1
рабочий год

Для того, чтобы реализовать автоматизированный запуск ранее представленных процедур, необходимо использовать специализированное расширение pg_cron – планировщик заданий. Изначально необходимо включить данное расширение посредством команды, что указана на рисунке 13.

```
41 create extension if not exists pg_cron;
```

Рисунок 13 – Команда добавления расширения pg_cron

Далее были созданы следующие планировщики (рис 20, 21), заданий, что осуществляют свой запуск по следующим правилам:

1. TaskUpdateDaysOffOnceAMonth – запуск в начале каждого месяца 1 числа;
2. TaskUpdateDaysVacationsUserAfterOneYearWorks – запуск спустя один рабочий год сотрудника.

```
24 select
25 cron.schedule (
26     'TaskUpdateDaysOffOnceAMonth',
27     '0 0 1 * *', /*в 0 минут, в 12 часов, в 1 день месяца, каждый месяц, в любой день недели*/
28     'CALL UpdateDaysOffOnceAMonth()'
29 );
```

Рисунок 14 – cron - TaskUpdateDaysOffOnceAMonth

```
32 select
33 cron.schedule (
34     'TaskUpdateDaysVacationsUserAfterOneYearWorks',
35     '0 0 * * *', /*в 0 минут, в 12 часов, каждый день месяца, каждый месяц, в любой день недели*/
36     'CALL UpdateDaysVacationsUserAfterOneYearWorks()'
37 );
```

Рисунок 15 – cron – TaskUpdateDaysVacationsUserAfterOneYearWorks

2.4 Тестирование серверной части

На данном этапе выполнения процессов, связанных с разработкой мобильного программного обеспечения, было необходимо осуществить процесс, целью которого являлось произвести полноценное тестирование серверной части приложения “FlyV”.

2.4.1 Тест-план

На данном этапе было необходимо произвести процесс, целью которого является реализация процесса, что заключается в разработке специализированного тест-плана для серверной части приложения “FlyV”.

Рассмотрим перечень пунктов, что содержит тест-план:

1. Функционал серверной части, что будет подвержен тестированию;
2. Критерии качества;
3. Перечень документации, что будет использоваться для тестирования серверной части мобильного программного продукта;
4. Стратегия тестирования серверной части приложения;
5. Ресурсы.

Функционал серверной части.

Рассмотрим функционал серверной части мобильного программного продукта “FlyV”, что будет подвержен тестированию, по причине необходимости, выявить несоответствия работоспособности функционала, которые в последствии могут возникнуть в ходе создание программного кода мобильного приложения:

1. Вывод информации из сущности “Cities”;
2. Вывод информации из сущности “Substitutions”;
3. Вывод информации из сущности “Modules”;
4. Вывод информации из сущности “ModulesEmployees”;
5. Вывод информации из сущности “ReasonsAbsences”;
6. Вывод и обновление информации в сущности “Employees”;
7. Вывод, обновление и добавление информации в сущность “AbsencesEmployees”;
8. Процедура “UpdateDaysOffOnceAMonth”;
9. Процедура “UpdateDaysVacationsUserAfterOneYearWorks”;
10. Планировщик заданий “TaskUpdateDaysVacationsUserAfterOneYearWorks”;
11. Планировщик заданий “TaskUpdateDaysOffOnceAMonth”.

Критерии качества.

На данном этапе было необходимо составить список критериев, что будут эксплуатироваться для реализации процесса, что заключается в оценке качества серверной части мобильного программного продукта “FlyV”:

1. Функциональность:

- Весь ранее указанный функционал является работоспособным;

2. Надёжность:

- Функционал серверной части приложения должен производить задуманное в период осуществления работы, исключая сценарии возникновения различных дефектов;

- Должны присутствовать механизмы защиты, предоставляющие возможность обрабатывать различные исключительные ситуации.

3. Удобство сопровождения:

- Должна присутствовать возможность расширения и сопровождения серверной части мобильного программного продукта.

4. Эффективность:

- Время отклика от сервера должно находиться в пределах нормы.

Перечень тестовой документации.

По причине того, что список тестовой документации, будет рассмотрен не единожды и для того, чтобы исключить избыточность материала, тестовую документацию, что будет эксплуатироваться в целях проведения эффективного тестирования серверной части мобильного программного продукта “FlyV”, смотреть в Приложение В.

Стратегия тестирования.

На данном этапе выполнения поставленной задачи, было необходимо произвести создание специализированной стратегии тестирования серверной части ранее указанного мобильного приложения (табл 10).

Таблица 10. Стратегия тестирования серверной части приложения “FlyV”

Тестирование	Процедуры тестирования
Функциональное	<ol style="list-style-type: none">1. Проверить результат работы запроса на вывод всей информации из сущности “Cities”;2. Проверить результат работы запроса на вывод всей информации из сущности “Substitutions”;3. Проверить результат работы запроса на вывод всей информации из сущности “Modules”;4. Проверить результат работы запроса на вывод всей информации из сущности “Employees”;5. Проверить результат работы запроса на вывод всей информации из сущности “ReasonsAbsences”;6. Проверить результат работы запроса на вывод всей информации из сущности “AbsencesEmployees”;7. Проверить результат работы запроса на вывод всей информации из сущности “ModulesEmployees”;8. Проверить результат работы запроса на обновление информации в сущности “Employees”;

	<p>9. Проверить результат работы запроса на обновление информации в сущности “AbsencesEmployees”;</p> <p>10. Проверить результат работы запроса на добавление информации в сущность “AbsencesEmployees”;</p> <p>11. Проверить результат работы процедуры “UpdateDaysOffOnceAMonth”;</p> <p>12. Проверить результат работы процедуры “UpdateDaysVacationsUserAfterOneYearWorks”;</p> <p>13. Проверить результат работы планировщика заданий “TaskUpdateDaysOffOnceAMonth”;</p> <p>14. Проверить результат работы планировщика заданий “TaskUpdateDaysVacationsUserAfterOneYearWorks”.</p>
--	--

Ресурсы.

Рассмотрим перечень ресурсов, что необходимы для успешной реализации процесса тестирования серверной части мобильного ПП.

Технические ресурсы:

1. Среда тестирования: Supabase;
2. База данных: реализованная база данных “FlyV”, посредством использования ресурсов сервиса Supabase.

Программные ресурсы:

1. Supabase SQL Editor: встроенный редактор в Supabase, что предназначен для написания и выполнения SQL-запросов.

Человеческие ресурсы:

1. Тестировщик;
2. Разработчик.

Временные ресурсы:

1. Время на реализацию всех ранее описанных тестов к серверной части программного продукта.

2.4.2 Тестирование

На данном этапе разработки документации, по тестированию серверной части, был произведён процесс тестирования, ранее описанного функционала в тест-плане. Рассмотрим составленные тестовые наборы:

1. Тестовый набор: – “Запросы на select”;
2. Тестовый набор: – “Запросы на update”;
3. Тестовый набор: – “Запрос на insert”;
4. Тестовый набор: – “Запросы на использование процедур”;
5. Тестовый набор: – “Планировщики заданий”.

Для того, чтобы избежать избыточности материала, будут рассмотрены не все тест-кейсы. Для того, чтобы рассмотреть все составленные тест-кейсы по серверной части, смотреть Приложение Г.

Тестовый набор: – “Запросы на select”

Предварительные условия: для того, чтобы осуществить тест-кейсы из данного тестового набора, необходимо открыть сервис Supabase, иметь доступ к базе данных “FlyV” и открыть SQL Editor.

Тест-кейс, вывод (select) информации из сущности “AbsencesEmployees” (табл 11).

Таблица 11. Тест-кейс – 6

Шаги	Ожидаемый результат
В пустую строку SQL Editor ввести следующую команду: select * from "AbsencesEmployees"	В выбранной строке будет введена команда: select * from "AbsencesEmployees"
Нажать комбинацию клавиш CTRL + Enter	Пользователю будут выведены все данные, что хранятся в сущности “AbsencesEmployees”

Тестовый набор: – “Запросы на update”

Предварительные условия: для того, чтобы осуществить тест-кейсы из данного тестового набора, необходимо открыть сервис Supabase, иметь доступ к базе данных “FlyV” и открыть SQL Editor.

Тест-кейс, обновление (update) информации в сущности “Employees” (табл 12).

Таблица 12. Тест-кейс – 1

Шаги	Ожидаемый результат
В пустую строку SQL Editor ввести следующую команду: update "Employees" set number_phone = '+79200754345' where full_name = 'Астапчик Дмитрий Алексеевич'	В выбранной строке будет введена команда: update "Employees" set number_phone = '+79200754345' where full_name = 'Астапчик Дмитрий Алексеевич'
Нажать комбинацию клавиш CTRL + Enter	Пользователь обновит информацию в сущности “Employees”

Тестовый набор: – “Запросы на insert”

Предварительные условия: для того, чтобы осуществить данный тест-кейс, необходимо открыть сервис Supabase, иметь доступ к базе данных “FlyV” и открыть SQL Editor.

Тест-кейс, добавление (insert) информации в сущность “AbsencesEmployees” (табл 13).

Таблица 13. Тест-кейс – 1

Шаги	Ожидаемый результат
В пустую строку SQL Editor ввести следующую команду: insert into "AbsencesEmployees" (reson_id, employee_id, begin_date, amount_day) values ('a7b00882-9224-4ac9-8639- d7b147bde60e', '9bb7606d-6e02-4ae1- a5da-7e95c86e8f8c', '2024-11-12', 15)	В выбранной строке будет введена команда: insert into "AbsencesEmployees" (reson_id, employee_id, begin_date, amount_day) values ('a7b00882-9224-4ac9-8639- d7b147bde60e', '9bb7606d-6e02-4ae1- a5da-7e95c86e8f8c', '2024-11-12', 15)
Нажать комбинацию клавиш CTRL + Enter	Пользователь добавит запись в сущность “AbsencesEmployees”

2.5 Разработка мобильного приложения

На данном этапе были произведены процессы, целью которых являлось, проектирование архитектуры, написание программного кода и отладка, что совместно составляет разработку мобильного приложения “FlyV”.

Для исполнения цели, что заключается в анализе программного кода, разработанного мобильного программного продукта, необходимо смотреть проект, что содержится в каталоге project следующего репозитория <http://gogs.ngknn.ru:3000/Freps/FlyV>.

Рассмотрим результат осуществлённой деятельности.

2.5.1 Настройка проекта

Изначально, по причине необходимости эксплуатировать специализированные ресурсы, нужно было реализовать процесс, целью которого является подключение различных зависимостей, что предоставят возможность эффективно произвести функционал, соответствующий указанным требованиям. Данные требования были ранее рассмотрены в теоретической части.

Перечень, того, что необходимо добавить, настроить для осуществления эффективной разработки:

1. Добавить зависимости Supabase;
2. Добавить зависимость для навигации;
3. Добавить зависимость для отрисовки AsyncImage;
4. Добавить зависимости для проведения тестирования;
5. Настроить плагин сериализации;
6. Добавить разрешения на использование интернета в приложении;
7. Добавить разрешения для работы с текущим интернет-соединением;
8. Добавить разрешения для работы с системным календарём на телефоне пользователя.

Рассмотрим результат произведённой деятельности (рис 16-18).

```

//Supabase
implementation(libs.ktor.client.okhttp)
implementation(platform(libs.supabase.bom))
implementation(libs.postgrest.kt)
implementation(libs.realtime.kt)
implementation(libs.storage.kt)

//Navigation
implementation(libs.androidx.navigation.compose)

//Coil
implementation(libs.coil.compose)

//Тестирование.
testImplementation(libs.kotlinx.coroutines.test) //Нужно для использования специализирован
testImplementation(libs.mockk) //Для того, чтобы я мог заглушки на методы ставить

```

Рисунок 16 – Добавление зависимостей

```

plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.kotlin.android)
    kotlin("plugin.serialization") version "1.9.0"
}

```

Рисунок 17 – Настройка плагина сериализации

```

<!--Интернет-->
<uses-permission android:name="android.permission.INTERNET" />
<!--Для работы с текущим соединением-->
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

<!--Для работы с календарём-->
<uses-permission android:name="android.permission.WRITE_CALENDAR" />
<uses-permission android:name="android.permission.READ_CALENDAR" />

```

Рисунок 18 – Добавление разрешений

2.5.2 Настройка визуального стиля проекта

На данном этапе была произведена первичная настройка визуального стиля проекта, что включает в себя следующий перечень задач:

1. Добавление шрифтов;
2. Добавление цветовой палитры приложения;
3. Добавление специализированных иконок в формате .xml (аналог .svg);
4. Настройка логотипа мобильного приложения.

Рассмотрим результат осуществлённой деятельности (рис 19-)

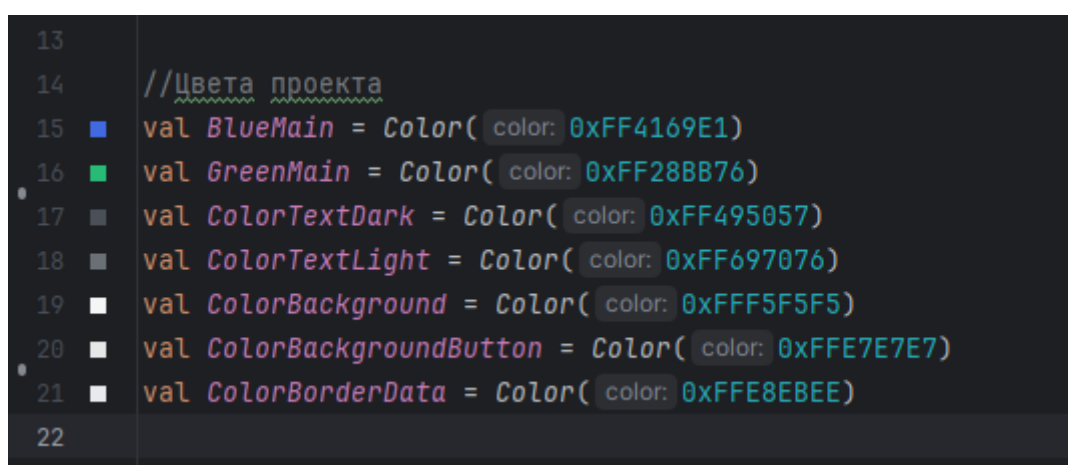


Рисунок 19 – Color.kt, цветовая палитра проекта

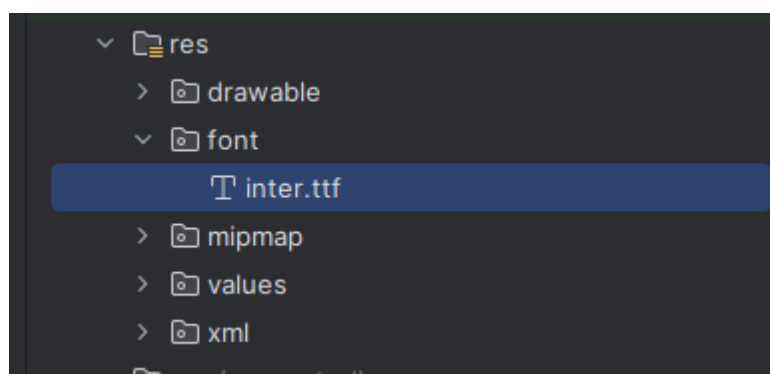


Рисунок 20 – res, font, добавленный шрифт inter.ttf

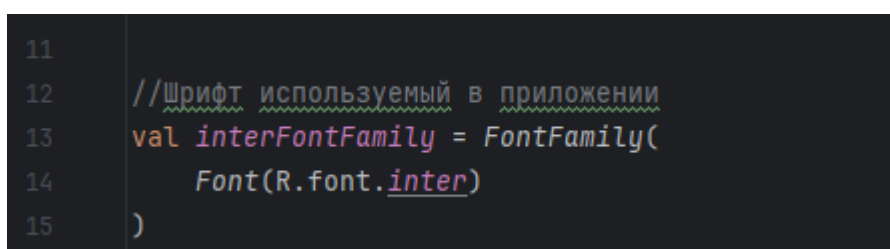


Рисунок 21 – Type.kt, шрифт используемый в приложении

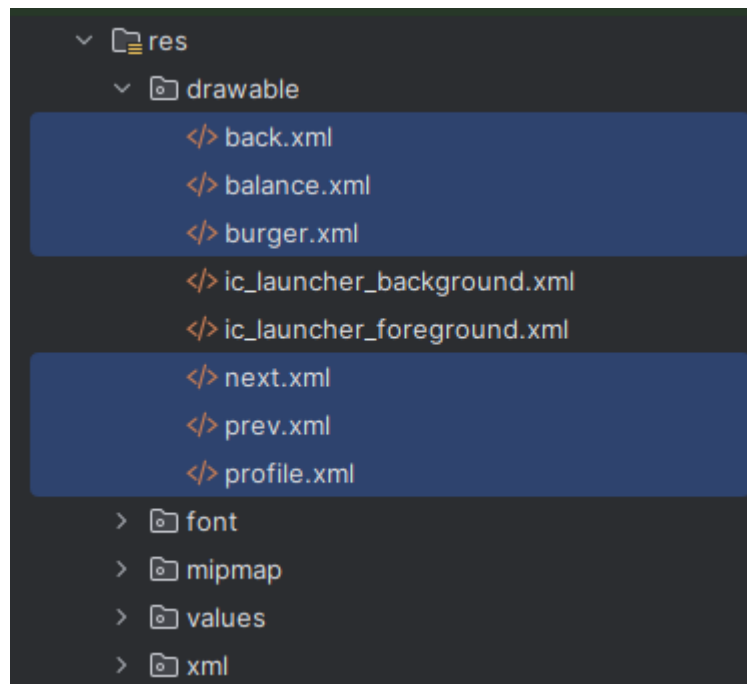


Рисунок 22 – res, drawable, иконки, используемые в мобильном приложении

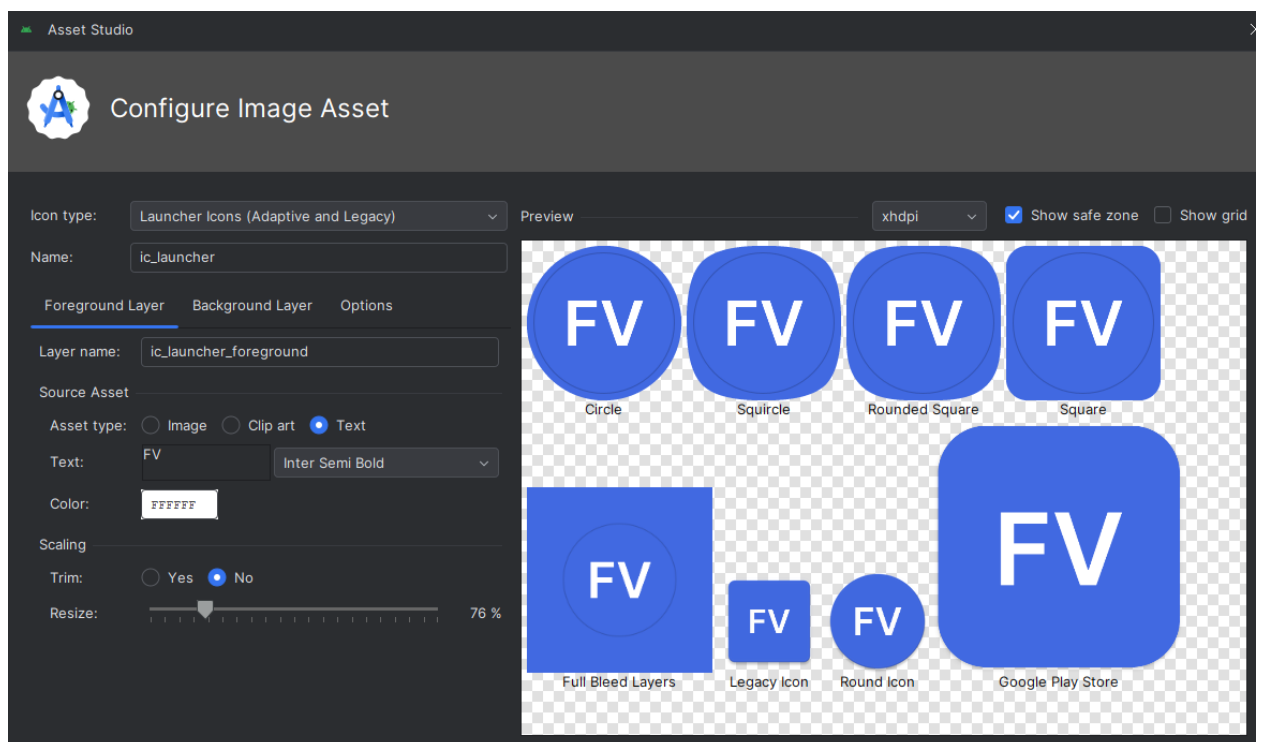


Рисунок 23 – Результат добавления логотипа

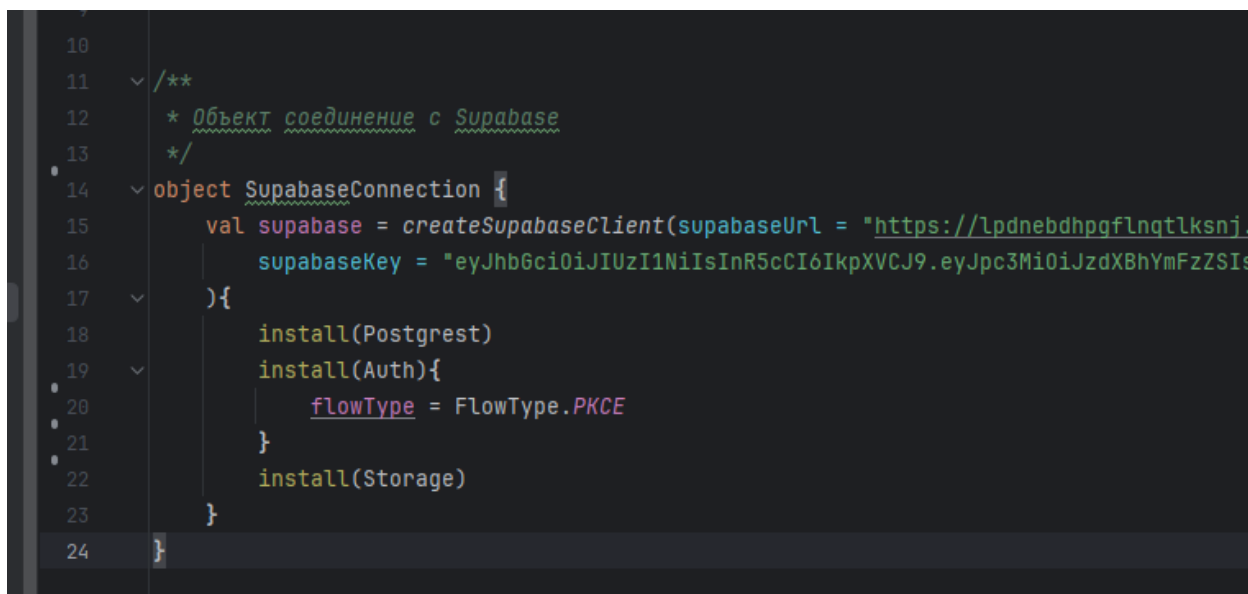
2.5.3 Подключение к серверной части

В качестве серверной части мобильного приложения “FlyV” используется специализированный комплекс Supabase. Рассмотрим, что было необходимо произвести для того, чтобы в проекте была предоставлена возможность взаимодействия с серверной частью:

1. Настройка соединения с сервером;
2. Создание моделей сущностей, для работы с данными в ходе проекта.

Для того, чтобы избежать избыточности иллюстрирующего материала, будут рассмотрены две модели сущностей базы данных. Все модели сущностей базы данных проекта, можно рассмотреть в репозитории, в следующем каталоге проекта: `project/.../Models/DataBase/Entities`.

Рассмотрим результат осуществлённой деятельности (рис 24-26).



```
10
11  /**
12   * Объект соединение с Supabase
13   */
14  object SupabaseConnection {
15      val supabase = createSupabaseClient(supabaseUrl = "https://lpdnebdhpgflnqtlksnj.
16          supabaseKey = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIs
17      ){
18          install(Postgrest)
19          install(Auth){
20              flowType = FlowType.PKCE
21          }
22          install(Storage)
23      }
24  }
```

Рисунок 24 – Объект, предназначенный для осуществления соединения с сервером

```

2
3 import kotlinx.serialization.SerialName //Помогает правильно указать название, потому что в базе snake_case, а Kotlin camelCase
4 import kotlinx.serialization.Serializable
5
6
7 @Serializable
8 data class Employee(
9     val id : String,
10     @SerialName("full_name") val fullName : String,
11     @SerialName("city_id") val cityId : String,
12     @SerialName("hire_date") val hireDate : String,
13     @SerialName("days_off") val daysOff : Int,
14     @SerialName("days_vacations") val daysVacation : Int,
15     @SerialName("number_phone") val numberPhone : String,
16     @SerialName("is_works") val isWorks : Boolean,
17     @SerialName("email_personal") val email : String?,
18     @SerialName("url_photo_profile") val urlPhotoProfile : String?
19 )
20

```

Рисунок 25 – Employee, модель сущности “Сотрудники”

```

6
7
8 @Serializable
9 data class AbsenceEmployee(
10     val id : String,
11     @SerialName("reason_id") val reasonId : String,
12     @SerialName("employee_id") val employeeId : String,
13     @SerialName("begin_date") val beginDate : String,
14     @SerialName("amount_day") val amountDay : Int
15 )
16

```

Рисунок 26 – AbsenceEmployee, модель сущности “Отсутствия и сотрудники”

2.5.4 Работа с серверной частью

Исходя из необходимости производить различные операции с данными, что хранятся на сервере, было необходимо разработать специализированный функционал, что предоставит возможность произвести данную задачу. Для реализации поставленного задания, было необходимо произвести создание запросов к серверу с клиента. По причине необходимости реализовать данную задачу, были разработаны специализированные классы, что содержат отдельные виды запросов к серверу. Рассмотрим данную структуру (рис 27):

1. Запросы, связанные с аутентификацией пользователя в рамках разрабатываемой системы (Класс: Auth);
2. Запросы, предназначенные для получения данных с сервера (Класс: Get);
3. Запросы, предназначенные для обновления данных на сервере (Класс: Update);
4. Запросы, предназначенные для добавления данных на сервер (Класс: Insert);
5. Запросы, что предназначены для удаления данных с сервера (Класс: Delete).

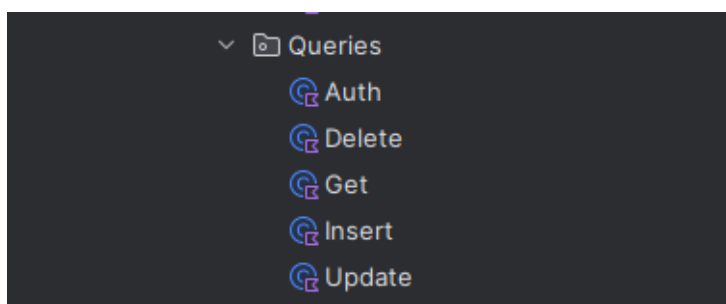


Рисунок 27 – Структура созданных запросов

Запросы были разделены на различные классы, в целях создания структуры, предоставляющей возможность пользователю, не ознакомленному с проектом, быстро разобраться с содержимым.

Рассмотрим содержимое некоторых классов, что используются для осуществления запросов к серверу, посредством использования ранее созданного объекта соединения.

В классе Get, содержатся запросы, предоставляющие возможность получить данные с сервера, посредством эксплуатации метода select в соответствии с определёнными условиями (рис 28).

```
/**
 * Запрос на получение отпусков пользователя по его uuid и uuid причины
 */
suspend fun getAbsencesEmployeesByIdUserAndReasonId(idUser : String, idReason: String) : List<AbsenceEmployee>{
    var listAbsenceEmployee = listOf<AbsenceEmployee>()
    try {
        listAbsenceEmployee = db
            .from( table: "AbsencesEmployees")
            .select(){
                filter {
                    eq( column: "reason_id", idReason)
                    eq( column: "employee_id", idUser)
                }
            }
            .decodeList<AbsenceEmployee>()
        Log.d( tag: "GetAbsencesEmployeesByIdUserAndReasonId", msg: "SuccessFetch")
    }
    catch (e : Exception){
        Log.d( tag: "ExceptionGetAbsencesEmployeesByIdUserAndAbsencesId", msg: "$e")
    }
    return listAbsenceEmployee
}
```

Рисунок 28 – Пример запроса на получение данных об отпусках определённого пользователя по специализированной причине

Суть данного метода заключается в возвращении списка отпусков конкретного пользователя по определённой причине, посредством эксплуатации методов, предназначенных для реализации данного типа задачи.

В классе Update, содержатся запросы, предоставляющие возможность обновить данные на сервере, посредством эксплуатации метода update, в соответствии с определёнными условиями (рис 29).

```
80
81 |≡
82 /**
83  * Запрос на обновление количества отпускных дней пользователя
84  */
85 suspend fun updateDaysVacationsByUserId(idUser : String, daysVacations : Int){
86     try {
87         db.from( table: "Employees")
88             .update(
89                 {
90                     set("days_vacations", daysVacations)
91                 }
92             ){
93                 filter {
94                     eq( column: "id", idUser)
95                 }
96             }
97         Log.d( tag: "UpdateDaysVacationsByUserId", msg: "SuccessUpdate")
98     }
99     catch (e: Exception){
100         Log.d( tag: "ExceptionUpdateDaysVacationsByUserId", msg: "$e")
101     }
102 }
```

Рисунок 29 – Пример запроса на обновление количества отпускных дней определённого пользователя

Суть данного метода, заключается в обработке переданного экземпляра пользователя, посредством поиска данной записи по id в базе данных, и в последствии установки нового значения в поле “days_vacations”

В классе Auth, содержатся различные функции, что были разработаны и созданы в целях необходимости вести жизненный цикл сессии пользователя в системе (рис 30).

```
/**
 * Функция авторизации пользователя по введённым данным
 */
suspend fun authorization(insertEmail : String, insertPassword : String){
    db.auth.signInWith(Email){
        email = insertEmail
        password = insertPassword
    }
}
```

Рисунок 30 – Пример функции, для проведения аутентификации пользователя в системе

Суть данной функции заключается в необходимости произвести аутентификацию пользователя по его данным. В случае отрицательного результате аутентификации функция “выбросит” исключение, что в последствии в другом модуле обрабатывается специализированным образом. В случае успешного результата, пользователю будет выдан токен аутентификации.

Класс Insert, был разработан в целях необходимости производить добавление данных в сущность базы данных “Отсутствия и сотрудники”. Класс Delete был разработан в целях необходимости, производить удаление данных из сущности БД “Отсутствия и сотрудники”.

В качестве примеров, были рассмотрены не все запросы, в целях исключения избыточности материала. Все реализованные запросы смотреть в репозитории проекта по следующему каталогу: project/.../Models/DataBase/Queries.

2.5.5 Views и ViewModels

На данном этапе был произведён процесс, целью которого является разработка и создание окон мобильного программного продукта и бизнес-логики, что соответствует определённому окну.

Рассмотрим структуру созданных окон (View) и соответствующей им логики (ViewModel) (рис 31, 32).

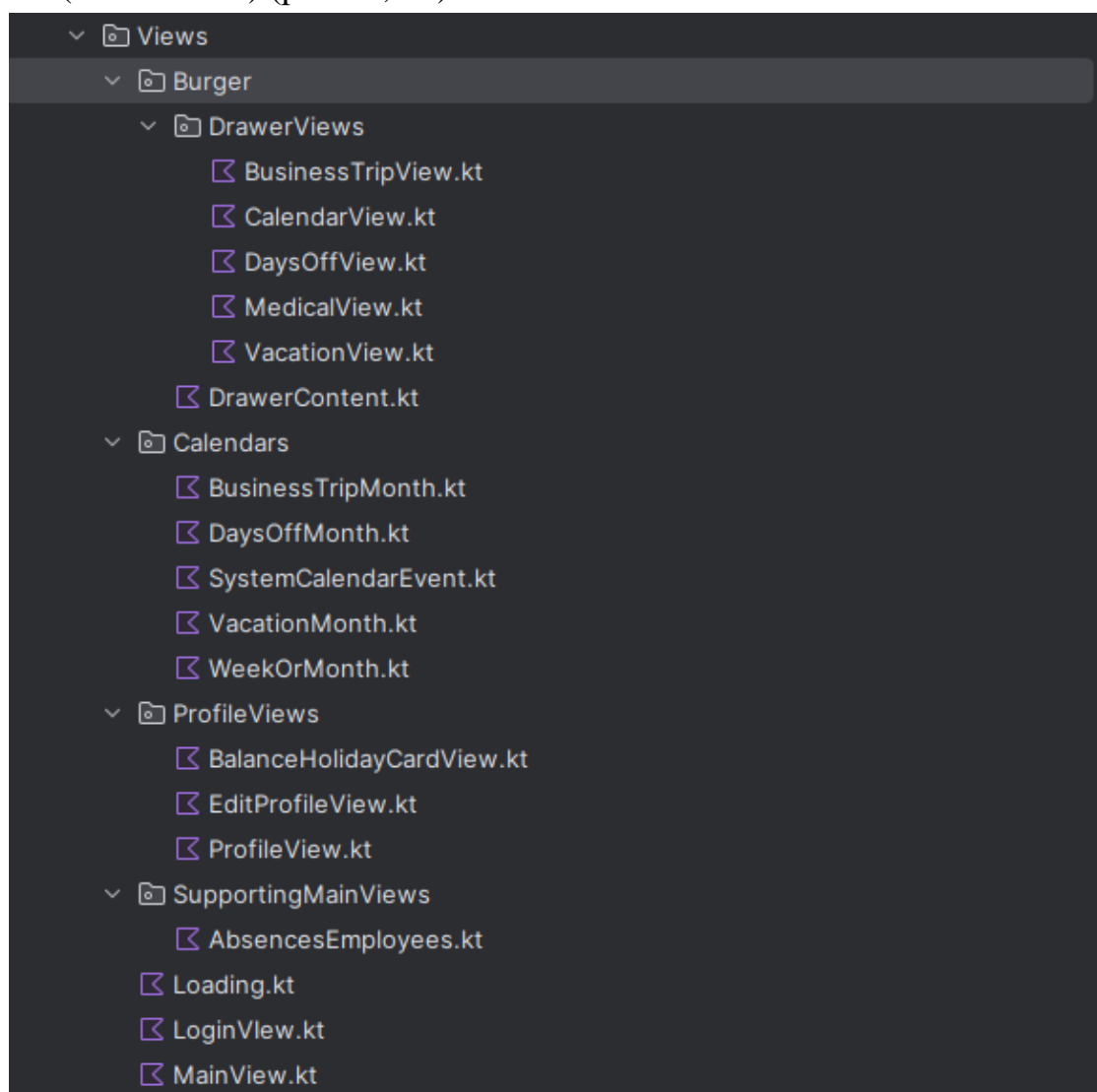


Рисунок 31 – Структура созданных окон

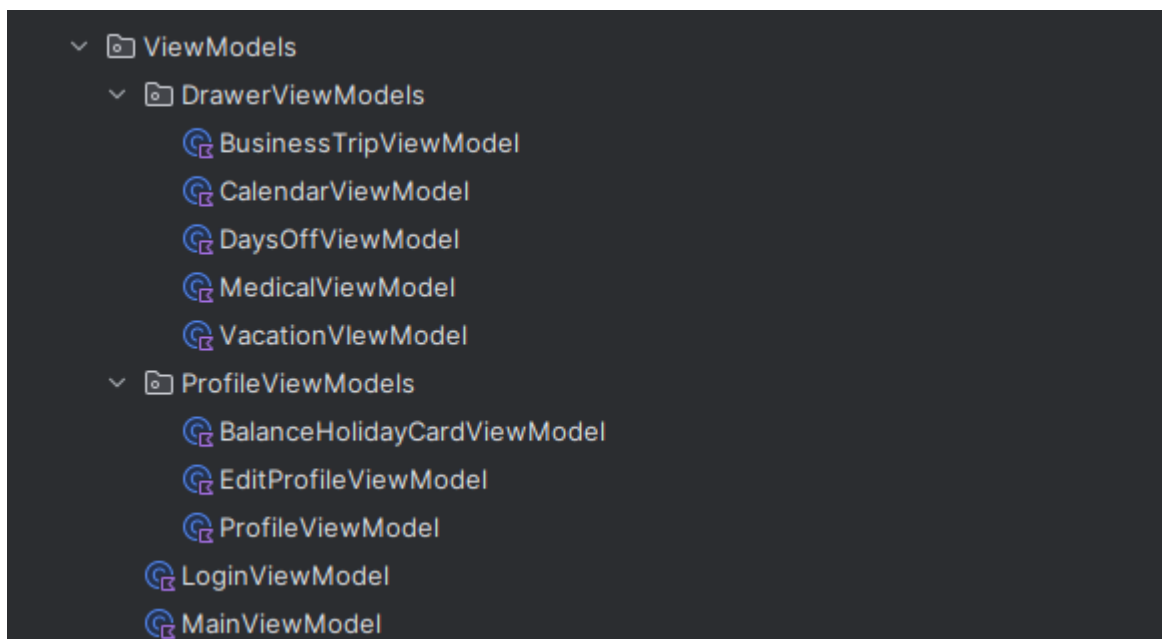


Рисунок 32 – Структура созданной логики для окон

В целях необходимости избежать избыточности материала, полноценно, с использованием иллюстрирующего материала будут рассмотрены два окна, что были предназначены для осуществления главного функционала мобильного программного продукта. При необходимости анализа программного кода всех созданных окон, смотреть содержимое проекта в репозитории.

Следующим шагом, в целях сохранения структуры, окна (View) и соответствующая им логика (ViewModel) будут рассмотрены совместно. Для каждого окна были добавлены обработчики в виде проверки на текущее интернет-соединение.

Главное окно (рис 33, 34):

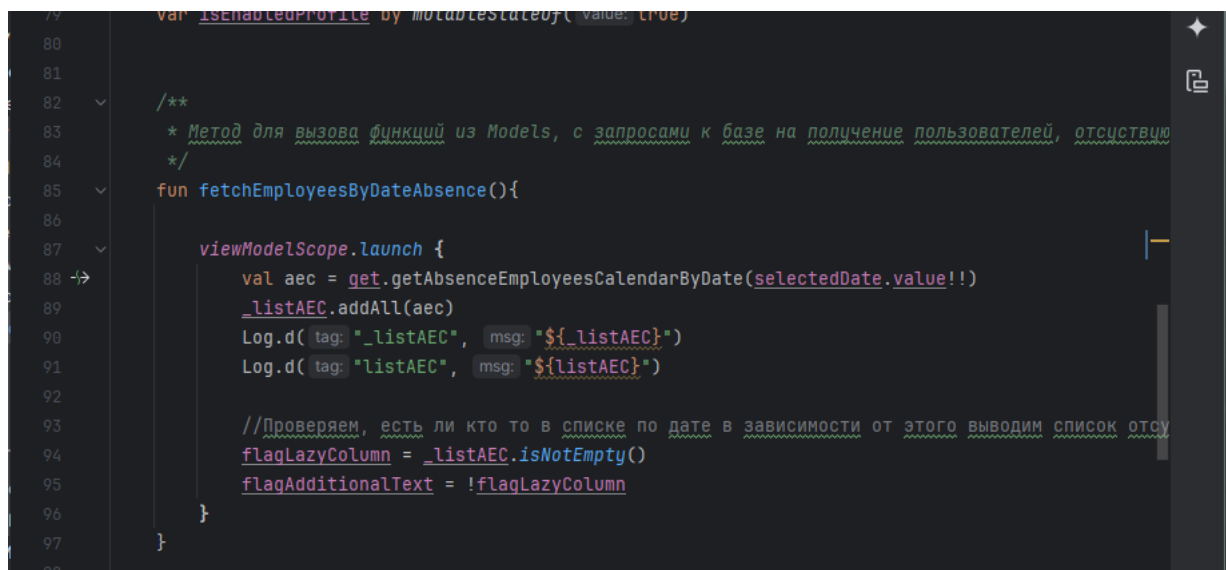
1. MainView.kt – данное окно предназначено для отображения основного меню, с которым пользователь взаимодействует каждый раз, при входе в программный продукт. На данном окне расположены профиль, текущий месяц, неделя, календарь, бургер-меню, сотрудники, что могут отсутствовать в определённый выбранный день, отпуск пользователя, что может быть через 3-1 день или конец отпуска, в случаях, если пользователь, авторизованный в системе, уже находится в отпуске. Также присутствует

возможность открытия бургер-меню. Аналогично, из данного окна, можно перейти во все окна, что содержатся в бургер-меню и окно профиля;

2. AbsencesEmployees.kt – данное окно, является вспомогательным и предоставляет возможность вывести в список отсутствующих пользователей в специализированном формате, в выбранный день;

3. WeekOrMonth.kt – данное окно, является календарём на неделю в контексте данного окна, и представляет собой список в строчку в виде кнопок с днями текущей недели;

4. MainViewModel – класс, что предназначен для реализации бизнес-логики главного окна. В данном окне, производится расчёт текущей недели, дней на неделю. Аналогично в данном классе производится определение, находится ли пользователь текущей сессии в отпуске, или отпуск находится в диапазоне от 3 до 1 дней. Данный класс предназначен для обработки данных, вытягиваемых из базы, для вывода пользователей, что отсутствуют в выбранный день.



```
79 var isEnabledProfile by mutableStateOf(value: true)
80
81
82 /**
83  * Метод для вызова функций из Models, с запросами к базе на получение пользователей, отсутствующих
84  */
85 fun fetchEmployeesByDateAbsence(){
86
87     viewModelScope.launch {
88         val aec = getAbsenceEmployeesCalendarByDate(selectedDate.value!!)
89         _listAEC.addAll(aec)
90         Log.d( tag: "_listAEC", msg: "${_listAEC}")
91         Log.d( tag: "listAEC", msg: "${listAEC}")
92
93         //Проверяем, есть ли кто то в списке по дате в зависимости от этого выводим список отсутствующих
94         flagLazyColumn = _listAEC.isNotEmpty()
95         flagAdditionalText = !flagLazyColumn
96     }
97 }
98
```

Рисунок 33 – Метод для получения отсутствующих пользователей, в классе MainViewModel

```

144
145 //Текущая неделя, месяц, дни
146 Column(
147     verticalArrangement = Arrangement.spacedBy(10.dp)
148 ) {
149     Text(text = "Текущая неделя",
150         fontFamily = interFontFamily,
151         fontSize = 32.sp,
152         color = ColorTextDark,
153         fontWeight = FontWeight.Medium,
154         textAlign = TextAlign.Left
155     )
156     Text(text = "${viewModel.month}, ${viewModel.dayBeginWeek}-${viewModel.dayEndWeek}",
157         fontFamily = interFontFamily,
158         fontSize = 20.sp,
159         color = ColorTextLight,
160         fontWeight = FontWeight.Medium,
161         textAlign = TextAlign.Left,
162         modifier = Modifier.padding(start = 2.dp)
163     )
164 }

```

Рисунок 34 – Код для отображения текущей недели и месяца в MainView.kt

Окно оформления отпусков (рис 35-39):

1. VacationView.kt – данное окно предназначено для оформления отпусков пользователем, на данном окне можно рассмотреть календарь на месяц, и переключить данные месяц на следующий в целях оформления отпуска, с возможностью оформления отпуска посредством выбора дней и нажатия на кнопку. Аналогично можно рассмотреть баланс отдыха, в связи с необходимостью удобно и быстро получить информацию о том, сколько у пользователя на данный момент есть отпускных дней;
2. BalanceHolidayCardView.kt – данное окно предназначено для отображения количества доступных отпускных дней, отпускных дней за стаж, отгулов, а также сколько на данный момент запланировано отпускных дней;
3. BalanceHolidayCardViewModel – класс, осуществляющий все необходимые расчёты для отображения данных на окне BalanceHolidayCardView.kt;
4. VacationMonth.kt – данное окно, является календарём на месяц в контексте данного окна, и представляет собой список в строчку в виде кнопок с днями текущей недели, что предоставляет возможность выбрать диапазон из двух дней, где первый начало отпуска, а второй конец. Аналогично данный календарь отображает дни, когда отпуск был запланирован;

5. VacationViewModel – класс, что предназначен для осуществления бизнес-логики окна VacationView.kt. В данном классе, присутствуют функции определения текущего месяца, года, и обработчик, что предоставляет возможность определить и учесть все возможные варианты, когда пользователь не имеет возможности оформить отпуск (не хватает баланса дней, планируется не за неделю, планируется во время замены другого сотрудника на рабочем месте, планируется во время назначенной командировки пользователю, планируется во время уже запланированного отпуска);

6. SystemCalendarEvent.kt – функция, что предназначена для открытия системного календаря на телефоне и добавления события отпуск, на выбранный период дат.

```
87         color = ColorTextLight
88     )
89     LazyRow(
90         modifier = Modifier.fillMaxWidth(),
91         horizontalArrangement = Arrangement.spacedBy(28.dp)
92     ) {
93         items(count = amountDaysInCalendar){ index ->
94
95             val currentDate = beginDate.plusDays(index.toLong())
96             val dayMonth = currentDate.dayOfMonth
97             val dayOfWeek = currentDate
98                 .dayOfWeek
99                 .getDisplayName(
100                     TextStyle.SHORT,
101                     Locale(language: "ru")
102                 ).replaceFirstChar { it.uppercase() }
103
104             //Эта прекрасная переменная для того, чтобы если выбранная дата совпадала с currentDate,
105             // то отрисовывалась окантовка синяя, в общем индексы шляпа
106             // (а эта переменная каждый раз отработывает, когда я нажимаю на кнопку, ведь экран перерисовывается)
107             val isSelected = when (currentDate) {
108                 firstSelectedDayInCalendar.value -> true
109                 secondSelectedDayInCalendar.value -> true
110                 else -> false
111             }
112
113
114             var isPlanned = false
115             for (it in firstAndLastDaysVacation){
116                 // Проверка, запланирована ли дата
117                 if(currentDate in it[0] .. it[1]){
118                     isPlanned = true
119                     break
120                 }
121             }
122         }
```

Рисунок 35 – Код календаря, где осуществляется обработка действий пользователя VacationMonth.kt


```

213     }
214     Spacer(modifier = Modifier.height(40.dp))
215     Column(
216         verticalArrangement = Arrangement.spacedBy(20.dp)
217     ) {
218         //Календарь
219         CalendarForVacation(
220             viewModel.beginDayMonth,
221             viewModel.endDayMonth,
222             viewModel.listFirstAndLastDaysVacation,
223             viewModel.resetState,
224             datesSelectedCallback = { firstDayVacation: LocalDate?, lastDayVacation: LocalDate? ->
225                 viewModel.isEnabledPlannedFirst = !(firstDayVacation == null || lastDayVacation == null)
226                 viewModel.firstSelectedDate.value = firstDayVacation
227                 viewModel.lastSelectedDate.value = lastDayVacation
228                 Log.d(tag: "RangeDates", msg: "${viewModel.firstSelectedDate.value} - ${viewModel.lastSelectedDate.value}")
229                 if(viewModel.isEnabledPlannedFirst){
230                     viewModel.amountDaysPlanned = ChronoUnit.DAYS.between(viewModel.firstSelectedDate.value, viewModel.lastSelectedDate.value).toInt() + 1//учитываем
231                     //Вызываем проверку все возможных условий
232                     viewModel.checkPossibilityVacation()
233                 }
234             }
235         )
236     }
237 }
238
239

```

Рисунок 36 – Код вызова календаря в окне оформления отпусков

VacationView.kt

```

176     }
177
178     //Получаем id пользователя которого должны заменять
179     val idUserSubstitution =
180         if(substitution!!.employeeFirstId == ProfileCache.profile.userInfo!!.id) substitution!!.employeeSecondId
181         else substitution!!.employeeFirstId
182     Log.d(tag: "idUserSubstitution", idUserSubstitution)
183
184     val listPlanned : List<AbsenceEmployee> = get.getAbsencesEmployeesByUserId(idUserSubstitution)
185     var isRangeNotTouch : Boolean = true
186
187     for(it in listPlanned){
188         //Проверяем (истина), что отпуск кончается раньше начала даты отсутствия или начинается после даты окончания отсутствия подменяемого сотрудника
189         if(!(lastSelectedDate.value!! < convertStringToLocalDate(it.beginDate) ||
190             firstSelectedDate.value!! > convertStringToLocalDate(it.beginDate).plusDays( daysToAdd: it.amountDay.toLong()-1))){
191             isRangeNotTouch = false
192             break
193         }
194     }
195
196     //Проверяем подошёл ли выбранный диапазон
197     if(isRangeNotTouch){
198         var isRangeVacationsNotTouch : Boolean = true
199         for(it in listFirstAndLastDaysVacation){
200             //Проверяем, что последний день отпуска выбранного заканчивается ранее первого дня отпуска, что был уже запланирован,
201             // и первый день выбранного отпуска больше дня конца запланированного отпуска
202             if(!(lastSelectedDate.value!! < it[0] ||
203                 firstSelectedDate.value!! > it[1])){
204                 isRangeVacationsNotTouch = false
205                 break
206             }
207         }
208     }
209

```

Рисунок 37 – Код обработки возможных сценариев оформления отпуска в

классе VacationViewModel

```

Column(
    verticalArrangement = Arrangement.spacedBy(13.dp)
) {
    //Ежегодный отпуск
    Column(
        verticalArrangement = Arrangement.spacedBy(6.dp)
    ) {
        Text(
            text = "ЕЖЕГОДНЫЙ ОТПУСК",
            fontSize = 10.sp,
            fontFamily = interFontFamily,
            color = ColorTextDark,
            fontWeight = FontWeight.Bold
        )
        Text(
            text = buildAnnotatedString {
                withStyle(style = SpanStyle(color = ColorTextLight)) {
                    append("Можешь запланировать: ")
                }
                withStyle(style = SpanStyle(color = ColorTextDark, fontWeight = FontWeight.Bold)) {
                    append(
                        if(viewModel.daysVacation == 0) { "- " }
                        else { "${viewModel.daysVacation} дн." }
                    )
                }
            },
            fontSize = 15.sp,
            fontFamily = interFontFamily
        )
    }
}

```

Рисунок 38 – Код, отображающий количество отпускных дней, на окне
BalanceHolidayCardView.kt

```

10 ~ /**
11  * Функция для открытия календаря на телефоне, для того, чтобы создать событие об отпуске
12  */
13 ~ fun openCalendarEvent(context: Context, firstDateEvent: Calendar, lastDateEvent: Calendar) {
14
15     //Устанавливаем время 12 часов ночи для первого дня события
16     firstDateEvent.set(Calendar.HOUR_OF_DAY, 0)
17     firstDateEvent.set(Calendar.MINUTE, 0)
18     firstDateEvent.set(Calendar.SECOND, 0)
19     firstDateEvent.set(Calendar.MILLISECOND, 0)
20
21     //Устанавливаем время без одной минуты 12 часов для последнего дня события
22     lastDateEvent.set(Calendar.HOUR_OF_DAY, 23)
23     lastDateEvent.set(Calendar.MINUTE, 59)
24     lastDateEvent.set(Calendar.SECOND, 59)
25     lastDateEvent.set(Calendar.MILLISECOND, 0)
26
27     val intent = Intent(Intent.ACTION_INSERT).apply {
28         type = "vnd.android.cursor.item/event"
29         putExtra(CalendarContract.Events.TITLE, value: "Отпуск") // Название события
30
31         putExtra(CalendarContract.EXTRA_EVENT_BEGIN_TIME, firstDateEvent.timeInMillis) // Время начала отпуска
32         putExtra(CalendarContract.EXTRA_EVENT_END_TIME, lastDateEvent.timeInMillis) // Время окончания отпуска
33
34         putExtra(CalendarContract.Events.DESRIPTION, value: "Описание события: отпуск") // Описание
35     }
36
37     try {
38         context.startActivity(intent)
39     }
40     catch (e: Exception) {
41         Toast.makeText(context, text: "Не удалось открыть календарь", Toast.LENGTH_SHORT).show()
42         Log.d(tag: "ExceptionOpenCalendarEvent", msg: "$e")
43     }
44 }

```

Рисунок 39 – Код, функции оформления события в календаре

Окно авторизации:

1. `LoginView.kt` – окно, где присутствует два поля, что предназначены для ввода логина и пароля пользователя. На окне присутствует логотип и кнопка, что предоставляет возможность начать процесс аутентификации данных, введенных пользователем. На кнопке, присутствует специализированный обработчик, что блокирует кнопку, если количество нажатий в несколько секунд превысило допустимое количество;
2. `LoginViewModel` – класс, предназначенный для осуществления бизнес-логики окна авторизации. В данном окне, производится валидация данных пользователя посредством использования определенных запросов из ранее рассмотренных классов, с запросами;
3. `Loading.kt` – вспомогательное окно, что предназначено для реализации окна загрузки, в момент авторизации пользователя.

Окно профиля:

1. `ProfileView.kt` – окно, где отображаются пользовательские данные, присутствует возможность рассмотреть баланс отдыха, перейти в окно редактирования, выйти из аккаунта;
2. `ProfileViewModel` – класс, где осуществляется специализированная бизнес-логика, предоставляющая возможность получать актуальные данные о пользователе;

Окно редактирования профиля:

1. `EditProfileView.kt` – окно, где отображаются поля для ввода новых данных о пользователе (можно изменить: город, почту, номер телефона). Аналогично присутствует кнопка, что позволяет возможность осуществить процесс обновления данных;
2. `EditProfileViewModel` – класс, где производится бизнес-логика окна редактирования профиля. В данном классе присутствует обработчик, что различает все комбинации обновления данных. Аналогично, в результате

обновления данных производится обновления данных, хранящихся во время выполнения программы.

Календарь:

1. `CalendarView.kt` – окно, где предоставлена возможность, использовать календарь и рассмотреть, кто в какой день отсутствует по определённой причине. Присутствует возможность переключения месяцев.
2. `WeekOrMonth.kt` – данное окно, является календарём на месяц в контексте данного окна, и представляет собой список в строчку в виде кнопок с днями текущей недели;
3. `CalendarViewModel` – класс, что содержит логику работы и обработчики для окна `CalendarView.kt`.

Отгулы:

1. `DaysOffView.kt` – окно, где предоставлена возможность оформить отгул пользователю. На окне присутствует специализированный календарь для оформления отгулов. Аналогично можно переключать месяцы;
2. `DaysOffMonth.kt` – календарь для оформления отгулов. Данный календарь предоставляет возможность выбрать один день для отгула. Аналогично на данном календаре отображаются оформленные отгулы;
3. `DaysOffViewModel` – класс, где содержится логика работы окна `DaysOffView.kt`. Аналогично присутствует обработчик на выбор дня для отгула (нельзя оформить отгул, если: не хватает баланса, отгул оформляется не за день и более).

Больничный:

1. `MedicalView.kt` – окно, что предназначено для открытия или закрытия больничного. На данном окне расположена информация о количества дней больничного и информация о статусе больничного на данный момент. Аналогично присутствует кнопка, посредством которой осуществляется открытие или закрытие больничного.

2. `MedicalViewModel` – класс, где разработана логика работы окна больничного.

Командировка:

1. `BusinessTripView.kt` – окно, что предназначено для отображения на календаре командировки, в случае её наличия. Аналогично пользователю будет выведено специализированное сообщение на окне, что командировка идёт в такой период дней и если командировки нет, пользователю будет выведено, что в ближайшее время нет командировок;

2. `BusinessTripViewModel` – класс, где разработана и создана специализированная логика, работы окна командировки.

Бургер-меню:

1. `DrawerContent.kt` – окно, что является бургер-меню и предоставляет возможность осуществлять переходы на окна `CalendarView.kt`, `VacationView.kt`, `DaysOffView.kt`, `MedicalView.kt`, `BusinessTripView.kt`.

2.5.6 Дополнительный необходимый функционал

На данном этапе будут рассмотрен функционал, что был создан дополнительно, в целях необходимости реализовать основной. Рассматриваемые функции, содержатся по следующему пути в репозитории проекта: `project/.../Models`

Рассмотрим данный функционал:

1. `Navigation.kt` – файл, содержащий функцию, что предоставляет возможность осуществить навигацию между всеми окнами программного продукта. Аналогично в данной функции были добавлены анимации для осуществления переходов между окнами приложения;
2. `BetweenDate.kt` – файл, содержащий функцию, что предназначена для определения, находился ли дата в заданном диапазоне дат;
3. `Network.kt` – файл, содержащий функцию, что предназначена для определения текущего интернет-соединения;
4. `UserState.kt` – скрытый класс, что содержит состояния загрузки, успеха, ошибки;
5. `LocalDateConverter.kt` – файл, содержащий функцию, что предоставляет возможность преобразовать тип `String` в `LocalDate`;
6. `LocalDateToCalendar.kt` – файл, содержащий функцию, что предоставляет возможность преобразовать тип `LocalDate` в `Calendar`;
7. `ParsingFullNameEmployee.kt` – файл, содержащий функцию, что предоставляет возможность распарсить строку с ФИО, специализированным образом;
8. `SessionLifeCycleTracking` – класс, что предназначен для отслеживания Ж.Ц сессии пользователя. Предоставляет возможность восстановить сессию пользователя в случае, если пользователь закрыл приложение менее чем на 5 минут;
9. `SessionCache.kt` – файл, что содержит функции, предназначенные для сохранения и восстановления данных о пользовательской сессии из памяти телефона;

10. `ClearData.kt` – файл, содержащий функцию для очистки данных о пользователе, что применяется в случае, если пользователь выйдет из своего аккаунта;

11. `ProfileCache` – объект специализированного `data-class`, что фиксирует определённую информацию о пользователе в период существования сессии пользователя;

12. `AbsenceEmployeeCalendar` – `data-class`, что является моделью для отображения данных об отсутствующих пользователях на следующих окнах: `MainView.kt`, `CalendarView.kt`

2.6 Тестирование мобильного приложения

На данном этапе реализации программного продукта “FlyV”, был произведён процесс тестирования разработанного приложения посредством эксплуатации ui-testing и unit-testing. Аналогично был составлен тест-план, тестовые наборы и тест-кейсы.

2.6.1 Разработка и реализация тестов (Android Studio)

На данном этапе были разработаны и использованы пять ui-test и unit-test.

В качестве примера будет рассмотрен один unit-test и ui-test (рис 40, 41).

```
/**
 * Тест проверяющий корректность работы конвертера String в LocalDate
 */
@Test
fun convertStringToLocalDateIsSuccess(){
    val expected = LocalDate.of( year: 2024, month: 11, dayOfMonth: 29)

    val stringDate = "2024-11-29"
    val result = convertStringToLocalDate(stringDate)

    assertEquals(expected, result)
}
```

Рисунок 41 – Unit-test, предназначенный проверить работоспособность функции преобразования типа String в LocalDate

```
/**
 * Проверка, на отображение экрана редактирования
 */
@Test
fun editProfileViewIsDisplayed(){
    //Устанавливаем экран, что будет тестироваться
    composeTestRule.setContent {
        EditProfileScreen(rememberNavController())
    }

    composeTestRule.onNodeWithText( text: "Редактирование профиля")
    composeTestRule.onNodeWithText( text: "Номер")
    composeTestRule.onNodeWithText( text: "Email")
    composeTestRule.onNodeWithText( text: "Город")
    composeTestRule.onNodeWithText( text: "Редактировать")
}
```

Рисунок 42 – UI-test, предназначенный проверить, что все элементы окна EditProfileView.kt отображаются

Рассмотрим результат проведения полного тестирования, посредством использования ранее составленных тестов (рис 43,44).

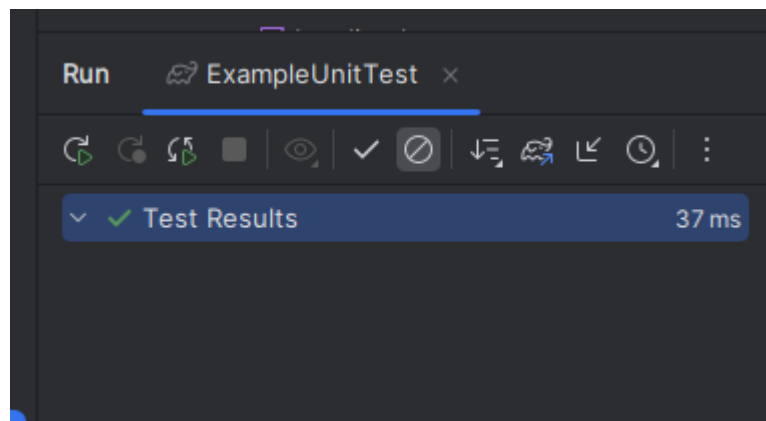


Рисунок 43 – Результат проведения Unit-testing

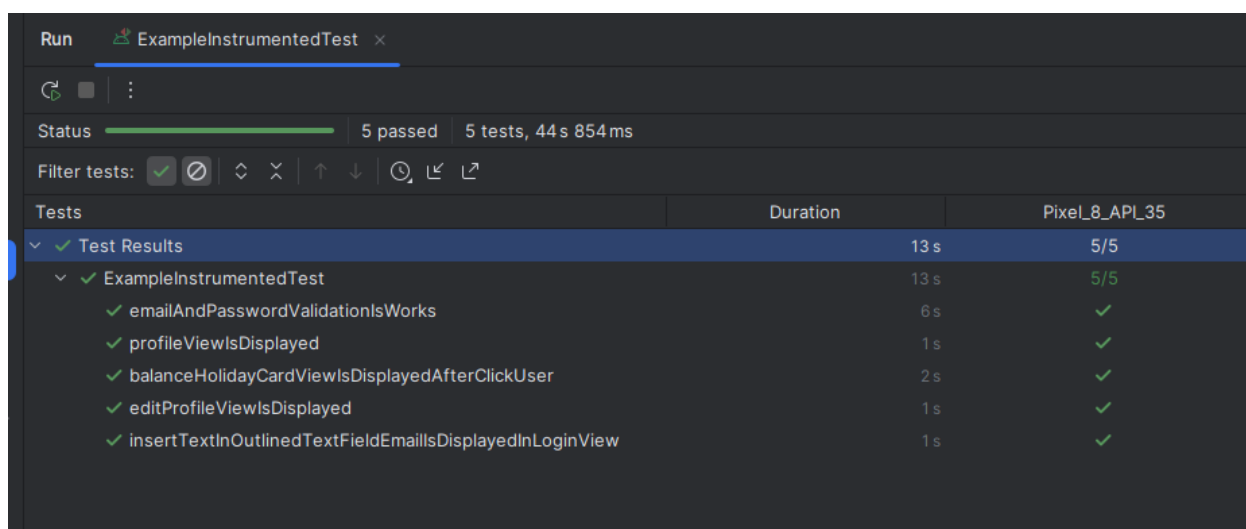


Рисунок 44 – Результат проведения UI-testing

2.6.2 Тест-план

На данном этапе было необходимо произвести процесс, целью которого является реализация процесса, что заключается в разработке специализированного тест-плана для разработанного продукта “FlyV”.

Рассмотрим перечень пунктов, что содержит тест-план:

1. Функционал мобильного программного продукта, что будет подвержен тестированию;
2. Критерии качества;
3. Перечень документации, что будет использоваться для тестирования серверной части мобильного программного продукта;
4. Стратегия тестирования серверной части приложения;
5. Ресурсы.

Функционал мобильного приложения.

Рассмотрим функционал мобильного приложения, что будет подвержен тестированию, по причине необходимости, выявить несоответствия работоспособности функционала мобильного программного продукта “FlyV”:

1. Метод, предназначенный для преобразования типа String в LocalDate (convertStringToLocalDate);
2. Метод, предназначенный для преобразования типа LocalDate в Calendar (localDateToCalendar);
3. Метод, предназначенный для определения, находится ли дата в заданном диапазоне дат (dateBetweenBeginDateAndEndDate);
4. Метод, предназначенный для обновления пользовательских данных, при переходе на окно профиля (refreshData);
5. Класс, где присутствует возможность отображать карточку баланса отдыха пользователя (isShowCardBalanceHoliday);
6. Отображение элементов на экране профиля пользователя системы (ProfileScreen);
7. Отображение элементов на экране редактирования профиля (EditProfileScreen);
8. Отображение карточки баланса отдыха, после действий пользователя, направленных на активацию карточки (BalanceHolidayCardScreen);
9. Изменение содержимого полей ввода на окне авторизации, после действий пользователя, направленных на ввод данных (LoginScreen);
10. Валидация вводимых данных пользователем, на окне авторизации (LoginScreen).

Критерии качества.

На данном этапе было необходимо составить список критериев, что будут эксплуатироваться для реализации процесса, что заключается в оценке качества мобильного программного продукта “FlyV”:

1. Функциональность:

- Весь ранее указанный функционал является работоспособным;

2. Надёжность:

- Функционал мобильного приложения должен производить задуманное в период осуществления работы, исключая сценарии возникновения различных дефектов;

- Должны присутствовать механизмы защиты, предоставляющие возможность обрабатывать различные исключительные ситуации.

3. Удобство сопровождения:

- Должна присутствовать возможность расширения и сопровождения функционала мобильного программного продукта.

4. Эффективность:

- Время выполнения поставленных задач, должно находиться в пределах нормы.

Перечень тестовой документации смотреть в приложение В.

Стратегия тестирования.

На данном этапе выполнения поставленной задачи, было необходимо произвести создание специализированной стратегии тестирования мобильного программного продукта “FlyV” (табл 14).

Таблица 14. Стратегия тестирования мобильного приложения

Тестирование	Процедуры тестирования
Unit-testing	<ol style="list-style-type: none">1. Проверить результат работы метода <code>convertStringToLocalDate</code>;2. Проверить результат работы метода <code>localDateToCalendar</code>;3. Проверить результат работы метода <code>dateBetweenBeginDateAndEndDate</code>;4. Проверить результат работы метода <code>refreshData</code>;5. Проверить, что карточка баланса отдыха изначально скрыта.
UI-testing	<ol style="list-style-type: none">1. Проверить, отображение элементов на экране профиля пользователя системы;2. Проверить отображение элементов на экране редактирования профиля пользователя системы;3. Проверить, отображение карточки баланса отдыха после действий пользователя, направленных на активацию карточки;4. Проверить, изменение содержимого полей ввода на окне авторизации, после действий пользователя направленных на ввод данных;5. Проверить, валидацию данных на окне авторизации.

Ресурсы.

Рассмотрим перечень ресурсов, что необходимы для успешной реализации процесса тестирования серверной части мобильного ПП.

Программные ресурсы:

2. Среда тестирования: Android Studio;
3. Разработанный функционал.

Человеческие ресурсы:

3. Тестировщик;
4. Разработчик.

Временные ресурсы:

2. Время на реализацию всех ранее описанных тестов программному продукту.

2.6.3 Тестирование

На данном этапе разработки документации по тестированию мобильного приложения был произведён процесс тестирования, ранее описанного функционала в тест-плане. Рассмотрим составленные тестовые наборы:

1. Тестовый набор: – “Unit-testing”;
2. Тестовый набор: – “UI-testing”.

Для того, чтобы исключить избыточность материала, в качестве примера будут рассмотрены один Unit-test, и один UI-test. Все тесты смотреть в Приложение Д.

Тестовый набор: – “Unit-testing”

Предварительные условия: для того, чтобы осуществить данный вид тест-кейсов, необходимо открыть Android Studio, с разработанным программным продуктом. В последствии необходимо открыть класс ExampleUnitTest.

Тест-кейс, преобразование типа данных String в LocalDate (табл 3).

Таблица 15. Тест-кейс 1

Шаги	Ожидаемый результат
Оставить доступным только тест <code>convertStringToLocalDateIsSuccess</code>	Все тесты, не относящиеся к указанному, будут закомментированы
Нажать ПКМ на класс <code>ExampleUnitTest</code>	Появится специализированное меню
В открывшемся меню нажать на пункт: Run ‘ExampleUnitTest’	Пользователь увидит успешный результат преобразования типа String в LocalDate

Тестовый-набор: – “UI-testing”

Предварительные условия для UI-test: для того, чтобы осуществить данный вид тест-кейсов, необходимо открыть Android Studio, с разработанным программным продуктом. В последствии необходимо открыть класс ExampleInstrumentedTest.

Тест-кейс, отобразить все элементы окна профиля пользователя (ProfileScreen)

Таблица 16. Тест-кейс 1

Шаги	Ожидаемый результат
Оставить доступным только тест profileViewIsDisplayed	Все тесты, не относящиеся к указанному, будут закомментированы
Нажать ПКМ на класс ExampleInstrumentedTest	Появится специализированное меню
В открывшемся меню нажать на пункт: Run ‘ExampleUnitTest’	Пользователь увидит, успешный результат того, все элементы окна пользователя (ProfileScreen) отображаются

2.7 Оценка эффективности использования “FlyV”

В результате завершения создания и проведения тестирования программного обеспечения, данный продукт был выпущен в релиз (рис 45, 46).

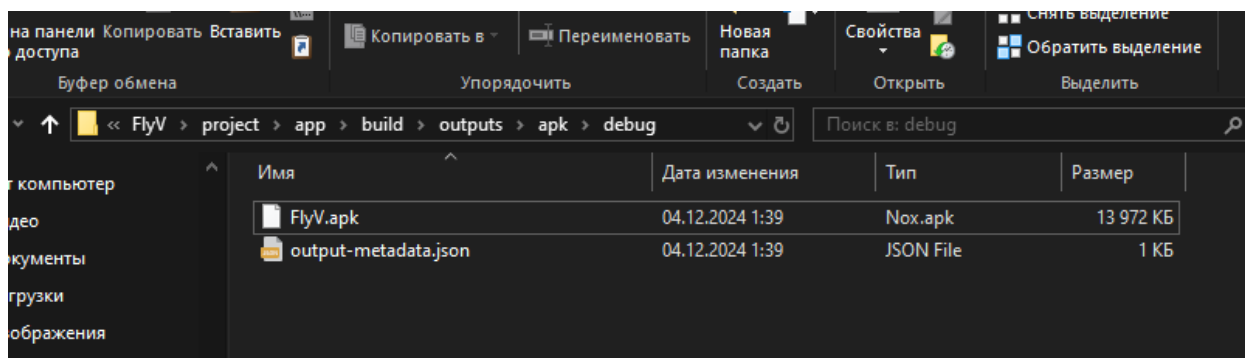


Рисунок 45 – Результат выпуска “FlyV” в релиз

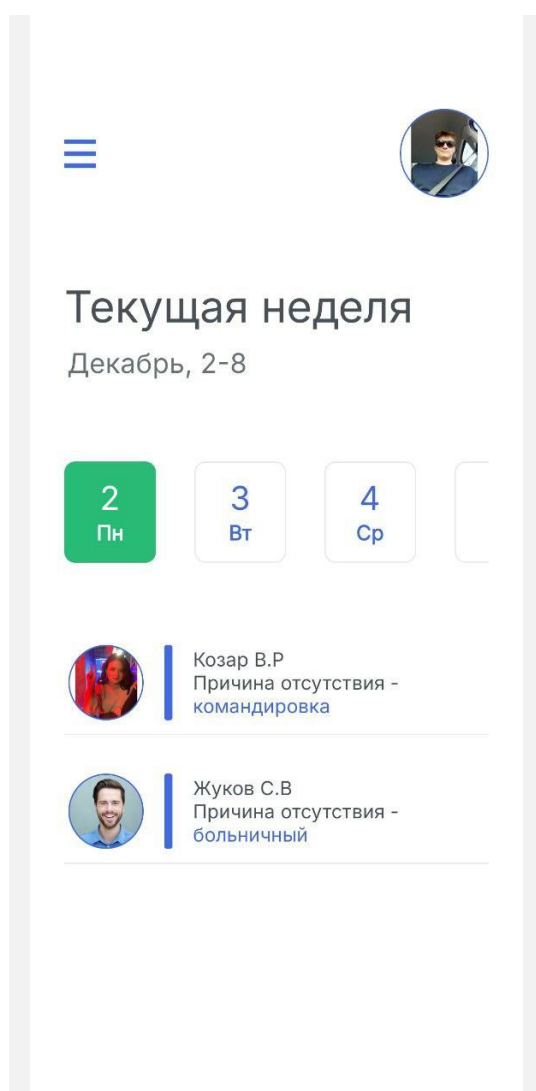


Рисунок 46 – Результат выпуска “FlyV” в релиз

Разработанный программный продукт “FlyV” значительно улучшает процессы управления отпускными периодами на предприятии, обеспечивая автоматизацию и удобство для сотрудников и руководства. Реализация системы позволила упростить оформление отпусков, отгулов и больничных, исключив необходимость использования бумажных заявок или отдельных согласований. Теперь сотрудники могут выбрать нужный период и отправить заявку напрямую через мобильное приложение, что значительно экономит время и снижает вероятность ошибок. Руководство предприятия, в свою очередь, получает доступ к актуальной информации о состоянии кадровых ресурсов, включая данные о сотрудниках, находящихся в отпуске, на больничном, в командировке или на отгуле.

Добавлены функции расчёта и отображения доступных отпускных дней, а также проверки условий предоставления отпусков, таких как наличие достаточного баланса или необходимость замещения.

Важным аспектом является внедрение системы авторизации и верификации, которая обеспечивает доступ только для уполномоченных пользователей, тем самым защищая данные от несанкционированного доступа. Программный продукт также предоставляет возможность удалённой работы, что особенно актуально в современных условиях. Благодаря функционалу отслеживания отпусков и отсутствий приложение способствует повышению прозрачности и управляемости кадровых ресурсов. В результате, “FlyV” не только сокращает затраты времени на обработку заявок, но и улучшает общую эффективность управления персоналом предприятия.

ЗАКЛЮЧЕНИЕ

В результате осуществлённой деятельности была разработана и создана система управления отпускными периодами (СУОП) под названием “FlyV”. Данная система предоставляет возможность эффективно и удобно производить процесс планирования отпуска, отгула, отслеживать наличие командировок, открывать и закрывать больничный. Аналогично, данная система предоставляет актуальную информацию о графиках отсутствия сотрудников, что значительно повышает прозрачность и эффективность управления кадровыми ресурсами.

В ходе разработки мобильного программного продукта, особое внимание было уделено тестированию и отладке разрабатываемого приложения. Были протестированы две важные части продукта: серверная часть и клиентская часть.

Разработанная система продемонстрировала свою эффективность, сокращая затраты времени на выполнение рутинных операций и минимизируя вероятность ошибок, связанных с человеческим фактором. Мобильный программный продукт “FlyV” повышает уровень автоматизации процессов управления отпусками, отгулами, больничными. Аналогично автоматизированы процессы отслеживания командировок и отсутствий кадрового ресурса. Разработанная система обеспечивает возможность удалённой и удобной работы с ранее указанными видами данных, что является актуальным решением в современных компаниях.

ПРИЛОЖЕНИЕ А

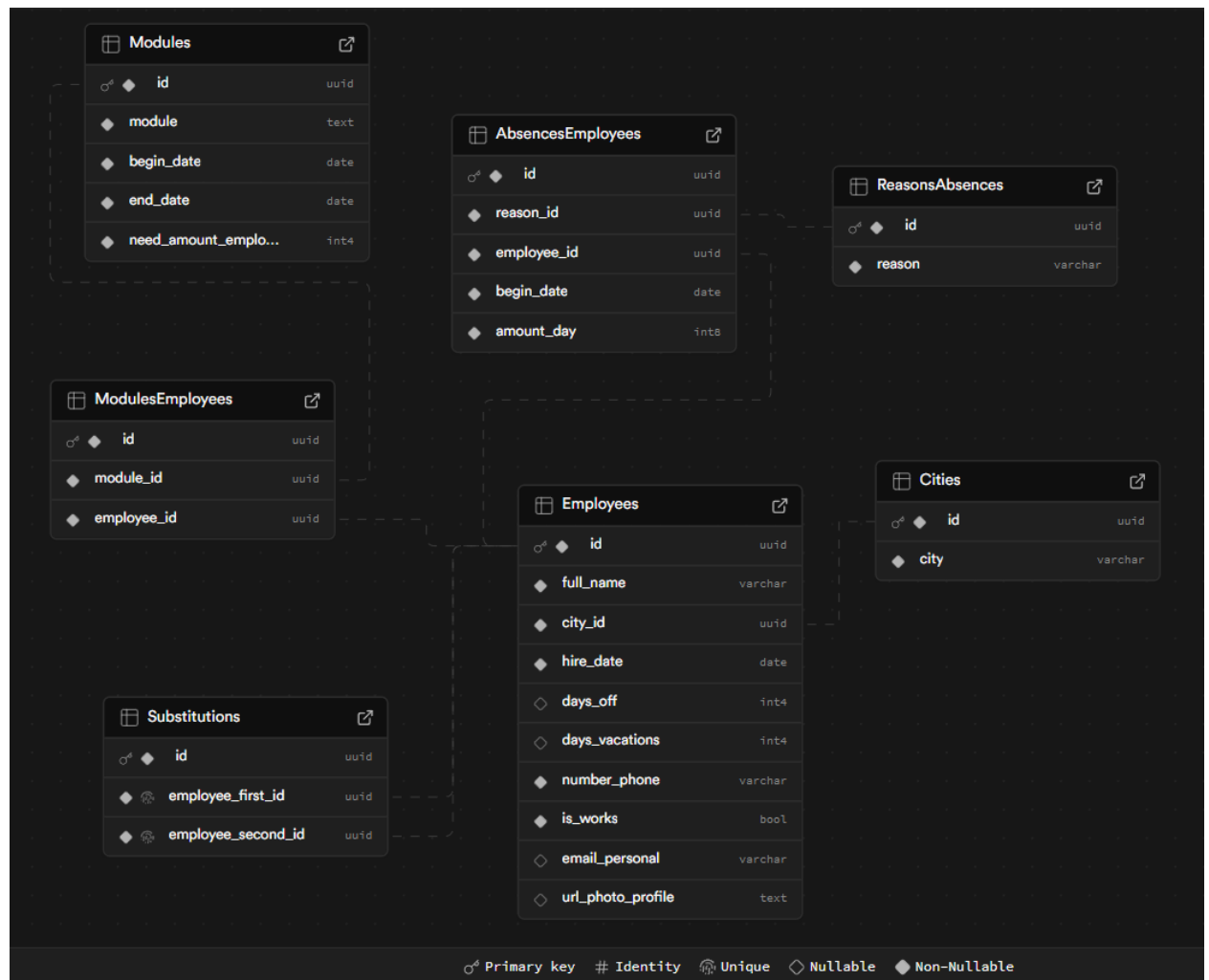


Рисунок 1 – Даталогическая модель базы данных “FlyV”

ПРИЛОЖЕНИЕ Б

Table Editor

frepsik's Org Free / FlyV / Enable branching

Filter Sort Insert

schema: public

New table

Search tables...

AbsencesEmployees

Cities

Employees

Modules

ModulesEmployees

ReasonsAbsences

Substitutions

	id uuid	city varchar	
	1a40aa9a-818d-4b8f-869f-9a73e4489635	Казань	
	ac645104-91ce-4d4d-87aa-9f5c9a9f2735	Москва	
	b55d9df7-ef65-4194-a298-69857efeaace	Калининград	
	c1e0242c-2d40-46d0-80e6-0af678756141	Нижний Новгород	
	f9b02111-c99f-47ca-8d12-845ab16eff17	Санкт-Петербург	

Рисунок 1 – Результат создания сущности “Города”

Table Editor

frepsik's Org Free / FlyV / Enable branching

Filter Sort Insert

schema: public

New table

Search tables...

AbsencesEmployees

Cities

Employees

Modules

ModulesEmployees

ReasonsAbsences

Substitutions

	id uuid	module text	begin_date date	end_date date	need_amount_employees int4
	00eb9bff-a96e-4c4d-b6b3-4052bd52a18:	Графика и визуальные эффекты	2024-10-01	2024-10-30	4
	55be76da-76e2-461e-aadd-7b683a126ac6	Игровой движок и физика	2024-11-01	2024-12-01	5
	79c92fd5-8f2f-4c37-950f-fd1b1afb552c	Звуки и музыка	2024-12-05	2024-12-30	2
	c4345560-1cdc-4519-8b76-20224735ef20	Анимация и кинематика	2024-11-01	2024-11-18	2

Рисунок 2 – Результат создания сущности “Модули”

Table Editor

frepsik's Org Free / FlyV / Enable branching

Filter Sort Insert

schema: public

New table

Search tables...

AbsencesEmployees

Cities

Employees

Modules

ModulesEmployees

ReasonsAbsences

Substitutions

	id uuid	module_id uuid	employee_id uuid	
	22ea8a95-3ecb-44a6-aaad-8ece61ff0c	00eb9bff-a96e-4c4d-b6b3-4052b...	b2cdea26-4794-477a-b3aa-d2059...	
	30ee0314-162d-4f93-a9a3-16ca8ad1b8b1	00eb9bff-a96e-4c4d-b6b3-4052b...	402cf10c-7c62-46f1-b6f4-187a277...	
	36d02ff3-4882-4b1d-a896-4200c654ae4i	00eb9bff-a96e-4c4d-b6b3-4052b...	efb54877-8421-40ff-8a49-79282c...	
	3f2bf3fe-2131-4bb8-845f-8a3ef87fe26e	00eb9bff-a96e-4c4d-b6b3-4052b...	369ef26a-8ca7-4bd7-b0b0-3758b...	

Рисунок 3 – Результат создания сущности “Модули и сотрудники”

Table Editor

frepsik's Org Free / FlyV / Enable branching

schema: public

+ New table

Search tables...

- AbsencesEmployees
- Cities
- Employees
- Modules
- ModulesEmployees
- ReasonsAbsences
- Substitutions

Filter Sort Insert

Auth policy

<input type="checkbox"/>	id uuid	employee_first_id uuid	employee_second_id uuid	+
<input type="checkbox"/>	2807be1b-e2a0-49eb-bec5-475cf45be53c	402cf10c-7c62-46f1-b6f4-187a277...	32438aeb-8014-420b-822f-ed9a1c...	
<input type="checkbox"/>	416b2fe1-5097-48a8-9c17-f6355d7db6e1	9bb7606d-6e02-4aet-a5da-7e95c...	b2cdea26-4794-477a-b3aa-d2059...	
<input type="checkbox"/>	6b840a93-1646-4833-8621-957064f3d33c	369ef26a-8ca7-4bd7-b0b0-3758b...	e0a6dc16-d4d3-45e2-9ff5-6ad35...	
<input type="checkbox"/>	b127e9be-8c29-46a9-a838-cfa3fd5effe8	547da52d-02fd-42e8-9da5-75f74e...	efb54877-8421-40ff-8a49-79282c...	

Рисунок 4 – Результат создания сущности “Замены”

Table Editor

frepsik's Org Free / FlyV / Enable branching

schema: public

+ New table

Search tables...

- AbsencesEmployees
- Cities
- Employees
- Modules
- ModulesEmployees
- ReasonsAbsences
- Substitutions

Filter Sort Insert

<input type="checkbox"/>	id uuid	reason varchar	+
<input type="checkbox"/>	336e77aa-2a3b-425a-85fe-76e5f76ab0ac	Отпуск	
<input type="checkbox"/>	a02fb81d-b9e2-4079-b1ee-ba0798959127	Больничный	
<input type="checkbox"/>	a7b00882-9224-4ac9-8639-d7b147bde60	Командировка	
<input type="checkbox"/>	b7e2097f-cdca-4883-b5b7-df186b812114	Отгул	

Рисунок 5 – Результат создания сущности “Причины отсутствия”

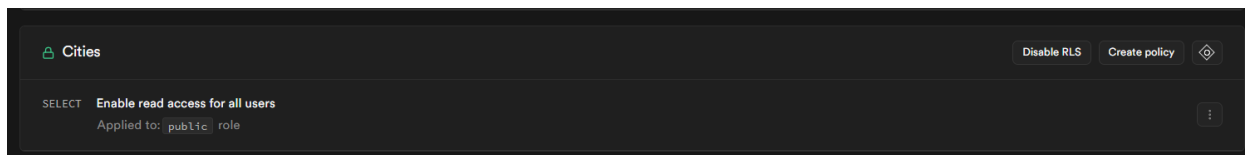


Рисунок 6 – Результат добавления политик для сущности “Города”

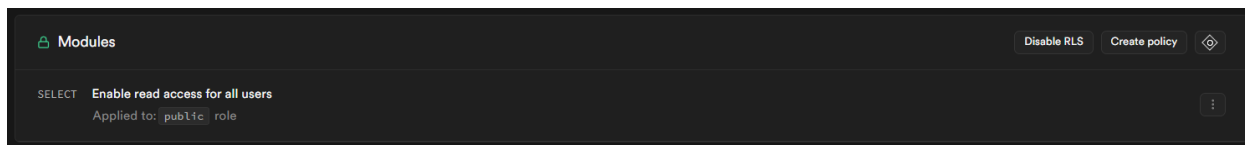


Рисунок 7 – Результат добавления политик для сущности “Модули”

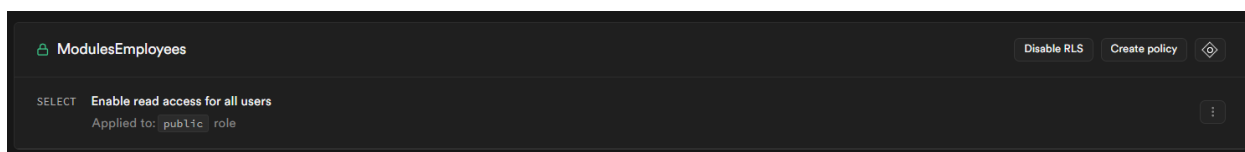


Рисунок 8 – Результат добавления политик для сущности
“Модули и сотрудники”

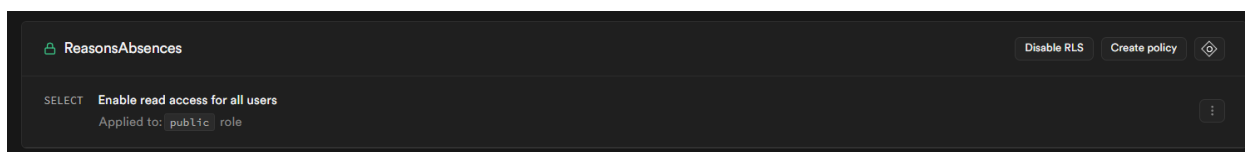


Рисунок 9 – Результат добавления политик для сущности
“Причины отсутствия”

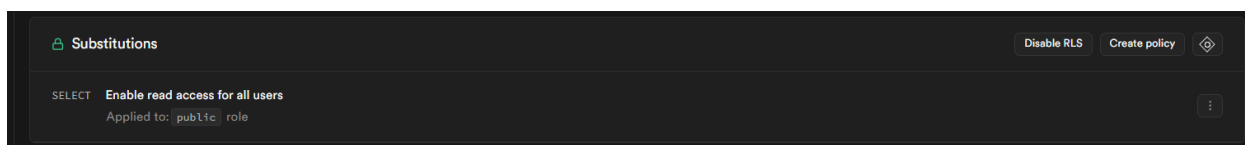


Рисунок 10 – Результат добавления политик для сущности “Замены”

ПРИЛОЖЕНИЕ В

Таблица 1. Перечень тестовой документации

Вид документации	Краткая сводка информация о документации
Тест-план	Документ, описывающий весь объем работ по тестированию
Критерии качества	Набор информации, определяющий требования к качеству
Тест-кейс	Создание конкретных сценариев тестирования, описывающих шаги и ожидаемый результат
Стратегия тестирования	Набор информации, предназначенный для определения основных направлений и методов тестирования

ПРИЛОЖЕНИЕ Г

Тестовый набор: – “Запросы на select”

1. Тест-кейс, вывод (select) информации из сущности “Cities” (табл 1).

Таблица 1. Тест-кейс – 1

Шаги	Ожидаемый результат
В пустую строку SQL Editor ввести следующую команду: select * from "Cities"	В выбранной строке будет введена команда: select * from "Cities"
Нажать комбинацию клавиш CTRL + Enter	Пользователю будут выведены все данные, что хранятся в сущности “Cities”

2. Тест-кейс, вывод (select) информации из сущности “Substitutions” (табл 2).

Таблица 2. Тест-кейс – 2

Шаги	Ожидаемый результат
В пустую строку SQL Editor ввести следующую команду: select * from "Substitutions"	В выбранной строке будет введена команда: select * from "Substitutions"
Нажать комбинацию клавиш CTRL + Enter	Пользователю будут выведены все данные, что хранятся в сущности “Substitutions”

3. Тест-кейс, вывод (select) информации из сущности “Modules” (табл 3).

Таблица 3. Тест-кейс – 3

Шаги	Ожидаемый результат
В пустую строку SQL Editor ввести следующую команду: select * from "Modules"	В выбранной строке будет введена команда: select * from "Modules"
Нажать комбинацию клавиш CTRL + Enter	Пользователю будут выведены все данные, что хранятся в сущности “Modules”

4. Тест-кейс, вывод (select) информации из сущности “Employees” (табл 4).

Таблица 4. Тест-кейс – 4

Шаги	Ожидаемый результат
В пустую строку SQL Editor ввести следующую команду: select * from "Employees"	В выбранной строке будет введена команда: select * from "Employees"
Нажать комбинацию клавиш CTRL + Enter	Пользователю будут выведены все данные, что хранятся в сущности “Employees”

5. Тест-кейс, вывод (select) информации из сущности “ReasonsAbsences” (табл 5).

Таблица 5. Тест-кейс – 5

Шаги	Ожидаемый результат
В пустую строку SQL Editor ввести следующую команду: select * from "ReasonsAbsences"	В выбранной строке будет введена команда: select * from "ReasonsAbsences"
Нажать комбинацию клавиш CTRL + Enter	Пользователю будут выведены все данные, что хранятся в сущности “ReasonsAbsences”

6. Тест-кейс, вывод (select) информации из сущности “ModulesEmployees” (табл 7).

Таблица 6. Тест-кейс – 7

Шаги	Ожидаемый результат
В пустую строку SQL Editor ввести следующую команду: select * from "ModulesEmployees"	В выбранной строке будет введена команда: select * from "ModulesEmployees"
Нажать комбинацию клавиш CTRL + Enter	Пользователю будут выведены все данные, что хранятся в сущности “ModulesEmployees”

Тестовый набор: – “Запросы на update”

1. Тест-кейс, обновление (update) информации в сущности “AbsencesEmployees” (табл 11).

Таблица 7. Тест-кейс – 2

Шаги	Ожидаемый результат
В пустую строку SQL Editor ввести следующую команду: update "AbsencesEmployees" set amount_day = 9 where employee_id = 'e0a6dc16-d4d3-45e2-9ff5-6ad35ed0421a'	В выбранной строке будет введена команда: update "AbsencesEmployees" set amount_day = 9 where employee_id = 'e0a6dc16-d4d3-45e2-9ff5-6ad35ed0421a'
Нажать комбинацию клавиш CTRL + Enter	Пользователь обновит информацию в сущности “AbsencesEmployees”

Тестовый набор: – “Запросы на использование процедур”

Предварительные условия: для того, чтобы осуществить тест-кейсы из данного тестового набора, необходимо открыть сервис Supabase, иметь доступ к базе данных “FlyV” и открыть SQL Editor.

1. Тест-кейс, запуск процедуры “UpdateDaysOffOnceAMonth” (табл 8).

Таблица 8. Тест-кейс – 1

Шаги	Ожидаемый результат
В пустую строку SQL Editor ввести следующую команду: call UpdateDaysOffOnceAMonth()	В выбранной строке будет введена команда: call UpdateDaysOffOnceAMonth()
Нажать комбинацию клавиш CTRL + Enter	Пользователь запустит реализацию процедуры на обновление баланса отгулов у пользователей системы

“UpdateDaysVacationsUserAfterOneYearWorks” (табл 9).

Таблица 9. Тест-кейс – 2

Шаги	Ожидаемый результат
В пустую строку SQL Editor ввести следующую команду: call UpdateDaysVacationsUserAfterOneYearWorks ()	В выбранной строке будет введена команда: call UpdateDaysVacationsUserAfterOneYearWorks ()
Нажать комбинацию клавиш CTRL + Enter	Пользователь запустит реализацию процедуры на обновление банка отпускных дней, для пользователей, что отработали год со дня устройства

Тестовый набор: – “Планировщики заданий”

1. Тест-кейс, проверка работоспособности планировщиков заданий:
“TaskUpdateDaysOffOnceAMonth”,
“TaskUpdateDaysVacationsUserAfterOneYearWorks” (табл 10).

Предварительные условия: для того, чтобы осуществить данный тест-кейс, необходимо открыть сервис Supabase, иметь доступ к базе данных “FlyV” и открыть Table Editor.

Таблица 10. Тест-кейс – 1

Шаги	Ожидаемый результат
В доступном окне, необходимо нажать на выпадающий список “schema”	Пользователю будет выведен список всех существующих схем и доступен для нажатия
В выпадающем списке выбрать “cron”	Пользователь откроет схему “cron”
Доступном окне, нажать на сущность “job_run_details”	Пользователю будет выведено содержимое сущности “job_run_details”, в которой фиксируются все запуски существующих планировщиков заданий

ПРИЛОЖЕНИЕ Д

Тестовый набор: – “Unit-testing”

1. Тест-кейс, преобразование типа данных LocalDate в Calendar (табл 1).

Таблица 1. Тест-кейс 2

Шаги	Ожидаемый результат
Оставить доступным только тест <code>convertLocalDateToCalendarIsSuccess</code>	Все тесты, не относящиеся к указанному, будут закомментированы
Нажать ПКМ на класс <code>ExampleUnitTest</code>	Появится специализированное меню
В открывшемся меню нажать на пункт: <code>Run 'ExampleUnitTest'</code>	Пользователь увидит успешный результат преобразования типа <code>LocalDate</code> в <code>Calendar</code>

2. Тест-кейс, определить, что дата, находится в диапазоне дат (табл 2).

Таблица 2. Тест-кейс 3

Шаги	Ожидаемый результат
Оставить доступным только тест <code>dateBetweenBeginDateAndEndDateIsSuccess</code>	Все тесты, не относящиеся к указанному, будут закомментированы
Нажать ПКМ на класс <code>ExampleUnitTest</code>	Появится специализированное меню
В открывшемся меню нажать на пункт: <code>Run 'ExampleUnitTest'</code>	Пользователь увидит, успешный результат того, что функция определит нахождение даты в диапазоне дат

3. Тест-кейс, обновить пользовательские данные (табл 3).

Таблица 3. Тест-кейс 4

Шаги	Ожидаемый результат
Оставить доступным только тест refreshProfileDataIsSuccess	Все тесты, не относящиеся к указанному, будут закомментированы
Нажать ПКМ на класс ExampleUnitTest	Появится специализированное меню
В открывшемся меню нажать на пункт: Run 'ExampleUnitTest'	Пользователь увидит, успешный результат того, что функция refreshData обновляет пользовательские данные

4. Тест-кейс, проверить, что изначально при открытии окна профиля пользователя карточка баланса отдыха скрыта (табл 4).

Таблица 4. Тест-кейс 5

Шаги	Ожидаемый результат
Оставить доступным только тест cardBalanceHolidayIsClosed	Все тесты, не относящиеся к указанному, будут закомментированы
Нажать ПКМ на класс ExampleUnitTest	Появится специализированное меню
В открывшемся меню нажать на пункт: Run 'ExampleUnitTest'	Пользователь увидит, успешный результат того, что карточка баланса отдыха пользователя системы исходно, является закрытой

Тестовый набор: – “UI-testing”

1. Тест-кейс, отобразить все элементы окна редактирования пользователя (EditProfileScreen) (табл 5)

Таблица 5. Тест-кейс 2

Шаги	Ожидаемый результат
Оставить доступным только тест editProfileViewIsDisplayed	Все тесты, не относящиеся к указанному, будут закомментированы
Нажать ПКМ на класс ExampleInstrumentedTest	Появится специализированное меню
В открывшемся меню нажать на пункт: Run ‘ExampleUnitTest’	Пользователь увидит, успешный результат того, все элементы окна редактирования пользовательских данных (EditProfileScreen) отображаются

2. Тест-кейс, проверить отображение карточки баланса отдыха пользователя в окне ProfileScreen, после действий пользователя, направленных на её активацию (табл 6).

Таблица 6. Тест-кейс 3

Шаги	Ожидаемый результат
Оставить доступным только тест balanceHolidayCardViewIsDisplayedAfterClickUser	Все тесты, не относящиеся к указанному, будут закомментированы
Нажать ПКМ на класс ExampleInstrumentedTest	Появится специализированное меню
В открывшемся меню нажать на пункт: Run ‘ExampleUnitTest’	Пользователь увидит, успешный результат того, что карточка баланса отдыха после нажатия пользователя на кнопку отображается

3. Тест-кейс, проверить, что вводимые данные пользователем в поля “Логин” и “Пароль”, на окне авторизации, отображаются (табл 7)

Таблица 7. Тест-кейс 4

Шаги	Ожидаемый результат
Оставить доступным только тест <code>insertTextInOutlinedTextFieldEmailIsDisplayedInLoginView</code>	Все тесты, не относящиеся к указанному, будут закомментированы
Нажать ПКМ на класс <code>ExampleInstrumentedTest</code>	Появится специализированное меню
В открывшемся меню нажать на пункт: Run ‘ <code>ExampleUnitTest</code> ’	Пользователь увидит, успешный результат того, что внесённые данные пользователем отображаются

4. Тест-кейс, проверить, что осуществляется валидация данных на окне авторизации (табл 8)

Таблица 8. Тест-кейс 5

Шаги	Ожидаемый результат
Оставить доступным только тест emailAndPasswordValidationIsWorks	Все тесты, не относящиеся к указанному, будут закомментированы
Нажать ПКМ на класс ExampleInstrumentedTest	Появится специализированное меню
В открывшемся меню нажать на пункт: Run 'ExampleUnitTest'	Пользователь увидит, успешный результат того, что валидация данных срабатывает