УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«МИНСКИЙ ГОСУДАРСТВЕННЫЙ КОЛЛЕДЖ ЦИФРОВЫХ ТЕХНОЛОГИЙ»

Специальность: 2-39 03 02

«Программируемые мобильные системы»

ОТЧЕТ

ПО ПРЕДДИПЛОМНОЙ ПРАКТИКЕ

Учащийся

группы 75МС курс IV (Л.И.Рыжук)

Руководитель практики (А.А.Шавейко)

Отметка:

Минск, 2025

РЕФЕРАТ

Отчёт по преддипломной практике 62 с., 7 р., 52 рис., 7 табл., 12 источников, 2 приложения.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ХИМЧИСТКА, УСЛУГА, ПОЛЬЗОВАТЕЛЬ, АДМИНИСТРАТОР, ТОВАР, ПОИСК, КОРЗИНА ЗАКАЗОВ, ОФОРМЛЕНИЕ ЗАКАЗА, KOTLIN, ОХРАНА ТРУДА, ЭКОНОМИКА, ЭНЕРГОСБЕРЕЖЕНИЕ.

Объектом исследования является процесс заказа услуги в химчистке.

Целью проекта является разработка мобильного приложения для химчистки, в котором клиент сможет заказать необходимую услугу.

Программа разрабатывалась при помощи Android Studio на языке Kotlin.

Цель работы – разработать мобильное приложение с каталогом услуг, с помощью которого клиент может заказать необходимые ему услуги на нужную дату и время. Заказ клиента должен формироваться автоматически. Предусмотреть личный кабинет клиента: список оформленных услуг. Пользовательский интерфейс приложения должен обеспечивать наглядное, интуитивно понятное представление структуры, размещенной на нем информации, быстрый и логичный переход к разделам и страницам. Модификация разделов и содержимого разделов должна осуществляться посредством администраторского интерфейса.

Результатом работы стало мобильное приложение для организации, занимающейся услугами химчистки, для потенциальных клиентов.

Основные эксплуатационные показатели: простота использования, понятный интерфейс, интуитивное размещение информации.

Степень внедрения – программа используется любым пользователем, который хочет оформить услугу в химчистке.

Эффективность программного средства: позволяет уменьшить время затрат на поиск и заказ необходимой услуги или товара в химчистке.

Автор проекта подтверждает, что приведенный в отчете расчётно-аналитический материал правильно и объективно отражает состояние разрабатываемого объекта, все заимствованные из литературных и других источников теоретические и методологические положения и концепции сопровождаются ссылками на их авторов.

Содержание

[Введение 4](#_Toc164085019)

[1 Объектно-ориентированный анализ и проектирование приложения 6](#_Toc164085020)

[1.1 Назначение и цели создания мобильного приложения 6](#_Toc164085021)

[1.2 Проектирование модели 9](#_Toc164085022)

[2 Проектирование мобильного приложения 13](#_Toc164085023)

[2.1 Требования к мобильному приложению 13](#_Toc164085024)

[2.2 Структура мобильного приложения 13](#_Toc164085025)

[2.3 Проектирование макета мобильного приложения 15](#_Toc164085026)

[2.5 Защита и сохранность данных 20](#_Toc164085027)

[2.6 Организация и ведение информационной базы (модели) 21](#_Toc164085028)

[3 Реализация мобильного приложения 24](#_Toc164085029)

[3.1 Описание используемых функций и процедур 24](#_Toc164085030)

[4 Описание мобильного приложения 28](#_Toc164085031)

[4.1 Общие сведения 28](#_Toc164085032)

[4.2 Описание разделов приложения 28](#_Toc164085033)

[5 Методика испытаний 41](#_Toc164085034)

[5.1 Технические требования 41](#_Toc164085035)

[5.2 Функциональное тестирование 41](#_Toc164085036)

[6 Вопросы охраны труда 46](#_Toc164085037)

[6.1 Охрана труда для офисных работников 46](#_Toc164085038)

[7 Энерго- и ресурсосбережение 52](#_Toc164085039)

[Заключение 54](#_Toc164085040)

[Список использованных источников 55](#_Toc164085041)

[Приложение А Фрагменты текста программы 56](#_Toc164085042)

[Приложение Б Диаграммы UML 60](#_Toc164085043)

ВВЕДЕНИЕ

В современных условиях розничной торговли эффективное взаимодействие между магазинами и поставщиками является критически важным для успешного функционирования всей товаропроводящей цепи. С развитием мобильных технологий все более актуальными становятся специализированные приложения, позволяющие автоматизировать ключевые бизнес-процессы в режиме реального времени как для розничных точек, так и для их поставщиков.

**Предметная область приложения** охватывает комплексную систему управления взаимоотношениями между магазинами и поставщиками. Разрабатываемая платформа включает два взаимосвязанных модуля: для магазинов розничной торговли и для поставщиков товаров.

Для **магазинов** система предоставляет функционал управления складским учетом с группировкой товаров, поиском и фильтрацией; оформления заказов у поставщиков с отслеживанием статусов; управления профилем магазина и генерации сопроводительной документации. Особенностью является возможность как заказа товаров у поставщиков, так и самостоятельного создания товаров на складе.

Для **поставщиков** система включает управление товарными партиями с возможностью массового добавления; обработку входящих заказов от магазинов; контроль остатков товаров; управление профилем поставщика и просмотр отзывов от магазинов.

**Актуальность разработки** обусловлена необходимостью создания единой платформы, которая объединяет розничные магазины и поставщиков, обеспечивая прозрачность цепочки поставок, сокращение времени обработки заказов и минимизацию ошибок при учете товародвижения.

**Целью курсового проекта** является разработка мобильного приложения для работы поставщиков и магазинов, которое упростит магазинам поиск поставщиков и ведение склада, а поставщикам оборот товаров и контроль клиентов.

# 1 Объектно-ориентированный анализ и проектирование приложения

# 1.1 Назначение и цели создания мобильного приложения

В условиях динамично развивающегося рынка розничной торговли и оптовых поставок ключевым фактором конкурентоспособности бизнеса становится скорость, точность и прозрачность всех этапов товародвижения. Традиционные механизмы взаимодействия между магазинами и поставщиками, основанные на телефонных переговорах, обмене электронными письмами и бумажном документообороте, демонстрируют свою неэффективность: они подвержены человеческим ошибкам, приводят к потерям времени на согласование деталей, создают информационный хаос и затрудняют контроль над исполнением обязательств. Особенно остро эти проблемы ощущаются в сегменте малого и среднего бизнеса, где ограниченность ресурсов не позволяет внедрять комплексные, но дорогостоящие и сложные в обслуживании ERP-системы класса SAP или «1С:Предприятие». В результате предприниматели вынуждены работать с устаревшими, неавтоматизированными процессами, что снижает их операционную эффективность, увеличивает логистические издержки и негативно сказывается на общей управляемости бизнеса.

Разрабатываемое мобильное приложение «Помощник поставщикам и магазинам» призвано стать цифровым мостом, преодолевающим эти коммуникационные и операционные разрывы. Оно представляет собой не просто инструмент для учета, а целостную B2B-платформу, спроектированную с учетом реальных потребностей двух основных групп пользователей – розничных торговых точек и оптовых поставщиков. Основное назначение приложения заключается в создании единого, структурированного и безопасного цифрового пространства, в котором все этапы взаимодействия – от поиска товара до получения накладной – становятся интуитивно понятными, автоматизированными и отслеживаемыми в режиме реального времени. Это позволяет трансформировать разрозненные, рутинные операции в слаженный, технологичный бизнес-процесс.

Главной целью создания мобильного приложения является разработка доступного, высокофункционального и удобного решения, которое кардинально упростит и оптимизирует все аспекты взаимодействия между магазинами и поставщиками, что в конечном итоге позволит пользователям сосредоточиться на стратегическом развитии своего бизнеса, а не на решении текущих операционных проблем. Конкретные цели проекта можно детализировать следующим образом:

1. Автоматизация и стандартизация процесса закупок. Ликвидировать хаос в процессе поиска товаров и оформления заказов путем предоставления единого централизованного каталога товарных партий от множества поставщиков. Внедрение интеллектуальных инструментов (расширенный поиск по названию, фильтрация по цене, поставщику) позволит менеджерам по закупкам находить нужные позиции за считанные минуты, сравнивать условия и формировать оптимальные заказы, экономя до 80% времени по сравнению с ручными методами.
2. Цифровизация документооборота и повышение юридической прозрачности исключит ошибки, связанные с ручным заполнением документов, и ускорит бухгалтерское сопровождение операций. Приложение должно автоматически генерировать электронные накладные на основе данных заказа, которые будут доступны для просмотра, сохранения и печати непосредственно в интерфейсе. Это обеспечивает бесшовный переход от коммерческой договоренности к официальному документу, снижая риски недопонимания и конфликтов.
3. Обеспечение сквозной прозрачности и управляемости цепочки поставок. Создать систему, в которой каждый участник в любой момент времени имеет четкое представление о статусе заказа. Для магазина это означает возможность отслеживать этапы: «оформлен», «отправлен», «получен», для поставщика – видеть входящие заявки, планировать сборку и отгрузку, а также контролировать историю взаимодействия с каждым клиентом. Такая прозрачность минимизирует количество уточняющих звонков и повышает уровень доверия между партнерами.
4. Создание интегрированной системы складского и товарного учета. Предоставить магазину не просто журнал заказов, а полноценный инструмент управления товарными запасами. После получения заказа товары автоматически попадают на «виртуальный склад» приложения, где с ними можно выполнять все необходимые операции: устанавливать розничную цену, корректировать остатки, формировать отчеты. Это превращает приложение из инструмента закупок в центральный узел управления торговыми операциями.
5. Формирование среды для обратной связи и повышения качества сервиса. Внедрить репутационный механизм в виде системы отзывов магазинов о работе поставщиков. Это создает здоровую конкурентную среду, где поставщики мотивированы работать качественнее и оперативнее, а магазины получают дополнительный критерий для выбора партнера, основанный на реальном опыте коллег.
6. Обеспечение мобильности и независимости от рабочего места. Дать владельцам бизнеса, управляющим и менеджерам возможность контролировать ключевые процессы в режиме 24/7 с помощью смартфона или планшета. Возможность подтвердить заказ, проверить остатки на складе или посмотреть накладную в любой точке – это новый стандарт операционной гибкости для современного бизнеса.

Детализированный перечень возможностей, предоставляемых приложением:

Для магазинов (розничных точек):

* Прохождение процедуры регистрации и ведение детализированного профиля с указанием реквизитов, адреса, контактов.
* Интуитивный доступ к агрегированному каталогу товарных партий от всех зарегистрированных поставщиков.
* Формирование заказанной корзины с выбором необходимых товаров и указанием требуемых количеств.
* Автоматическое формирование электронной накладной (счет-фактуры) по каждому подтвержденному заказу с возможностью ее просмотра, экспорта в распространенные форматы (PDF) и прямой печати на мобильном принтере.
* Полнофункциональный модуль складского учета: автоматическое оприходование товаров из полученных заказов, ручное добавление позиций (например, от локальных поставщиков), редактирование цен, списание при продажах, контроль текущих остатков в режиме реального времени.
* Централизованная панель управления заказами с полной историей, фильтрацией по статусам («в обработке», «доставляется», «завершены») и датам.
* Возможность оставить структурированный отзыв о работе поставщика с оценкой по ключевым параметрам: скорость обработки заказа, соответствие товара описанию, состояние поставки.

Для поставщиков (оптовых компаний):

* Создание и редактирование корпоративного профиля с размещением информации о компании, ассортименте, условиях работы.
* Расширенный инструментарий для управления товарным каталогом: массовая загрузка партий через CSV-файлы, поштучное добавление с детальным описанием, загрузка качественных фотографий, оперативное обновление цен и остатков.
* Специализированная панель для обработки входящих заказов с мгновенными push-уведомлениями о новых заявках, возможностью подтверждения, изменения состава заказа (по согласованию) и отметки о факте отгрузки.
* Инструменты аналитики для контроля собственных товарных запасов и анализа продаж.
* Раздел для просмотра и анализа отзывов, полученных от магазинов-клиентов, что позволяет проводить работу над ошибками и улучшать сервис.

Таким образом, данное мобильное приложение позиционируется не как очередной учетный модуль, а как стратегическая цифровая платформа для малого и среднего бизнеса в сфере торговли. Она призвана ликвидировать существующие операционные неэффективности, создать новое качество взаимодействия между контрагентами и стать драйвером роста для своих пользователей, что в условиях цифровой трансформации экономики делает проект исключительно актуальным и востребованным.

# Проектирование модели

**Для разрабатываемого мобильного приложения входной информацией будет служить ввод данных поставщиком при добавлении новой товарной партии, при поиске магазином товаров, вносимая магазином информация при регистрации и оформлении заказа.  
 Выходной информацией для данного проекта являются результат добавления новых товарных партий поставщиком, формирование каталога товаров, автоматическое создание накладных, а также формирование истории заказов и инструментов складского учёта для магазина.  
 Данный проект выполняет следующие основные функции:**

* добавление, удаление, редактирование товарных партий поставщиком;
* оформление заказа магазином;
* формирование корзины заказов;
* автоматическая генерация электронных накладных;
* управление складскими остатками и ценами магазином;
* обработка и изменение статуса заказов поставщиком;
* ведение истории операций для обеих сторон;
* система отзывов магазинов о поставщиках.

Функциональная модель представлена на рисунке 1.1.

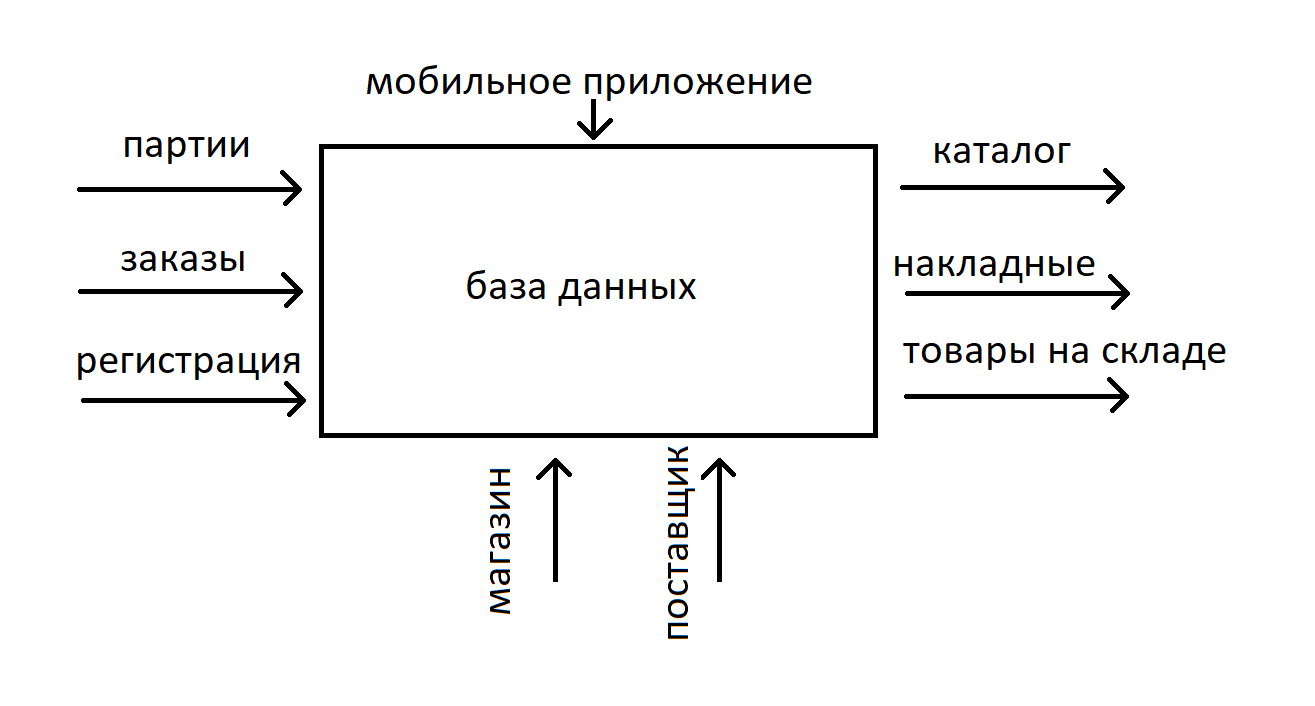


Рисунок 1.1 - функциональная модель

Суть диаграммы вариантов использования заключается в представлении проектируемой системы как набора сущностей или актеров, которые взаимодействуют с системой через различные сценарии использования.

В  сценариях описываются не только взаимодействия между пользователями и сущностями, но также реакции сущностей на получение конкретных сообщений от пользователей и их восприятие за пределами самих сущностей. Кроме того, варианты использования могут включать описание особенностей реализации сервиса и учет различных исключительных ситуаций, таких как обработка ошибок системы. Общее множество вариантов использования должно исчерпывающим образом определять все возможные аспекты ожидаемого поведения системы.

На диаграмме вариантов использования представлены основные функции:

* зарегистрироваться;
* авторизоваться;
* посмотреть поставщиков;
* управлять складом;
* добавить партию товара;
* искать партии;
* оформить заказ;

Диаграмма вариантов использования представлена в графической части на первом листе.

Диаграмма объектов служит для визуализации статической структуры модели системы в рамках объектно-ориентированного программирования. Эта диаграмма может отображать разнообразные взаимосвязи между конкретными экземплярами классов и объектами предметной области, а также предоставляет информацию о их внутренней структуре и типах отношений. Примечательно, что временные аспекты функционирования системы не учитываются на данной диаграмме. Таким образом, диаграмма объектов представляет собой дополнительный этап в развитии концептуальной модели проектируемой системы.

Диаграмма классов для проектируемой системы представлена в графической части на втором листе.

Основные классы проекта:

* + StatelessWidget;
  + StatefulWidget;
  + Scaffold;
  + MaterialApp.

В данной проектируемой системе представлены следующие компоненты:

* + Main.dart
  + Suppliermenu.dart
  + Shopmenu.dart
  + Login.dart
  + Managermenu.dart

# 2 Проектирование мобильного приложения

# 2.1 Требования к мобильному приложению

Разрабатываемое мобильное приложение должно предоставлять пользователю интуитивно понятный и удобный интерфейс для максимального упрощения взаимодействия с программой.

Все элементы в приложении должны быть четко видимы, избегая размытых форм, схожих цветов элементов и фона, чтобы предотвратить напряжение зрения. Создание контрастного окружения не только повышает удобство восприятия информации, но также придает интерфейсу ощущение стабильности и спокойствия, что способствует легкости в понимании и использовании приложения.

Язык интерфейса приложения – русский.

В приложении будет использован стандартный шрифт Roboto.

Дизайн приложения должен быть адаптирован для корректного отображения при следующих разрешениях экрана:

* 768x1024;
* 720x1280;
* 1024x600;
* 1280x752;
* 1280x800.

Разрабатываемое мобильное приложение должно соответствовать гайдлайну Google Material Design.

# 2.2 Структура мобильного приложения

В состав логической структуры входят следующие компоненты.

Первым компонентом структуры является экран авторизации (экран приложения с полями для ввода логина и пароля пользователя).

Вторым компонентом структуры является меню поиска поставок продукции магазином (экран приложения со списком партий товаров).

Третьим компонентом структуры является экран заказов магазина (экран приложения с историей заказов).

Четвертым компонентом структуры является экран аккаунта магазина товара (экран с фотографией, описанием и данными аккаунта).

Пятым компонентом структуры является экран склада магазина (экран приложения где отображаются продукты, полученные магазином).

Шестым компонентом структуры является экран заказов поставщика (экран на котором отображаются заказы, которые должен отправить поставщик).

Седьмым компонентом структуры является экран добавления партии товара поставщиком (экран приложения со всеми партиями товара поставщика и добавлением новых).

Восьмым компонентом структуры приложения является экран редактирования профиля поставщика (экран с фотографией, описанием и данными аккаунта).

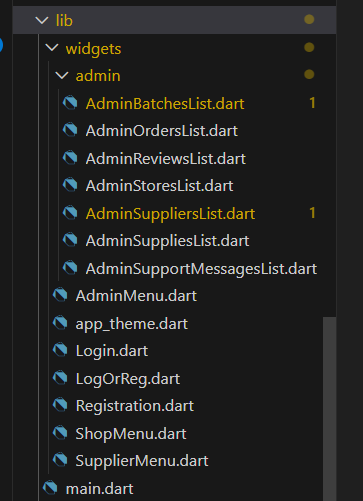
Физическая структура приложения представлена на рисунке 2.1.  


Рисунок 2.1 – Физическая структура приложения

При запуске приложения первым делом загружается стартовый экран, на котором расположены кнопки «Войти» и «Зарегистрироваться». Этот экран является точкой входа для всех пользователей и позволяет сразу определить дальнейший путь работы с системой. Перед началом полноценного использования приложения необходимо пройти процедуру регистрации: пользователь вводит логин, пароль, адрес и описание, а также выбирает тип аккаунта – магазин или поставщик. После завершения регистрации нужно войти в аккаунт, используя логин и пароль. Только после этого становятся доступны все функции, которые соответствуют выбранному типу учетной записи. Таким образом, процесс регистрации и входа обеспечивает персонализацию и разграничение возможностей для разных категорий пользователей.

После входа в приложение в роли магазина пользователь попадает на экран склада. Здесь отображаются все товары, которые принадлежат данному магазину. Пользователь может просматривать подробную информацию о каждом товаре, редактировать описание, изменять количество и характеристики. Также предусмотрена возможность продать часть товара или полностью списать его, если срок годности истек. Это позволяет вести учет и поддерживать актуальность ассортимента. Склад становится центральным инструментом управления товарами, где можно контролировать остатки, следить за сроками и оперативно принимать решения.

Если пользователь переходит в раздел «Поставки», он видит список партий товаров, размещенных поставщиками. Каждая партия сопровождается подробной информацией, включая описание, цену и отзывы других покупателей. Пользователь может изучить ассортимент, сравнить разные предложения и принять решение о заказе. При оформлении заказа предоставляется выбор количества партий, отображается цена за одну партию и итоговая стоимость всего заказа. Такой подход делает процесс закупки прозрачным и удобным, позволяя магазину планировать расходы и формировать ассортимент в соответствии с потребностями.

После оформления заказа пользователь попадает на экран истории заказов. Здесь отображаются все партии, которые уже были доставлены и приняты. По каждому заказу можно просмотреть подробную информацию, включая дату доставки, состав партии и отзывы. Дополнительно доступна функция генерации электронной накладной, готовой к печати, что облегчает документооборот. В истории также отображаются новые поставки, которые только что были заказаны. Пользователь может изучить их детали и при необходимости отменить заказ. Если заявка уже собрана поставщиком и доставлена, магазин нажимает кнопку «Получить». После этого система предлагает назначить цену для поштучной продажи, и товар автоматически перемещается на склад. Таким образом, весь процесс – от заказа до поступления товара на склад – становится последовательным, понятным и удобным для управления.

Структура экранов приложения представлена на рисунке 2.2.

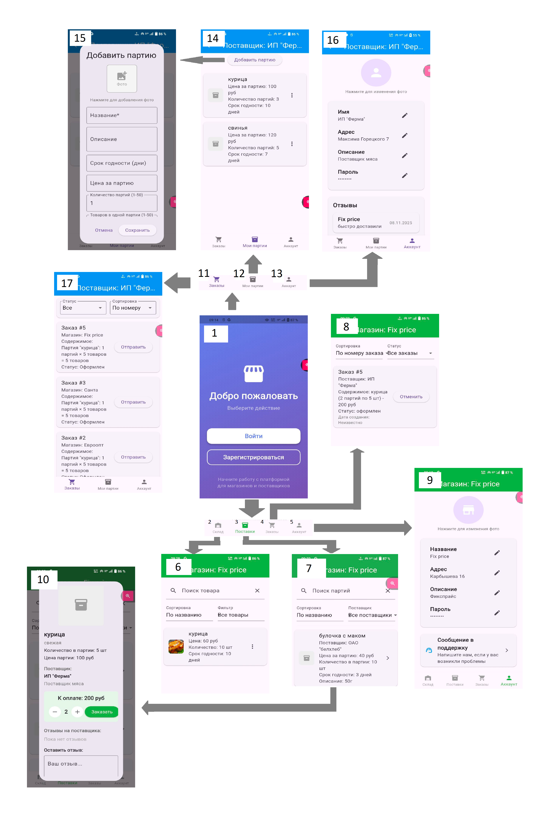


Рисунок 2.2 – Структура экранов

Когда поставщик входит в приложение, на его главном экране сразу же отображаются все новые заказы, которые необходимо подготовить и отправить. Это решение позволяет ему моментально видеть актуальные заявки и без промедления приступать к их обработке, не тратя время на поиск или дополнительные действия. Таким образом, процесс становится максимально удобным и прозрачным: поставщик получает полную картину текущих задач и может сразу организовать работу, распределить ресурсы и ускорить выполнение заказов.

После того как заказ отправлен, поставщику открывается возможность более детально работать с партиями товаров. В приложении он может просматривать уже существующие партии, анализировать их состав, состояние и сроки годности, а также добавлять новые партии для расширения ассортимента. Такой функционал обеспечивает гибкое и удобное управление товарными запасами, позволяет своевременно обновлять складскую информацию и поддерживать актуальность предложений. Благодаря этому поставщик может оперативно реагировать на потребности магазинов, формировать новые предложения и поддерживать высокий уровень обслуживания.

Кроме того, поставщик имеет возможность редактировать свой аккаунт. В настройках он может менять пароль, обновлять имя и корректировать адрес, что дает возможность поддерживать учетные данные в актуальном состоянии. Это особенно важно для тех случаев, когда меняются контактные данные или требуется повысить уровень безопасности учетной записи. Возможность оперативно вносить изменения в личную информацию делает работу с приложением более надежной и удобной, а также обеспечивает гибкость в управлении профилем. В итоге поставщик получает полный набор инструментов для эффективного взаимодействия с системой: от обработки заказов до управления партиями и актуализации учетных данных.

# 2.3 Проектирование макета мобильного приложения

Одним из ключевых этапов в разработке приложения является создание макета. На начальных этапах разработки макеты применяются для формирования прототипа приложения и визуальной демонстрации работы будущего продукта. Прототип представляет собой упрощенное изображение приложения, состоящее из последовательности нарисованных экранов. Внешний вид прототипа аналогичен тому, каким будет разрабатываемое приложение. Прототип должен включать основные элементы управления, эффективно моделировать переходы между экранами, обеспечивая наглядную демонстрацию внешнего вида и функционала будущего приложения.

Дизайн интерфейса мобильного приложения выполнен в стиле Material Design. Material Design – дизайн программного обеспечения и приложений операционной системы Android от компании Google.

Модели экранов были сделаны при помощи веб инструмента Figma.

Экран загрузки содержит:

* логотип приложения;
* название приложения.

Макет экрана загрузки представлен на рисунке 2.3.



Рисунок 2.3 – Макет экрана загрузки

Экран регистрации содержит:

* поля логин, пароль, адрес, описание;
* выбор типа аккаунта
* кнопку «Выход»;
* кнопку «зарегистрироваться»;

Макет экрана регистрации представлен на рисунке 2.4.

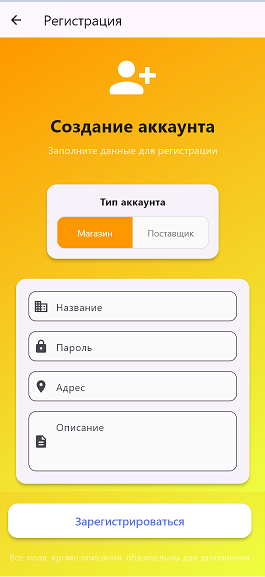


Рисунок 2.5 – Макет экрана регистрации

Экран входа содержит:

* поля логин, пароль;
* выбор типа аккаунта
* кнопку «Вход»;
* кнопку «Выход»;

Макет экрана входа представлен на рисунке 2.5.

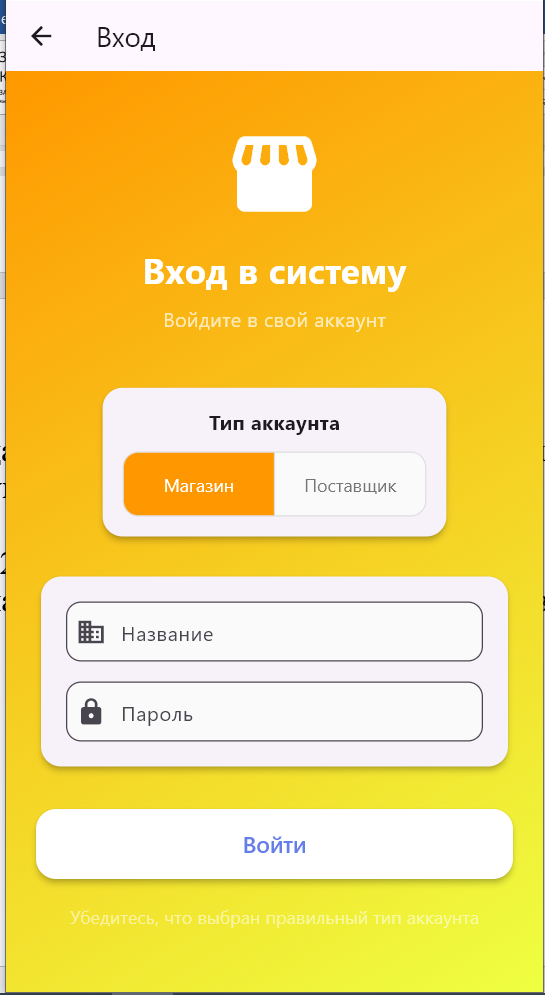


Рисунок 2.5 – экран входа

Главный экран меню магазина содержит содержит:

* поле поиска, сортировки и фильтрации партий;
* партии товаров;
* кнопки меню на нижней панели;
* кнопку «Выход»;

Макет экрана меню магазина представлен на рисунке 2.6.

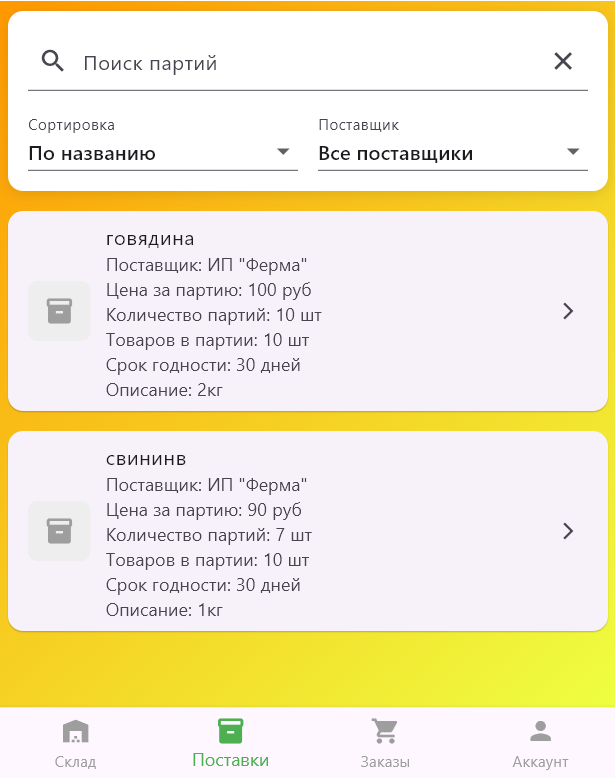


Рисунок 2.6 – заказ партии

Главный экран меню поставщика содержит:

* поле поиска, сортировки и фильтрации партий;
* партии товаров;
* кнопки меню на нижней панели;
* кнопку «Выход»;

Макет экрана меню магазина представлен на рисунке 2.7.

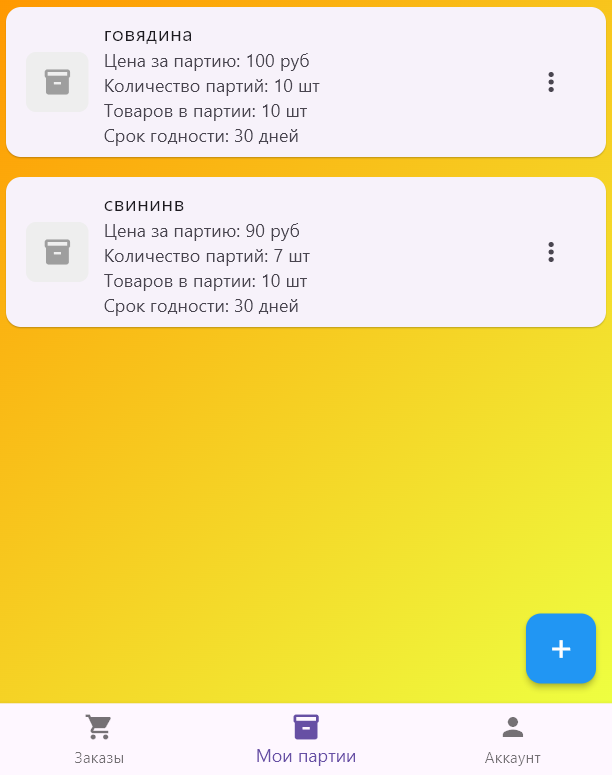


Рисунок 2.7 – макет экрана меню поставщика

Главный экран меню администратора базы данных содержит:

* кнопки для управления сущностями базы данных;
* кнопки для настройки логина и пароля администратора;
* кнопку для настройки подключения к серверу;
* кнопку «Выход»;

Макет экрана меню админа представлен на рисунке 2.8

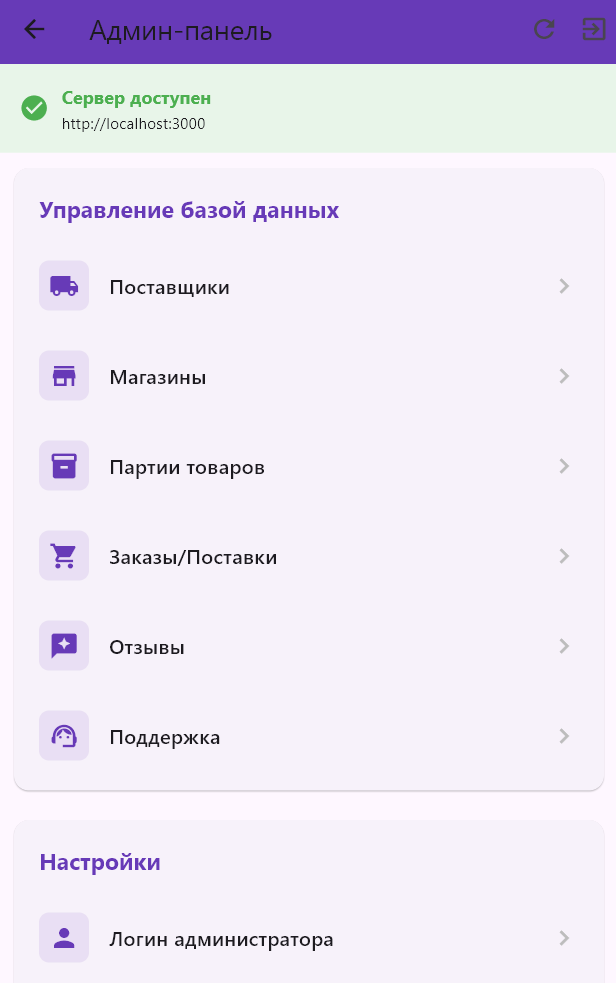


Рисунок 2.8 – маню админа БД

2.4 Программно-технические средства, необходимые для разработки приложения

Инструментами разработки будут являться:

* операционная система Windows 10;
* среда разработки Visual Studio code;
* язык программирования dart;
* язык программирования JavaScript;
* интегрированная среда Android SDK;
* Node.js.
* Prisma ORM;

Для реализации проекта было выбрано современное кроссплатформенное средство разработки Flutter, основанное на языке программирования Dart. Это решение обусловлено его высокой производительностью, богатым набором встроенных компонентов и возможностью создания нативных интерфейсов под Android с минимальными затратами времени. Flutter позволяет разрабатывать приложения с единым кодом, обеспечивая при этом гибкость в дизайне, адаптивность под различные размеры экранов и высокую отзывчивость интерфейса. Благодаря архитектуре, основанной на виджетах, разработчик получает полный контроль над визуальным представлением и логикой взаимодействия, что особенно важно при создании интуитивно понятного интерфейса для пользователей без технической подготовки.

Серверная часть проекта реализована с использованием платформы Node.js – асинхронной, событийно-ориентированной среды выполнения JavaScript, которая обеспечивает высокую производительность и масштабируемость. Node.js идеально подходит для обработки большого количества одновременных запросов, что важно при работе с заказами, регистрацией пользователей и обменом данными между магазинами и поставщиками. В качестве инструмента для взаимодействия с базой данных на сервере используется Prisma ORM (Рисунок 2.9) – современный объектно-реляционный маппер, предоставляющий удобный и типизированный интерфейс для работы с данными. Prisma позволяет описывать модели данных в декларативной форме, автоматически генерировать SQL-запросы, управлять миграциями схемы базы данных и обеспечивать целостность информации. Это значительно упрощает реализацию бизнес-логики, повышает надёжность и ускоряет разработку серверной части.

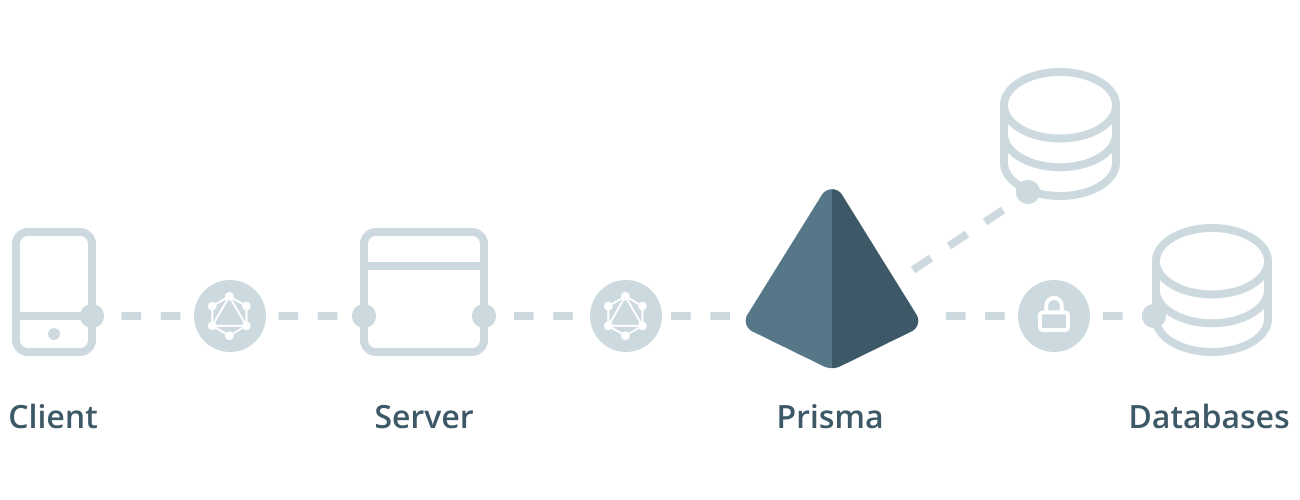


Рисунок 2.9 – принцип работы Prisma ORM

В качестве основной среды разработки используется Visual Studio Code – лёгкий, но при этом мощный и гибкий редактор кода, обладающий широкими возможностями расширения, настройки и интеграции с современными инструментами разработки. VS Code активно используется в профессиональной среде благодаря своей кроссплатформенности, высокой производительности и богатому набору функций, которые делают его идеальным выбором для реализации мобильных, веб- и серверных приложений. В контексте текущего проекта он обеспечивает удобную интеграцию с Flutter SDK, позволяя быстро запускать сборку, выполнять горячую перезагрузку, отслеживать изменения в UI и проводить отладку в реальном времени. С помощью встроенного эмулятора и поддержки горячей перезагрузки (hot reload) можно мгновенно видеть изменения в UI при редактировании кода, что ускоряет итерации и позволяет оперативно тестировать гипотезы.

Visual Studio Code предоставляет расширенные возможности для организации удалённого доступа к локальным сервисам посредством функции проброса портов. Эта функция позволяет опубликовать локальный сервер, запущенный на произвольном порту (например, localhost:3000), через защищённый HTTPS-канал, предоставляя глобальную ссылку, доступную извне. Такая ссылка формируется через облачную инфраструктуру Microsoft и обеспечивает безопасный доступ к локальному окружению без необходимости ручного конфигурирования туннелей, прокси или внешнего хостинга.

Данная возможность особенно актуальна при разработке фронтенд-приложений на основе Flutter Web, React, Vue или при тестировании API на Node.js, FastAPI, Express и других фреймворках. Проброс порта позволяет валидировать интерфейс и сетевую логику на реальных устройствах, включая мобильные браузеры, а также демонстрировать промежуточные версии заказчику или команде. Это существенно ускоряет цикл обратной связи, облегчает кросс-платформенное тестирование и позволяет выявлять ошибки, связанные с CORS, авторизацией, передачей токенов, обработкой заголовков и взаимодействием с внешними API.

Таким образом, функция проброса портов в Visual Studio Code выступает не просто как вспомогательный инструмент, а как полноценный механизм облачного доступа к локальному окружению, интегрированный в архитектуру разработки. Она способствует гибкой коллаборации, ускоряет тестирование, упрощает демонстрацию и делает локальную разработку доступной в глобальном контексте – без потери безопасности, производительности и контроля.

Для тестирования логики и поведения компонентов поддерживается запуск юнит-тестов и виджет-тестов непосредственно из редактора. Расширения, такие как Dart Test Explorer, позволяют запускать отдельные тесты, отслеживать покрытие и получать подробные отчёты об ошибках. Это делает VS Code не только средой для написания кода, но и полноценной платформой для проектирования, тестирования и отладки прототипов, что особенно важно на этапе активной разработки и валидации пользовательского опыта.

Для тестирования и отладки API-интерфейсов используется инструмент Postman (Рисунок 2.10) – мощное средство для создания, отправки и анализа HTTP-запросов. С его помощью можно моделировать различные сценарии взаимодействия клиента и сервера, проверять корректность обработки запросов, анализировать структуру ответов, выявлять ошибки и документировать API. Postman позволяет создавать коллекции запросов, проводить автоматизированное тестирование, использовать переменные окружения и сценарии, что делает его незаменимым инструментом при разработке и проверке функционала регистрации, авторизации, оформления заказов, обработки заявок и генерации накладных.

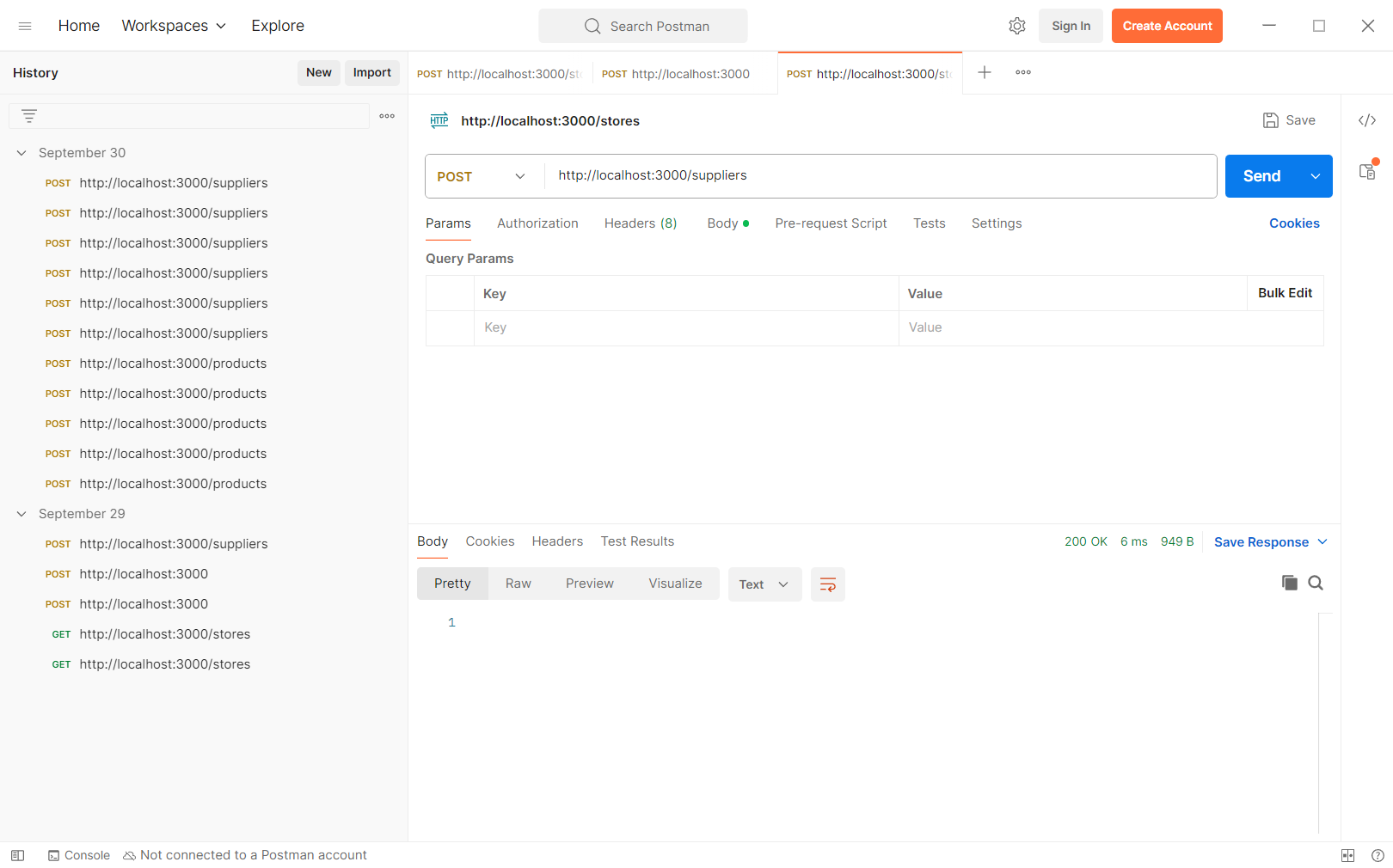


Рисунок 2.10 ­– окно программы postman

Для хранения данных на клиентской стороне используется встроенная реляционная база данных SQLite, которая зарекомендовала себя как надёжное, стабильное и широко применяемое решение в сфере разработки мобильных и десктопных приложений. Её архитектура позволяет сохранять информацию непосредственно на устройстве пользователя, что критически важно для обеспечения автономной работы приложения в условиях нестабильного или отсутствующего интернет-соединения. SQLite поддерживает полноценный набор функций, характерных для современных СУБД: транзакции, индексы, сложные запросы с фильтрацией, сортировкой и агрегацией, что делает её идеальной для реализации функционала складского учёта, фиксации остатков, хранения информации о товарах, заказах, пользователях, поставщиках, а также для ведения истории операций и аналитики. Благодаря своей лёгкости и отсутствию необходимости в отдельном сервере, SQLite обеспечивает высокую производительность и простоту интеграции, особенно в мобильных приложениях, где ресурсы ограничены.

Локальное хранение данных позволяет реализовать полноценный офлайн-режим, что критически важно для магазинов, работающих в удалённых регионах, на выездных точках продаж, в помещениях с ограниченным доступом к сети или в условиях нестабильной связи. Это обеспечивает непрерывность бизнес-процессов, возможность фиксации операций в реальном времени и последующую синхронизацию с сервером при восстановлении соединения. Такой подход повышает надёжность системы, снижает зависимость от внешних факторов и делает приложение более устойчивым к сбоям.

Система контроля версий построена на базе Git, а репозиторий проекта размещён на платформе GitHub, что обеспечивает прозрачность разработки, возможность командной работы, отслеживание изменений, управление ветками, проведение ревью и документирование процесса. GitHub предоставляет мощные инструменты для автоматизации сборки, тестирования и деплоя, включая интеграцию с CI/CD-пайплайнами, что позволяет ускорить выпуск новых версий, повысить качество кода и обеспечить стабильность релизов. Такой подход способствует масштабируемости проекта, упрощает сопровождение и открывает возможности для интеграции с внешними сервисами и платформами.

Выбор указанных инструментальных средств обусловлен их надёжностью, популярностью в профессиональной среде, активным сообществом, богатой документацией, регулярными обновлениями и высокой степенью совместимости между собой. Это позволяет реализовать все поставленные задачи проекта, обеспечить стабильную работу приложения, удобный и интуитивно понятный интерфейс, безопасность хранения данных, возможность масштабирования и адаптации под различные сценарии использования. Использование проверенных технологий снижает риски при разработке, упрощает сопровождение, ускоряет внедрение новых функций и обеспечивает долгосрочную устойчивость решения. Такой подход делает создаваемое приложение актуальным для малого и среднего бизнеса, нуждающегося в доступных, надёжных и эффективных инструментах автоматизации торговли, логистики, аналитики и управления клиентскими данными.

# 2.5 Защита и сохранность данных

Для работы приложения используется база данных SQLite вместе с Prisma ORM, что обеспечивает эффективное и безопасное взаимодействие с локальными данными. SQLite – это легковесная реляционная база данных, которая хранится в виде одного файла на устройстве и не требует отдельного сервера. Благодаря своей компактности и автономности, она идеально подходит для мобильных приложений, особенно на платформе Android. SQLite поддерживает стандартный SQL и широко применяется для хранения пользовательских данных, кэширования, офлайн-доступа и других задач.

Prisma ORM – это современный инструмент для работы с базами данных, ориентированный на разработчиков, использующих TypeScript и JavaScript. Он предоставляет удобный способ взаимодействия с базой данных через автогенерируемый клиент, который позволяет выполнять запросы к данным с помощью типизированного API. Prisma состоит из нескольких компонентов: Prisma Client, Prisma Migrate и Prisma Studio. Prisma Client – это основной интерфейс для работы с базой данных, Prisma Migrate – инструмент для управления миграциями, а Prisma Studio – визуальный редактор данных.

Структура базы данных (Рисунок 2.11) организована в виде взаимосвязанных таблиц, каждая из которых отвечает за хранение информации по ключевым сущностям бизнес-логики: поставщики, товары, магазины, партии товара и склады. Такая модель обеспечивает логическую целостность, упрощает выполнение запросов, позволяет реализовать фильтрацию, сортировку, агрегацию и аналитическую обработку данных.

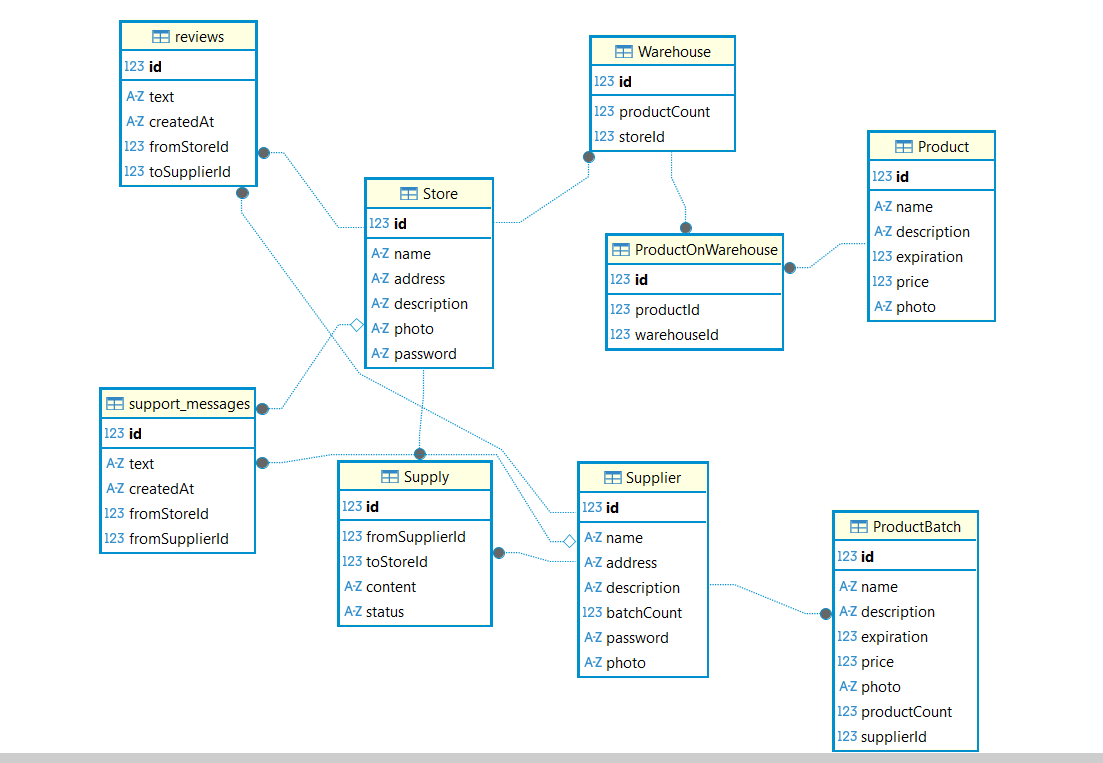


Рисунок 2.11 – схема базы данных

# 2.6 Организация и ведение информационной базы (модели)

Все данные проекта будут храниться в гугл-сервисе «Firebase» в формате JSON.

В используемой базе данных хранятся следующие объединения данных: история, пользователи, ассортимент. Для удобства они представлены в виде таблиц 2.1 – 2.5, соответственно «магазины», «партия товара», «продукт», «отзыв», «сообщение в поддержку».

Таблица 2.1 – Структура таблицы «магазины»

|  |  |  |
| --- | --- | --- |
| **Имя поля** | **Назначение** | **Тип данных** |
| id | Уникальный идентификатор магазина | Int |
| name | Название магазина | String |
| address | Адрес магазина | String |
| description | Описание магазина | String |
| photo | Фото магазина | String? |
| password | Пароль магазина | String |

Таблица 2.2 – структура таблицы «партия товара»

|  |  |  |
| --- | --- | --- |
| id | Уникальный идентификатор партии | Int |
| name | Название партии | String |
| description | Описание партии | String |
| expiration | Срок годности | Int |
| price | Цена партии | Float |
| photo | Фото партии | String? |
| itemsPerBatch | Количество товаров в одной партии | Int |
| quantity | Количество партий | Int |
| supplierId | Идентификатор поставщика | Int |

Таблица 2.3 – Структура таблицы «продукт»

|  |  |  |
| --- | --- | --- |
| **Имя поля** | **Назначение** | **Тип данных** |
| id | Уникальный идентификатор товара | Int |
| name | Название товара | String |
| description | Описание товара | String |
| expiration | Срок годности (в днях) | Int |
| price | Цена за единицу | Float |
| photo | Фото товара | String? |

Таблица 2.4 – Структура таблицы «отзыв»

|  |  |  |
| --- | --- | --- |
| **Имя поля** | **Назначение** | **Тип данных** |
| id | Уникальный идентификатор отзыва | Int |
| text | Текст отзыва | String |
| createdAt | Дата создания | DateTime |
| fromStoreId | Идентификатор магазина | Int |
| toSupplierId | Идентификатор поставщика | Int |

Таблица 2.4 – Структура таблицы «сообщение в поддержку»

|  |  |  |
| --- | --- | --- |
| **Имя поля** | **Назначение** | **Тип данных** |
| id | Уникальный идентификатор сообщения | Int |
| text | Содержание сообщения | String |
| createdAt | Дата создания | DateTime |
| fromStoreId | Идентификатор магазина (если есть) | Int? |
| fromSupplierId | Идентификатор поставщика (если есть) | Int? |

3 Реализация мобильного приложения

# 3.1 Описание используемых функций и процедур

Приложение имеет интуитивно понятный графический интерфейс с современным дизайном. Для навигации используется система вкладок ButtonNavigationBar, позволяющая пользователям легко переключаться между основными разделами. Цветовая схема выдержана в градиентных тонах, задаваемых в GlobalConfig, что обеспечивает единый визуальный стиль. Программа состоит из нескольких взаимосвязанных экранов (окон), каждый из которых предназначен для выполнения конкретных задач пользователя.

Для реализации поставленных задач было использовано множество функций и методов. Основные из них:

* Авторизация и аутентификация (вход для магазина, поставщика и администратора);
* Регистрация нового аккаунта (магазина или поставщика);
* Управление складом: просмотр, добавление, редактирование и продажа товаров;
* Просмотр и заказ партий товаров у поставщиков;
* Управление заказами (поставками): отслеживание статуса, отмена, получение и генерация накладных;
* Управление профилем: изменение данных, фото, настройки темы и отправка сообщений в поддержку;
* Администрирование системы через отдельное меню.

Рассмотрим основные функции проекта.

Функция «Авторизация пользователя» реализует процесс входа в систему. Пользователь выбирает тип аккаунта, «Магазин» или «Поставщик», вводит название и пароль. После нажатия кнопки «Войти» отправляется POST-запрос на сервер для проверки учетных данных. Система также проверяет наличие обновлений приложения (статус 426) и предоставляет специальный доступ для администратора через жестко заданные или сохраненные в настройках логин и пароль. В случае успеха пользователь перенаправляется в соответствующее меню ShopMenu или SupplierMenu, а данные сессии сохраняются для автоматического входа. Пример метода авторизации представлен ниже.

Future<void> \_login() async {

*// Проверка данных администратора*

final String? savedAdminName = await GlobalConfig.getSetting(GlobalConfig.adminUsernameKey);

final String? savedAdminPassword = await GlobalConfig.getSetting(GlobalConfig.adminPasswordKey);

final String adminName = savedAdminName ?? 'admin';

final String adminPassword = savedAdminPassword ?? 'admin';

if (nameController.text.trim() == adminName && passwordController.text.trim() == adminPassword) {

*// Вход как администратор*

Navigator.pushReplacement(context, MaterialPageRoute(builder: (context) => const AdminMenu()));

return;

}

*// Вход как магазин или поставщик*

final String endpoint = isSupplier ? '/suppliers/login' : '/stores/login';

final response = await http.post(

Uri.parse('$baseUrl$endpoint'),

headers: {'Content-Type': 'application/json', 'X-App-Version': appVersion},

body: jsonEncode({'name': nameController.text, 'password': passwordController.text}),

);

if (response.statusCode == 200) {

final user = jsonDecode(response.body);

await GlobalConfig.saveUserData(isSupplier ? 'supplier' : 'store', user); *// Сохранение сессии*

*// Перенаправление в основное меню*

Navigator.pushReplacement(context, MaterialPageRoute(builder: (context) => isSupplier ? SupplierMenu(supplier: user) : ShopMenu(store: user)));

} else if (response.statusCode == 401) {

\_showError('Неверное имя пользователя или пароль');

}

}

Функция «Добавить товар на склад» (для магазина) осуществляет процесс расширения собственной номенклатуры. При нажатии на плавающую кнопку «+» во вкладке «Склад» открывается диалоговое окно. Пользователь заполняет название, описание, цену и срок годности товара, а также может загрузить его изображение через галерею устройства.

После нажатия кнопки «Добавить» отправляется POST-запрос на соответствующий эндпоинт (/warehouses/{storeId}/products). Изображение конвертируется в формат Base64 и передается в составе JSON-объекта. При успешном выполнении запроса список товаров на складе автоматически обновляется. Пример метода загрузки данных представлен ниже.

Future<void> \_createProductOnWarehouse() async {

*// ... (получение данных из полей ввода и изображения)*

String? photoData;

if (selectedImage != null) {

final bytes = await selectedImage!.readAsBytes();

final base64Image = base64Encode(bytes);

final mimeType = \_getMimeType(imageType);

photoData = 'data:$mimeType;base64,$base64Image'; *// Формирование data URI*

}

final response = await http.post(

Uri.parse('$baseUrl/warehouses/${widget.store['id']}/products'),

headers: {'Content-Type': 'application/json'},

body: jsonEncode({

'name': nameController.text.trim(),

'description': descriptionController.text.trim(),

'price': price,

'expiration': expiration,

'photo': photoData, *// Передача изображения в запросе*

}),

);

if (response.statusCode == 200) {

\_loadWarehouseProducts(); *// Обновление UI*

ScaffoldMessenger.of(context).showSnackBar(const SnackBar(content: Text('Товар добавлен на склад')));

}

}

Функция «Создание заказа на поставку» реализует процесс оформления заказа выбранной партии товара у поставщика. Во вкладке «Поставки» пользователь просматривает доступные партии, сгруппированные по характеристикам. При нажатии на карточку партии открывается диалоговое окно с детальной информацией, отзывами на поставщика и возможностью оставить свой отзыв. Пользователь выбирает количество партий с помощью счетчика. После нажатия кнопки «Заказать» формируется и отправляется POST-запрос на создание заказа (/orders/create). В запросе передаются ID партии, магазина, поставщика и выбранное количество. Созданный заказ появляется во вкладке «Заказы» со статусом «оформлен». Пример метода создания заказа представлен ниже.

Future<void> \_createOrder() async {

final response = await http.post(

Uri.parse('$baseUrl/orders/create'),

headers: {'Content-Type': 'application/json'},

body: jsonEncode({

'batchId': widget.batch['id'], *// ID конкретной партии*

'storeId': widget.storeId,

'supplierId': widget.batch['supplierId'],

'quantity': \_quantity, *// Выбранное пользователем количество*

}),

);

if (response.statusCode == 200) {

widget.onOrderCreated(); *// Колбэк для обновления списка заказов*

ScaffoldMessenger.of(context).showSnackBar(const SnackBar(content: Text('Заказ создан')));

}

}

Функция «Получение заказа и добавление товара на склад» выполняется, когда статус заказа меняется на «отправлен». Магазин может подтвердить получение товара, нажав кнопку «Получить» в карточке заказа. При этом открывается диалоговое окно, где необходимо установить итоговую цену продажи для товара и при необходимости загрузить новое фото (если фото не выбрано, используется изображение из партии). После подтверждения отправляется POST-запрос (/orders/receive), который изменяет статус заказа на «получено», а также создает соответствующее количество единиц товара на складе магазина с указанной ценой. Это ключевая функция, связывающая логистику поставок с управлением складскими запасами.

Future<void> \_receiveOrder() async {

*// ... (получение цены и конвертация изображения)*

final response = await http.post(

Uri.parse('$baseUrl/orders/receive'),

headers: {'Content-Type': 'application/json'},

body: jsonEncode({

'supplyId': widget.supply['id'],

'pricePerItem': price,

'photo': photoData, *// Новое или унаследованное фото*

}),

);

if (response.statusCode == 200) {

widget.onOrderReceived(); *// Колбэк для обновления заказов и склада*

ScaffoldMessenger.of(context).showSnackBar(const SnackBar(content: Text('Заказ получен, товары добавлены на склад')));

}

}

# 4 Описание мобильного приложения

# 4.1 Общие сведения

Целью дипломного проекта является разработка мобильного приложения для автоматизации поставок и управления складом B2B-сектора, с каталогом товарных партий, с помощью которого магазины могут заказывать нужные им товары у поставщиков, а поставщики управлять своими предложениями.\*\*

Проектируемое приложение должно быть простым и максимально понятным для использования. Администратор имеет возможность управления системой через отдельное меню, поставщик – добавления и управления товарными партиями, а магазин – приобретения этих партий и управления собственным складом.

Главным назначением мобильного приложения «Система поставок B2B» является возможность оформления заказов, отслеживания поставок и управления товарными запасами в любое время и в любом месте.

Все ресурсы приложения (кроме иконки приложения) создаются и управляются программно с использованием виджетов Flutter.

Установка приложения будет осуществляться из apk–файла.

Запуск приложения осуществляется путём нажатия на его значок.

# 4.2 Описание разделов приложения

Приложение предназначено для упрощения взаимодействия между поставщиками и магазинами, обеспечивая удобный и структурированный процесс обмена товарными партиями. Основная функциональность заключается в том, что поставщики могут размещать партии товаров в системе, указывая их состав, количество и другие параметры, необходимые для последующей обработки. После размещения партии становятся доступными для магазинов, которые, в свою очередь, имеют возможность просматривать доступные предложения, оформлять заказы на нужные партии и получать товары для дальнейшей реализации.

Магазины заказывают партии с целью последующей розничной продажи, реализуя товары поштучно. Такой подход позволяет оптимизировать товарооборот, снизить издержки на логистику и упростить процесс закупки. Кроме того, приложение предоставляет магазинам возможность получать накладные на оформленные заказы, что способствует прозрачности документооборота и упрощает бухгалтерский учёт.

Работа с приложением начинается с окна входа и регистрации (Рисунок 4.1). При наличии аккаунта нужно в него войти, а при отсутствии – создать.

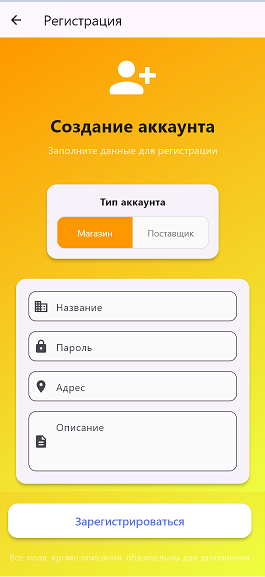
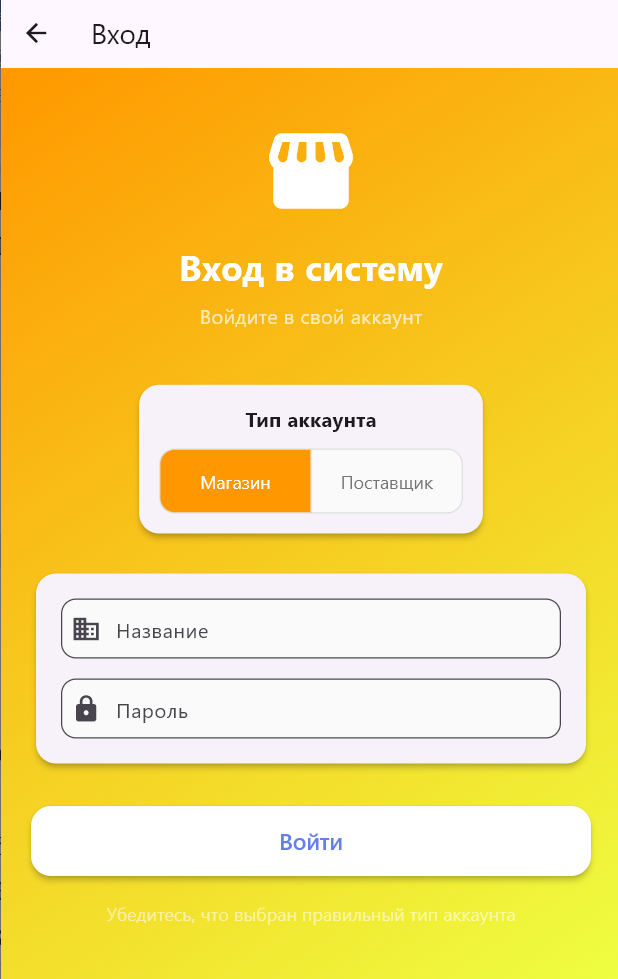


Рисунок 4.1 – вход и регистрация

После завершения регистрации администратор магазина попадает в раздел склада. На данном этапе склад еще пустой, и для того чтобы наполнить его товарами, необходимо перейти в раздел, посвященный поставкам, где можно оформить заказ на нужные позиции (Рисунок 4.2). В карточках каждого товара отображается его название, подробное описание, стоимость и фотография, что позволяет легко ориентироваться в ассортименте. Для удобства работы предусмотрены функции сортировки партий, возможность их фильтрации по различным параметрам, а также поиск, который помогает быстро находить нужные товары среди доступных вариантов.

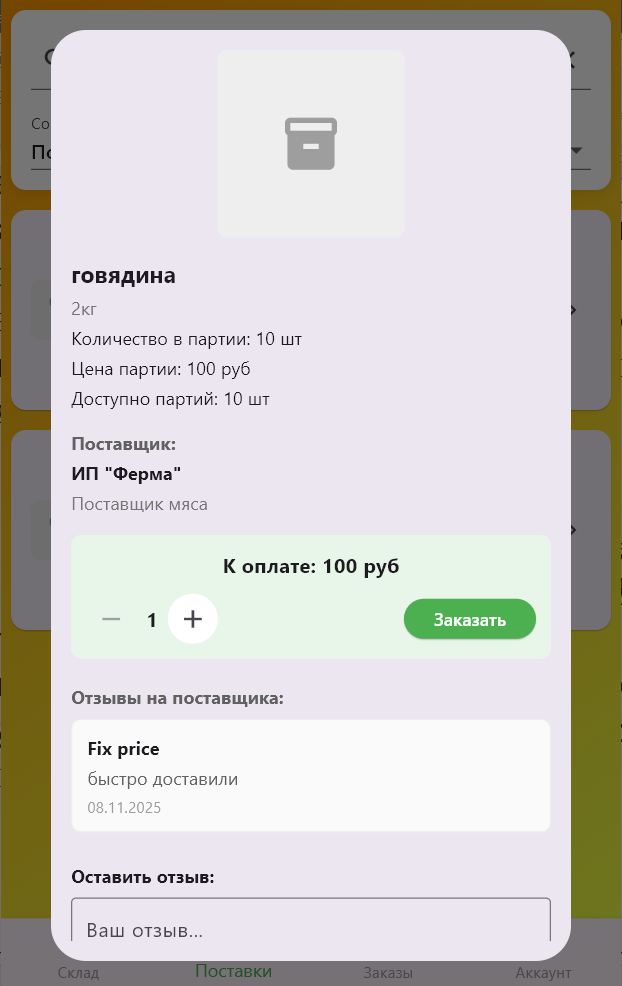


Рисунок 4.2 – заказ партии

Все заказы, включая оформленные, отправленные и полученные, отображаются в меню заказов (Рисунок 4.3). Для удобства работы предусмотрены функции сортировки и фильтрации, позволяющие быстро находить нужные позиции и управлять списком заказов. После того как заказ оформлен, магазину остается лишь ожидать момента, когда поставщик примет его и начнет дальнейшую обработку.

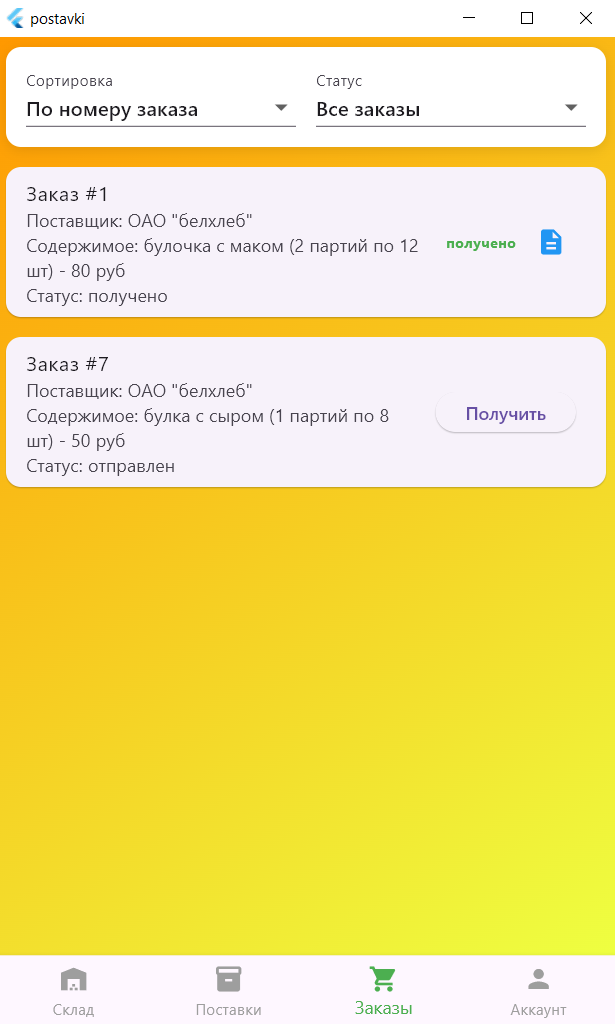


Рисунок 4.3 – меню заказов

Когда поставщик авторизуется и входит в приложение, на его экране автоматически отображаются все новые заказы, которые необходимо подготовить и отправить (Рисунок 4.4). Таким образом, он сразу видит актуальные заявки и может оперативно приступить к их обработке, не тратя время на поиск или дополнительные действия.

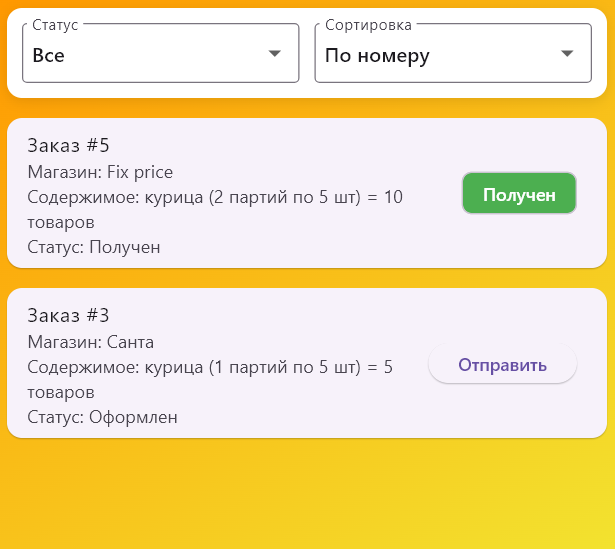


Рисунок 4.4 – отправка заказа

После того как заказ отправлен, поставщик получает возможность работать с партиями товаров. В приложении он может просматривать уже существующие партии, анализировать их состав и состояние, а также добавлять новые партии для расширения ассортимента (Рисунок 4.5). Такой функционал обеспечивает удобное управление товарными запасами и позволяет оперативно обновлять складскую информацию.

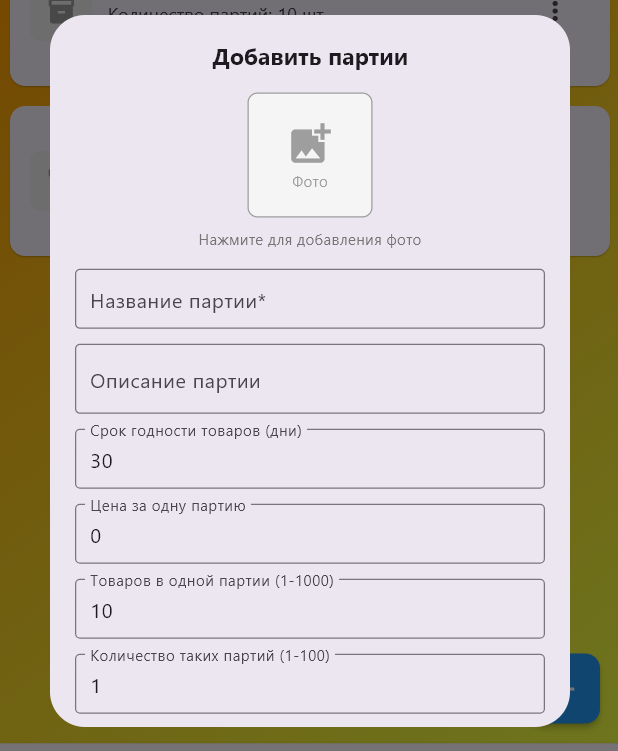
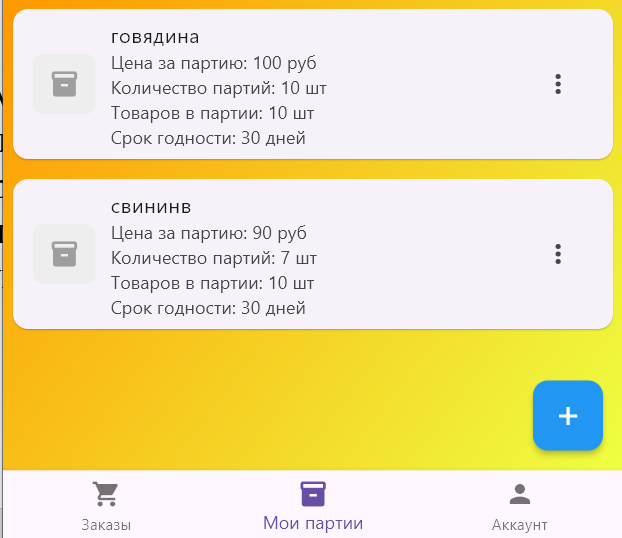


Рисунок 4.5 – просмотр и создание партий

Поставщик имеет возможность редактировать свой аккаунт: менять пароль, обновлять имя и корректировать адрес, что позволяет поддерживать актуальные учетные данные и оперативно вносить необходимые изменения в личную информацию.

После того как поставщик отправил заказ, и доставка была выполнена, магазин получает возможность принять его. После подтверждения приема заказ автоматически добавляется на склад, где он становится доступным для дальнейшего использования и управления (Рисунок 4.6).

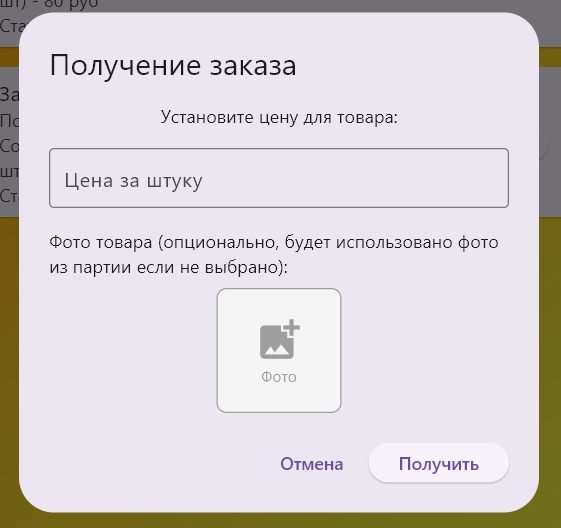


Рисунок 4.6 – получение заказа

После того как заказ был принят магазином, становится доступна дополнительная функция – просмотр автоматически сформированной накладной. Этот документ создается системой без необходимости ручного ввода данных, что значительно упрощает процесс учета. Кроме того, магазин может распечатать накладную, используя встроенные в andriod инструменты, что обеспечивает удобство ведения документации и позволяет быстро оформить все необходимые бумаги (Рисунок 4.7).

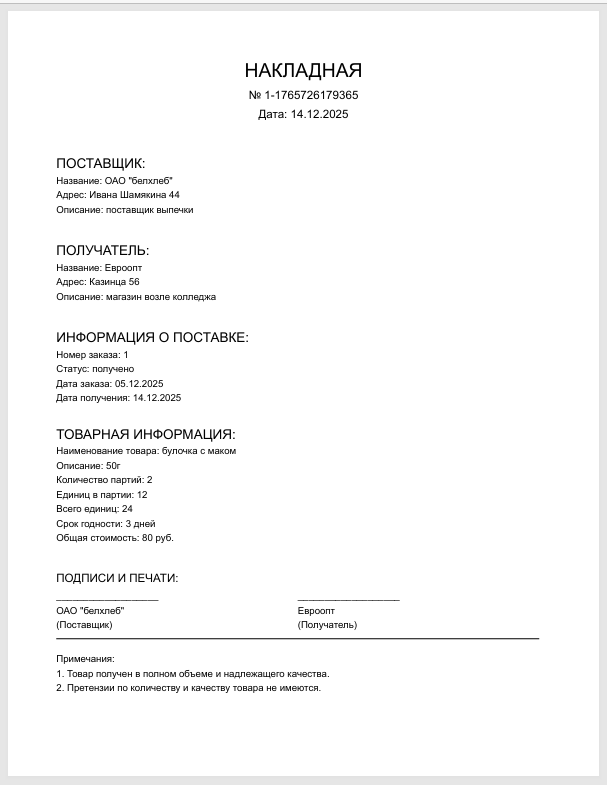


Рисунок 4.7 – накладная

В процессе розничной торговли магазин получает полный доступ к управлению складом. В этом разделе предусмотрены функции создания новых товаров, редактирования уже существующих позиций и их последующей продажи. Такой подход обеспечивает удобное ведение учета, позволяет оперативно обновлять ассортимент и контролировать движение товаров на складе, что делает процесс торговли более прозрачным и эффективным (Рисунок 3.8).

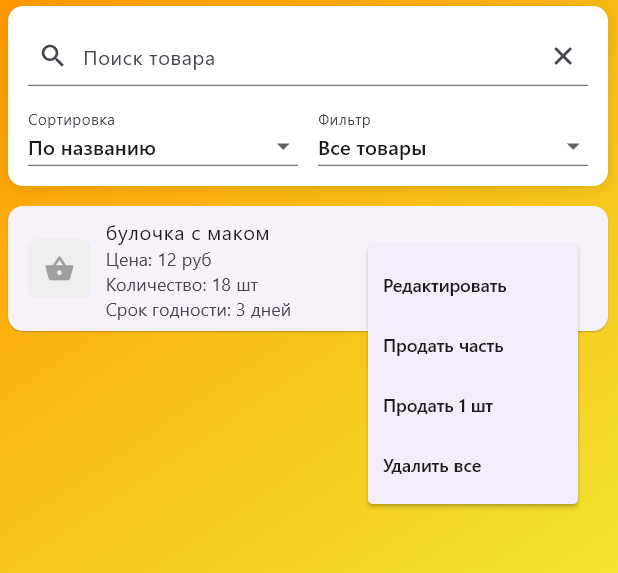


Рисунок 4.8 – управление складом.

Магазин имеет возможность оставить отзыв о поставщике, который затем будет отображаться в его интерфейсе. Таким образом, поставщик сможет увидеть полученный отзыв и учитывать его в дальнейшей работе, что способствует более прозрачному взаимодействию и повышает качество сотрудничества (Рисунок 4.9).

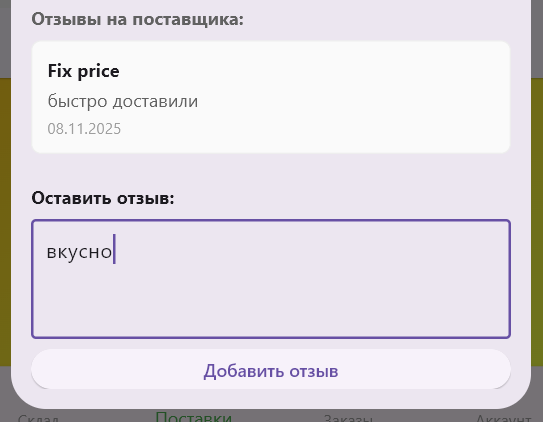
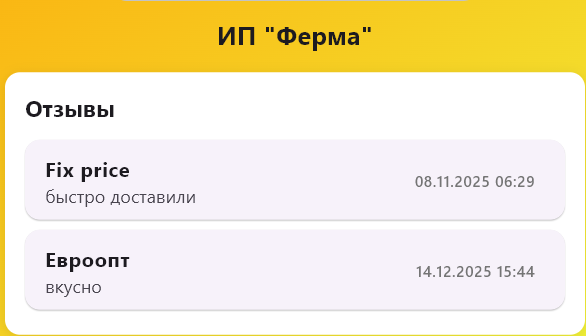


Рисунок 4.9 – отправление и просмотр отзыва

Во вкладке аккаунта для каждого пользователя предусмотрен полноценный набор функций, позволяющих управлять своим профилем и настраивать приложение под личные предпочтения. В верхней части отображается фотография, которая служит визуальным символом и помогает быстро идентифицировать владельца профиля. При необходимости её можно изменить, загрузив новое изображение. Под фотографией указано имя аккаунта, которое также можно редактировать, чтобы оно соответствовало актуальным данным или личным пожеланиям.

Ниже расположены основные элементы управления: кнопка для обращения в службу поддержки, где пользователь может отправить отчет о проблемах, в случае их возникновения; кнопка настроек, открывающая доступ к расширенным параметрам приложения; а также кнопка выхода из аккаунта, позволяющая завершить текущую сессию и обеспечить безопасность данных, особенно если устройство используется несколькими людьми.

В разделе настроек приложения открываются дополнительные возможности. Здесь можно изменить личные данные аккаунта, такие как имя, адрес электронной почты или другие сведения, чтобы поддерживать их актуальность. При необходимости доступна функция удаления аккаунта, которая полностью стирает информацию и отключает профиль от системы. Меню аккаунта и настроек представлены на рисунке 4.10.

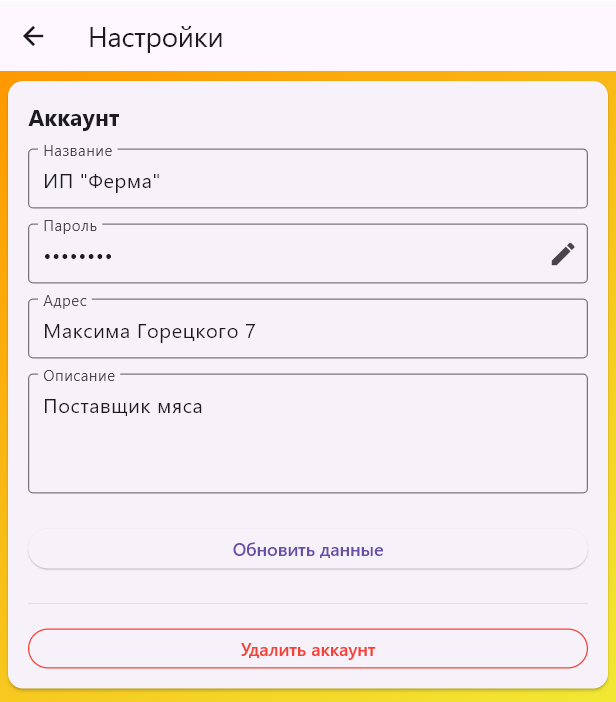
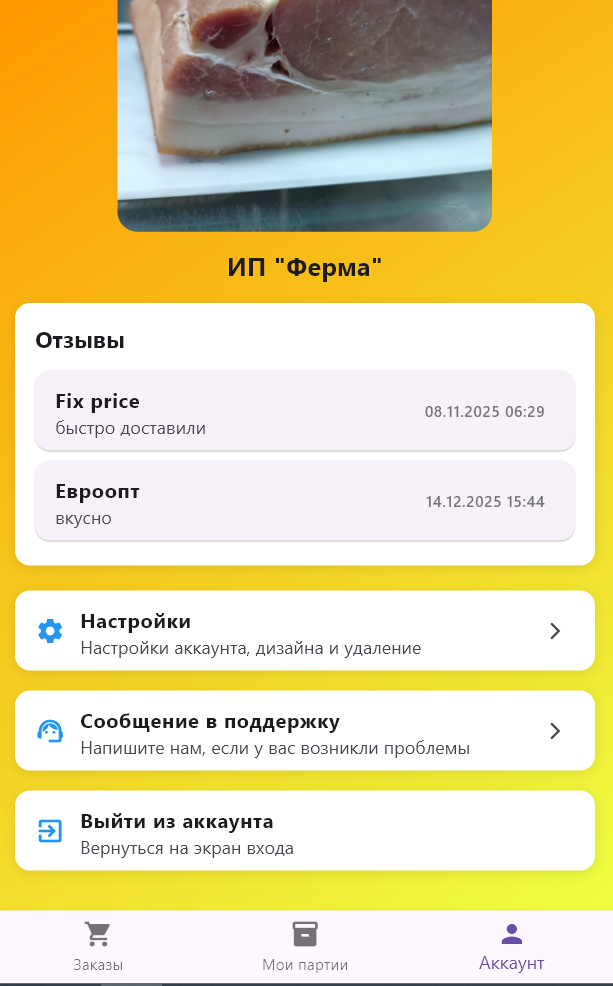


Рисунок 4.10 – меню аккаунта и настроек.

Кроме того, предусмотрена возможность смены цветовой схемы интерфейса, что делает работу с приложением более комфортной и индивидуальной. Доступно несколько вариантов цветов, из которых пользователь может создать градиент для фона приложения на свой вкус. Настройка цветовой схемы и варианты цветов представлены на рисунке 4.11.

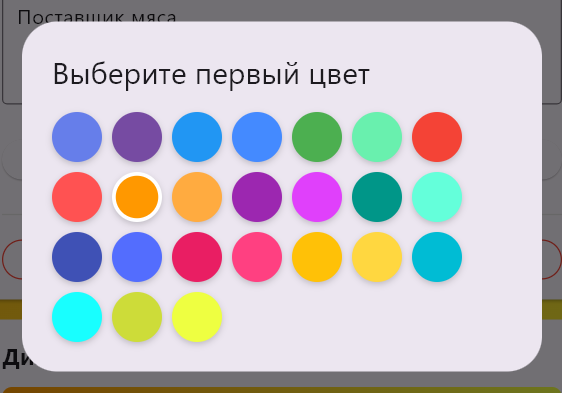
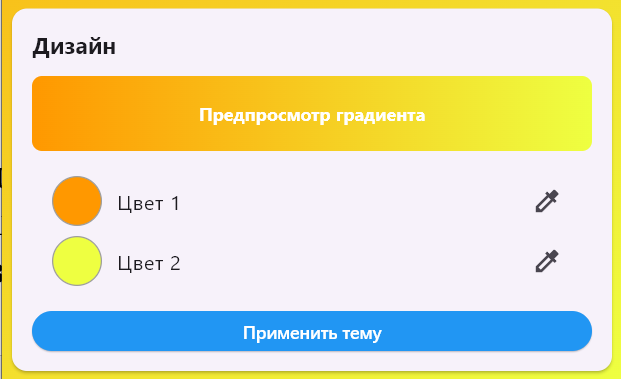


Рисунок 4.11 – смена цветовой схемы и варианты цветов

При вводе стандартной пары имени и пароля admin/admin администратор базы данных получает доступ к специальному меню, которое открывает перед ним полный набор инструментов для управления системой. В этом меню собраны все ключевые сущности базы, включая поставщиков, магазины, партии товаров, заказы и другие связанные элементы, что позволяет администратору контролировать и корректировать работу приложения в целом. Здесь же предусмотрена возможность изменения собственных учетных данных: администратор может сменить логин и пароль, чтобы повысить уровень безопасности и защитить доступ от посторонних. Дополнительно в меню доступна функция изменения адреса сервера, что особенно важно в случае, если прежний адрес становится недоступным или требуется перенести систему на новый сервер. Таким образом, данное меню служит центральной точкой управления, где администратор получает все необходимые средства для поддержания актуальности данных, обеспечения стабильной работы и гибкой настройки параметров базы. Главное меню администратора представлено на рисунке 4.12.

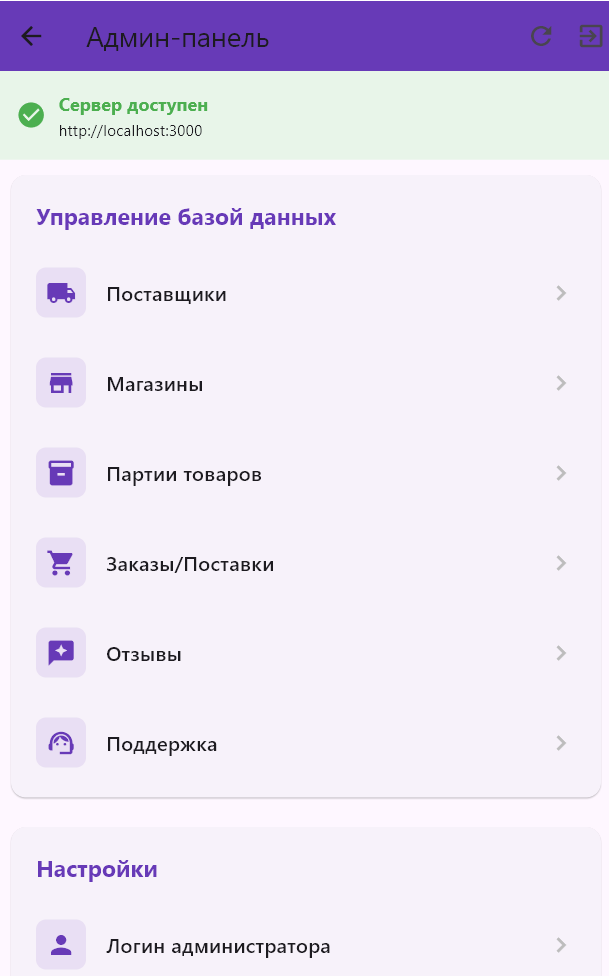


Рисунок 4.12 – главное меню администратора

В меню администратора находятся 6 основных кнопок управления базой данных:

* Управление магазинами
* Управление поставщиками
* Управление партиями товаров
* Управление заказами
* Управление отзывами
* Управление сообщениями в поддержку

При нажатии на одну из этих кнопок, открывается соответствующее меню, например, управление магазинами, в нем можно просмотреть список всех магазинов, просмотреть подробную информацию о каждом и, при необходимости, отредактировать данные или удалить аккаунт. Пример управления магазинами изображен на рисунке 4.13.

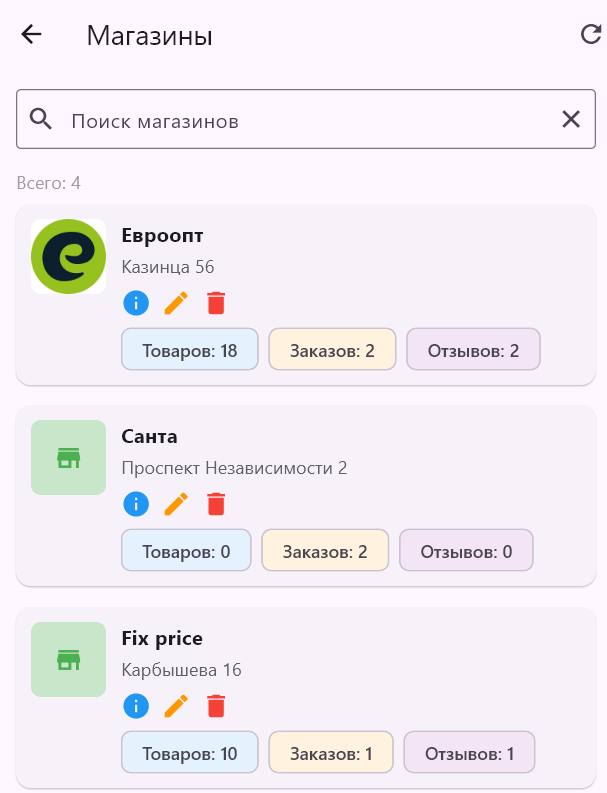


Рисунок 4.13 – управление магазинами

Приложение демонстрирует связный, логичный и завершённый контур работы, охватывающий все ключевые этапы взаимодействия между магазином и поставщиком. Магазин получает возможность оформлять и отслеживать заказы, принимать поставки и вести розничные операции в единой системе, а поставщик оперативно видит поступающие заявки, организует отгрузку и поддерживает актуальность товарных партий. Такой подход обеспечивает непрерывность процессов и исключает разрывы в информационном обмене.

Сквозные инструменты – сортировка, фильтры, поиск и автоматическая генерация накладных – значительно снижают объём ручных операций и вероятность ошибок. Это повышает скорость реакции на изменения, делает учёт прозрачным и управляемым. Простые карточки товара с ключевыми атрибутами обеспечивают наглядную навигацию по ассортименту, а единый раздел заказов устраняет разрозненность данных и ускоряет принятие решений.

Важным элементом является взаимная обратная связь через систему отзывов. Она укрепляет доверие и дисциплину процессов: поставщик получает сигнал о качестве взаимодействия, а магазин – дополнительный инструмент влияния на уровень сервиса. Конструкция ролей и чёткая стадийность жизненного цикла заказа создают предсказуемость и прозрачность: каждому участнику очевидно текущее состояние процесса и следующий шаг. В результате формируется устойчивая операционная среда, где каждое действие фиксируется и оставляет однозначный след в системе.

Итоговый эффект выражается в управляемости, прозрачности и снижении транзакционных издержек. Магазин быстрее формирует, принимает и реализует товар, а поставщик точнее планирует отгрузки и поддерживает актуальность ассортимента. Система масштабируется по объёму операций благодаря формализованным стадиям и унифицированным интерфейсам, оставаясь понятной и компактной в ежедневной работе.

# 5 Методика испытаний

# 5.1 Технические требования

Для выполнения тестирования данного приложения на устройствах Android необходимы следующие технически требования:

* процессор 1,7 ГГц;
* минимальный объем оперативной памяти – 1 ГБ;
* свободное место 50 МБ.
* операционная система Android 10 или более поздние версии.

# 5.2 Функциональное тестирование

Во время тестирования будут проверяться все функции, доступные в приложении: регистрация, авторизация, проверка создания партии товара, проверка создания заказа, поиск партий товаров, работа панели администратора, изменение цветовой схемы приложения. После чего, все собранные и найденные ошибки будут исправлены. результат тестирования представлен в таблице 5.1.

Страница авторизации представлена на рисунке 5.1.

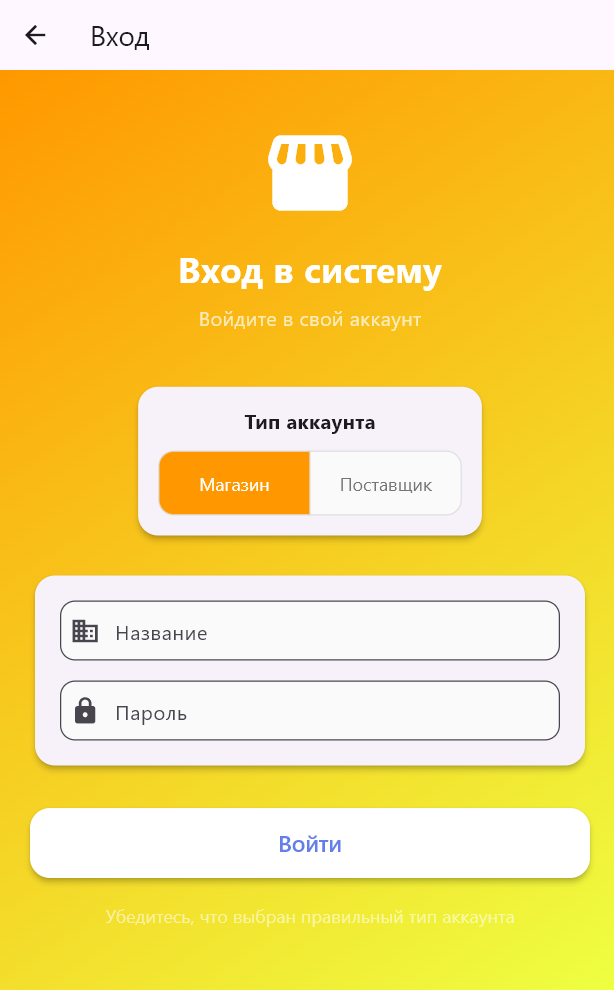


Рисунок 5.1 – Страница авторизации

Таблица 5.1 – Функциональное тестирование

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название тест-кейса | Описание | Ожидаемый результат | Фактический результат | Скриншот |
| Проверка системы регистрации | 1 Открыть приложение  2 Зарегистрировать нового пользователя | Сообщение об успешной регистрации | Сообщение об успешной регистрации | Рисунок 5.2 |
| Продолжение таблицы 5.1 | | | | |
| Название тест-кейса | Описание | Ожидаемый результат | Фактический результат | Скриншот |
| Проверка системы авторизации | 1. Открыть приложение 2. Зайти под созданным ранее пользователем 3. Выйти из учетной записи | Успешное выполнение функций входа и выхода | Успешное выполнение функций входа и выхода | Рисунки 5.3 – 5.4 |
| Поиск партий товара | 1. Открыть приложение 2. Войти под учетной записью 3. Ввести запрос в поле поиска | Вывод результата поиска | Вывод результата поиска | Рисунок 5.5 |
| Работа панели администратора | 1. Открыть приложение 2. Войти под учетной записью админа 3. Попробовать что-то изменить | Изменение отзыва | Изменение отзыва | Рисунок 5.6 |
| Изменение цветовой схемы приложения | 1 Открыть приложение  2 Войти под учетной записью администратора  3 Перейти в настройки  4 изманить цвета | Изменение градиента фона | Изменение градиента фона | Рисунок 5.7 |

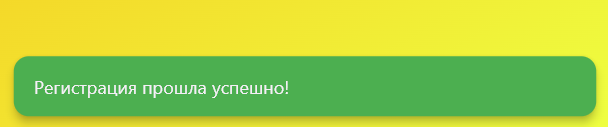


Рисунок 5.2 – Сообщение успешной регистрации.

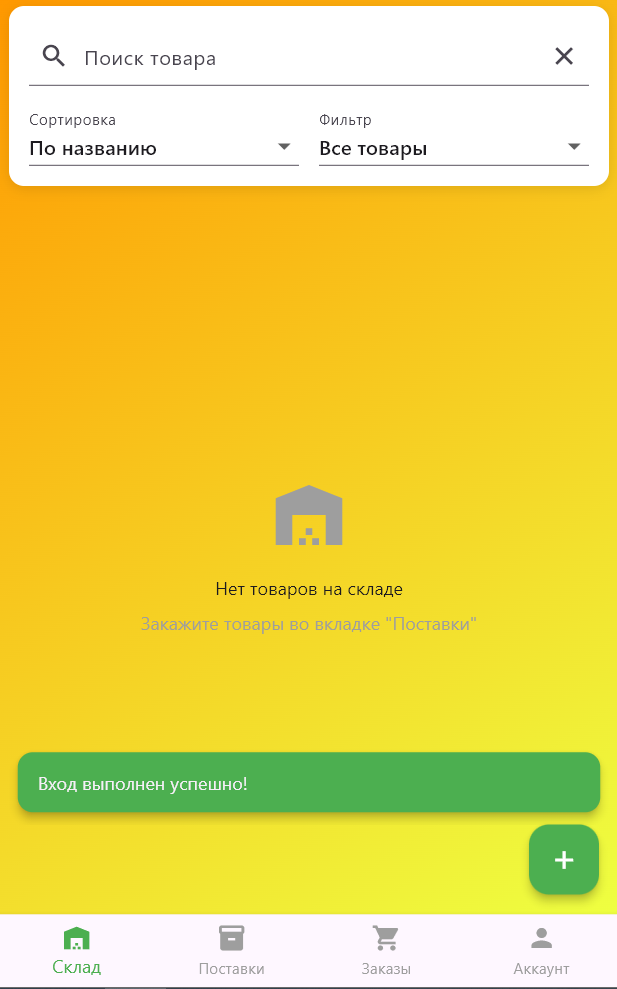


Рисунок 5.3 – Вход в учётную запись



Рисунок 5.4 – Выход из учётной записи



Рисунок 5.5 – поиск партий товаров

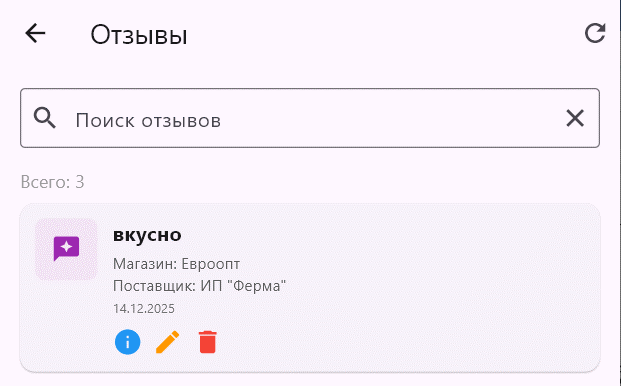
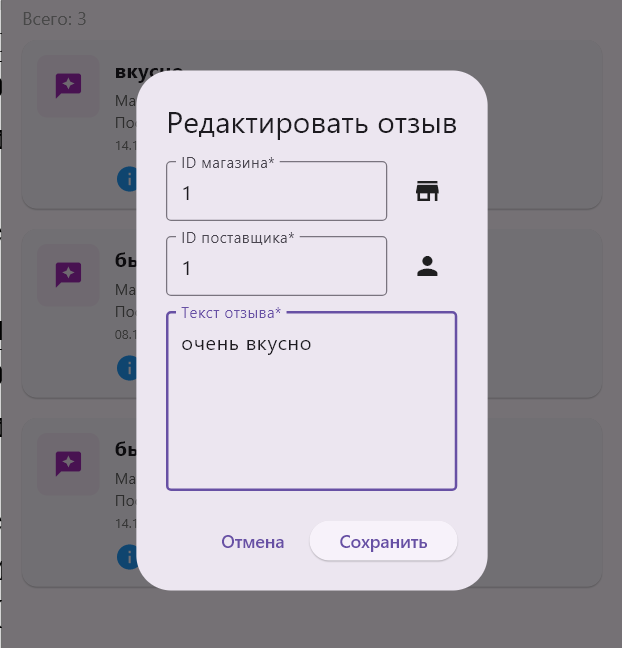
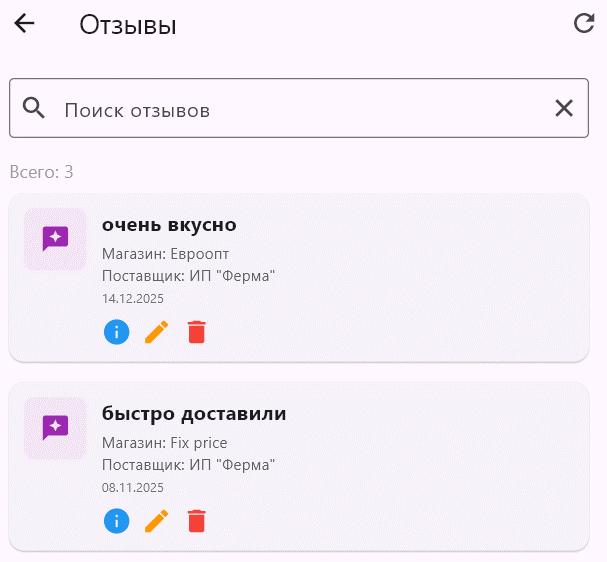
  

Рисунок 5.6 – изменение отзыва

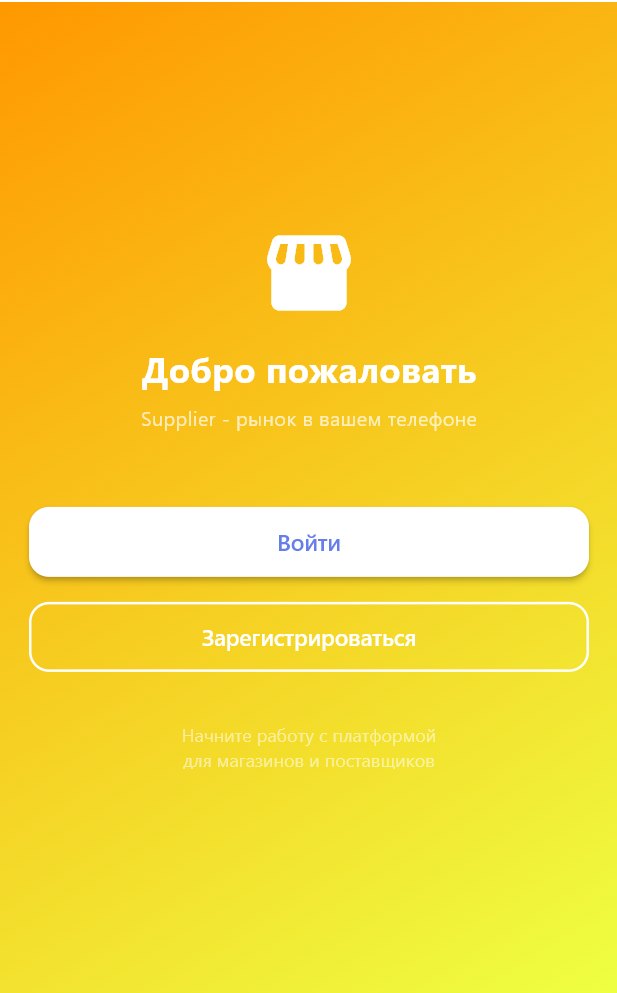
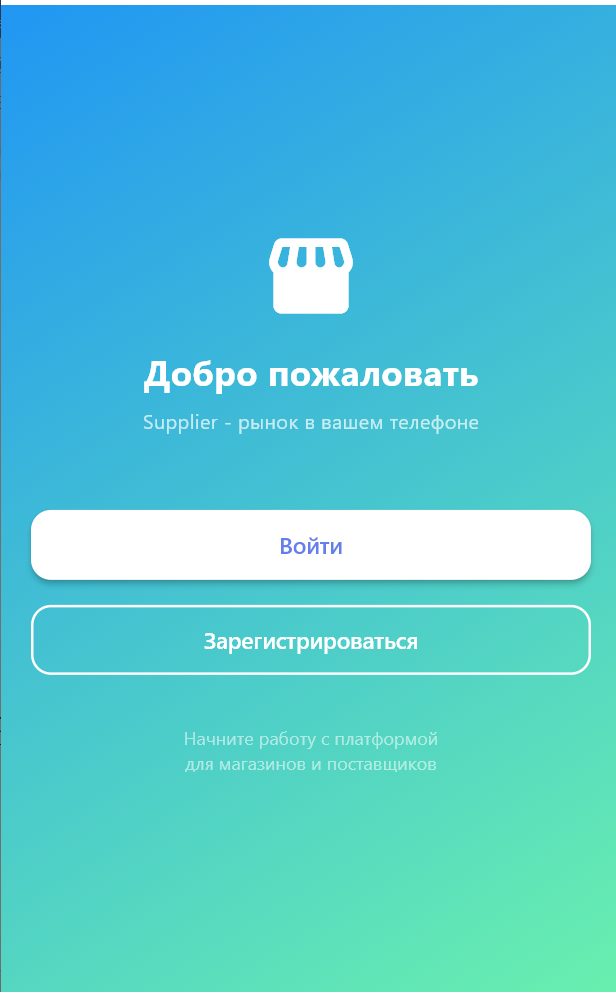
 

Рисунок 5.7 – изменение градиента фона

В ходе проверки были протестированы все ключевые функции приложения, включая регистрацию и авторизацию пользователей, создание партий товаров и заказов, поиск партий, работу панели администратора, а также возможность изменения цветовой схемы интерфейса. Все выявленные ошибки были зафиксированы и устранены, что позволило повысить стабильность и надёжность работы системы. Таким образом, приложение подтвердило корректность реализации основных функций и готовность к дальнейшему использованию.

### 6 Вопросы охраны труда

## 6.1 Классификация производственных травм. Меры по предупреждению производственного травматизма

Производственный травматизм представляет собой не просто статистический показатель или перечень несчастных случаев, а сложный социально-экономический феномен, пронизывающий все сферы трудовых отношений и являющийся своего рода лакмусовой бумажкой для оценки зрелости любой организации. По своей сути, это совокупность травм, полученных работниками при исполнении трудовых обязанностей под воздействием опасных производственных факторов, однако за этим сухим определением скрывается огромный пласт проблем – от инженерных просчётов и организационных упущений до глубоких психологических и культурных установок, существующих как внутри коллектива, так и в обществе в целом. Особенно важно понимать, что в условиях современного офиса, который зачастую воспринимается как пространство, лишённое явных угроз, природа травматизма меняется, становясь более изощрённой и коварной. Здесь на первый план выходят не столько внезапные и очевидные происшествия, сколько постепенно накапливающиеся последствия не эргономичных условий, хронического психоэмоционального напряжения и систематического пренебрежения, казалось бы, мелкими правилами безопасности. Именно поэтому анализ и классификация травм должны выходить за рамки простого каталогизирования, становясь инструментом глубокой диагностики корпоративного здоровья. Разработка же эффективных мер по предупреждению травматизма – это уже задача стратегического планирования, требующая комплексного подхода, где технические решения неразрывно связаны с управленческими компетенциями и работой по формированию нового типа сознания у каждого сотрудника. Цель данного раздела – не только провести систематизацию известных видов производственных травм и предложить набор профилактических мероприятий, но и показать внутреннюю логику их взаимосвязи, продемонстрировав, как грамотная классификация становится фундаментом для построения целостной и живейшей системы безопасности, способной адаптироваться к меняющимся условиям и предвосхищать риски, а не просто реагировать на уже случившееся. Такая система напрямую определяет не только сохранение физического и психического здоровья персонала – ключевого ресурса в экономике знаний, – но и напрямую влияет на финансовую стабильность, операционную эффективность и, в конечном итоге, на деловую репутацию компании, делая инвестиции в охрану труда одним из наиболее обоснованных и окупаемых направлений корпоративных расходов.

Исторически подход к производственному травматизму прошёл долгий путь от полного его восприятия как неизбежной «издержки производства» до современного понимания в качестве управляемого параметра, поддающегося контролю и минимизации. Сегодня мы понимаем, что каждая травма – это результат стечения конкретных обстоятельств, цепочки событий, которую можно и нужно разорвать. Первым шагом на этом пути является детальная классификация, служащая языком для описания проблемы. Классифицировать травмы можно по множеству оснований, и каждое из них высвечивает новые грани. Основным и наиболее содержательным является классификация по характеру воздействия травмирующего фактора:

* Механические травмы: ушибы об угол стола, порезы о край папки, растяжения и переломы при падении со стремянки, падения на ровном месте, травмы от падающих предметов (мониторов, полок).
* Термические травмы: ожоги паром от кофемашины, кипятком, контакт с нагретой поверхностью принтера, обморожение в неотапливаемых помещениях.
* Электрические травмы: электроудар от неисправной розетки или повреждённого кабеля, поражение током от самодельного удлинителя.
* Химические травмы: ожоги и отравления от агрессивных чистящих средств, тонера для принтеров, паров корректирующей жидкости.
* Психофизиологические травмы: синдром запястного канала, хронические боли в спине и шее, компьютерный зрительный синдром, профессиональный стресс, эмоциональное выгорание.

Механические травмы, будучи самыми распространёнными даже в офисе, включают в себя огромный спектр случаев: от банальных ушибов об угол стола или порезов пальца о край металлической папки до более серьёзных инцидентов, таких как растяжения связок и переломы при падении со стремянки во время замены перегоревшей лампы или попытки достать архивную папку с верхней полки переполненного стеллажа. Отдельного внимания заслуживают травмы, связанные с падением на одном уровне, которые часто кажутся нелепыми, но могут иметь тяжёлые последствия – поскользнуться можно не только на явно мокром полу после уборки, но и на отполированном до зеркального блеска покрытии в холле или на незаметно упавшей скрепке; споткнуться – о незакреплённый край ковролина, о низкий, но не обозначенный порог между помещениями или, что встречается сплошь и рядом, о хаотично протянутый через проход сетевой кабель от принтера или обогревателя. К механическим же относятся травмы от падающих предметов: падение монитора со слишком узкой подставки, обрушение карниза или полки, перегруженной документами, неаккуратное обращение с тяжёлой коробкой бумаги, которую один сотрудник может уронить на ногу другому. Термические травмы в офисном контексте часто связаны с использованием бытовой техники – ожог паром от кофемашины или кипятком из чайника, контакт с нагретой поверхностью принтера после длительной печати, реже – обморожение при работе в неотапливаемом подвальном помещении архива или серверной, где сотрудник проводит время в поисках нужного оборудования. Электрические травмы представляют собой особую категорию опасности, так как их последствия могут быть катастрофическими, а причина часто кроется в незаметной на первый взгляд неисправности: это может быть электроудар от неисправной розетки, от повреждённого кабеля питания системного блока, от самодельного удлинителя, собранного без соблюдения элементарных норм, или от пробоя изоляции в старом мониторе. Химические травмы, хотя и менее типичны для обычного офиса, вполне возможны при неправильном использовании средств для уборки, содержащих агрессивные щёлочи или кислоты, при заправке картриджей для лазерных принтеров, когда тонер может попасть в дыхательные пути или на слизистые оболочки глаз, или при работе с корректирующими жидкостями на основе токсичных растворителей, пары которых при регулярном вдыхании способны вызвать хроническое отравление. Наконец, наиболее коварной и сложной для идентификации является категория психофизиологических травм и их отдалённых последствий, которые формально могут не регистрироваться как несчастный случай на производстве, но по своему разрушительному воздействию на здоровье и трудоспособность человека ничуть не уступают, а порой и превосходят физические повреждения. Сюда относится весь спектр профессиональных заболеваний «белых воротничков»: синдром запястного канала от постоянной работы с компьютерной мышью, хронические боли в шее и спине, вызванные длительным статическим напряжением в нефизиологичной позе за неэргономичным рабочим местом, компьютерный зрительный синдром, ведущий к прогрессированию близорукости и астенопии. Но что ещё важнее – это профессиональный стресс, эмоциональное выгорание, тревожные и депрессивные расстройства, которые развиваются под гнётом хронической перегрузки, цейтнота, неясности поставленных задач, конфликтных отношений в коллективе или с руководством. Эти состояния не только снижают качество жизни сотрудника, но и резко повышают риск ошибок, невнимательности и, как следствие, тех самых внезапных физических травм, создавая порочный круг.

Для полноты картины и ведения учёта травмы также классифицируют по степени тяжести исхода:

* Микротравмы (лёгкая царапина, синяк) – часто не регистрируются, но важны для анализа повторяющихся ситуаций.
* Лёгкие травмы (несложный перелом пальца, растяжение) – влекут временную утрату трудоспособности без стойких последствий.
* Тяжёлые травмы (сложные переломы, ожоги, потеря зрения) – приводят к длительной нетрудоспособности или инвалидности.
* Травмы со смертельным исходом.

Это разделение имеет не только медицинское, но и важнейшее статистическое и юридическое значение. Микротравмы, такие как лёгкая царапина или синяк, часто остаются незарегистрированными, однако их систематический анализ может указать на повторяющиеся опасные ситуации. Лёгкие травмы влекут за собой временную утрату трудоспособности, но не оставляют стойких последствий – сюда относится, например, несложный перелом пальца или растяжение лодыжки. Тяжёлые травмы – это те, что приводят к длительной нетрудоспособности, к инвалидности или значительной утрате профессиональных функций: тяжёлые ожоги, сложные переломы, травмы глаза с потерей зрения, серьёзные черепно-мозговые травмы. Отдельной, самой трагичной графой стоят травмы со смертельным исходом. Ещё один важный критерий – количество пострадавших, разделяющее происшествия на одиночные и групповые, где пострадавших двое и более. Групповой несчастный случай всегда является тревожным сигналом о серьёзных системных сбоях в организации безопасности труда, будь то отравление угарным газом из-за неисправности системы вентиляции или падение неустойчивой конструкции, задевшее нескольких человек. Кроме того, для понимания обстоятельств полезно рассматривать классификацию по виду происшествия, что соотносится с общепринятыми формами учёта: падение с высоты, падение на одном уровне, воздействие движущихся или падающих предметов, поражение электрическим током, воздействие экстремальных температур или вредных веществ. Такой многоракурсный взгляд позволяет перевести разговор о травматизме из плоскости единичных несчастий в плоскость управляемых процессов, каждый элемент которого может быть подвергнут анализу и корректировке.

Но сама по себе классификация – это лишь карта местности. Чтобы начать по ней движение, необходимы методы анализа, которые превращают сырые данные в полезную информацию. Одним из старейших является статистический метод, который выявляет общие тенденции через расчёт коэффициентов частоты и тяжести травматизма за длительный период. Он даёт общую картину, показывает, улучшается ситуация или ухудшается, но мало что говорит о конкретных причинах. Более наглядный топографический метод предлагает нанести места всех происшествий на план офиса – это позволяет визуально выделить «горячие точки», участки повышенной опасности: скользкий пол у кулера с водой, узкий проход между стеллажами, зона с вечно перегруженными розетками. Самый же глубокий и ценный – монографический метод, который предполагает скрупулёзное расследование каждого отдельного случая, даже самого незначительного. Здесь изучается всё: состояние оборудования и инструментов в момент происшествия, условия освещённости и микроклимата, квалификация и психофизиологическое состояние пострадавшего, соответствие его действий инструкциям, действия коллег и руководителя. Цель – выстроить цепочку событий и докопаться до коренной причины, которая часто лежит не в плоскости личной неосторожности, а в области плохой организации, недостаточного обучения или неверной расстановки приоритетов руководством. Наконец, экономический метод переводит проблему на язык, понятный любому руководителю, подсчитывая как прямые затраты (оплата больничного, штрафы контролирующих органов, страховые выплаты), так и косвенные, которые часто в разы больше: простой работы, затраты на срочный найм и обучение нового сотрудника, падение морального духа в коллективе, ущерб репутации. Этот метод наглядно показывает, что инвестиции в профилактику – это не благотворительность, а выгодное вложение.

На основе такого всестороннего анализа строится комплексная система мер по предупреждению травматизма, которая должна быть именно системой, а не набором разрозненных действий. Эта система стоит на трёх фундаментальных опорах: технические, организационные и социально-психологические меры, которые взаимно усиливают друг друга. Технические меры направлены на физическое устранение или ослабление опасности. Это, прежде всего, проектирование и поддержание безопасной среды: правильная планировка помещений без узких проходов и слепых зон, устройство полов с противоскользящим покрытием, надёжное крепление всех навесных конструкций, грамотная разводка электропроводки с использованием кабель-каналов и коробов, исключающая появление «паутины» из удлинителей. Это также выбор безопасного оборудования: офисная мебель со скруглёнными кромками и регулируемыми параметрами, оргтехника с необходимыми защитными кожухами и блокировками (например, шредер, который не работает при открытом лотке), исправные стремянки с широкими ступенями и противоскользящим покрытием. Сюда же относится обеспечение средствами коллективной и индивидуальной защиты – от антистатических браслетов для работы с чувствительной электроникой до удобных нескользящих ковриков у рабочих мест. Однако сколько бы ни было вложено в технику, её эффективность сводится к нулю без грамотных организационных мер, которые создают рамки для безопасного поведения. Центральным элементом здесь является нормативная база: чёткие, конкретные и актуальные инструкции по охране труда для каждого процесса – от работы с копировальным аппаратом до правил эвакуации. Но инструкции мертвы без обучения. Поэтому следующая критически важная составляющая – это многоуровневая система инструктирования: вводный инструктаж при приёме на работу, первичный – непосредственно на рабочем месте, повторные – не реже чем раз в полгода, а также внеплановые и целевые при изменении условий, введении нового оборудования или после происшествий. Формальный подход здесь губителен; необходимо использовать интерактивные форматы: тренинги, разбор реальных кейсов, практические занятия по оказанию первой помощи. Организация самого рабочего места, его аттестация или специальная оценка условий труда (СОУТ) – это тоже организационная задача, призванная привести физические параметры (освещённость, шум, электромагнитные поля) в соответствие с нормативами и подобрать эргономичную мебель. Не менее важен постоянный контроль и аудит: плановые и внезапные проверки состояния техники и помещений, ведение предписанных журналов, регулярные совещания по вопросам безопасности. Особую роль играет обязательное расследование каждого инцидента, включая микротравмы, так как это – бесценный источник информации о скрытых дефектах системы. Все эти организационные усилия должны быть подкреплены социально-гигиеническими и психологическими мерами, направленными на человека. Борьба с производственным утомлением – это не просто слова, а введение регламентированных перерывов, организация зон отдыха, поощрение производственной гимнастики. Создание здорового психологического климата, где нет места моббингу и выгоранию, где царит атмосфера взаимного уважения и поддержки, напрямую снижает уровень стресса, а значит, и вероятность ошибок по невнимательности. И, наконец, кульминацией всех усилий является целенаправленная пропаганда и формирование культуры безопасности. Это долгий процесс, в результате которого безопасное поведение становится внутренней потребностью, а не внешним принуждением. Это достигается через постоянное информирование (плакаты, бюллетени, корпоративный портал), через вовлечение сотрудников в обсуждение проблем безопасности, через признание и поощрение тех, кто демонстрирует ответственное отношение. Когда работник сам заинтересован в том, чтобы его рабочее место было безопасным, когда он не стесняется указать коллеге на нарушение, система приобретает невероятную устойчивость и способность к саморегуляции.

Вся эта многослойная деятельность имеет прочную правовую основу, которая в нашей стране определяется Трудовым кодексом, федеральными законами, такими как закон о специальной оценке условий труда, и многочисленными подзаконными актами, санитарными правилами и нормами. Соблюдение этих норм – не добровольное пожелание, а прямая обязанность работодателя. При этом экономическое обоснование профилактики неоспоримо: затраты на создание безопасных условий, на обучение, на качественное оборудование несопоставимо ниже, чем потенциальные расходы на ликвидацию последствий одной серьёзной аварии, которые включают в себя не только прямые компенсации и штрафы, но и колоссальные репутационные потери, судебные издержки, падение производительности и уход ключевых специалистов. Таким образом, грамотно выстроенная система профилактики травматизма – это не центр затрат, а инвестиция в будущее компании, в её человеческий капитал, в её устойчивость и доброе имя. Она превращает охрану труда из скучной бюрократической процедуры в динамичный и жизненно важный бизнес-процесс, интегрированный во все аспекты деятельности организации. В конечном счёте, работа в этом направлении – это проявление не просто юридической ответственности, но и высшей степени управленческой зрелости и социальной ответственности бизнеса, где ценность человеческой жизни и здоровья является непреложным приоритетом, лежащим в основе любого успеха.

# 7 Энерго- и ресурсосбережение

Энергосбережение – это комплекс методов и приемов, направленных на эффективное использование топливно-энергетических ресурсов планеты. Основная задача энергосбережения заключается не только в сохранении ресурсов, непосредственно связанных с производством энергии, но и в уменьшении косвенного воздействия на этот процесс.

Эта взаимосвязь широко известна: большее количество произведенной энергии влечет за собой увеличение выбросов парниковых газов в атмосферу, что, в конечном счете, приводит к необратимым изменениям климата. Современные технологии и инновации в области оборудования позволяют предпринимать меры по сохранению планеты, не ущемляя при этом личный комфорт и безопасность.

Кроме морального удовлетворения от участия в сохранении окружающей среды, энергосбережение приносит и конкретные материальные выгоды. Внедрение эффективных систем управления освещением снижает расходы на электроэнергию. К тому же, бережное отношение к оборудованию продлевает срок его службы, что экономит на ремонте или замене вышедших из строя приборов.

Сегодня освещение можно автоматизировать с помощью разнообразной сенсорной техники:

* cветорегуляторы (диммеры) плавно регулируют интенсивность свечения ламп, что помогает управлять потреблением электроэнергии;
* датчики движения реагируют на инфракрасное излучение и управляют освещением в зависимости от перемещения, особенно эффективны в местах, где освещение естественным светом недостаточно;
* датчики присутствия более чувствительны к малейшим движениям идеально подходят для офисов и аудиторий, автоматически управляя освещением даже при длительном пребывании человека в покое;
* таймеры следят за временем работы света, обеспечивая его выключение или включение в указанные промежутки времени;

Помимо использования датчиков, научно-технический прогресс предоставил энергосберегающие источники света, доступные каждому потребителю:

* люминесцентные и энергосберегающие лампы, обладающие длительным ресурсом работы и позволяющие существенно снизить энергопотребление;
* светодиодные лампы, являющиеся экологически чистыми и энергоэффективными источниками света, обладающими высокой долговечностью и низким энергопотреблением.

Эффективные методы энергосбережения включают в себя не только использование новейших технологий, но и постоянное осознанное отношение к рациональному потреблению электроэнергии.

В рамках энергосберегающих мероприятий для ПК можно применять несколько стратегий, чтобы снизить энергопотребление и сделать использование компьютеров более эффективным:

* управление питанием. Включение режима ждущего режима (sleep mode) или режима ожидания (hibernate) в периоды бездействия ПК позволяет существенно экономить энергию. Установка коротких временных интервалов до активации этих режимов при неактивном использовании компьютера поможет снизить потребление энергии. - Использование функции автоматического выключения монитора или жестких дисков при длительном простое также способствует снижению энергопотребления;
* выбор энергоэффективных компонентов.  При выборе компьютерных компонентов (например, процессоров, видеокарт, блоков питания) стоит учитывать их энергоэффективность. Выбор компонентов с пониженным энергопотреблением может значительно снизить расход электроэнергии при работе ПК;
* регулярное обновление и настройка программного обеспечения.  Регулярные обновления операционных систем и программного обеспечения могут содержать оптимизации энергопотребления. Проверка и настройка энергосберегающих функций в программных настройках ПК также помогут оптимизировать использование ресурсов;
* использование средств управления энергопотреблением.  Установка специализированных программ и приложений, которые мониторят и управляют энергопотреблением компьютера, может помочь в автоматизации процесса оптимизации энергосбережения;
* проведение регулярного технического обслуживания.  Поддержание чистоты внутренних компонентов ПК и проверка их работоспособности способствуют более эффективной работе устройства, что в свою очередь может снизить его энергопотребление. Интеграция этих методов в повседневную работу на ПК позволяет не только снизить затраты на электроэнергию, но и сделать использование компьютера более эффективным и экологически ориентированным.

# ЗАКЛЮЧЕНИЕ

В проекте разработано мобильное приложение для химчистки «Dry Clean». Приложение обеспечивает удобный и эффективный способ заказа услуг и покупки товаров.

В мобильном приложении разработан удобный, понятный и простой в использовании интерфейс. Приложение реализовано полностью в соответствии с поставленной задачей дипломного проекта. Приложение было протестировано и отлажено.

В приложении реализованы следующие задачи:

* реализован механизм заказа услуг на необходимую дату и время;
* реализован механизм автоматического формирования заказа;
* организована пользовательская и администраторская части;
* реализован механизм автоматического изменения статуса заказа;
* реализован интерфейс приложения, обеспечивающий наглядное, интуитивно понятное представление структуры;
* реализован механизм модификации содержимого разделов посредством администраторского интерфейса.

Программный продукт готов к практическому использованию. Данное приложение может быть дополнено и модернизировано.

В проекте рассмотрен вопрос по охране труда «Охрана труда для офисных работников» и выполнена графическая часть: диаграмма вариантов использования, диаграмма классов и карта пользователя.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Химчитска [Электронный ресурс]. – Режим доступа: https:// medium.com/@sonyagoldfr/чистим-всё-31d854ab6ad3. – Дата доступа: 19.01.2024.
2. Cleex.by [Электронный ресурс]. – Режим доступа: https:// Cleex.by. – Дата доступа: 19.01.2024.
3. Belhimchistka.by [Электронный ресурс]. – Режим доступа: https:// belhimchistka.by. – Дата доступа: 19.01.2024.
4. Material Design [Электронный ресурс]. – Режим доступа: https:// google-developer-training.github.io/android-developer-fundamentals-course-concepts-v2/unit-2-user-experience/lesson-5-delightful-user-experience/5-2-c-material-design/5-2-c-material-design. – Дата доступа: 19.01.2024.
5. Figma [Электронный ресурс]. – Режим доступа: https://www.figma.com. – Дата доступа: 19.01.2024.
6. Обзор обновлений и новых функций Windows 10 [Электронный ресурс]. – Режим доступа: https://www.microsoft.com/ru-ru/windows/features. – Дата доступа: 19.01.2024.
7. Варакин, М.В. Разработка мобильных приложений под Android / М.В. Варакин. – М.: УЦ «Специалист» при МГТУ им. Н. Э. Баумана, 2012. – 128 c.
8. Медникс, З. Программирование под Android. / З. Медникс, Л. Дорнин. –СПб.: Издательство Питер, 2012. – 250 с.
9. Леива, А. Kotlin for Android Developers. ­ Бермингем: Издательство Packt Publishing, 2017. – 364 с.
10. Firebase [Электронный ресурс]. – Режим доступа: https://Firebase.com. – Дата доступа: 19.01.2024.
11. Охрана труда: учеб. / В. Г. Андруш, Л. Т. Ткачёва, К. Д. Яшин – 2-е изд., испр. и доп. – Мн.: РИПО, 2021. – 334 с.
12. Энерго-и ресурсосбережение [Электронный ресурс]. – Режим доступа: http://studfile.net/preview/5631040/page:11. – Дата доступа: 19.01.2024.

# ПРИЛОЖЕНИЕ А

(справочное)

Фрагменты текста программы

//Оформление заказа

package com.example.applicationdryclean  
  
import android.content.Intent  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.widget.Toast  
import androidx.recyclerview.widget.LinearLayoutManager  
import com.example.applicationdryclean.Adaptar.PendingOrderAdapter  
import com.example.applicationdryclean.Model.AdminOrderDetails  
import com.example.applicationdryclean.databinding.ActivityPendingOrderBinding  
import com.google.firebase.database.DataSnapshot  
import com.google.firebase.database.DatabaseError  
import com.google.firebase.database.DatabaseReference  
import com.google.firebase.database.FirebaseDatabase  
import com.google.firebase.database.ValueEventListener  
  
class PendingOrderActivity : AppCompatActivity(), PendingOrderAdapter.OnItemClicked {  
 private lateinit var binding:ActivityPendingOrderBinding  
 private var listOfName:MutableList<String> = *mutableListOf*()  
 private var listOfTotalPrice: MutableList<String> = *mutableListOf*()  
 private var listOfImageFirstProductOrder:MutableList<String> = *mutableListOf*()  
 private var listOfOrderItem: ArrayList<AdminOrderDetails> = *arrayListOf*()  
 private lateinit var database: FirebaseDatabase  
 private lateinit var databaseOrderDetails: DatabaseReference  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 binding = ActivityPendingOrderBinding.inflate(*layoutInflater*)  
 setContentView(binding.*root*)  
 *//initialization of database* database = FirebaseDatabase.getInstance()  
 *//initialization of databaseReference* databaseOrderDetails = database.*reference*.child("OrderDetails")  
  
 getOrderDetails()  
  
 binding.backButton.setOnClickListener {finish()  
 }}  
  
 private fun getOrderDetails() {  
 *//retrieve order details from Firebase database* databaseOrderDetails.addListenerForSingleValueEvent(object : ValueEventListener {  
 override fun onDataChange(snapshot: DataSnapshot) {  
 for (orderSnapshot in snapshot.*children*){  
 val adminOrderDetails = orderSnapshot.getValue(AdminOrderDetails::class.*java*)  
 adminOrderDetails?.*let* {listOfOrderItem.add(it)  
 }}  
 addDataToListForRecyclerView()  
 }  
  
 override fun onCancelled(error: DatabaseError) {  
  
 }  
 })  
 }  
  
 private fun addDataToListForRecyclerView() {  
 for (orderItem in listOfOrderItem){  
  
 *//add data to respective list for populating the recyclerview* orderItem.userName?.*let* {listOfName.add(it) }orderItem.totalPrice?.*let* {listOfTotalPrice.add(it) }orderItem.productImages?.*filterNot* {it.*isEmpty*()}?.*forEach*{listOfImageFirstProductOrder.add(it)  
 }}  
 setAdapter()  
 }  
  
 private fun setAdapter() {  
 binding.pendingOrderRecyclerView.*layoutManager* = LinearLayoutManager(this)  
 val adapter = PendingOrderAdapter(this, listOfName,listOfTotalPrice,listOfImageFirstProductOrder,this)  
 binding.pendingOrderRecyclerView.*adapter* = adapter  
 }  
  
 override fun onItemClickListener(position: Int) {  
 val intent = Intent(this, OrderDetailsActivity::class.*java*)  
 val userOrderDetails = listOfOrderItem[position]  
 intent.putExtra("userOrderDetails",userOrderDetails)  
 startActivity(intent)  
 }  
  
 override fun onItemAcceptClickListener(position: Int) {  
 *//handle item acceptance and update database* val childItemPushKey = listOfOrderItem[position].itemPushKey  
 val clickItemOrderReference = childItemPushKey?.*let*{database.*reference*.child("OrderDetails").child(it)  
 }clickItemOrderReference?.child("orderAccepted")?.setValue(true)  
 updateOrderAcceptStatus(position)  
 }  
  
 override fun onItemDispatchClickListener(position: Int) {  
 *//handle item dispatch and update database* val dispatchItemPushKey = listOfOrderItem[position].itemPushKey  
 val dispatchItemOrderReferance =  
 database.*reference*.child("CompletedOrder").child(dispatchItemPushKey!!)  
 dispatchItemOrderReferance.setValue(listOfOrderItem[position])  
 .addOnSuccessListener {deleteThisItemFromOrderDetails(dispatchItemPushKey)  
 }}  
  
 private fun deleteThisItemFromOrderDetails(dispatchItemPushKey: String) {  
 val adminOrderDetailsItemsReference = database.*reference*.child("OrderDetails").child(dispatchItemPushKey)  
 adminOrderDetailsItemsReference.removeValue()  
 .addOnSuccessListener {Toast.makeText(this, "Заказ отправлен!", Toast.*LENGTH\_SHORT*).show()  
 }.addOnFailureListener {Toast.makeText(this, "Заказ не отправлен!", Toast.*LENGTH\_SHORT*).show()  
 }}  
  
 private fun updateOrderAcceptStatus(position: Int) {  
 *//update order acceptance in user's BuyHistory and OrderDetails* val userIdOfClickedItem = listOfOrderItem[position].userUid  
 val pushKeyOfClickedItem = listOfOrderItem[position].itemPushKey  
 val buyHistoryReference =  
 database.*reference*.child("user").child(userIdOfClickedItem!!)  
 .child("BuyHistory").child(pushKeyOfClickedItem!!)  
 buyHistoryReference.child("orderAccepted").setValue(true)  
 databaseOrderDetails.child(pushKeyOfClickedItem).child("orderAccepted").setValue(true)  
 }  
  
}

# ПРИЛОЖЕНИЕ Б

(обязательное)

Диаграммы UML

