**Государственное бюджетное профессиональное образовательное учреждение «Нижегородский Губернский колледж»**

Методическая комиссия «Информатика и вычислительная техника»

Допущен к защите:

преподаватель

\_\_\_\_\_\_\_\_\_\_\_\_ Ю. С. Мамшева

«4» декабря 2024 г.

**ОТЧЕТ**

**ПО РАЗРАБОТКЕ МОБИЛЬНОГО ПРИЛОЖЕНИЯ «WarehouseEmployee»**

МДК 01.03 Разработка мобильных приложений

Руководитель: \_\_\_\_\_\_\_\_\_\_ Ю.С. Мамшева 04.12.2024 г.

Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Г.А. Куклев 04.12.2024 г.

Специальность, группа: 09.02.07, 31П

Нижний Новгород

2024 г.

**РАЗРАБОТКА КОНЦЕПТА ПРИЛОЖЕНИЯ И СОСТАВЛЕНИЕ ТЕХНИЧЕСКОГО ЗАДАНИЯ**

Современные склады сталкиваются с растущим объемом работы, необходимостью повышения производительности и оптимизации логистических процессов. Развитие мобильных технологий открывает уникальные возможности для создания решений, которые могут изменить подход к работе персонала на складе. Приложение, призванное улучшить взаимодействие сотрудников, повысить эффективность и прозрачность выполнения задач, приобретает особую актуальность. Удобный инструмент для распределения задач и оперативной коммуникации внутри команды способен существенно повысить производительность труда, сократить время выполнения заказов, минимизировать риски и повысить общую эффективность всего складского процесса. В условиях конкуренции и постоянного роста объемов работы, разработка такого приложения позволит компаниям оставаться конкурентоспособными, оптимизировать расходы и повысить уровень удовлетворенности персонала.

Исходя от актуальности, были поставлены следующие задачи, которые приложение должно решать:

1. Предоставление необходимой информации о задаче
2. Расчет времени до выполнения задачи
3. Коммуникация между работниками с помощью сообщений
4. Отображение оптимального пути выполнения задачи

Также необходимо разработать приложение не только для обычных сотрудников, но и от тех, которые являются главными на смене. Для них нужно добавить функционал, который позволит ввести отчетность о работниках, которые пришли на смену (журнал посещаемости). У главных по смене немного изменяется отображение данных о задачах, что описано в составленном техническом задании. В нем также было прописано где будет разрабатываться дизайн, само приложение, какие паттерны использоваться и другие основные требования «[См. ПРИЛОЖЕНИЕ А. Техническое задание](#_ПРИЛОЖЕНИЕ_А)».

**РАЗРАБОТКА ДИЗАЙНА ПРИЛОЖЕНИЯ**

Исходя из концепции приложения, а именно то, что оно создается для рабочих, должно ускорять и упрощать работу, было принято решение делать дизайн не сильно ярким, пестрым, вызывающим, чтобы не отвлекать работников от работы. По такой же логике выбирались шрифты для приложения. Однако для удобства работников были разработаны две темы приложения – темная и светлая.

**Выбор цветовой палитры**

**Для темной темы:**

**Главный цвет:** #424141 (Задний фон приложения и элементов)



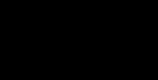
**Вторичный цвет:** #686666 (Фон элементов списков, списка)

****

**Акцентный цвет:** #FBF4F4 (Фон кнопок, области заголовок)

****

**Дополнительный цвет:** #000000 (Текст)

****

**Для светлой темы:**

**Главный цвет:** #DBDBDB (Задний фон приложения и элементов)

****

**Вторичный цвет:** # 9B9999 (Фон элементов списков, списка)

****

**Акцентный цвет:** #006BA7 (Фон кнопок, области заголовок, некоторый текст)

****

**Дополнительный цвет:** #FFFFF (Текст)

****

**Выбор шрифта**

**Для заголовков:** Magnet Trial Bold.

**Для текста:** Seravek Regular/Bold

**Макет приложения:**

Макет был создан в сервисе «Figma». Макет доступен по ссылке - <https://www.figma.com/design/I0BU5cw0tcnEDkCUU53PKI/Untitled?node-id=0-1&t=CpNYu23TgVZ4WGKr-1>

По итогу разработки были созданы дизайны следующих экранов:

1. Авторизация – первый экран, который видит пользователь



Рисунок 1. Экран авторизации

1. Журнал посещаемости – экран, который видит главный по смене после авторизации.

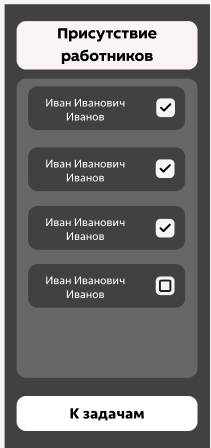


Рисунок 2. Экран журнала посещаемости

1. Задачи – экран, который видит обычный работник после авторизации. Главным работник увидит экран после журнала посещаемости. Если зашел обычный работник, то будут отображаться те задачи, в которых он участвует. Если главный по смене, то задачи, в которых он главный. Также на экране есть таймер, который начинает отсчет от самой близкой задачи по времени выполнения, относительно времени на телефоне. Если время прошло, то рамка задачи станет зеленым. Если все задачи прошли, то вместо таймера будет надпись, что сегодня задач больше нет.

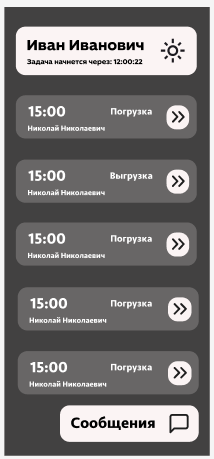


Рисунок 3. Экран задач

1. Задача погрузки – экран, если работник перешел в задачу с типом «Погрузка». С этого экрана можно посмотреть оптимальный путь выполнения задачи, написать сообщение, посмотреть информацию о ячейке склада. В колонке «Товар - Ячейка» показано таким образом, что какой товар необходимо погрузить в ячейку.



Рисунок 4. Экран задачи погрузки

1. Задача выгрузки – экран, который имеет такое же назначение, функционал, как экран «Задача погрузки», однако открывается для задач с типом «Выгрузка» и отображение колонки «Товар - Ячейка» показывает из какой ячейки необходимо выгрузить товар.



Рисунок 5. Экран задачи выгрузки

1. Ячейка – экран, который открывается из экрана задач погрузки/выгрузки. Содержит в себе информацию о ячейке.



Рисунок 6. Экран ячейки

1. Выбор работника для сообщения – дополнение к экрану «Задачи» и «Задача погрузки/выгрузки» (в зависимости от роли). Если в приложение зашел главный по смене, то:

* На экране «Задачи» будут отображаться обычные работники, которые участвуют в выполнении сегодняшних задач.
* На экране «Задача погрузка/выгрузка» будут отображаться те работники, которые выполняют конкретно эту задачу.

Если зашел обычный работник, то:

* На экране «Задачи» будут отображаться все работники, которые являются ответственными по задачам.
* На экране «Задача погрузка/выгрузка» будет открываться сразу чат с ответственными по задаче



Рисунок 7.1. Экран выбора работника для сообщения

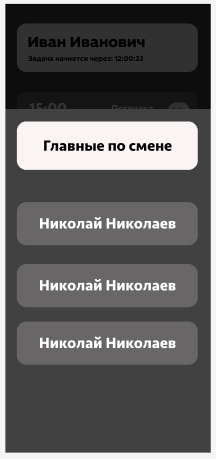


Рисунок 7.2. Экран выбора работника для сообщения

1. Чат – экран, в котором работники ведут коммуникацию между собой.



Рисунок 8. Экран чата

1. Оптимальный путь – экран, на котором отображается изображение оптимального пути решения задачи

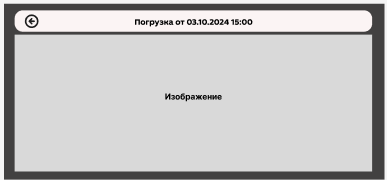


Рисунок 9. Экран оптимального пути

**РАЗРАБОТКА СЕРВЕНОЙ ЧАСТИ ПРИЛОЖЕНЯ**

Было решено реализовать серверную часть приложения через сервис «Supabase»

Supabase – это бесплатный аналог Firebase, полифункциональная платформа, объединяющая в себе несколько важных программных решений и упрощающая их реализацию до предельно примитивного уровня, чтобы даже новички в разработке могли спокойно добавить в свои приложения или сайты такие функции, как авторизация, хранилище файлов, обновление контента на сайте в реальном времени и т.п.

Таблица 1

Список сущностей

|  |  |  |
| --- | --- | --- |
| № | Название | Назначение |
| 1 | Работники | Перечень лиц, работающих на складе. |
| 2 | Роли | Перечень должностей работников на складе. |
| 3 | Склады | Перечень складов предприятия |
| 4 | Секции | Перечень секций склада |
| 5 | Ячейки | Перечень ячеек секций склада, который хранят в себе продукты |
| 6 | Продукты | Перечень товаров, которые хранятся на складе |
| 7 | Категории продуктов | Перечень категорий продуктов |
| 8 | Задачи | Перечень задач, который должны выполняться на складах |
| 9 | Категории задач | Перечень категорий задач |
| 10 | Задачи и продукты | Перечень продуктов, который участвуют в задаче и ячейки склада, в которые будут помещены/выгружены продукты |
| 11 | Задачи и работники | Перечень работников, которые должны выполнить задачу |
| 12 | Чаты | Перечень всех чатов сотрудников |
| 13 | Сообщения | Перечень сообщений сотрудников |
| 14 | Рабочие смены | Перечень рабочих смен на складах |
| № | Название | Назначение |
| 15 | Рабочие смены и сотрудники | Перечень сотрудников, которые должны прийти на смену |

Для каждой таблицы (сущности) были описаны ее атрибутов. Атрибут на физическом уровне – это колонки таблицы и выражает определенное свойство объекта.

Таблица 2

Список атрибутов таблицы «Работники»

| Ключевое поле | Название | Назначение |
| --- | --- | --- |
| ПК (первичный ключ) | Идентификатор работника | Ключевое поле, предназначенное для однозначной идентификации каждой записи в таблице. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждому работнику. Это целое число. |
|  | Идентификатор роли | Внешний ключ, ссылающийся на таблицу должностей. Определяет должность работника. |
|  | Имя |  |
|  | Фамилия |  |
|  | Отчество |  |
|  | Идентификатор склада | Внешний ключ, ссылающийся на таблицу складов. Определяет, к какому складу привязан работник. |

Таблица 3

Список атрибутов таблицы «Должности»

| Ключевое поле | Название | Назначение |
| --- | --- | --- |
| ПК (первичный ключ) | Идентификатор роли | Ключевое поле, предназначенное для однозначной идентификации каждой роли. |
|  | Название роли |  |

Таблица 4

Список атрибутов таблицы «Склады»

| Ключевое поле | Название | Назначение |
| --- | --- | --- |
| ПК (первичный ключ) | Идентификатор склада | Ключевое поле, предназначенное для однозначной идентификации каждого склада. |
|  | Название склада |  |
|  | Расположение |  |

Таблица 5

Список атрибутов таблицы «Ячейки»

| Ключевое поле | Название | Назначение |
| --- | --- | --- |
| ПК (первичный ключ) | Идентификатор ячейки | Ключевое поле, предназначенное для однозначной идентификации каждой ячейки. |
| ВК | Идентификатор секции | Внешний ключ, ссылающийся на таблицу секций. Указывает в какой секции находится ячейка |
| ВК | Идентификатор продукта | Внешний ключ, ссылающийся на таблицу продуктов. Указывает какой продукт хранится в ячейке |
|  | Название ячейки |  |
|  | Максимальное количество продукции |  |
|  | Количество продукции |  |
|  | Вес продукции |  |
|  | Сокращенное название |  |

Таблица 6

Список атрибутов таблицы «Секции»

| Ключевое поле | Название | Назначение |
| --- | --- | --- |
| ПК (первичный ключ) | Идентификатор секции | Ключевое поле, предназначенное для однозначной идентификации каждой секции. |
|  | Название секции |  |
| ВК | Идентификатор склада | Внешний ключ, ссылающийся на таблицу складов. Указывает на каком складе находится секция. |
|  | Сокращенное название |  |

Таблица 7

Список атрибутов таблицы «Продукты»

| Ключевое поле | Название | Назначение |
| --- | --- | --- |
| ПК (первичный ключ) | Идентификатор продукта | Ключевое поле, предназначенное для однозначной идентификации каждого продукта. |
|  | Название продукта |  |
|  | Артикул |  |
|  | Вес |  |
| ВК | Идентификатор категории | Внешний ключ, ссылающийся на таблицу категорий продуктов. Указывает категорию продукта. |

Таблица 8

Список атрибутов таблицы «Категории продуктов»

| Ключевое поле | Название | Назначение |
| --- | --- | --- |
| ПК (первичный ключ) | Идентификатор | Ключевое поле, предназначенное для однозначной идентификации каждой категории продуктов. |
|  | Название категории |  |

Таблица 9

Список атрибутов таблицы «Задачи»

| Ключевое поле | Название | Назначение |
| --- | --- | --- |
| ПК (первичный ключ) | Идентификатор задачи | Ключевое поле, предназначенное для однозначной идентификации каждой задачи. |
| ВК | Идентификатор категории задачи | Внешний ключ, ссылающийся на таблицу категорий задач. Указывает на категорию задачи. |
| ВК | Идентификатор ответственного работника | Внешний ключ, ссылающийся на таблицу работников. Указывает ответственного за задачу работника. |
|  | Дата создание |  |
|  | Путь к оптимальному изображению |  |
|  | Дата выполнения задачи |  |
|  | Завершена |  |

Таблица 10

Список атрибутов таблицы «Категории задач»

| Ключевое поле | Название | Назначение |
| --- | --- | --- |
| ПК (первичный ключ) | Идентификатор категории | Ключевое поле, предназначенное для однозначной идентификации каждой категории задач. |
|  | Название категории |  |

Таблица 11

Список атрибутов таблицы «Задачи и продукты»

| Ключевое поле | Название | Назначение |
| --- | --- | --- |
| ПК (первичный ключ) | Идентификатор задачи с продуктами | Ключевое поле, предназначенное для однозначной идентификации привязки задачи к продукту. |
| ВК | Идентификатор задачи | Внешний ключ, ссылающийся на таблицу задач. Указывает на выполняемую задачу. |
| ВК | Идентификатор продукта | Внешний ключ, ссылающийся на таблицу продуктов, указывает на продукт, который участвует в задаче. |
| ВК | Идентификатор ячейки | Внешний ключ, ссылающийся на таблицу ячеек. Указывает в какую / из какой ячейки будет использован продукт. |

|  |  |  |
| --- | --- | --- |
| Ключевое поле | Название | Назначение |
|  | Количество продукта |  |
|  | Позиция в оптимальном сортировочном ряду |  |

Таблица 12

Список атрибутов таблицы «Задачи и работники»

| Ключевое поле | Название | Назначение |
| --- | --- | --- |
| ПК (первичный ключ) | Идентификатор задачи с работниками | Ключевое поле, предназначенное для однозначной идентификации записи о привязке работника к задаче. |
| ВК | Идентификатор задачи | Внешний ключ, ссылающийся на таблицу задач. Указывает на выполняемую задачу. |
| ВК | Идентификатор работника | Внешний ключ, ссылающийся на таблицу работников. Указывает на работников, которые должны выполнить задачу |
|  | Завершен работником |  |

Таблица 13

Список атрибутов таблицы «Чаты»

| Ключевое поле | Название | Назначение |
| --- | --- | --- |
| ПК (первичный ключ) | Идентификатор чата | Ключевое поле, предназначенное для однозначной идентификации каждого чата. |

|  |  |  |
| --- | --- | --- |
| Ключевое поле | Название | Назначение |
| ВК | Идентификатор первого работника | Внешний ключ, ссылающийся на таблицу работников. Указывает на первого собеседника в чате. |
| ВК | Идентификатор второго работника | Внешний ключ, ссылающийся на таблицу работников. Указывает на второго собеседника в чате. |

Таблица 14

Список атрибутов таблицы «Сообщения»

| Ключевое поле | Название | Назначение |
| --- | --- | --- |
| ПК (первичный ключ) | Идентификатор сообщения | Ключевое поле, предназначенное для однозначной идентификации каждого сообщения в чате. |
| ВК | Идентификатор чата | Внешний ключ, ссылающийся на таблицу чатов. Указывает в каком чате было написано сообщение. |
| ВК | Идентификатор отправителя | Внешний ключ, ссылающийся на таблицу работников. Указывает какой работник написал сообщение. |
|  | Содержание сообщения |  |

Таблица 15

Список атрибутов таблицы «Рабочие смены»

| Ключевое поле | Название | Назначение |
| --- | --- | --- |
| ПК (первичный ключ) | Идентификатор смены | Ключевое поле, предназначенное для однозначной идентификации каждой смены. |
|  | Дата смены |  |
| ВК | Идентификатор склада | Внешний ключ, ссылающийся на таблицу складов. Указывает в каком складе проходит смена |
| ВК | Идентификатор основного работника смены | Внешний ключ, ссылающийся на таблицу работников. Указывает на работника, который главный на смене. |

Таблица 16

Список атрибутов таблицы «Рабочие смены и сотрудники»

| Ключевое поле | Название | Назначение |
| --- | --- | --- |
| ПК (первичный ключ) | Идентификатор смены с работником | Ключевое поле, предназначенное для однозначной идентификации записи о привязке работника к смене. |
| ВК | Идентификатор работника | Внешний ключ, ссылающийся на таблицу работников. Указывает на работника, который должен прийти на смену. |

|  |  |  |
| --- | --- | --- |
| Ключевое поле | Название | Назначение |
| ВК | Идентификатор смены | Внешний ключ, ссылающийся на таблицу смен. Указывает на конкретную смену. |

Таблица 17

Список связей

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название связи | Сущности, участвующие в связи | Назначение |
| 1 | 1:M | Работники – Роли | Один работник имеет одну должность |
| 2 | 1:М | Работники - Чаты | Один работник может находиться в нескольких чатах |
| 3 | 1:М | Чаты – Сообщения | В одном чате несколько сообщений |
| 4 | 1:M | Работники - Сообщения | Один работник может написать несколько сообщений |
| 5 | M:М | Работники - Задачи | Одну задачу могут выполнять несколько работников. Один работник может выполнять несколько задач |
| 6 | М:М | Задачи - Продукты | В одной задаче участвует несколько продуктов. Один продукт может находиться в разных задачах |

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название связи | Сущности, участвующие в связи | Назначение |
| 7 | 1:M | Задачи – Категории задач | Одна категория может характеризовать несколько задач |
| 8 | 1:М | Продукты – Категории продуктов | Одна категория может характеризовать несколько продуктов |
| 9 | 1:M | Работники - Склады | На одном складе работает много работников |
| 10 | 1:M | Склады - Секции | В одном складе несколько секций |
| 11 | 1:M | Секции - Ячейки | В одной секции несколько ячеек |
| 12 | 1:M | Ячейки - Продукты | Один продукт может храниться в нескольких ячейках |
| 13 | 1:M | Рабочие смены - склад | На одном складе может быть несколько смен |
| 14 | M:М | Рабочие смены - Работники | На одной смене несколько работников. Один работник может быть в нескольких сменах |

Таблица 18

Список атрибутов таблицы «Рабочие смены и работники»

| № | Название | Идентификатор | Тип | Не пусто | Ограничение |
| --- | --- | --- | --- | --- | --- |
| 1 | Идентификатор | id | Строковой | Да | ПК (первичный ключ) |
| 2 | Идентификатор работника | id\_worker | Числовой | Да | FK (внешний ключ) |
| 3 | Идентификатор смены | id\_work\_shift | Числовой | Да | FK (внешний ключ) |

Таблица 19

Список атрибутов таблицы «Рабочие смены»

| № | Название | Идентификатор | Тип | Не пусто | Ограничение |
| --- | --- | --- | --- | --- | --- |
| 1 | Идентификатор | id | Числовой | Да | ПК (первичный ключ) |
| 2 | Дата смены | date\_shift | Дата | Да |  |
| 3 | Идентификатор Главного работника смены | id\_main\_shift\_worker | Строковой | Да | FK (внешний ключ) |

Таблица 20

Список атрибутов таблицы «Работники»

| № | Название | Идентификатор | Тип | Не пусто | Ограничение |
| --- | --- | --- | --- | --- | --- |
| 1 | Идентификатор | id | Числовой | Да | ПК (первичный ключ) |
| 2 | Идентификатор роли | id\_role | Числовой | Да | FK (внешний ключ) |
| 3 | Имя | first\_name | Текстовый | Да |  |
| 4 | Фамилия | last\_name | Текстовый | Да |  |
| 5 | Отчество | patronymic | Текстовый | Да |  |
| 6 | Идентификатор склада | id\_warehouse | Числовой | Да | FK (внешний ключ) |

Таблица 21

Список атрибутов таблицы «Роли»

| № | Название | Идентификатор | Тип | Не пусто | Ограничение |
| --- | --- | --- | --- | --- | --- |
| 1 | Идентификатор | id | Числовой | Да | ПК (первичный ключ) |
| 2 | Название роли | name\_role | Текстовый | Да |  |

Таблица 22

Список атрибутов таблицы «Чаты»

| № | Название | Идентификатор | Тип | Не пусто | Ограничение |
| --- | --- | --- | --- | --- | --- |
| 1 | Идентификатор | id | Числовой | Да | ПК (первичный ключ) |
| 2 | Идентификатор первого работника в чате | worker\_first | Строковый | Да | FK (внешний ключ) |
| 3 | Идентификатор второго работника в чате | worker\_second | Строковый | Да | FK (внешний ключ) |

Таблица 23

Список атрибутов таблицы «Сообщения и чаты»

| № | Название | Идентификатор | Тип | Не пусто | Ограничение |
| --- | --- | --- | --- | --- | --- |
| 1 | Идентификатор | id | Числовой | Да | ПК (первичный ключ) |
| 2 | Идентификатор чата | id\_chat | Числовой | Да | FK (внешний ключ) |
| 3 | Идентификатор работника, отправившего сообщение | id\_worker\_sender | Строковый | Да | FK (внешний ключ) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Название | Идентификатор | Тип | Не пусто | Ограничение |
| 4 | Содержание сообщения | content\_message | Текстовый | Да |  |

Таблица 24

Список атрибутов таблицы «Склады»

| № | Название | Идентификатор | Тип | Не пусто | Ограничение |
| --- | --- | --- | --- | --- | --- |
| 1 | Идентификатор | id | Числовой | Да | ПК (первичный ключ) |
| 2 | Название склада | name\_warehouse | Текстовый | Да |  |
| 3 | Локация | location | Текстовый | Да |  |

Таблица 25

Список атрибутов таблицы «Секции»

| № | Название | Идентификатор | Тип | Не пусто | Ограничение |
| --- | --- | --- | --- | --- | --- |
| 1 | Идентификатор | id | Числовой | Да | ПК (первичный ключ) |
| 2 | Название секции | name\_section | Текстовый | Да |  |
| 3 | Идентификатор склада | id\_warehouse | Числовой | Да | FK (внешний ключ) |
| 4 | Сокращенное название | abbreviated\_name | Текстовый | Да |  |

Таблица 26

Список атрибутов таблицы «Ячейки»

| № | Название | Идентификатор | Тип | Не пусто | Ограничение |
| --- | --- | --- | --- | --- | --- |
| 1 | Идентификатор | id | Числовой | Да | ПК (первичный ключ) |
| 2 | Идентификатор секции | id\_section | Числовой | Да | FK (внешний ключ) |
| 3 | Идентификатор продукта | id\_product | Числовой | Да | FK (внешний ключ) |
| 4 | Название ячейки | name\_cell | Текстовый | Да |  |
| 5 | Кол-во продукта в ячейке | count\_product\_in\_cell | Числовой | Да |  |
| 6 | Вес продукта в ячейке | weight\_product\_in\_cell | Числовой | Да |  |
| 7 | Сокращенное название | abbreviated\_name | Текстовый | Да |  |

Таблица 27

Список атрибутов таблицы «Продукты»

| № | Название | Идентификатор | Тип | Не пусто | Ограничение |
| --- | --- | --- | --- | --- | --- |
| 1 | Идентификатор | ID | Числовой | Да | ПК (первичный ключ) |
| 2 | Название продукта | Product\_Name | Текстовый | Да |  |
| 3 | Артикул | Article | Текстовый | Да |  |
| 4 | Вес | Weight | Числовой | Да |  |
| 5 | Идентификатор категории продукта | ID\_Product\_Category | Числовой | Да | FK (внешний ключ) |

Таблица 28

Список атрибутов таблицы «Категории продуктов»

| № | Название | Идентификатор | Тип | Не пусто | Ограничение |
| --- | --- | --- | --- | --- | --- |
| 1 | Идентификатор | id | Числовой | Да | ПК (первичный ключ) |
| 2 | Название категории | name\_category | Текстовый | Да |  |

Таблица 29

Список атрибутов таблицы «Категории задач»

| № | Название | Идентификатор | Тип | Не пусто | Ограничение |
| --- | --- | --- | --- | --- | --- |
| 1 | Идентификатор | id | Числовой | Да | ПК (первичный ключ) |
| 2 | Название категории | name\_categories | Текстовый | Да |  |

Таблица 30

Список атрибутов таблицы «Задачи»

| № | Название | Идентификатор | Тип | Не пусто | Ограничение |
| --- | --- | --- | --- | --- | --- |
| 1 | Идентификатор | id | Числовой | Да | ПК (первичный ключ) |
| 2 | Идентификатор категории задачи | id\_category\_task | Числовой | Да | FK (внешний ключ) |
| 3 | Идентификатор работника, ответственного за задачу | id\_responsible\_worker | Строковой | Да | FK (внешний ключ) |
| 4 | Дата создания | date\_create | Дата | Да |  |
| 5 | Путь к изображению | img\_optimal\_path | Текстовый | Да |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Название | Идентификатор | Тип | Не пусто | Ограничение |
| 6 | Дата выполнения | date\_execution\_task | Дата | Да |  |
| 7 | Завершена | is\_completed | Логический | Да |  |

Таблица 31

Список атрибутов таблицы «Задачи и работники»

| № | Название | Идентификатор | Тип | Не пусто | Ограничение |
| --- | --- | --- | --- | --- | --- |
| 1 | Идентификатор | id | Числовой | Да | ПК (первичный ключ) |
| 2 | Идентификатор задачи | id\_task | Числовой | Да | FK (внешний ключ) |
| 3 | Идентификатор работника, выполнившего задачу | id\_worker\_compited | Числовой | Да | FK (внешний ключ) |
| 4 | Завершен работником | Is\_worker\_complited | Логический | Да |  |

Таблица 32

Список атрибутов таблицы «Задачи и продукты»

| № | Название | Идентификатор | Тип | Не пусто | Ограничение |
| --- | --- | --- | --- | --- | --- |
| 1 | Идентификатор | id | Числовой | Да | ПК (первичный ключ) |
| 2 | Идентификатор задачи | id\_task | Числовой | Да | FK (внешний ключ) |
| 3 | Идентификатор продукта | id\_product | Числовой | Да | FK (внешний ключ) |
| 4 | Идентификатор ячейки | id\_cell | Числовой | Да | FK (внешний ключ) |
| 5 | Кол-во продукта в задаче | count\_product | Числовой | Да |  |
| 6 | Позиция продукта в оптимальной схеме | position\_in\_optimal\_scheme | Числовой | Да |  |

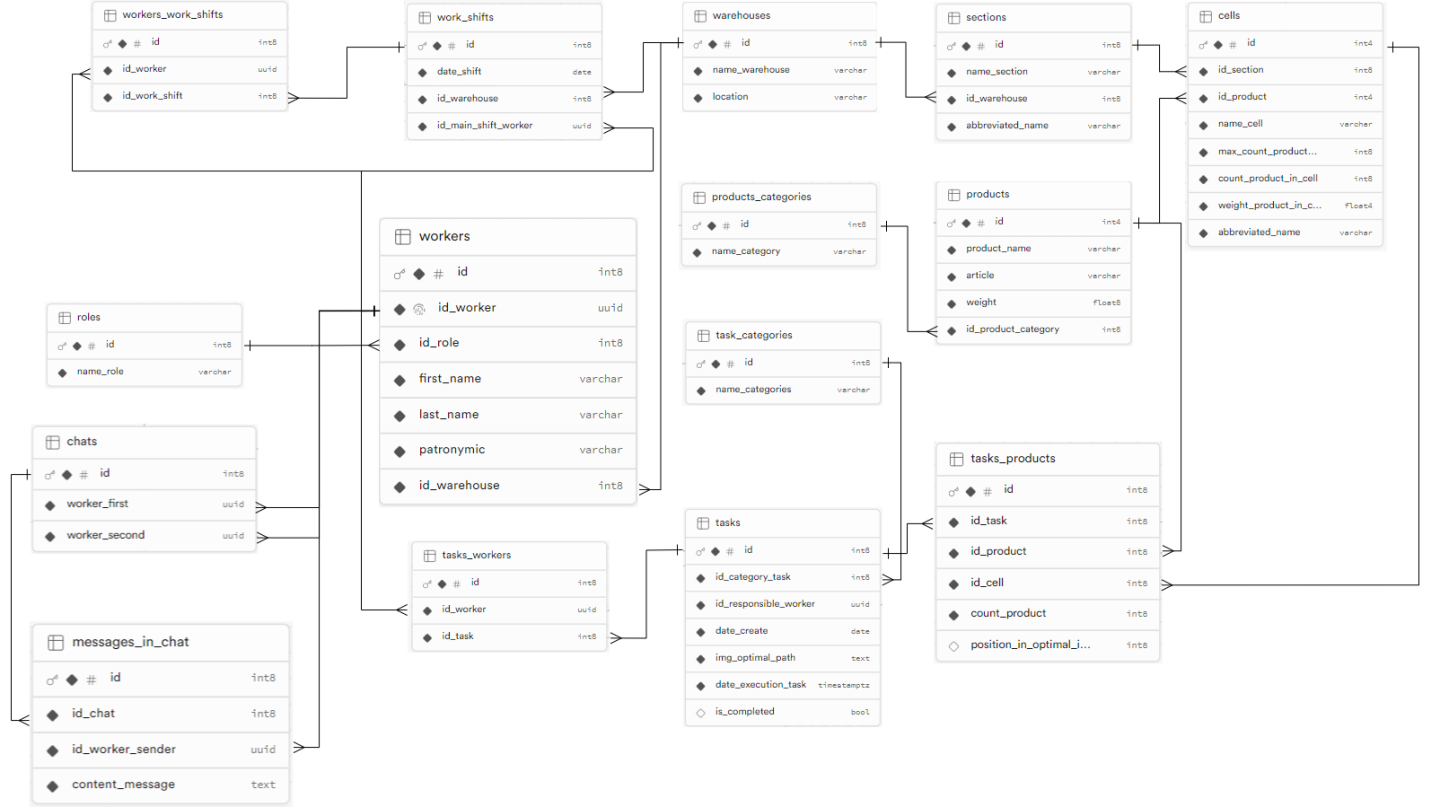


Рисунок 1. ER-диаграмма БД

Триггеры

1. Подсчет веса всех продуктов в таблице «Cell» при изменении количества продукта в ячейке.

|  |
| --- |
| begin      new.weight\_product\_in\_cell := new.count\_product\_in\_cell \* (SELECT weight FROM public."Product" WHERE id = new.id\_product);      RETURN new;  end; |

1. Обновляет выполнимость задачи на true, когда все работники выполнили свою задачу в таблице tasks\_workers

|  |
| --- |
| BEGIN    -- Проверяем, есть ли еще работники, которые не завершили задачу    IF NOT EXISTS (      SELECT 1      FROM tasks\_workers      WHERE id\_task = NEW.id\_task      AND is\_worker\_complited = FALSE    ) THEN      -- Если все работники завершили задачу, обновляем статус в tasks      UPDATE tasks      SET is\_completed = TRUE      WHERE id = NEW.id\_task;    END IF;    RETURN NEW;  END; |

1. Проверяет запись в таблице Workers\_work\_shift, чтобы один и тот же сотрудник не был записан на ту же смену (т.е. нет повторяющихся записей)

|  |
| --- |
| BEGIN    IF EXISTS (      SELECT 1      FROM workers\_work\_shifts      WHERE id\_work\_shift = NEW.id\_work\_shift AND id\_worker = NEW.id\_worker    ) THEN      RAISE EXCEPTION 'Работник уже записан на эту смену';    END IF;    RETURN NEW;  END; |

**ТЕСТИРОВАНИЕ СЕРВЕРНОЙ ЧАСТИ ПРИЛОЖЕНИЯ**

Перед тестированием был разработан тест-план

1. **Цели и задачи тестирования**
2. Цели

* Проверка работоспособность БД приложения
* Проверка работоспособность авторизации

1. Задачи

* Убедиться в работе сервиса Supabase
* Убедиться в разработанной БД на сервисе Supabase (выборка, вставка данных, обновление)
* Убедиться в корректной работе триггеров БД
* Убедиться в работе механизма авторизации Supabase

1. **Объекты тестирования**
2. Таблица «users» в схеме «auth»
3. Все сущности (таблицы) схемы «public»
4. Механизм авторизации
5. Механизм работы с БД
6. Триггеры БД
7. **Методы тестирования**
8. Функциональное тестирование

* Обработка ошибок
* Авторизация пользователь
* Работа с данными

1. **Ресурсы и инструменты тестирования**

Для тестирования необходим один тестировщик, ПК, созданная БД на сервисе Supabase, реализованные триггеры, заполненные таблицы данными. Тестирование будет происходить в «Visual Studio Code» через «Supabase CLI».

Supabase CLI - это Локальная разработка, развертывание на платформе Supabase и настройка рабочих процессов CI/CD.

1. **Планирование и оценка времени**

Функциональное тестирование:

Начало: 28.10.2024

Конец (приблизительно): 01.11.2024

1. **Критерии завершения тестирования:**
2. Все тестовые случаи, определенные в тестовых наборах, успешно пройдены.
3. Все обнаруженные ошибки исправлены, а их исправления проверены.
4. Сервер соответствует требованиям разработки

Далее были разработаны тестовые наборы и тест-кейсы

1. **Тестовые наборы**
2. Авторизация пользователей

|  |  |
| --- | --- |
| Тест-кейсы | |
| Название | Ожидаемый результат |
| Авторизация с корректным логином и паролем | Сервер вернет uuid пользователя |
| Авторизация с пустым логином и паролем | Сервер вернет null |
| Авторизация с некорректным логином | Сервер вернет null |

1. Выборка данных

|  |  |
| --- | --- |
| Тест-кейсы | |
| Название | Ожидаемый результат |
| Выборка работников на сегодняшнюю смену | Сервер вернет список пользователей, которые должны прийти на сегодняшнюю смену |
| Выборка задач пользователя на сегодняшний день | Сервер вернет список сегодняшних задач пользователя |
| Выборка ячеек, участвующих в задаче | Сервер вернет список ячеек, участвующих в задаче |
| Выборка продуктов, участвующих в задаче | Сервер вернет список продуктов, участвующих в задаче |

1. Триггеры ([см. СЕРВЕРНАЯ ЧАСТЬ ПРИЛОЖЕНИЯ](../СЕРВЕРНАЯ%20ЧАСТЬ%20ПРИЛОЖЕНИЯ.docx))

|  |  |
| --- | --- |
| Тест-кейсы | |
| Название | Ожидаемый результат |
| Подсчет веса всех продуктов в таблице «cells» при изменении количества продукта в ячейке | Сервер автоматически изменит вес продукта в ячейке |
| Название | Ожидаемый результат |
| Добавление одинаковой записи в таблицу «workers\_work\_shifts» | Сервер выведет ошибку о том, что работник уже записан на эту смену |
| Обновление выполнения работником задачи в таблице «tasks\_workers» | Сервер автоматически изменит в таблице «tasks» у задачи поле «is\_completed» на true |
| Обновление количества продуктов в ячейке при выполнении задачи | Сервер автоматически обновит количество продуктов в таблице «cells» в соответствии категории задачи |

1. **Тест-кейсы**

* Название: Авторизация с корректным логином и паролем

Предварительные условия: в БД существует пользователь с телефоном «79290509332» и паролем «123» (проверяется именно зашифрованный пароль), открыт Supabase в браузере

Шаги:

1. Открыть «SQL Editor»
2. Ввести запрос «SELECT \* FROM auth.users WHERE phone = '79290509332' and encrypted\_password = '$2a$10$V6z25.1LqNpQNsa6Vomgp.m8U/.UC743rp1FJ3yhvxtweqaV.FEDO'»
3. Нажать кнопку «Run»

Ожидаемый результат: сервер вернет пользователя с uuid «4202d2e7-5164-4cd1-aa16-e38815846b7c»

* Название: Авторизация с пустым логином и паролем

Предварительные условия: открыт Supabase в браузере

Шаги:

1. Открыть «SQL Editor»
2. Ввести запрос «SELECT \* FROM auth.users WHERE phone = ' ' and encrypted\_password = ' '»
3. Нажать кнопку «Run»

Ожидаемый результат: сервер вернет null

* Название: Авторизация с некорректным логином

Предварительные условия: открыт Supabase в браузере

Шаги:

1. Открыть «SQL Editor»
2. Ввести запрос «SELECT \* FROM auth.users WHERE phone = '7929050933333332' and encrypted\_password = '$2a$10$V6z25.1LqNpQNsa6Vomgp.m8U/.UC743rp1FJ3yhvxtweqaV.FEDO'»
3. Нажать кнопку «Run»

Ожидаемый результат: сервер вернет null

* Название: Выборка работников на смену «30.10.2024»

Предварительные условия: открыт Supabase в браузере, в таблице «workers\_work\_shifts» существуют записи на дату «30.10.2024» и смена для склада, где id равен 1

Шаги:

1. Открыть «SQL Editor»
2. Ввести запрос «SELECT id\_worker FROM public.workers\_work\_shifts

WHERE id\_work\_shift = (SELECT id FROM public.work\_shifts WHERE date\_shift = '2024-10-30' AND id\_warehouse = 1);»

1. Нажать кнопку «Run»

Ожидаемый результат: сервер вернет список работников с 1 склада, которые должны прийти на смену «30.10.2024»

* Название: Выборка задач пользователя на сегодняшний день

Предварительные условия: открыт Supabase в браузере, в таблице «tasks\_workers» существуют записи для пользователя с uuid «024b4df6-0695-44ac-9d52-c1a3d3be4771», где задачи должны быть выполнены «30.10.2024»

Шаги:

1. Открыть «SQL Editor»
2. Ввести запрос «SELECT id\_task FROM public.tasks\_workers WHERE id\_worker = '024b4df6-0695-44ac-9d52-c1a3d3be4771' AND id\_task IN ( SELECT id FROM public.tasks WHERE date\_trunc('day', date\_execution\_task) = '2024-10-30');»
3. Нажать кнопку «Run»

Ожидаемый результат: сервер вернет список задач, которые должны быть выполнены пользователем «30.10.2024»

* Название: Выборка ячеек, участвующих в задаче

Предварительные условия: открыт Supabase в браузере, в таблице «tasks\_cells» существуют записи, где id задачи равен 4

Шаги:

1. Открыть «SQL Editor»
2. Ввести запрос «SELECT \* FROM public.cells WHERE id IN ( SELECT id\_cell FROM public.tasks\_products WHERE id\_task = 4);»
3. Нажать кнопку «Run»

Ожидаемый результат: сервер вернет список ячеек, которые участвуют в задаче, где id равен 4

* Название: Выборка продуктов, участвующих в задаче

Предварительные условия: открыт Supabase в браузере, в таблице «tasks\_products» существуют записи, где id задачи равен 4

Шаги:

1. Открыть «SQL Editor»
2. Вставить запрос «SELECT \* FROM public.products WHERE id IN (SELECT id\_product FROM public.tasks\_products WHERE id\_task = 4);»
3. Нажать кнопку «Run»

Ожидаемый результат: сервер вернет список продуктов, которые участвуют в задаче, где id равен 4

* Название: Подсчет веса всех продуктов в таблице «cells» при изменении количества продукта в ячейке

Предварительные условия: открыт Supabase в браузере, в таблице «cells» существует запись, где id\_product – 1, count\_product\_in\_cell – 1, weight\_product\_in\_cell – 0.5

Шаги:

1. Открыть «SQL Editor»
2. Ввести запрос «update public.cells set count\_product\_in\_cell = 2 where id = 1»
3. Нажать кнопку «Run»
4. Открыть в «Table Editor» таблицу «cells»

Ожидаемый результат: сервер автоматически обновил количество продуктов в ячейке (2) и вес продуктов (1) в ячейке, где id равен 1.

* Название: Добавление одинаковой записи в таблицу «workers\_work\_shifts»

Предварительные условия: открыт Supabase в браузере, в таблице «workers\_work\_shift» существует запись, где id\_worker - 024b4df6-0695-44ac-9d52-c1a3d3be4771, а id\_work\_shift - 1

Шаги:

1. Открыть «SQL Editor»
2. Ввести запрос «insert into public.workers\_work\_shifts (id\_worker, id\_work\_shift) values ('024b4df6-0695-44ac-9d52-c1a3d3be4771', 1)»
3. Нажать кнопку «Run»

Ожидаемый результат: сервер вернет сообщение о том, что «Работник уже записан на эту смену» и не ставит новую запись

* Название: Обновление выполнения работником задачи в таблице «tasks\_workers»

Предварительные условия: открыт Supabase в браузере, в таблице «tasks» есть запись, где id равен 1 и is\_completed - false, в таблице «tasks\_workers» есть запись, где id\_worker – «024b4df6-0695-44ac-9d52-c1a3d3be4771», id\_tasl – 1, is\_worker\_complited – false

Шаги:

1. Открыть «SQL Editor»
2. Ввести запрос «update public.tasks\_workers set is\_worker\_complited = true where id\_worker = '024b4df6-0695-44ac-9d52-c1a3d3be4771' and id\_task = 1»
3. Нажать кнопку «Run»
4. Открыть в «Table Editor» таблицу «tasks»

Ожидаемый результат: сервер автоматически изменит is\_completed на true, где id\_task равен 1

* Название: Обновление количества продуктов в ячейке при выполнении задачи

Предварительные условия: открыт Supabase в браузере, в таблице «tasks» есть запись, где id равен 4 и is\_completed – false, в таблице «tasks\_products» есть запись, где id\_task – 4, id\_product – 1, id\_cell – 1, count\_product -–2, в таблице «cells» id – 1, id\_product – 1, count\_product\_in\_cell - 2

Шаги:

1. Открыть «SQL Editor»
2. Выполнить запрос «update public.tasks set is\_completed = true where id = 4»
3. Нажать кнопку «Run»
4. Открыть в «Table Editor» таблицу «cells»

Ожидаемый результат: сервер автоматически изменил количество продуктов в ячейке на 0 (он из 2 вычел количество продуктов в таблице «tasks\_producta»)

1. **Unit-тесты**

* Тесты авторизации пользователей

| № | | Название теста | | Описание | Ожидаемый результат | |
| --- | --- | --- | --- | --- | --- | --- |
| 11 | | Проверка существования пользователя | | Проверяет, что пользователь с номером телефона 79290509332 и зашифрованным паролем $2a$10$V6z25.1LqNpQNsa6Vomgp.m8U/.UC743rp1FJ3yhvxtweqaV.FEDO существует в базе данных. | id должен быть равен 4202d2e7-5164-4cd1-aa16-e38815846b7c. | |
| SELECT is(id, '4202d2e7-5164-4cd1-aa16-e38815846b7c', 'id должен быть равен 4202d2e7-5164-4cd1-aa16-e38815846b7c')  FROM auth.users  WHERE phone = '79290509332'  AND encrypted\_password ='$2a$10$V6z25.1LqNpQNsa6Vomgp.m8U/.UC743rp1FJ3yhvxtweqaV.FEDO'; | | | | | | |
| 2 | | Проверка пустого результата запроса | | Проверяет, что запрос с пустыми значениями номера телефона и пароля возвращает пустой результат. | Запрос должен вернуть пустой результат. | |
| SELECT is\_empty(    'SELECT id FROM auth.users WHERE phone = '' AND encrypted\_password = ''',    'Запрос должен быть пустым'); | | | | | | |
| 3 | Проверка отсутствия пользователя | | Проверяет, что пользователь с номером телефона 79290509329 и зашифрованным паролем $2a$10$/4dEFK05Rq/zJNinADjvVuUdRGMONL7keKfROj6xWIsW67Z4ocAiO не существует в базе данных. | | | Запрос не должен вернуть id 4202d2e7-5164-4cd1-aa16-e38815846b7c. |

|  |
| --- |
| SELECT isnt(id, '4202d2e7-5164-4cd1-aa16-e38815846b7c', 'id НЕ должен быть равен 4202d2e7-5164-4cd1-aa16-e38815846b7c')  FROM auth.users  WHERE phone = '79290509329'  AND encrypted\_password ='$2a$10$/4dEFK05Rq/zJNinADjvVuUdRGMONL7keKfROj6xWIsW67Z4ocAiO'; |
| Результаты тестов |
|  |

* Тесты выборки данных

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название теста | Описание | Ожидаемый результат |
| 1 | Проверка выборки работников на смену | Проверяет, что сервер вернет список работников, которые должны прийти «30-10-2024» на склад с id 1 | Сервер вернет список, состоящий из 3 записей |
| SELECT is(CAST(count(\*) as int), 3, 'Запрос должен вернуть 3 строки')  FROM public.workers\_work\_shifts  WHERE id\_work\_shift = (SELECT id FROM public.work\_shifts WHERE date\_shift = '2024-10-30' AND id\_warehouse = 1); | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Название теста | Описание | | | Ожидаемый результат |
| 2 | Проверка выборки задач пользователя | Проверяет, что сервер вернет список задач пользователя, которые должны будут выполнены «30-10-2024» | | | Сервер вернет список, состоящий из 3 записей |
| SELECT is(CAST(count(\*) as int), 3, 'Запрос должен вернуть 3 строки')  FROM public.tasks\_workers  WHERE id\_worker = '024b4df6-0695-44ac-9d52-c1a3d3be4771'  AND id\_task IN (      SELECT id      FROM public.tasks      WHERE date\_trunc('day', date\_execution\_task) = '2024-10-30'); | | | | | |
| 3 | Проверка выборки ячеек из задачи | | Проверяет, что сервер вернет список ячеек, которые записаны в задаче с id - 4 | Сервер вернет список, состоящий из 2 записей | |
| SELECT is(CAST(count(\*) as int), 2, 'Запрос должен вернуть 2 строки')  FROM public.cells  WHERE id IN (      SELECT id\_cell      FROM public.tasks\_products      WHERE id\_task = 4); | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название теста | Описание | Ожидаемый результат |
| 4 | Проверка выборки продуктов из задачи | Проверяет, что сервер вернет список продуктов, которые записаны в задаче с id 4 | Сервер вернет список, состоящий из 2 записей |
| SELECT is(CAST(count(\*) as int), 2, 'Запрос должен вернуть 2 строки')  FROM public.products  WHERE id IN (      SELECT id\_product      FROM public.tasks\_products      WHERE id\_task = 4); | | | |
| Результаты тестов | | | |
|  | | | |

* Тесты триггеров

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | | Название теста | | Описание | | Ожидаемый результат |
| 1 | | Проверка возврата функции check\_worker  \_on\_shift | | Проверяет, что в БД функция check\_worker  \_on\_shift возвращает trigger | | Сервер не вернет исключение |
| SELECT function\_returns('check\_worker\_on\_shift', 'trigger'); | | | | | | |
| 2 | | Проверка возврата функции task\_update\_cells | | Проверяет, что в БД функция task\_update\_cells возвращает trigger | | Сервер не вернет исключение |
| SELECT function\_returns('task\_update\_cells', 'trigger'); | | | | | | |
| № | | Название теста | | Описание | | Ожидаемый результат | |
| 3 | | Проверка возврата функции update\_task\_completed | | Проверяет, что в БД функция update\_task\_completedвозвращает trigger | | Сервер не вернет исключение | |
| SELECT function\_returns('update\_task\_completed', 'trigger'); | | | | | | | |
| 4 | | Проверка возврата функции update\_weight\_product\_in\_cell | | Проверяет, что в БД функция task\_update\_cells возвращает trigger | | Сервер не вернет исключение | |
| SELECT function\_returns('update\_weight\_product\_in\_cell', 'trigger'); | | | | | | | |
| Результаты тестов | | | | | | | |
|  | | | | | | | |

**РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ**

1. **Разработка экранов**

* Authorization – экран авторизации

Функционал:

1. Ввод телефона и пароля работника
2. Скрытие и показ пароля
3. Авторизация работника в системе через кнопку «Вход» (Если авторизация неуспешная или отсутствует интернет, то приложение уведомит работника об этом через Toast с текстом ошибки)

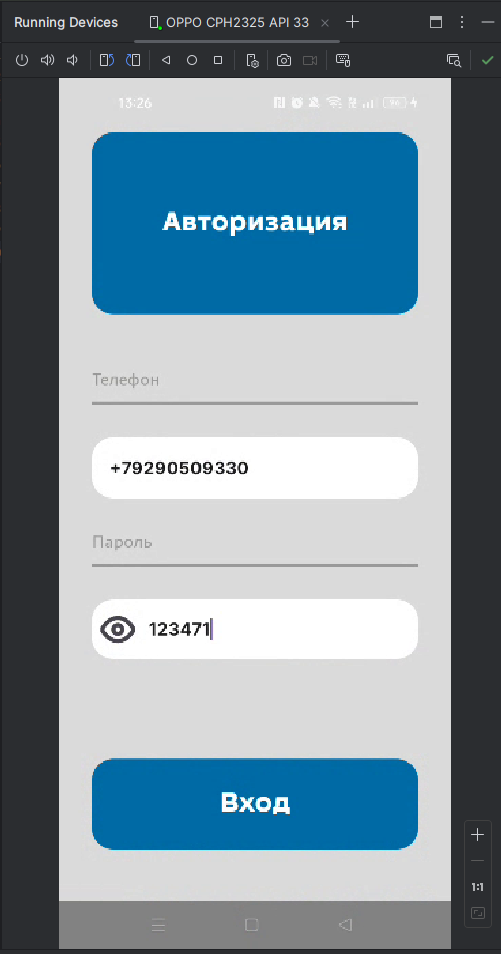


Рисунок 10. Экран авторизации

* Журнал посещаемости

Функционал:

1. Отображение работников, которые должны прийти конкретно на сегодняшнюю смену
2. Отметить тех, кто из работников пришел
3. Переход к задачам через кнопку «К задачам» (Перед переходом приложение спросит работника, уверен ли он в своих отметках, если да, то произойдет переход)

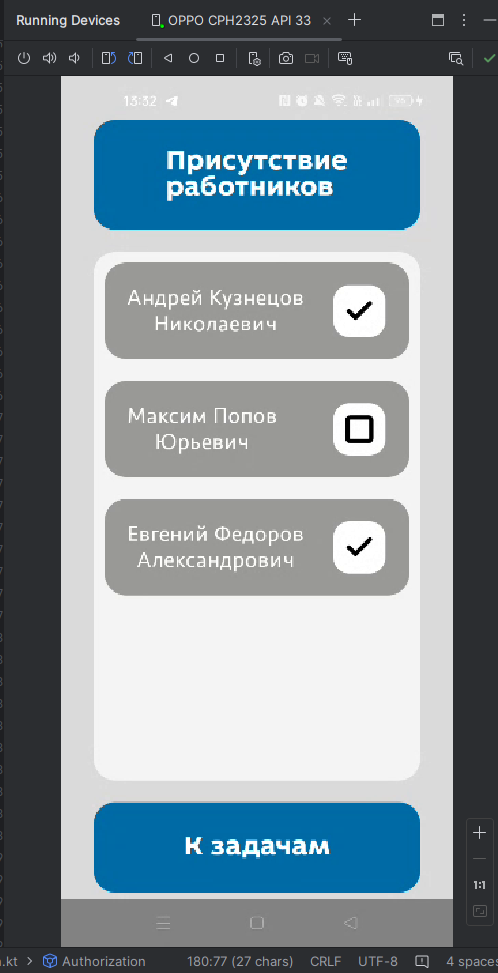


Рисунок 11. Журнал посещаемости (1)

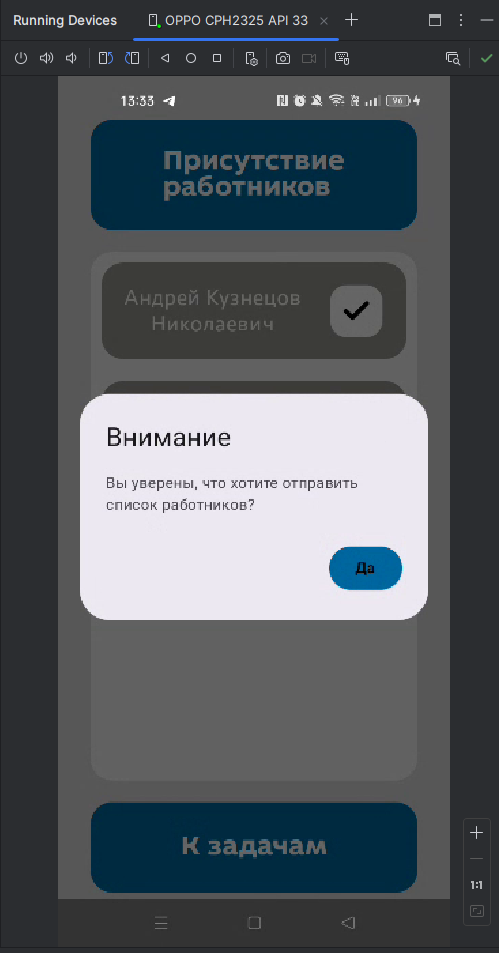


Рисунок 12. Журнал посещаемости (2)

* Задачи

Функционал:

1. Отображение задач конкретно на сегодняшний день в зависимости от роли работника
2. Таймер до выполнения следующей ближайшей задачи
3. Смена темы приложения
4. Переход к информации о задаче
5. Написать сообщение другому работнику

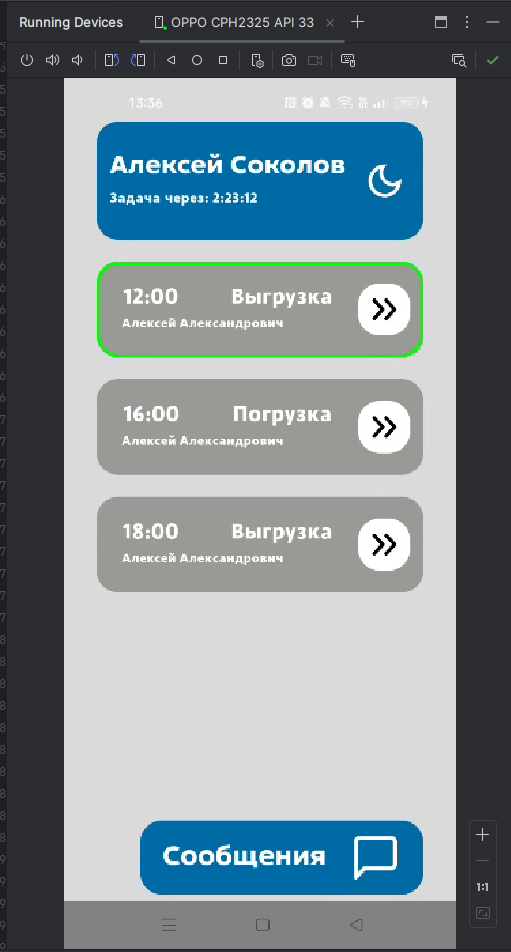


Рисунок 13. Задачи (1)

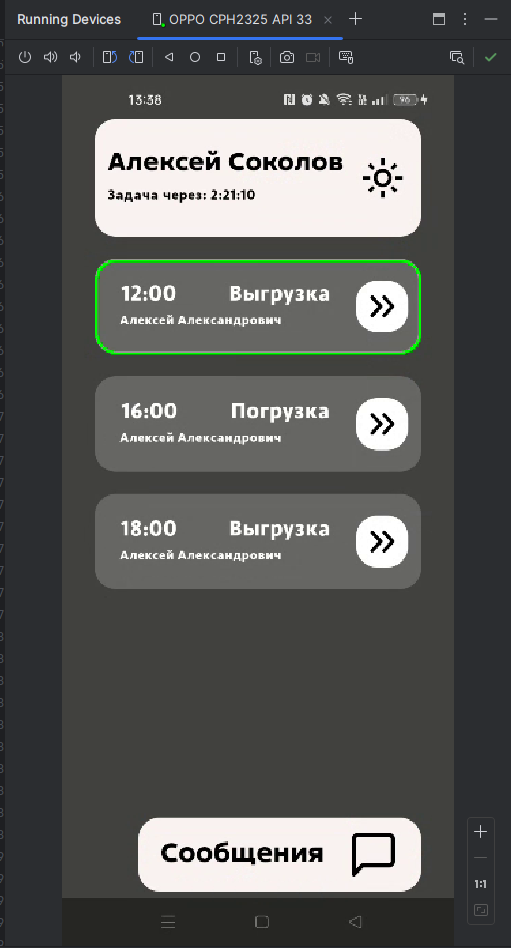


Рисунок 14. Задачи (2)

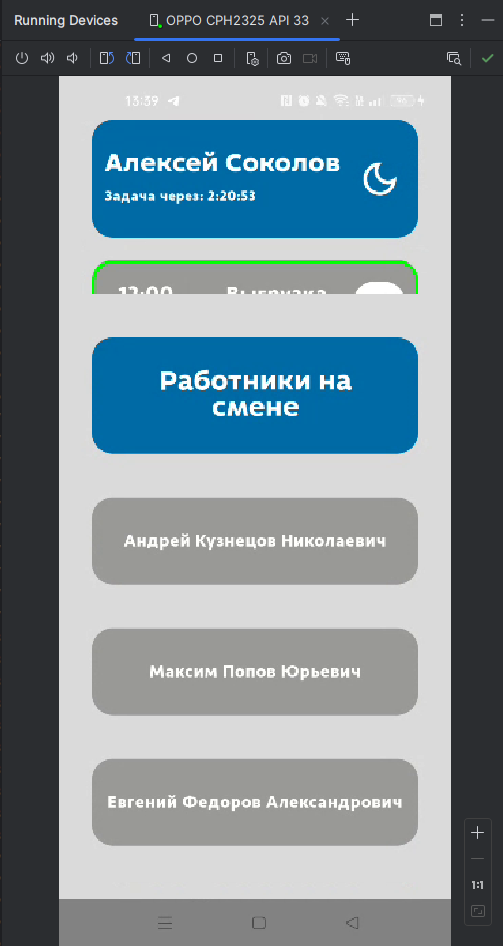


Рисунок 15. Задачи (3)

* Задача погрузки/выгрузки

Функционал:

1. Отображение информации задачи
2. Просмотри оптимального пути через кнопку «Посмотреть путь схематично»
3. Написать сообщению работнику относительно роли вошедшего в систему
4. Просмотр информации конкретной ячейки через элемент списка «Товар - Ячейка»

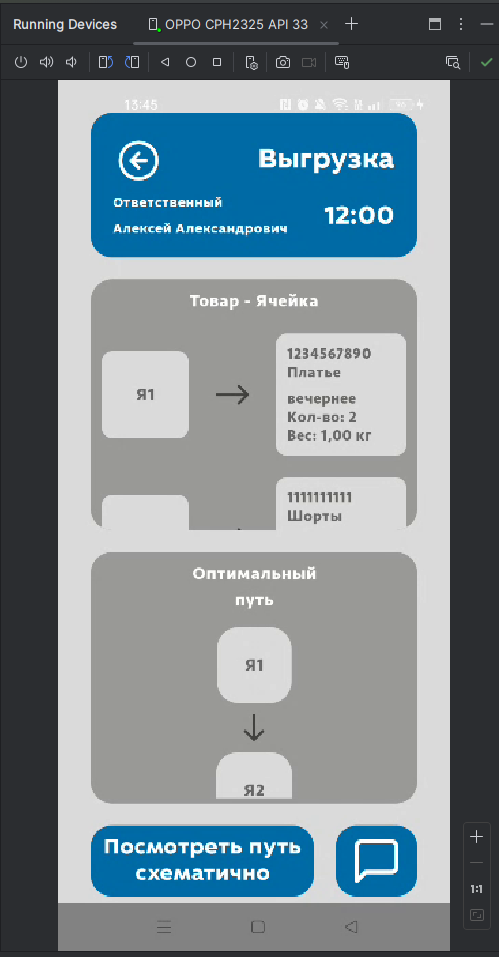


Рисунок 16. Задача выгрузки

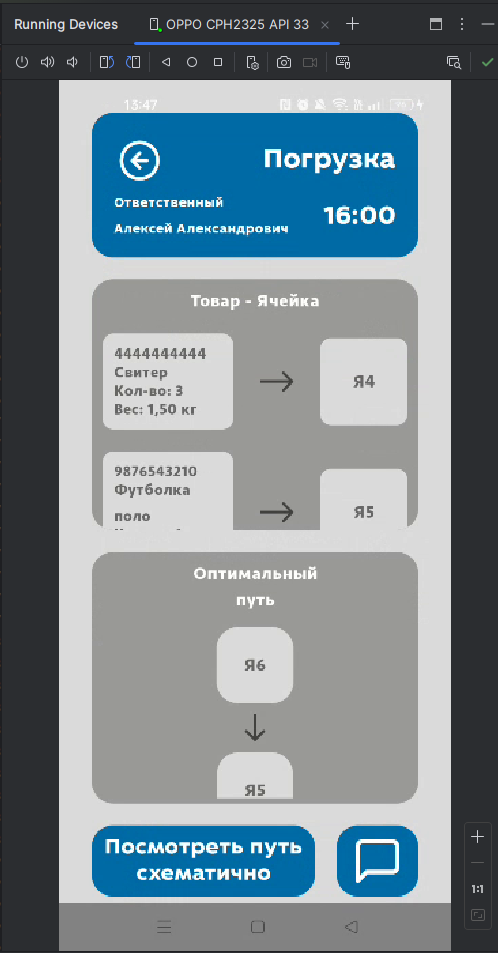


Рисунок 17. Задача погрузки

* Ячейка

Функционал:

1. Просмотр данных о ячейке



Рисунок 18. Экран ячейки

* Чат

Функционал:

1. Отправка сообщений другому работнику

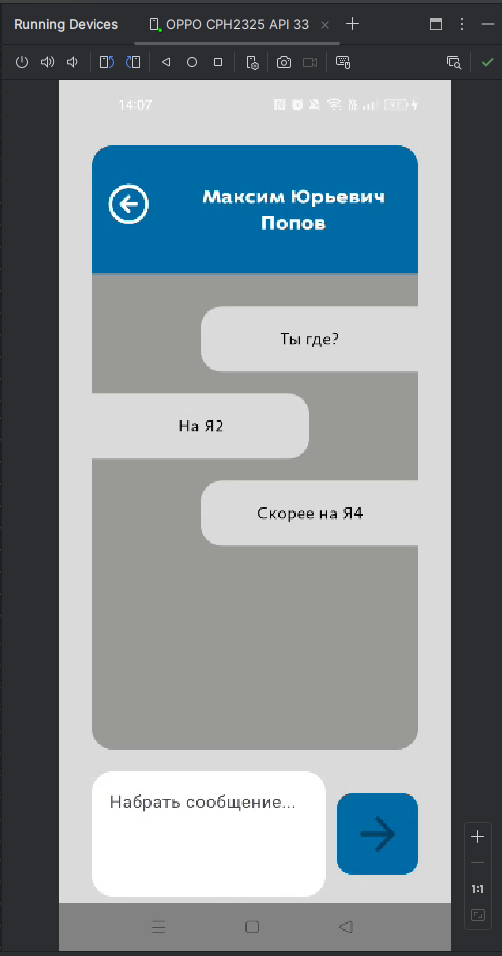


Рисунок 19. Экран чата

* Оптимальный путь

Функционал:

1. Просмотр изображения оптимального пути задачи

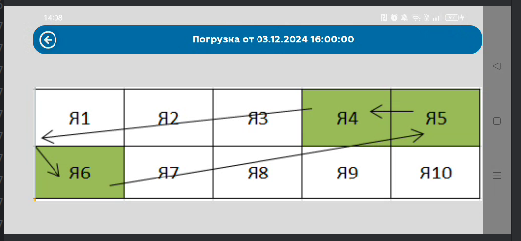


Рисунок 20. Экран оптимального пути

1. **Подключение библиотек, Supabase.**

Для работоспособности приложения были подключены следующие библиотеки

Листинг 1. Библиотеки

|  |
| --- |
| *implementation*(*libs*.*androidx*.*core*.*ktx*) *implementation*(*libs*.*androidx*.*lifecycle*.*runtime*.*ktx*) *implementation*(*libs*.*androidx*.*activity*.*compose*) *implementation*(platform(*libs*.*androidx*.*compose*.*bom*)) *implementation*(*libs*.*androidx*.*ui*) *implementation*(*libs*.*androidx*.*ui*.*graphics*) *implementation*(*libs*.*androidx*.*ui*.*tooling*.*preview*) *implementation*(*libs*.*androidx*.*material3*) *testImplementation*(*libs*.*junit*) *androidTestImplementation*(*libs*.*androidx*.*junit*) *androidTestImplementation*(*libs*.*androidx*.*espresso*.*core*) *androidTestImplementation*(platform(*libs*.*androidx*.*compose*.*bom*)) *androidTestImplementation*(*libs*.*androidx*.*ui*.*test*.*junit4*) *debugImplementation*(*libs*.*androidx*.*ui*.*tooling*) *debugImplementation*(*libs*.*androidx*.*ui*.*test*.*manifest*) //retrofit *implementation*("com.squareup.retrofit2:retrofit:2.9.0") *implementation*("com.squareup.retrofit2:converter-gson:2.9.0") //supabase lib *implementation*(platform(*libs*.*bom*)) *implementation*(*libs*.*postgrest*.*kt*) *implementation*(*libs*.*storage*.*kt*) *implementation*("io.github.jan-tennert.supabase:realtime-kt:3.0.2") *implementation*(*libs*.*auth*.*kt*) *implementation*(*libs*.*ktor*.*client*.*android*) *implementation*(*libs*.*ktor*.*client*.*core*) *implementation*(*libs*.*ktor*.*utils*) //hilt  *implementation*("com.google.dagger:hilt-android:2.52") *kapt*("com.google.dagger:hilt-compiler:2.52") *implementation*("androidx.hilt:hilt-navigation-compose:1.2.0")  //coil *implementation*("io.coil-kt.coil3:coil-compose:3.0.3") *implementation*("io.coil-kt.coil3:coil-network-okhttp:3.0.3") //test *testImplementation* ("org.jetbrains.kotlinx:kotlinx-coroutines-test:1.9.0") *testImplementation*("io.mockk:mockk:1.13.5") |

Подключение к Supabase было реализовано через модули hilt

Листинг 2. Подключение к Supabase

|  |
| --- |
| package com.example.warehouseemployee.data.objects  import com.example.warehouseemployee.BuildConfig import dagger.Module import dagger.Provides import dagger.hilt.InstallIn import dagger.hilt.components.SingletonComponent import io.github.jan.supabase.SupabaseClient import io.github.jan.supabase.auth.Auth import io.github.jan.supabase.auth.FlowType import io.github.jan.supabase.auth.*auth* import io.github.jan.supabase.createSupabaseClient import io.github.jan.supabase.postgrest.Postgrest import io.github.jan.supabase.postgrest.*postgrest* import io.github.jan.supabase.realtime.Realtime import io.github.jan.supabase.storage.Storage import io.github.jan.supabase.storage.*storage* import javax.inject.Singleton  @InstallIn(SingletonComponent::class) @Module object SupabaseContext {   @Provides  @Singleton  fun provideSupabaseClient(): SupabaseClient {  return *createSupabaseClient*(  supabaseUrl = BuildConfig.*SUPABASE\_URL*,  supabaseKey = BuildConfig.*SUPABASE\_ANON\_KEY* ) **{** install(Postgrest)  install(Auth) **{** flowType = FlowType.*PKCE* scheme = "app"  host = "supabase.com"  **}** install(Storage)  install(Realtime)  **}** }  //Провайдер постгри  @Provides  @Singleton  fun provideSupabaseDatabase(client: SupabaseClient): Postgrest {  return client.*postgrest* }  //Провайдер авторизации  @Provides  @Singleton  fun provideSupabaseAuth(client: SupabaseClient): Auth {  return client.*auth* }  //Провайдер хранилища  @Provides  @Singleton  fun provideSupabaseStorage(client: SupabaseClient): Storage {  return client.*storage* }  } |

1. **Логика приложения**

Приложение создавалось по паттерну MVVM, следовательно вся логика была прописана в ViewModel и репозиториях. Каждый экран имеет свою ViewModel. Репозитории разделялись по объекту работы, т.е. в одном хранились запросы, связанные только с задачами, другой связан только с рабочими.

Полным код можно посмотреть в репозитории на Gogs по ссылке: <http://gogs.ngknn.ru:3000/CoolVeryVery/MobileSoftwareWarehouseEmployee.git>

**ТЕСТИРОВАНИЕ МОБИЛЬНОГО ПРИЛОЖЕНИЯ**

Перед тестированием был составлен тест-план

1) Цели и задачи тестирования

1. Цели

• Проверка работоспособности мобильного приложения

• Проверка работоспособности авторизации

• Проверка работоспособности переходов между экранами

• Проверка работоспособности инициализации свойств во ViewModels

2. Задачи

• Убедиться в работе приложения Warehouseemployee

• Убедиться в корректном механизме авторизации в приложении Warehouseemployee

• Убедиться в правильной инициализации свойств во ViewModels приложения Warehouseemployee

• Убедиться в правильном переходе между экранами в приложении Warehouseemployee

2) Объекты тестирования

1. Экран Authorization

2. Экран VisitingWorkers

3. Экран TasksWorker

4. ViewModel AuthorizationViewModel

5. ViewModel TasksWorkerViewModel

6. ViewModel MessagesViewModel

7.

3) Методы тестирования

1. Unit-тесты – метод белого ящика

2. UI-тесты – метод черного ящика

4) Ресурсы и инструменты тестирования

Для тестирования необходим один тестировщик, ПК, полностью созданное и функционирующее приложение Warehouseemployee, заполненные таблицы БД данными. Тестирование будет происходить в «Android Studio»

5) Планирование и оценка времени

Функциональное тестирование:

Начало: 27.11.2024

Конец: 29.11.2024

6) Критерии завершения тестирования:

1. Все тестовые случаи, определенные в тестовых наборах, успешно пройдены.

2. Все обнаруженные ошибки исправлены, а их исправления проверены.

После были составлены тестовые наборы, тест-кейсы, unit-тесты, ui-тесты

1. **Тестовые наборы**
   1. **UI-тесты**
2. Экран Authorization

|  |  |
| --- | --- |
| Тест-кейсы | |
| Название | Ожидаемый результат |
| Отображение экрана Authorization | Приложение отобразило экран Authorization |
| Отображение сообщения о пустых полях ввода на экране | Приложение отобразило сообщение, что поля ввода пустые |
| Отображение сообщения о некорректных данных для авторизации | Приложение отобразило сообщение, что телефон или пароль неверный |

1. Экран VisitingWorkers

|  |  |
| --- | --- |
| Тест-кейсы | |
| Название | Ожидаемый результат |
| Отображение экрана VisitingWorkers после успешной авторизации главного по смене на экране Authorization | Приложение отобразило экран VisitingWorkers |

1. Экран TasksWorker

|  |  |
| --- | --- |
| Тест-кейсы | |
| Название | Ожидаемый результат |
| Отображение экрана TasksWorker после успешной авторизации работника на экране Authorization | Приложение отобразило экран TasksWorker |

**2.1. Unit-тесты**

1. AuthorizationViewModel

|  |  |
| --- | --- |
| Тест-кейсы | |
| Название | Ожидаемый результат |
| Работоспособность метода onPhoneChange | Метод onPhoneChange положило значение в свойство VM |
| Работоспособность метода onPasswordChange | Метод onPasswordChange положило значение в свойство VM |
| Инициализация свойства navigateTo | Свойство navigateTo инициализировалось как null |

2. TasksWorkerViewModel

|  |  |
| --- | --- |
| Тест-кейсы | |
| Название | Ожидаемый результат |
| Инициализация свойства taskList | Свойство taskList инициализировалось пустым списком |

3. MessagesViewModel

|  |  |
| --- | --- |
| Тест-кейсы | |
| Название | Ожидаемый результат |
| Инициализация свойства isError | Свойство isError инициализировалось как false |

1. Тест-кейсы

* Название: Отображение экрана Authorization

Шаги:

1. Открыть «Android Studio»

2. Открыть файл «ExampleInstrumentedTest»

3. Напротив функции Authorization\_IsDisplayed нажать зеленый треугольник (кнопка запуска теста)

Ожидаемый результат: функция вернет то, что экран Authorization отобразился

* Название: Отображение сообщения о пустых полях ввода на экране

Шаги:

1. Открыть «Android Studio»

2. Открыть файл «ExampleInstrumentedTest»

3. Напротив функции Authorization\_Toast\_EmptyTextFolder\_IsDisplayed нажать зеленый треугольник (кнопка запуска теста)

Ожидаемый результат: функция вернет то, что на экран Authorization отобразился сообщение о пустых полях

* Название: Отображение сообщения о некорректных данных для

Шаги:

1. Открыть «Android Studio»

2. Открыть файл «ExampleInstrumentedTest»

3. Напротив функции Authorization\_Toast\_IncorrectSignIn\_IsDisplayed нажать зеленый треугольник (кнопка запуска теста)

Ожидаемый результат: функция вернет то, что на экране Authorization отобразилось сообщение о том, что телефон или пароль неверный

* Название: Отображение экрана VisitingWorkers после успешной авторизации главного по смене на экране

Шаги:

1. Открыть «Android Studio»

2. Открыть файл «ExampleInstrumentedTest»

3. Напротив функции Authorization\_SignIn\_To\_VisitingWorkers нажать зеленый треугольник (кнопка запуска теста)

Ожидаемый результат: функция вернет то, что экран VisitingWorkers отобразился после успешной авторизации на экране Authorization

* Название: Отображение экрана TasksWorker после успешной авторизации работника на экране Authorization

Шаги:

1. Открыть «Android Studio»

2. Открыть файл «ExampleInstrumentedTest»

3. Напротив функции Authorization\_SignIn\_To\_TaskWorker нажать зеленый треугольник (кнопка запуска теста)

Ожидаемый результат: функция вернет то, что экран TasksWorker отобразился после успешной авторизации на экране Authorization

* Название: Работоспособность метода onPhoneChange

Шаги:

1. Открыть «Android Studio»

2. Открыть файл «ExampleUnitTest »

3. Напротив функции TestMethod\_AuthorizationViewModel\_onPhoneChange нажать зеленый треугольник (кнопка запуска теста)

Ожидаемый результат: функция вернет то, что значение поля в VM совпало с ожидаемым значением

* Название: Работоспособность метода onPasswordChange

Шаги:

1. Открыть «Android Studio»

2. Открыть файл «ExampleUnitTest »

3. Напротив функции TestMethod\_AuthorizationViewModel\_onPasswordChange нажать зеленый треугольник (кнопка запуска теста)

Ожидаемый результат: функция вернет то, что значение поля в VM совпало с ожидаемым значением

* Название: Инициализация свойства navigateTo

Шаги:

1. Открыть «Android Studio»

2. Открыть файл «ExampleUnitTest »

3. Напротив функции TestProperty\_AuthorizationViewModel\_NavigateToValue нажать зеленый треугольник (кнопка запуска теста)

Ожидаемый результат: функция вернет то, что в свойстве navigateTo лежит null

* Название: Инициализация свойства taskList

Шаги:

1. Открыть «Android Studio»

2. Открыть файл «ExampleUnitTest »

3. Напротив функции TestProperty\_TasksWorkerViewModel\_TaskListValuee нажать зеленый треугольник (кнопка запуска теста)

Ожидаемый результат: функция вернет то, что в свойстве taskList лежит пустой список

* Название: Инициализация свойства isError

Шаги:

1. Открыть «Android Studio»

2. Открыть файл «ExampleUnitTest »

3. Напротив функции TestProperty\_MessagesViewModel\_isErrorValue нажать зеленый треугольник (кнопка запуска теста)

Ожидаемый результат: функция вернет то, что в свойстве isError лежит false

1. UI-тесты

1. Экран Authorization

| № | Название теста | Описание | Ожидаемый результат |
| --- | --- | --- | --- |
| 1 | Отображение экрана Authorization | Проверяет, что приложение отображает экран Authorization | Приложение отобразило экран Authorization. |
| @Test fun Authorization\_IsDisplayed() {  composeTestRule.setContent {  val auth: Auth = client.auth  val postgrest: Postgrest = client.postgrest  val workerRep: WorkerRepository = WorkerRepositoryImpl(postgrest)  val authRep: AuthenticationRepository = AuthenticationRepositoryImpl(auth)  authVM = AuthorizationViewModel(authRep, workerRep)  TestNavigation(AuthorizationDestination.route)  }  composeTestRule.onNodeWithText("Авторизация").assertIsDisplayed() } | | | |
| 2 | Отображение сообщения о пустых полях ввода на экране | Проверяет, что приложение отображает на экране Authorization сообщение о пустых текстовых полях | Приложение отобразило сообщение, что поля ввода пустые |
| @Test fun Authorization\_Toast\_EmptyTextFolder\_IsDisplayed() {  composeTestRule.setContent {  val auth: Auth = client.auth  val postgrest: Postgrest = client.postgrest  val workerRep: WorkerRepository = WorkerRepositoryImpl(postgrest)  val authRep: AuthenticationRepository = AuthenticationRepositoryImpl(auth)  authVM = AuthorizationViewModel(authRep, workerRep)  TestNavigation(AuthorizationDestination.route)  }  composeTestRule.onNodeWithText("Вход").performClick()  composeTestRule.onNodeWithText("Поля не должны быть пустыми").isDisplayed() } | | | |
| 3 | Отображение сообщения о некорректных данных для авторизации | Проверяет, что приложение отображает на экране Authorization сообщение о неверном телефоне или пароле | Приложение отобразило сообщение, что телефон или пароль неверный. |
| @Test fun Authorization\_Toast\_IncorrectSignIn\_IsDisplayed() {  composeTestRule.setContent {  val auth: Auth = client.auth  val postgrest: Postgrest = client.postgrest  val workerRep: WorkerRepository = WorkerRepositoryImpl(postgrest)  val authRep: AuthenticationRepository = AuthenticationRepositoryImpl(auth)  authVM = AuthorizationViewModel(authRep, workerRep)  authVM.phoneValue = "+3219"  authVM.passwordValue = "123460"  TestNavigation(AuthorizationDestination.route)  }  composeTestRule.onNodeWithText("Вход").performClick()  composeTestRule.onNodeWithText("Неверный телефон или пароль").isDisplayed() } | | | |
| Результаты тестов | | | |
|  | | | |

|  |
| --- |
|  |

2. Экран VisitingWorkers

| № | Название теста | Описание | Ожидаемый результат |
| --- | --- | --- | --- |
| 1 | Отображение экрана VisitingWorkers после успешной авторизации главного по смене на экране Authorization | Проверяет, что приложение отображает экран VisitingWorkers после успешной авторизации на экране Authorization | Приложение отобразило экран VisitingWorkers |
| @Test fun Authorization\_SignIn\_To\_VisitingWorkers() {  composeTestRule.setContent {  val auth: Auth = client.auth  val postgrest: Postgrest = client.postgrest  val workerRep: WorkerRepository = WorkerRepositoryImpl(postgrest)  val authRep: AuthenticationRepository = AuthenticationRepositoryImpl(auth)  authVM = AuthorizationViewModel(authRep, workerRep)  authVM.phoneValue = "+79290509319"  authVM.passwordValue = "123460"  TestNavigation(AuthorizationDestination.route)   }  composeTestRule.onNodeWithText("Вход").performClick()  composeTestRule.onNodeWithText("К задачам").isDisplayed() } | | | |

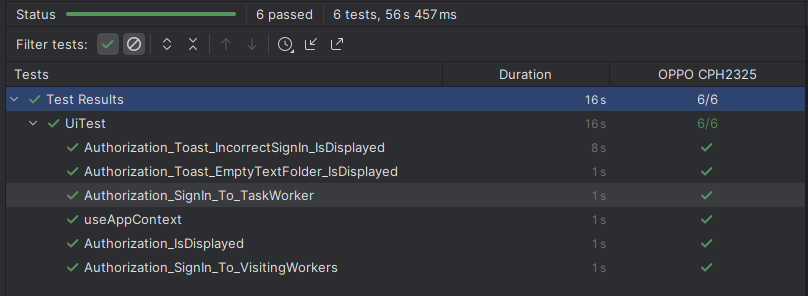
|  |
| --- |
| Результаты тестов |
|  |

3. Экран TasksWorker

| № | Название теста | Описание | Ожидаемый результат |
| --- | --- | --- | --- |
| 1 | Отображение экрана TasksWorker после успешной авторизации главного по смене на экране Authorization | Проверяет, что приложение отображает экран TasksWorker после успешной авторизации на экране Authorization | Приложение отобразило экран TasksWorker |
| @Test fun Authorization\_SignIn\_To\_TaskWorker() {  composeTestRule.setContent {  val auth: Auth = client.auth  val postgrest: Postgrest = client.postgrest  val workerRep: WorkerRepository = WorkerRepositoryImpl(postgrest)  val authRep: AuthenticationRepository = AuthenticationRepositoryImpl(auth)  authVM = AuthorizationViewModel(authRep, workerRep)  authVM.phoneValue = "+79290509325"  authVM.passwordValue = "123466"  TestNavigation(AuthorizationDestination.route)   }  composeTestRule.onNodeWithText("Вход").performClick()  composeTestRule.onNodeWithText("Сообщения").isDisplayed() } | | | |

|  |
| --- |
| Результаты тестов |
|  |

Прохождение всех тестов:



1. UNIT-тесты

1. AuthorizationViewModel

| № | Название теста | Описание | Ожидаемый результат |
| --- | --- | --- | --- |
| 1 | Работоспособность метода onPhoneChange | Проверяет, что приложение записало значение телефона через метод в AuthorizationViewModel | Метод onPhoneChange положило значение в свойство VM |
| @Test fun TestMethod\_AuthorizationViewModel\_onPhoneChange() {  val mockWorkerRepository =  mockk<WorkerRepository>(relaxed = true) // Также заменяем WorkerRepository на mock  val mockAuthRepository =  mockk<AuthenticationRepository>(relaxed = true) // Также заменяем WorkerRepository на mock  authVM = AuthorizationViewModel(mockAuthRepository, mockWorkerRepository)  authVM.onPhoneChange("+79290509325")  assertEquals("+79290509325", authVM.phoneValue) } | | | |
| 2 | Работоспособность метода onPasswordChange | Проверяет, что приложение записало значение пароля через метод в AuthorizationViewModel | Метод onPasswordChange положило значение в свойство VM |
| @Test fun TestMethod\_AuthorizationViewModel\_onPasswordChange() {  val mockWorkerRepository =  mockk<WorkerRepository>(relaxed = true) // Также заменяем WorkerRepository на mock  val mockAuthRepository =  mockk<AuthenticationRepository>(relaxed = true) // Также заменяем WorkerRepository на mock  authVM = AuthorizationViewModel(mockAuthRepository, mockWorkerRepository)  authVM.onPasswordChange("123466")  assertEquals("123466", authVM.passwordValue) } | | | |
| 3 | Инициализация свойства navigateTo | Проверяет, что приложение инициализировало значение свойства navigateTo в AuthorizationViewModel как null | Свойство navigateTo инициализировалось как null |
| @Test fun TestProperty\_AuthorizationViewModel\_NavigateToValue() {  val mockWorkerRepository =  mockk<WorkerRepository>(relaxed = true) // Также заменяем WorkerRepository на mock  val mockAuthRepository =  mockk<AuthenticationRepository>(relaxed = true) // Также заменяем WorkerRepository на mock  authVM = AuthorizationViewModel(mockAuthRepository, mockWorkerRepository)  assertEquals(null, authVM.navigateToValue) } | | | |

|  |
| --- |
| Результаты тестов |
|  |

2. TasksWorkerViewModel

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название теста | Описание | Ожидаемый результат |
| 1 | Инициализация свойства taskList | Проверяет, что приложение инициализировало значение свойства taskList в TasksWorkerViewModel пустым списком | Свойство taskList инициализировалось пустым списком |

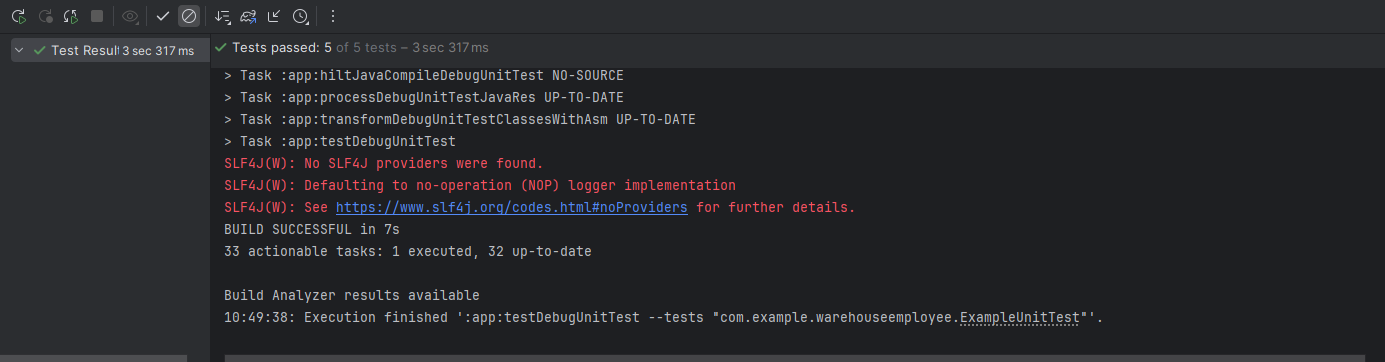
|  |
| --- |
| @Test fun TestProperty\_TasksWorkerViewModel\_TaskListValuee() {  val mockTaskRepository =  mockk<TaskRepository>(relaxed = true) // Также заменяем WorkerRepository на mock  taskVM = TasksWorkerViewModel(mockTaskRepository)  assertEquals(listOf<Task>(), taskVM.taskListValue)  } |
| Результаты тестов |
|  |

3. MessagesViewModel

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название теста | Описание | Ожидаемый результат |
| 1 | Инициализация свойства isError | Проверяет, что приложение инициализировало значение свойства isError в MessagesViewModel как false | Свойство isError инициализировалось как false |
| @Test fun TestProperty\_MessagesViewModel\_isErrorValue() {  val mockMessageRepository =  mockk<MessagesRepository>(relaxed = true) // Также заменяем WorkerRepository на mock  messageVM = MessagesViewModel(mockMessageRepository)  assertEquals(false, messageVM.isErrorValue)  } | | | |

|  |
| --- |
| Результаты тестов |
|  |

Прохождение всех тестов:



**ЗАКЛЮЧЕНИЕ**

В ходе работы были выполнены все поставленные задачи и соблюдены все паттерны разработки мобильного приложения, что упрощает понимание кода и улучшает навыки разработки. Был реализован полноценный дизайн всего приложения, серверная часть с необходимыми триггерами и таблицами и само мобильное приложение, которое корректно отрабатывает исключения, полностью выполняет запланированный функционал.

# ПРИЛОЖЕНИЕ А

**Мобильное приложение для работников склада**

**Актуальность приложения:**

Современный склад – это динамичная система, где от оперативности и точности действий работников зависит эффективность всей логистической цепочки. Мобильное приложение для работника склада становится неотъемлемой частью этой системы, повышая производительность, сокращая ошибки и оптимизируя процессы.

Приложение позволит работникам четко формировать свои задачи и корректнее их выполнять.

Приложение обеспечит точность и прозрачность действий, сводя к минимуму риск ошибок, которые могут привести к потерям товара, задержкам в доставке и финансовым убыткам.

Данные, собираемые приложением, дают возможность эффективнее контролировать работников и выполнение их задач.

**Основное назначение:**

Работник склада сможет просматривать необходимые операции, а именно:

* Во сколько начинается задача
* Какие ячейки склада используются
* Что в ячейках храниться (различная информация о товаре, вес и т.п.)
* Как оптимально пройти по ячейкам
* Просмотреть схематический план склада
* Отметить, что задача была выполнена

Также есть другая роль для главного по смене, управляющего и т.п.

В его функционал входит:

* Смотреть, кто работает на смене
* Отмечать, что работник пришел на смену
* Просматривать задачи
* Какие ячейки склада используются
* Что в ячейках храниться (различная информация о товаре, вес и т.п.)
* Писать работникам сообщения
* Как оптимально пройти по ячейкам
* Просмотреть схематический план склада

**Паттерн проектирования:**

Мобильное приложение будет спроектировано по паттерну MVVM.

MVVM четко разделяет логику приложения, что позволяет без труда разобраться в работе и корректно спроектировать само ПО.

Данный паттерн был выбран из-за ряда преимуществ проектирования через него:

* Разделение ответственности: Позволяет разделить приложение на отдельные части, что облегчает разработку, тестирование и сопровождение.
* Тестируемость: ViewModel можно легко тестировать, так как она не зависит от View.
* Повторное использование кода: ViewModel можно использовать в разных View.
* Чистая архитектура: MVVM помогает создавать приложения, которые легко поддерживать и масштабировать.

**Проектирование требований к серверной части приложения:**

Серверная часть будет развернута на Supabase. В БД должно содержаться все необходимые таблицы, связи, политики и авторизированные пользователи.

Также для корректной работы с сообщениями используется сервис «Resend»

Resend – это сервис для отправки транзакционных email-сообщений, который фокусируется на надежности, доставке и простоте использования. Он позволяет легко интегрировать отправку email-сообщений в приложение и гарантирует, что пользователи получают важные сообщения.

**Проектирование требований к мобильному приложению:**

Мобильное приложение должно иметь адекватный дизайн, но не перегружать интерфейс, чтобы не отвлекать сотрудников от работы. В мобильном приложении должны присутствовать экраны:

* Авторизации (будут рассматривать методы через почту или мобильный телефон)
* Список задач
* Информация конкретной задачи (во сколько выполнить задачу, какие ячейки используются, кто ответственный и прочее)
* Содержание груза в ячейке склада
* Экран оптимального пути по складу
* Схематическое изображение пути по складу
* Диалог с менеджером
* Список работающих сотрудников на смене

Мобильное приложение должно обрабатывать исключительный ситуации, такие как отсутствие интернета, некорректный ввод и другие возможные.

**Выбор для реализации ПО и обоснование выбора.**

1. Для реализации ПО был выбран Android Studio. Android Studio — это бесплатная IDE, которая обеспечивает гибкую и мощную систему сборки, которая позволяет легко управлять зависимостями, создавать разные версии приложения и настраивать процесс сборки. Она предоставляет множество инструментов, оптимизированных для разработки Android-приложений, таких как:

* Android Emulator
* Layout Editor
* Code Editor
* Отладчик

1. Для серверной части была выбрана платформа Supabase. Supabase - это мощный инструмент для разработки backend, который предлагает ряд преимуществ, делающих его привлекательным выбором для разработчиков.

Supabase предаставляет функционал:

* Авторизация и аутентификация пользователей,
* Функции реального времени (Real-time),
* Хранилище объектов
* ORM
* API для работы с данными

Также Supabase имеет открытою и обширную документацию, где показ весь возможный его функционал.

Для проектирования дизайна была выбрана платформа Figma

1. Figma — это мощный инструмент для дизайна, который предлагает богатый набор функций, удобный интерфейс и онлайн-коллаборацию, что делает его идеальным выбором для дизайнеров всех уровней. Его ключевые преимущества заключаются в простоте использования, доступности и гибкости, позволяя создавать высококачественные дизайны веб-сайтов, мобильных приложений, интерфейсов и других проектов. Также Figma предоставляет свой функционал для начинаний беспалтно.
2. Draw.io — это бесплатный онлайн-сервис, который позволяет создавать диаграммы, схемы, прототипы и многое другое.

Draw.io отлично подходит для:

* Создание диаграмм
* Визуализация данных
* Сотрудничество: Работа над диаграммами в режиме реального времени с другими пользователями
* Документация: Создание иллюстрированных документов для различных целей

1. Resend – это сервис для отправки транзакционных email-сообщений, который фокусируется на надежности доставки, простоте использования и удобстве. Он помогает легко интегрировать отправку email в приложение, чтобы пользователи получали важные сообщения, такие как подтверждение регистрации, восстановление пароля, оповещения о новых событиях и даже персонализированные маркетинговые рассылки. Resend использует интеллектуальные алгоритмы для оптимизации доставки, гарантируя, что сообщения дойдут до адресатов. Сервис предоставляет простой API и библиотеки, чтобы интегрировать его с приложением, а также инструменты для отслеживания доставки и диагностики проблем. Resend также предлагает аналитику, чтобы следить за открытием и кликами по сообщениям. Еще одним преимуществом является интеграция с Supabase, что позволяет отправлять email-сообщения прямо из приложения Supabase, автоматизируя процесс и персонализируя сообщения с помощью данных, хранящихся в базе. В целом, Resend, вместе с Supabase, позволяет создавать мощные и надежные системы email-маркетинга, улучшая взаимодействие с пользователями и оптимизируя процесс отправки email-сообщений.