**Государственное бюджетное профессиональное образовательное учреждение**

**«Нижегородский Губернский колледж»**

Методическая комиссия «Информатика и вычислительная техника»

Допущен к защите:

преподаватель

\_\_\_\_\_\_\_Л.В. Мухина,

\_\_\_\_\_\_\_Е.П. Голубева,

**ОТЧЕТ** **ПО**

**УЧЕБНОЙ ПРАКТИКЕ**

ПМ.01 РАЗРАБОТКА МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ КОМПЬЮТЕРНЫХ СИСТЕМ

Руководитель: \_\_\_\_\_\_\_\_\_\_ Мухина Л.В., Голубева Е.П 21.03.2025 г.

Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Зрячев Т.А 21.03.2025 г.

Специальность, группа: 09.02.07, 43П

Нижний Новгород

2025 г.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc193442251)

[1 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ 7](#_Toc193442252)

[1.1 Разработка, администрирование и защита баз данных 7](#_Toc193442253)

[1.1.2 Выбор СУБД 7](#_Toc193442254)

[1.1.3 Проектирование структуры базы данных 7](#_Toc193442255)

[1.1.4 Создание базы данных 11](#_Toc193442256)

[1.2 Разработка модулей программного обеспечения для компьютерных систем 11](#_Toc193442257)

[1.2.1 Описание задачи 11](#_Toc193442258)

[1.2.2 Выбор технологий 12](#_Toc193442259)

[1.2.3 Архитектура приложения 12](#_Toc193442260)

[1.2.4 Алгоритм расчета скидки 13](#_Toc193442261)

[1.2.5 Оформление интерфейса 13](#_Toc193442262)

[1.3 Сопровождение и обслуживание программного обеспечения компьютерных систем 14](#_Toc193442263)

[1.3.1 Разработка последовательного пользовательского интерфейса 14](#_Toc193442264)

[1.3.2 Добавление и редактирование данных о партнерах 15](#_Toc193442265)

[1.4 Осуществление интеграции программных модулей 17](#_Toc193442266)

[1.4.1 Вывод истории реализации продукции партнером 17](#_Toc193442267)

[1.4.2 Разработка метода расчета количества материала, необходимого для производства продукции 18](#_Toc193442268)

[1.4.3 Тестирование функции расчета количества материалов 20](#_Toc193442269)

[1.5 Тестирование редактирования партнера 21](#_Toc193442270)

[1.6 Построение UML диаграмм 22](#_Toc193442271)

[2 ПОДДЕРЖКА И ТЕСТИРОВАНИЕ ПРОГРАММНЫХ МОДУЛЕЙ 25](#_Toc193442272)

[2.1 Разработка и создание базы данных 25](#_Toc193442273)

[2.1.1 Предобработка данных и заполнение базы 28](#_Toc193442274)

[2.2 Разработка приложения 29](#_Toc193442275)

[2.2.1 Модуль авторизации 29](#_Toc193442276)

[2.2.2 Список товаров 31](#_Toc193442277)

[2.2.3 Просмотр и оформление заказа 33](#_Toc193442278)

[2.3 Тестирование программных решений 36](#_Toc193442279)

[2.3.1 Разработка метода расчета свободных временных интервалов для формирования оптимального графика работы сотрудников 36](#_Toc193442280)

[2.3.2 Разработка модульных тестов к библиотеке 40](#_Toc193442281)

[2.3.3 Тестирование добавления товара администратором 40](#_Toc193442282)

[3 РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ 42](#_Toc193442283)

[3.1 Верстка приложения 42](#_Toc193442284)

[3.2 Клиент-серверное взаимодействие 48](#_Toc193442285)

[3.2.1 Авторизация и регистрация 49](#_Toc193442286)

[3.2.2 Восстановление пароля 49](#_Toc193442287)

[3.2.3 Вывод продуктов 50](#_Toc193442288)

[3.2.4 Избранное 50](#_Toc193442289)

[3.2.5 Профиль 50](#_Toc193442290)

[ЗАКЛЮЧЕНИЕ 51](#_Toc193442291)

[ПРИЛОЖЕНИЯ 52](#_Toc193442292)

[Приложение А 52](#_Toc193442293)

[Приложение Б 54](#_Toc193442294)

[Приложение В 55](#_Toc193442295)

[Приложение Г 58](#_Toc193442296)

# ВВЕДЕНИЕ

В современных условиях развития информационных технологий разработка модулей программного обеспечения представляет собой ключевой этап создания качественного и эффективного программного продукта. Процесс разработки программных модулей требует комплексного подхода, включающего проектирование архитектуры, написание кода, его тестирование и оптимизацию, что обеспечивает создание надежного и масштабируемого программного обеспечения.

Настоящий отчет подготовлен по результатам выполнения модуля "ПМ.01 Разработка модулей программного обеспечения для компьютерных систем" и отражает процесс освоения практических навыков создания программных модулей. Практическая работа позволила получить ценный опыт в области разработки программного обеспечения, применить теоретические знания на практике и развить профессиональные компетенции, необходимые для успешной работы в сфере информационных технологий.

В ходе выполнения модуля были реализованы различные программные модули, проведена их оптимизация и тестирование. По результатам работы сделаны выводы о наиболее эффективных подходах к разработке и совершенствованию программных решений.

Цели и задачи

Основной целью выполнения данного модуля является формирование профессиональных компетенций в области разработки модулей программного обеспечения.

Задачи:

* освоить методологии и технологии разработки программных модулей;
* изучить принципы проектирования архитектуры программного обеспечения;
* применять современные языки и среды программирования при разработке модулей;
* проводить тестирование и отладку программного кода;
* организовать работу с системами контроля версий;
* документировать разработанные программные решения;
* оптимизировать производительность создаваемых модулей программного обеспечения.

# 1 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

## Разработка, администрирование и защита баз данных

Целью выполнения данного модуля является разработка подсистемы для работы с партнерами компании в рамках системы управления данными. Подсистема должна обеспечивать функционал для просмотра списка партнеров, добавления/редактирования данных о партнерах, а также просмотра истории реализации продукции партнером. Для достижения этой цели создана база данных (БД), соответствующая третьей нормальной форме, с обеспечением ссылочной целостности, согласованной схемой именования и необходимыми первичными и внешними ключами.

**1.1.1 Описание предметной области**

Производственная компания «Мастер пол» выпускает напольные покрытия и реализует продукцию через партнеров. Партнеры могут быть как розничными магазинами, так и оптовыми компаниями или интернет-магазинами. Взаимодействие с партнерами требует учета их данных, истории продаж, рейтинга и других параметров, влияющих на условия сотрудничества.

Основные требования к подсистеме:

* Хранение данных о партнерах (наименование, контактные данные, рейтинг и т.д.).
* Учет истории реализации продукции партнерами.
* Возможность добавления и редактирования данных о партнерах.

### 

### 1.1.2 Выбор СУБД

Для реализации базы данных была выбрана СУБД PostgreSQL из-за её надежности, поддержки сложных запросов и удобства работы с ER-диаграммами.

### 1.1.3 Проектирование структуры базы данных

На основе анализа предметной области были выделены следующие основные сущности:

* Типы продукции (product\_type) – справочник типов продукции.
* Типы партнеров (partner\_type) – справочник типов партнеров.
* Типы материалов (material\_type) – справочник типов материалов.
* Партнеры (partner) – хранит информацию о партнерах.
* Продукция (product) – хранит данные о продукции.
* Реализация продукции (partner\_products) – история реализации продукции партнерами.
* Единицы измерения (measurement\_unit) – справочник единиц измерения.
* Материалы (material) – хранит данные о материалах.

Описание таблиц:

1. product\_type

* id (PK, SERIAL): Уникальный идентификатор типа продукции.
* title (VARCHAR): Название типа продукции.
* coeff (FLOAT): Коэффициент для расчетов.

1. partner\_type

* id (PK, SERIAL): Уникальный идентификатор типа партнера.
* title (VARCHAR): Название типа партнера.

1. material\_type

* id (PK, SERIAL): Уникальный идентификатор типа материала.
* title (VARCHAR): Название типа материала.
* defect\_coeff (FLOAT): Коэффициент брака.

1. partner

* id (PK, SERIAL): Уникальный идентификатор партнера.
* partner\_type\_id (FK, INTEGER): Идентификатор типа партнера.
* title (VARCHAR): Наименование компании.
* address (VARCHAR): Юридический адрес.
* INN (VARCHAR): ИНН.
* director (VARCHAR): ФИО директора.
* phone (VARCHAR): Контактный телефон.
* email (VARCHAR): Электронная почта.
* logo (TEXT): URL логотипа.
* rating (FLOAT): Рейтинг партнера.

1. product

* id (PK, SERIAL): Уникальный идентификатор продукции.
* article (VARCHAR): Артикул продукции.
* product\_type\_id (FK, INTEGER): Идентификатор типа продукции.
* description (TEXT): Описание продукции.
* image (TEXT): URL изображения.
* title (VARCHAR): Название продукции.
* min\_cost (DECIMAL): Минимальная стоимость.
* length (FLOAT): Длина упаковки.
* width (FLOAT): Ширина упаковки.
* height (FLOAT): Высота упаковки.

1. partner\_products

* id (PK, SERIAL): Уникальный идентификатор записи.
* product\_id (FK, INTEGER): Идентификатор продукции.
* partner\_id (FK, INTEGER): Идентификатор партнера.
* quantity (INTEGER): Количество проданного товара.
* sale\_date (DATE): Дата продажи.

1. measurement\_unit

* id (PK, SERIAL): Уникальный идентификатор единицы измерения.
* title (VARCHAR): Название единицы измерения.

1. material

* id (PK, SERIAL): Уникальный идентификатор материала.
* material\_type\_id (FK, INTEGER): Идентификатор типа материала.
* title (VARCHAR): Название материала.
* pack\_count (INTEGER): Количество в упаковке.
* measurement\_unit\_id (FK, INTEGER): Идентификатор единицы измерения.
* description (TEXT): Описание материала.
* image (TEXT): URL изображения.
* price (DECIMAL): Стоимость.
* quantity\_in\_stock (INTEGER): Количество на складе.
* min\_quantity (INTEGER): Минимальное допустимое количество.

По указанной выше спецификации составлена ER-диаграмма разрабатываемой базы данных. Ее можно увидеть на рисунке 1.



Рисунок 1. ER-диаграмма реализуемой базы данных

### 1.1.4 Создание базы данных

На основе анализа предметной области и выделения ключевых сущностей был написан скрипт создания базы данных на диалекте SQL, совместимом с СУБД PostgreSQL. Скрипт включает создание таблиц, определение первичных и внешних ключей, а также настройку ограничений для обеспечения целостности данных.

Структура базы данных разработана с учетом требований к функционалу подсистемы, таких как хранение данных о партнерах, продукции, материалах и истории продаж. Все таблицы спроектированы в соответствии с третьей нормальной формой (3НФ), что минимизирует избыточность данных и обеспечивает их согласованность.

Ознакомиться со скриптом создания базы данных можно в приложении А.

## 1.2 Разработка модулей программного обеспечения для компьютерных систем

Целью выполнения данного модуля является разработка программного модуля для учета партнеров компании с использованием базы данных, созданной в рамках предыдущего модуля. Модуль должен обеспечивать функционал для просмотра списка партнеров, расчета индивидуальной скидки для каждого партнера и отображения информации из базы данных. Для реализации был выбран фреймворк Avalonia и язык программирования C#, что позволило создать кроссплатформенное приложение с современным пользовательским интерфейсом. Полный код разработанного приложения можно увидеть в репозитории проекта, находящемся по ссылке, представленной в приложении Б.

### 

### 1.2.1 Описание задачи

Задача заключается в разработке программного модуля, который:

* Подключается к базе данных, созданной в первом модуле.
* Отображает список партнеров с их данными (наименование, контактная информация, рейтинг и т.д.).
* Рассчитывает индивидуальную скидку для каждого партнера на основе общей суммы продаж за весь период работы.
* Обеспечивает единый стиль оформления интерфейса, соответствующий предоставленному руководству по стилю.

### 1.2.2 Выбор технологий

Для реализации приложения был выбран фреймворк Avalonia, который позволяет создавать кроссплатформенные приложения с современным UI. Язык программирования — C#, так как он хорошо интегрируется с Avalonia и поддерживает работу с базами данных через библиотеки, такие как Npgsql для PostgreSQL и EntityFramework для работы с сущностями различных баз данных в C# приложениях.

### 

### 1.2.3 Архитектура приложения

При разработке программного модуля был выбран паттерн проектирования MVVM (Model-View-ViewModel), который является стандартным подходом для создания приложений с использованием фреймворка Avalonia. Этот выбор обусловлен возможностью разделения ответственности, удобством тестирования MVVM приложений и согласованностью с требованиями задания.

Приложение состоит из следующих компонентов:

* Models – модели подключаемой базы данных и ее контекст.
* Views – представления
  + MainWindow – основное окно приложения, в котором происходит рендер всех остальных страниц, посредством их привязки к компоненту UserControl данной страницы.
  + ShowPartners – окно со списком партнеров
* ViewModels – интерфейсы между моделями и представлениями
  + MainWindowViewModel – главная view model, содержащая свойства для взаимодействия с базой данных, изменения заголовка окна приложения и изменения содержимого окна приложения.
  + ShowPartnersViewModel – содержит данные о партнерах, получаемые из БД и функции навигации к другим окнам приложения.

### 

### 1.2.4 Алгоритм расчета скидки

Скидка для партнера рассчитывается следующим образом:

* Если общая сумма продаж меньше 10,000 — скидка 0%.
* Если общая сумма продаж от 10,000 до 50,000 — скидка 5%.
* Если общая сумма продаж от 50,000 до 300,000 — скидка 10%.
* Если общая сумма продаж больше 300,000 — скидка 15%.

Алгоритм реализован в виде свойства в модели Partner, которое с помощью LINQ запроса находит сумму реализованного конкретным партнером товара, и с помощью условной конструкции switch приводит в соответствие диапазонам количества товаров скидку для поставщика. Код данного свойства изображен на рисунке 2.

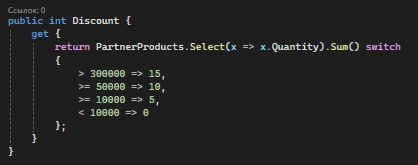


Рисунок 2. Код свойства, рассчитывающего скидку партнера

### 1.2.5 Оформление интерфейса

Интерфейс приложения выполнен в соответствии с руководством по стилю:

* Единый цветовой дизайн.
* Четкие заголовки и использование шрифта Segoe UI.
* Логотип компании размещен в верхней части главной формы.
* Установленные заголовки окон
* Использование языка разметки XAML

Интерфейс страницы списка партнеров представлен на рисунке 3.

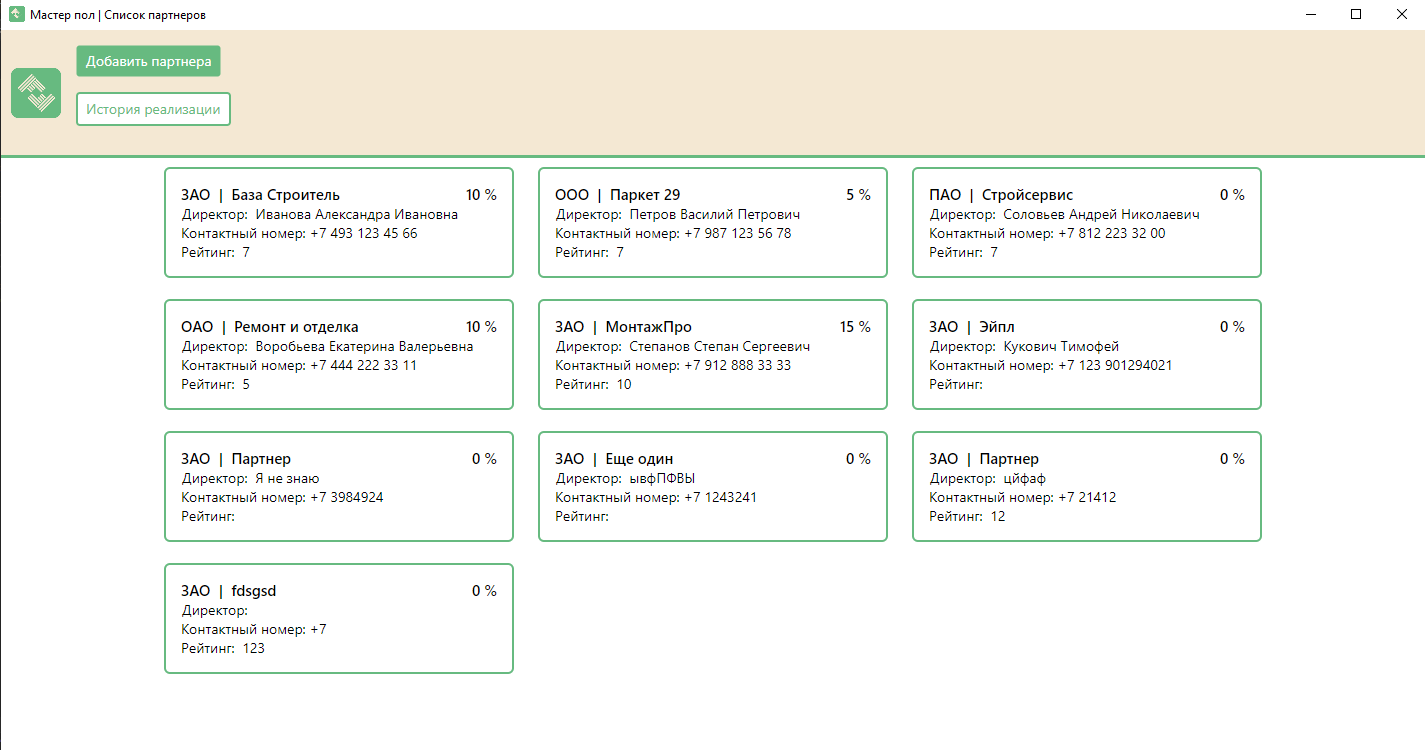


Рисунок 3. Интерфейс страницы списка партнеров

## 1.3 Сопровождение и обслуживание программного обеспечения компьютерных систем

Целью выполнения третьего модуля является разработка интерфейса программного модуля для работы с партнерами компании, обеспечивающего удобство взаимодействия пользователя с приложением. В рамках задания реализован последовательный пользовательский интерфейс с возможностью перемещения между окнами (страницами), добавлены функции добавления и редактирования данных о партнерах, а также реализована обработка исключительных ситуаций.

### 

### 1.3.1 Разработка последовательного пользовательского интерфейса

С данной целью было написано свойство Control в главной ViewModel (см. Рисунок 4), которое содержит текущий UserControl – контент страницы. Для изменения заголовков окон при перемещении между ними реализовано свойство WindowTitle, имеющее привязку к окну приложения в главном его представлении. Для обеспечения возможности навигации по нажатии кнопок ViewModel, отвечающие за взаимодействие с различными страницами, содержат методы, изменяющие свойство Control на страницу, на которую осуществляется переход.

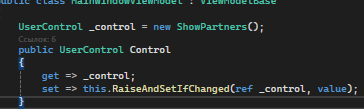


Рисунок 4. Свойство Control и пример назначения в качестве его значения страницы.

### 1.3.2 Добавление и редактирование данных о партнерах

Реализована ViewModel для редактирования и добавления данных о партнерах, в которой выбранное действие (добавление/редактирование) зависит от передачи объекта типа Partner в конструктор ViewModel. Разработанная форма показана на рисунке 5.

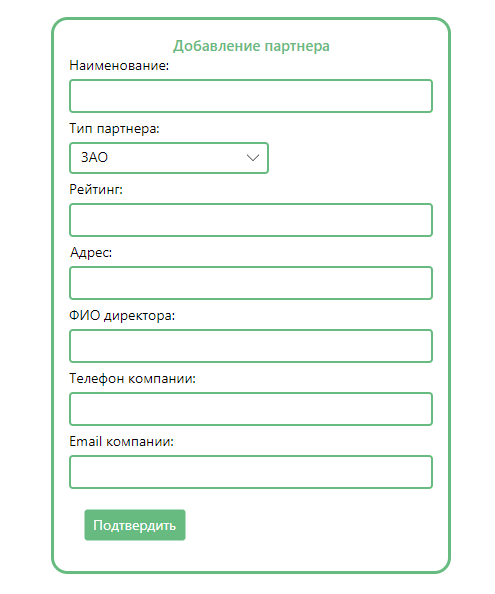


Рисунок 5. Форма добавления партнера

Связь с данными в базе осуществляется посредством привязки к полям формы соответствующих полей модели Partner, объект которой получается из базы в случае редактирования и создается в случае добавления партнера.

При этом поля, предлагающие выбор значения из списка, подгружают значения списка из базы данных и хранят выбранное значение в отдельном свойстве.

Некоторые атрибуты модели партнера экранируются на форме для предотвращения ошибок. В качестве примера такого поля можно привести рейтинг. Он должен содержать целочисленное значение, в связи с чем создано отдельное свойство для хранения строкового значения рейтинга, которое при валидации данных конвертируется в необходимый формат и присваивается партнеру.

Для валидации данных реализованы проверки каждого из полей формы (на пустое значение в случае со строковыми параметрами, на возможность конвертации в необходимый тип в случае с другими).

При этом, если введенные пользователем данные не прошли валидацию, отображается соответствующее сообщение об ошибке, как на рисунке 6.

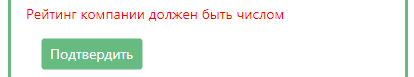


Рисунок 6. Вывод сообщения о неправильно введенных данных.

После внесения изменений или создания партнера пользователь информируется о статусе операции посредством MessageBox, как тот, что представлен на рисунке 7.

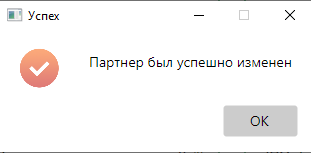


Рисунок 7. Окно, информирующее об успешном добавлении пользователя

## 1.4 Осуществление интеграции программных модулей

Целью выполнения четвертого модуля является расширение функционала системы для работы с партнерами компании путем добавления модуля вывода истории реализации продукции партнером и интеграции метода расчета количества материала, необходимого для производства продукции. В рамках задания реализованы:

* Функционал вывода истории продаж для конкретного партнера.
* Метод расчета количества материала с учетом возможного брака.

### 1.4.1 Вывод истории реализации продукции партнером

Вывод истории реализации продукции партнером выводится в отдельном окне приложения, перейти на которое можно из главного меню. На странице реализован список с возможностью выбора поставщика, информацию о реализации которого пользователь хочет посмотреть. При выборе определенного поставщика на окне отображаются все сделки партнера. Разработанный функционал продемонстрирован на рисунке 8.

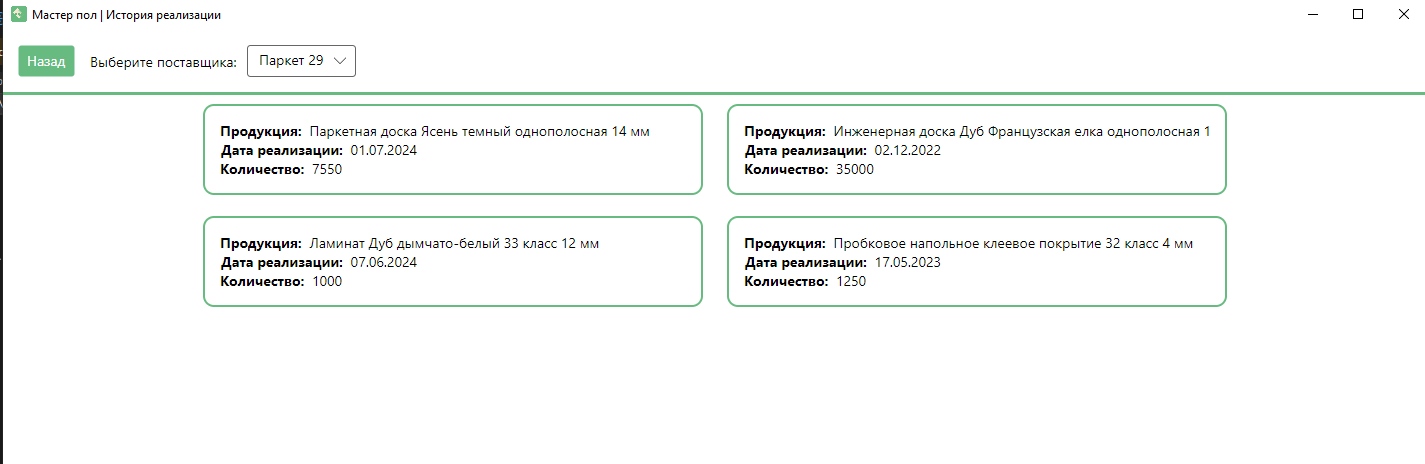


Рисунок 8. История реализации продукции партнером «Паркет 29»

На уровне алгоритма работы данного функционала главными являются свойство, отвечающее за хранение выбранного значения партнера, свойство со списком партнеров и функция получения истории реализации, фильтрующая таблицу PartnerProducts под выбранного партнера. Данная функция отрабатывает в момент изменения выбранного поставщика.

### 1.4.2 Разработка метода расчета количества материала, необходимого для производства продукции

Данный метод был разработан в библиотеке классов C# (полный код которой можно найти по ссылке в приложении Б, а также на рисунке 9).

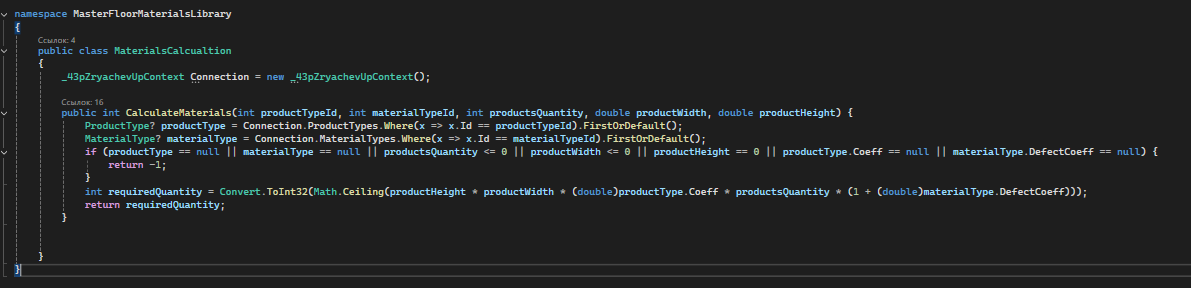


Рисунок 9. Метод расчета количества материалов

В классе, содержащем метод, объявлен объект контекста, отвечающий за соединение с базой данных и получение данных о материалах.

Разработанный метод содержит следующие входные параметры:

* productTypeId – идентификатор типа продукции, целочисленное значение.
* materialTypeId – идентификатор типа материала, целочисленное значение.
* productQuantity – количество продукции, которое необходимо получить, целочисленное значение.
* productWidth – ширина получаемой продукции, число с плавающей точкой.
* productHeight – высота получаемой продукции, число с плавающей точкой.

Объяснение работы алгоритма: в начале происходит попытка получения материала и продукции из базы данных по переданным идентификаторам, если они не найдены, переменным, хранящим их значение присваивается значение null. Далее происходит валидация всех переданных параметров, если какой-то из них не проходит проверку – метод возвращает -1.

Число необходимых материалов вычисляется как площадь продукции (произведение ширины и высоты), умноженная на коэффициент продукции и на количество необходимой продукции, увеличенная с учетом коэффициента брака определенного типа материала. Полученное значение округляется с избытком и конвертируется в целочисленное значение.

Реализованная программа показала правильные результаты при вызове разработанного метода в консольном приложении, в чем можно убедиться на рисунке 10.

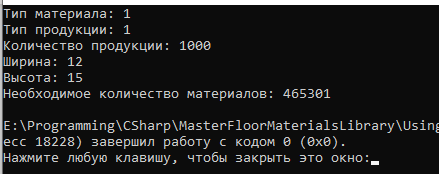


Рисунок 10. Проверка корректной работы реализованной функции расчета материалов

### 1.4.3 Тестирование функции расчета количества материалов

Для разработанной ранее функции написан проект тестов MSTest, содержащий 10 простых и 5 сложных модульных тестов для проверки работоспособности метода.

Разработанные тесты содержат различные типы проверок, среди которых:

* Проверка соответствия ожидаемых результатов полученным.
* Проверка возвращения позитивного значения количества материалов при корректных параметрах
* Проверка возвращения негативного значения количества материалов при некорректных параметрах
* Проверка возвращения ненулевого результата
* Проверка возвращения целочисленного значения количества материалов
* Проверка вызова исключений разных типов при некорректной работе метода.

Результат прохождения всех тестов представлен на рисунке 11.

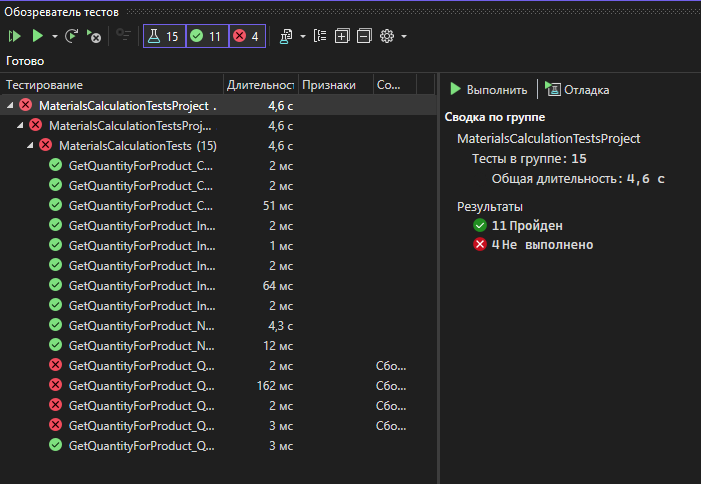


Рисунок 11. Сводка о прохождении тестов

Как можно увидеть, большинство тестов отработали корректно, а те из них, которые провалились, проверяют исключения, которые не генерируются из-за проверок входных параметров компилятором.

## 1.5 Тестирование редактирования партнера

Данный модуль направлен на проверку функционала редактирования информации о партнере в системе. Он включает в себя:

1. Валидацию данных

* Обязательные поля: название, адрес, директор, телефон, email.
* Корректность формата телефонного номера и email.
* Корректность рейтинга (числовое значение, неотрицательное).

1. Логику сохранения изменений

* Обновление данных партнера в базе при корректном вводе.
* Блокировку сохранения при наличии ошибок.

1. Работу с типами партнеров

* Выбор типа из списка, обработка случая пустого списка.

1. Обработку граничных условий

* Максимальная/минимальная длина полей.
* Спецсимволы в названии компании или адресе.

1. Интеграцию с базой данных

* Проверка сохранения/обновления данных.
* Обработка исключений (например, сбоев подключения).

Критерии оценки тестов:

* Успешное обновление данных при корректном вводе.
* Отображение ошибок при нарушении требований.
* Сохранение целостности данных (например, невозможность задать отрицательный рейтинг).
* Устойчивость к некорректным форматам телефонов/email.

Реализовано 5 тест кейсов, проверяющих функционал редактирования пользователя, один из которых представлен на рисунке 12.

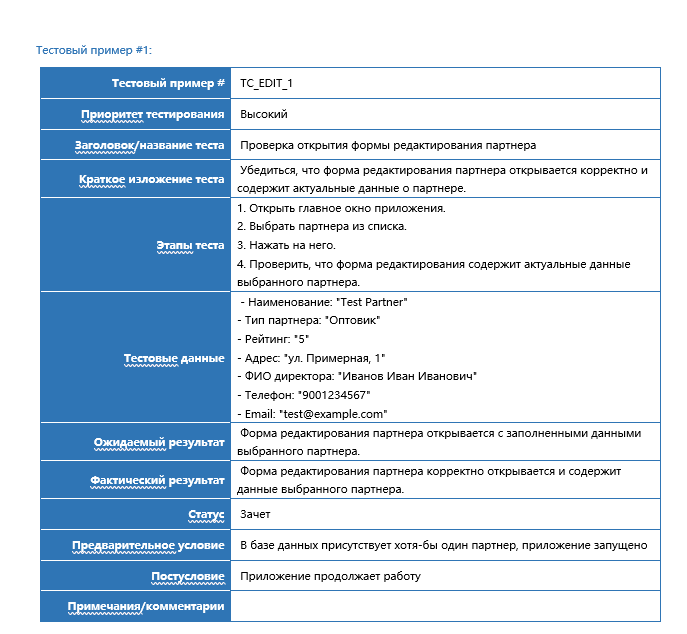


Рисунок 12. Тест кейс для проверки открытия формы редактирования

На примере представленного тест кейса можно увидеть, что каждый из них содержит уникальный идентификатор, приоритет тестирования, название теста, его краткое изложение, этапы, тестовые данные, ожидаемый и фактический результаты, статус, предварительное условие и постусловие.

Подробнее с разработанными тестами можно ознакомиться в репозитории, ссылка на который предоставлена в приложении Б.

По результатам написания тест-кейсов и проверки описанной в них функциональности каждый проверенный аспект отработал корректно, что может судить о безотказной работе тестируемого функционала.

## 1.6 Построение UML диаграмм

В качестве целей этого модуля поставлены: построение диаграммы использования (Use Case) для визуализации взаимодействия пользователей с системой, построение диаграммы деятельности (Activity) для описания ключевых процессов системы.

Основные требования:

Три роли пользователей: клиент, тренер, администратор.

Функции: регистрация, управление тренерами, подача заявок на тренировки, формирование планов, отслеживание прогресса.

В ходе реализации поставленных целей реализована диаграмма вариантов использования, представленная на рисунке 13.

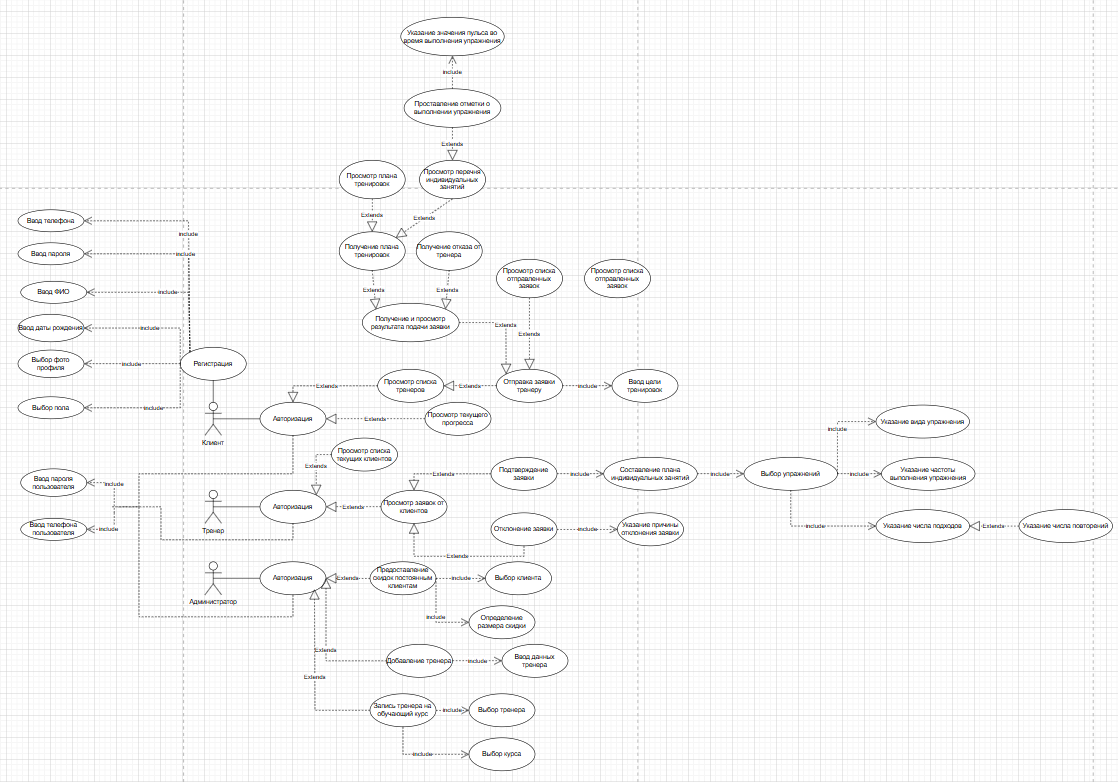


Рисунок 13. Диаграмма вариантов использования

Данная диаграмма в полной мере отображает пространство действий, доступное для обозначенных ролей.

Помимо этого, была реализована диаграмма деятельности, показанная на рисунке 14.

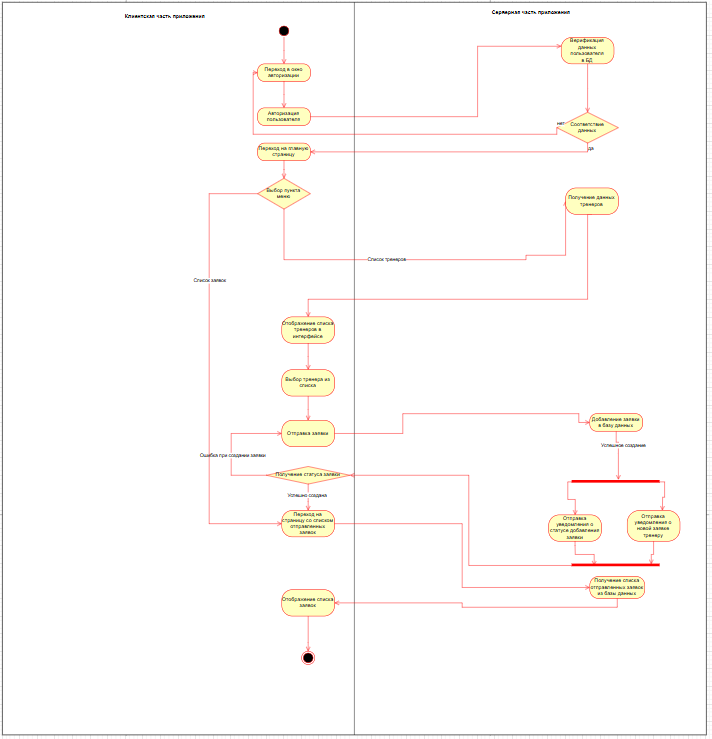


Рисунок 14. Диаграмма деятельности

Данная диаграмма позволяет отследить взаимосвязь серверной и клиентской части приложения фитнес центра на примере авторизации и создания заявки на тренера клиентом.

Разработанные диаграммы в полной мере отражают функционал приложения фитнес центра.

# 2 ПОДДЕРЖКА И ТЕСТИРОВАНИЕ ПРОГРАММНЫХ МОДУЛЕЙ

## 2.1 Разработка и создание базы данных

База данных создана в соответствии с предметной областью и предоставленными данными. В качестве СУБД выбрана PostgreSQL.

Созданы следующие таблицы:

1. role - хранит роли пользователей.

Столбцы:

* id: Уникальный идентификатор роли (первичный ключ).
* name: Название роли.

2. user - хранит информацию о пользователях.

Столбцы:

* id: Уникальный идентификатор пользователя (первичный ключ).
* surname, name, patronymic: ФИО пользователя.
* login: Логин пользователя.
* password: Пароль пользователя.
* role\_id: Ссылка на роль пользователя (внешний ключ на таблицу role).

3. manufacturer - хранит производителей товаров.

Столбцы:

* id: Уникальный идентификатор производителя (первичный ключ).
* name: Название производителя.

4. supplier - хранит поставщиков товаров.

Столбцы:

* id: Уникальный идентификатор поставщика (первичный ключ).
* name: Название поставщика.

5. category - хранит категории товаров.

Столбцы:

* id: Уникальный идентификатор категории (первичный ключ).
* name: Название категории.

6. measurement\_unit - хранит единицы измерения товаров.

Столбцы:

* id: Уникальный идентификатор единицы измерения (первичный ключ).
* name: Название единицы измерения.

7. product - хранит информацию о товарах.

Столбцы:

* article\_number: Артикул товара (первичный ключ).
* name: Название товара.
* category\_id: Ссылка на категорию товара (внешний ключ на таблицу category).
* manufacturer\_id: Ссылка на производителя товара (внешний ключ на таблицу manufacturer).
* supplier\_id: Ссылка на поставщика товара (внешний ключ на таблицу supplier).
* measurement\_unit\_id: Ссылка на единицу измерения товара (внешний ключ на таблицу measurement\_unit).
* cost: Стоимость товара.
* current\_discount: Текущая скидка на товар.
* max\_discount: Максимальная скидка на товар.
* quantity\_in\_stock: Количество товара на складе.
* description: Описание товара.
* photo: Фотография товара (хранится в виде бинарных данных).

8. status - хранит статусы заказов.

Столбцы:

* id: Уникальный идентификатор статуса (первичный ключ).
* name: Название статуса.

9. pickup\_point - хранит пункты выдачи заказов.

Столбцы:

* id: Уникальный идентификатор пункта выдачи (первичный ключ).
* address: Адрес пункта выдачи.

10. order - хранит информацию о заказах.

Столбцы:

* id: Уникальный идентификатор заказа (первичный ключ).
* status\_id: Ссылка на статус заказа (внешний ключ на таблицу status).
* date: Дата создания заказа (по умолчанию текущее время).
* delivery\_date: Дата доставки заказа.
* pickup\_point: Ссылка на пункт выдачи (внешний ключ на таблицу pickup\_point).
* user\_id: Ссылка на пользователя, сделавшего заказ (внешний ключ на таблицу user).
* pickup\_code: Код для получения заказа.

11. order\_products - хранит связь между заказами и товарами (многие ко многим).

Столбцы:

* order\_id: Ссылка на заказ (внешний ключ на таблицу order).
* product\_article\_number: Ссылка на товар (внешний ключ на таблицу product).
* product\_quantity: Количество товара в заказе.

Первичный ключ: Комбинация order\_id и product\_article\_number.

По указанной выше спецификации составлена ER-диаграмма разрабатываемой базы данных. Ее можно увидеть на рисунке 15.



Рисунок 15. ER-диаграмма созданной базы

Написан скрипт для создания базы данных, представленный в приложении В.

### 2.1.1 Предобработка данных и заполнение базы

Предобработка данных выполнена с применением программы Microsoft Excel. Данные внесены в базу средствами DBeaver. Заполненные таблицы представлены в Приложении Г.

## 2.2 Разработка приложения

Для реализации выбран язык программирования C# и фреймворк Avalonia.NET. Для подключения к базе данных используется Microsoft Entity Framework, в качестве провайдера Npgsql Entity Framework Core PostgreSQL. Код проекта предоставлен в репозитории (Приложение Б, часть 2).

### 2.2.1 Модуль авторизации

Был разработан модуль, отвечающий за авторизацию пользователей. На форме авторизации присутствуют поля ввода логина и пароля (См. Рисунок 1). Пользователь может авторизоваться под своей учетной записью или выполнить вход в гостевой аккаунт. По нажатии кнопки «Войти» происходит валидация полей формы, и, если валидация прошла успешно, в базе с помощью LINQ запроса ищется запись по введенным данным. Авторизация считается успешной если учетная запись найдена. В таком случае происходит переход на страницу списка товаров. При осуществлении входа в качестве гостя из базы получаются данные гостевой учетной записи. Id и название роли пользователя при авторизации сохраняются глобально и остаются доступными на протяжении всей пользовательской сессии.

Также предусмотрен механизм генерации и ввода капчи. Если пароль и логин не соответствуют данным пользователя, ему предлагается ввести данные заново и пройти капчу (См. Рисунок 2). Если капча не пройдена верно или данные все еще неправильные, пользователю предлагается подождать 10 секунд перед следующей попыткой ввода (См. Рисунок 3). Поля в этом случае очищаются и становятся недоступными для ввода. Капча отображается в элементе Canvas. При этом отрисовка элементов осуществляется следующим образом: в области всего холста рисуются линии случайного цвета и длины, символы отрисовываются в количестве 4 единиц, могут быть повернуты в сторону, имеют перечеркнутый вид.

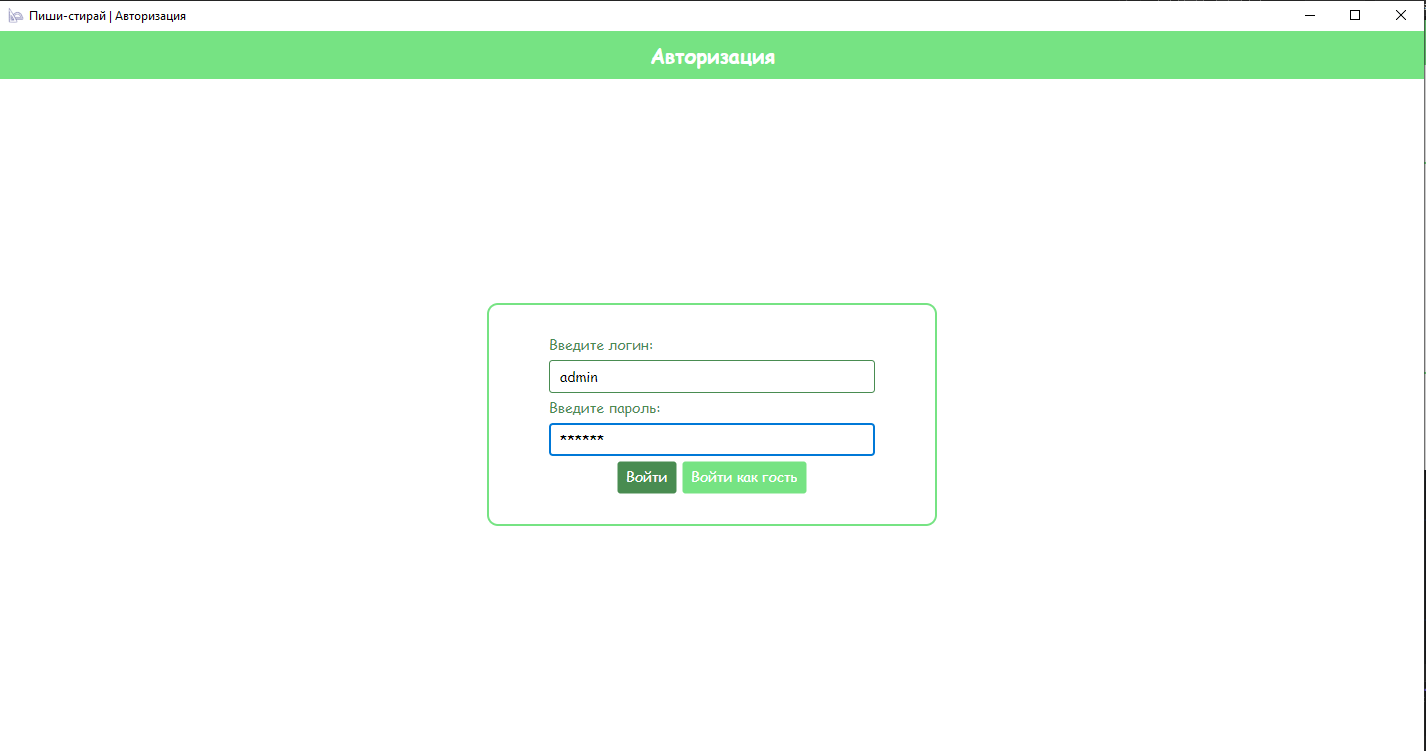


Рисунок 16. Форма авторизации

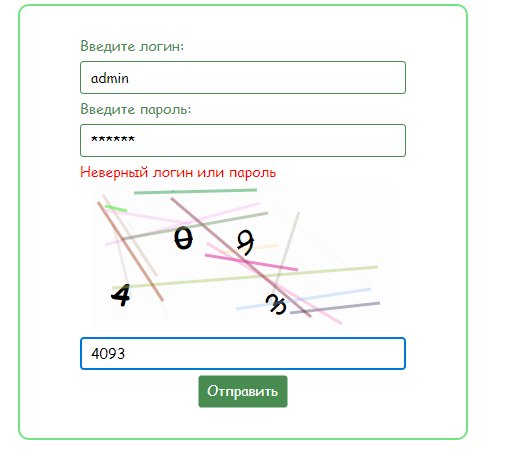


Рисунок 17. Механизм ввода и отображения капчи

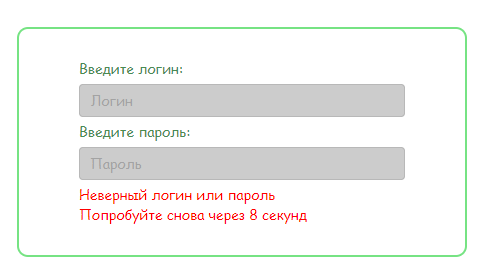


Рисунок 18. Блокировка ввода на 10 секунд при повторном неправильном вводе.

### 2.2.2 Список товаров

Реализована страница списка товаров. В нем выведены все необходимые поля (См. Рисунок 20), обеспечен условный рендеринг элементов (если скидка больше 15% ее фон зеленый, если есть скидка, цена на товар рассчитывается с учетом нее).

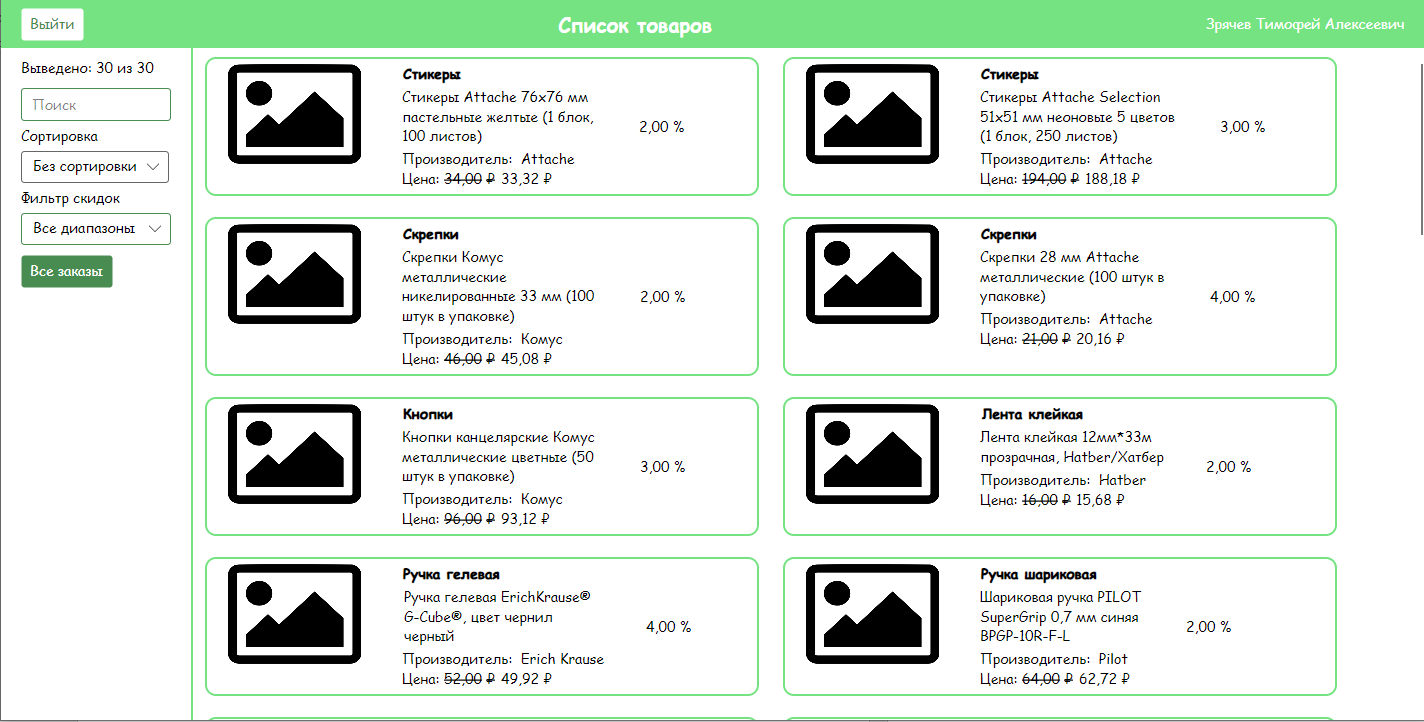


Рисунок 19. Список товаров

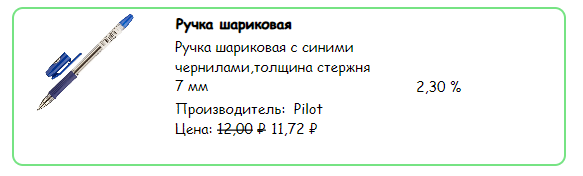


Рисунок 20. Карточка товара

Реализованы фильтры списка товаров. Среди них, поиск по названию, сортировка по стоимости (по возрастанию и по убыванию) и фильтрация по скидке. На рисунке 21 представлен вывод товаров с учетом выбранных фильтров.

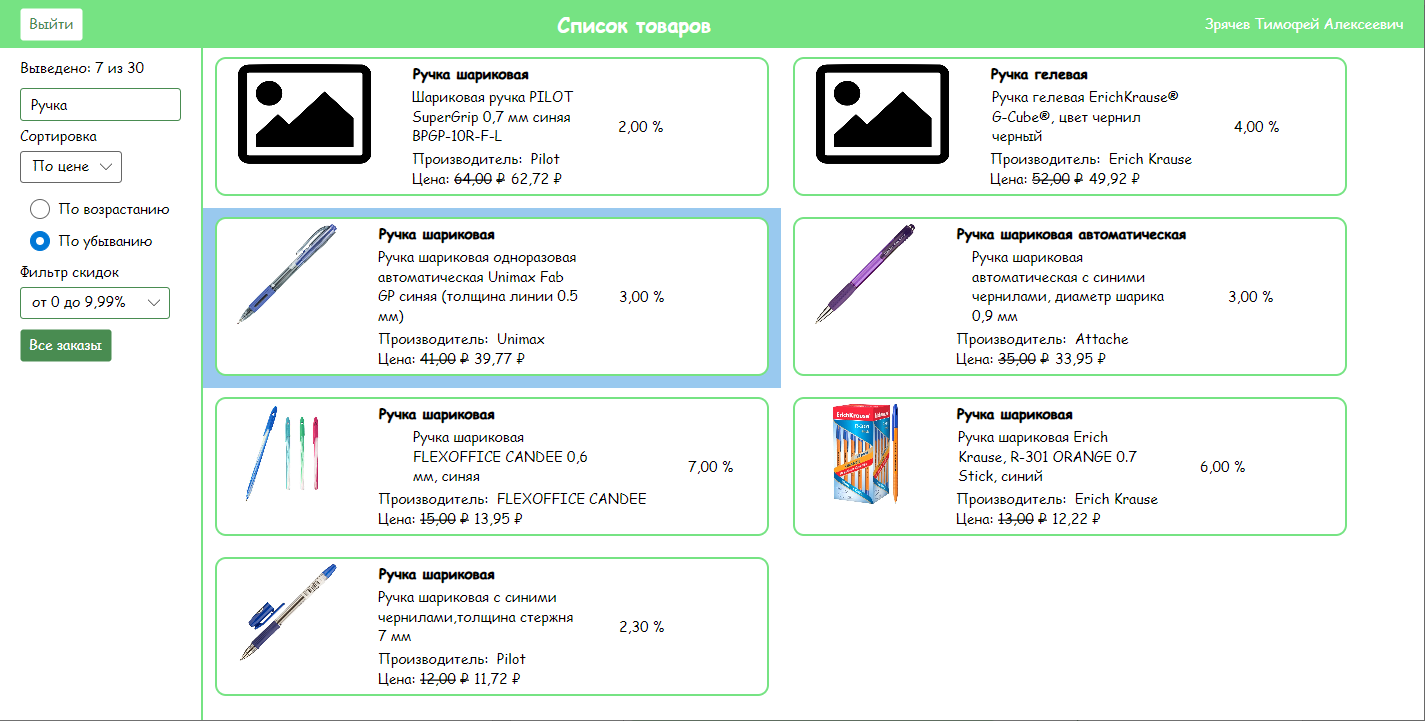


Рисунок 21. Вывод данных с применением фильтрации

Реализовано условное появление кнопки просмотреть заказ (См. Рисунок 21 слева), зависящее от наличия у текущего пользователя незавершенного заказа.

Реализовано добавление товара к заказу. Если уже был найден активный заказ, то товары добавляются к нему. В противном случае создается пустой активный заказ, куда и добавляются товары. Добавление происходит при выборе пункта добавить к заказу в контекстном меню, которое вызывается при клике правой кнопки мыши по товару (См. Рисунок 22).

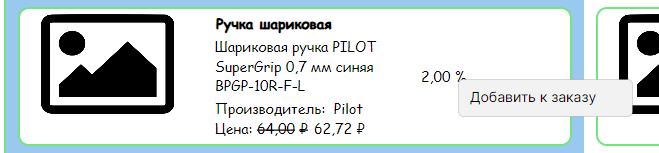


Рисунок 22. Контекстное меню добавления товара к заказу.

### 2.2.3 Просмотр и оформление заказа

Реализовано окно просмотра текущего заказа (Рисунок 1). На нем отображаются все товары, добавленные в заказ, их количество, а также общая информация о заказе: его номер, сумма со скидкой, скидка, пункт выдачи (который можно изменить). Количество товара в заказе также меняется по нажатию кнопок с иконками «+» и «-». Помимо этого, товар можно вообще убрать из заказа по нажатии кнопки с иконкой мусорного ведра или, сделав количество товара равным нулю. При этом, удаление товара из заказа не будет произведено, если пользователь не подтвердил это в message box.

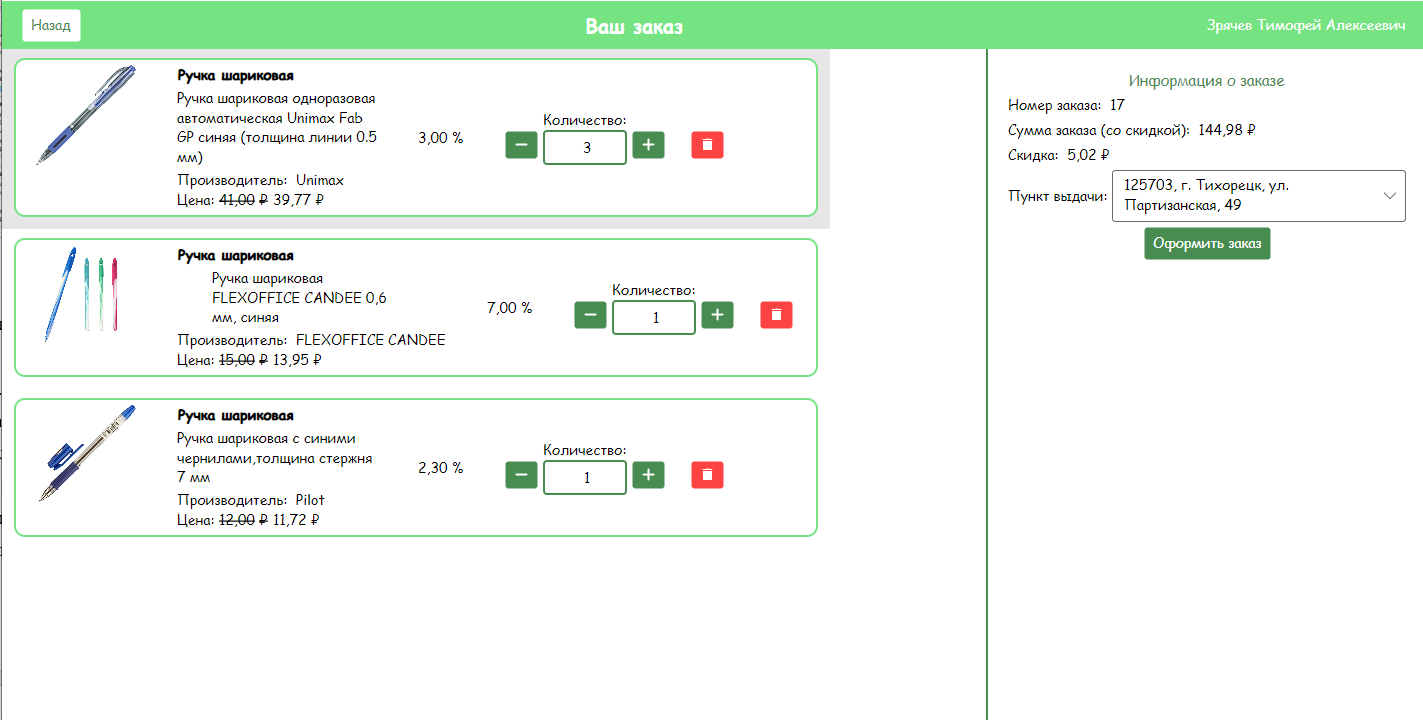


Рисунок 23. Окно просмотра заказа

По нажатии кнопки «Оформить заказ» формируется талон на получение заказа, заказ становится завершенным. Устанавливается дата получения заказа: если все товары из заказа присутствуют на складе в количестве более 3 единиц, срок доставки устанавливается равным 3 дням, в противном случае срок доставки равен 6 дням.

В талон выводится вся актуальная информация о заказе: его номер, ФИО клиента (если заказ оформил авторизованный пользователь), дата заказа, сумма заказа (без скидки), скидка, итоговая сумма заказа, пункт выдачи, код получения заказа (генерируется случайным образом) и состав заказа с наименованием товаров в нем, их стоимостью и количеством.

Генерируемый талон на получение заказа представлен на рисунке 24.

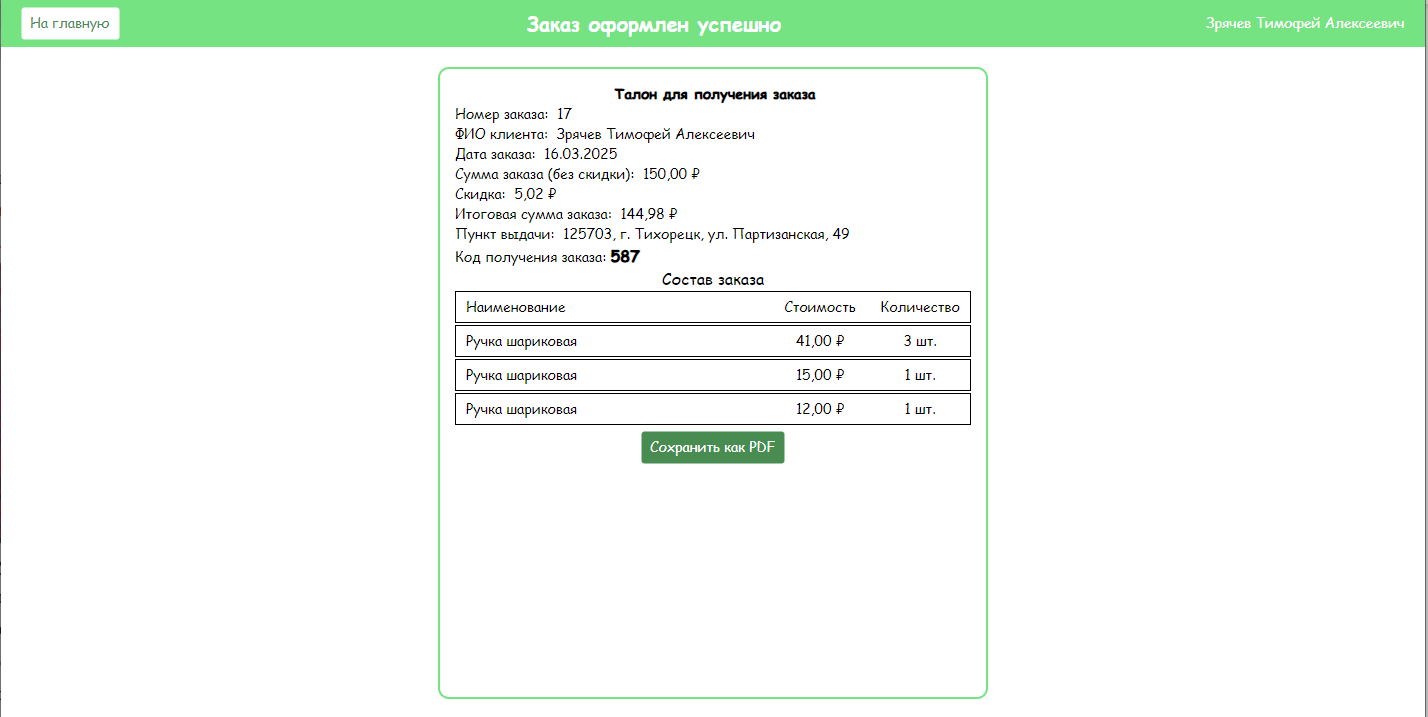


Рисунок 24. Талон на получение заказа

Пользователю предоставляется возможность сохранить талон на получение заказа в формате PDF по нажатии кнопки. При осуществлении этого действия пользователю выводится диалог выбора места сохранения файла, формируется и сохраняется pdf документ и заполняется данными текущего заказа. Сгенерированный PDF талон можно увидеть на рисунке 25.

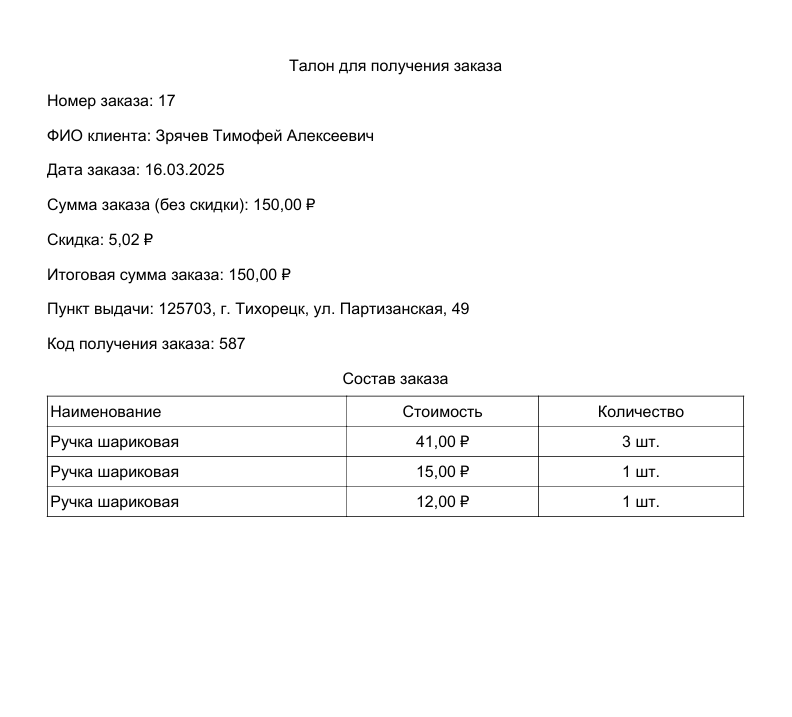


Рисунок 25. Талон в pdf формате

Окно просмотра заказов

Пользователям с ролью администратор и менеджер предоставляется возможность просмотра списка всех заказов. Переход в список осуществляется из окна списка товаров по нажатии кнопки «Все заказы». Реализованное окно можно увидеть на рисунке 26.

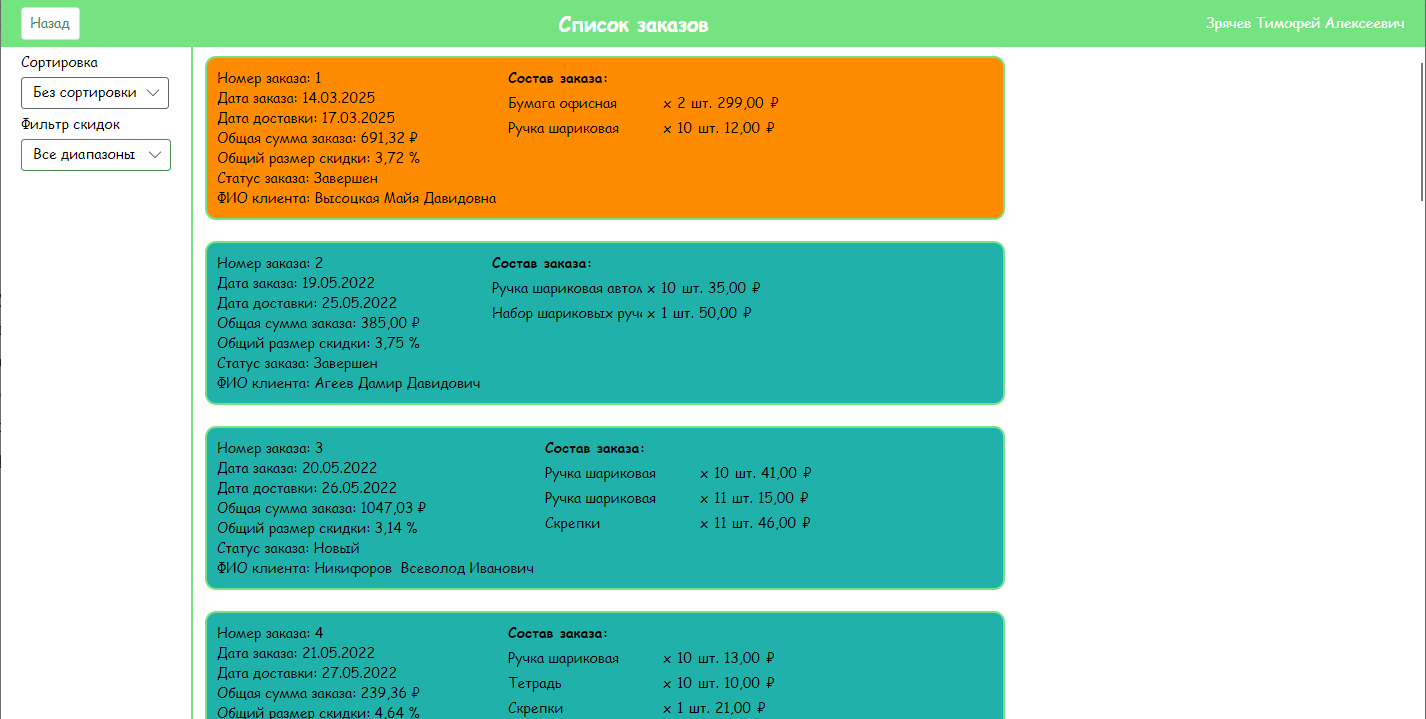


Рисунок 26. Список заказов

Как можно заметить, предусмотрена возможность фильтрации заказов по скидке и сортировка по стоимости (работу фильтров можно увидеть на рисунке 27). Если хотя бы один товар из заказа отсутствует на складе, фон заказа становится оранжевым, а если все товары из заказа есть на складе в количестве более 3 единиц – бирюзовым.

На карточке заказа отображена вся необходимая информация о нем: номер, дата оформления, дата доставки, общая сумма заказа, общий размер скидки, статус, ФИО клиента и состав заказа.

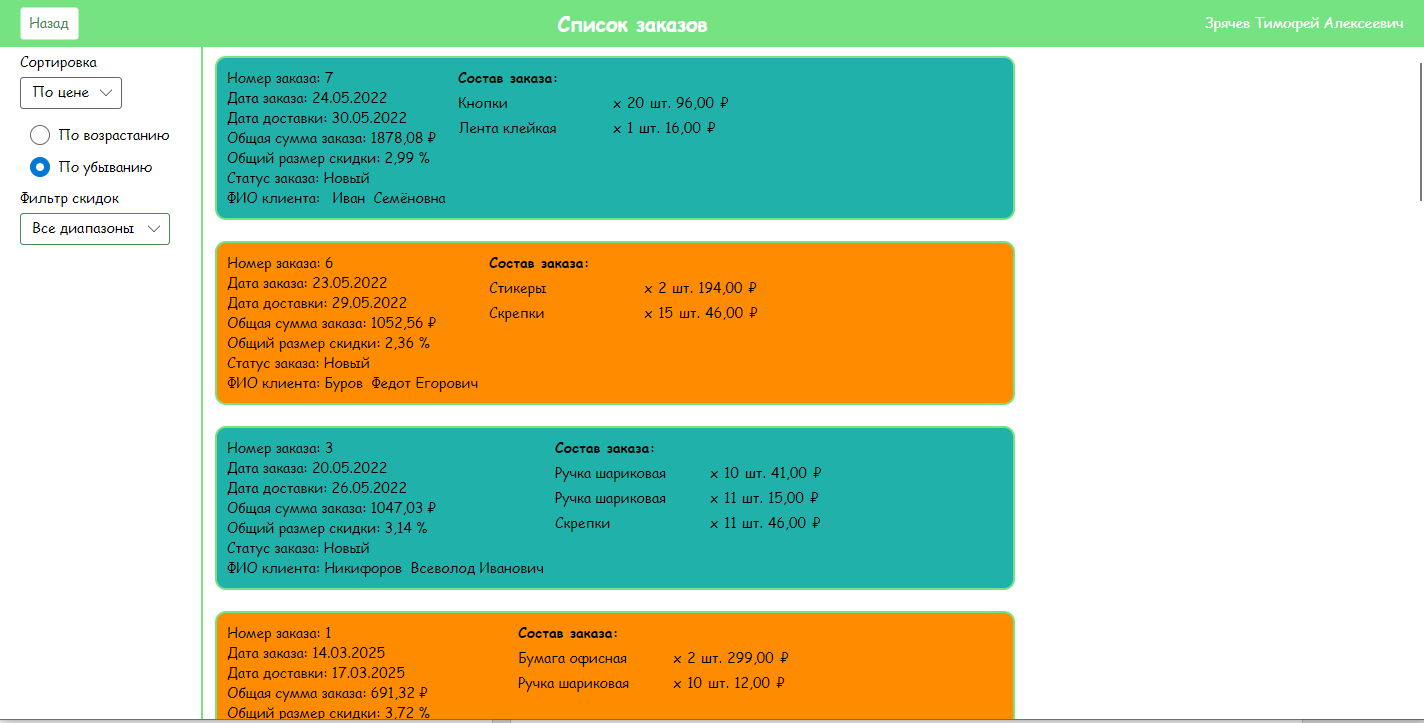


Рисунок 27. Отфильтрованный список заказов

## 2.3 Тестирование программных решений

### 2.3.1 Разработка метода расчета свободных временных интервалов для формирования оптимального графика работы сотрудников

Данный метод был разработан в библиотеке классов C# (полный код которой можно найти по ссылке в приложении Б, а также на рисунке 28).

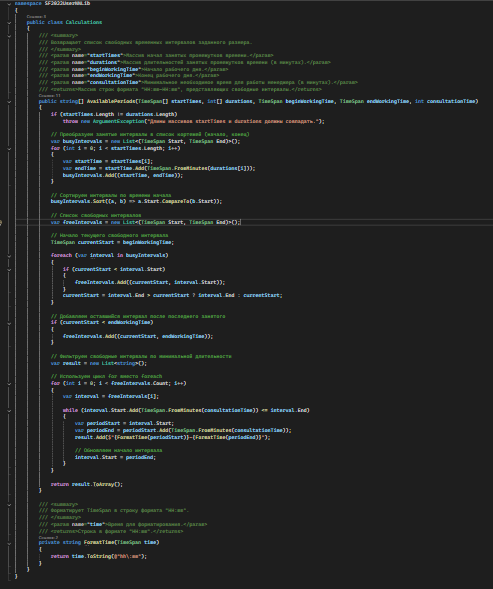


Рисунок 28. Метод расчета свободных временных интервалов

В классе, содержащем метод, реализован алгоритм, позволяющий определить список свободных временных интервалов заданного размера в графике сотрудника.

Описание разработки

Разработанная библиотека классов реализована в соответствии с требованиями к именованию и структуре. Она содержит один основной класс Calculations с методом AvailablePeriods, который выполняет расчет свободных временных интервалов.

Структура библиотеки:

Название библиотеки: SF2022UserNNLib.dll

Название класса: Calculations

Название метода: AvailablePeriods

Входные параметры

Метод AvailablePeriods принимает следующие входные параметры:

startTimes – массив начал занятых промежутков времени (TimeSpan[]).

durations – массив длительностей занятых промежутков времени в минутах (int[]).

beginWorkingTime – начало рабочего дня (TimeSpan).

endWorkingTime – конец рабочего дня (TimeSpan).

consultationTime – минимальное необходимое время для работы менеджера в минутах (int).

Выходные параметры

Метод возвращает массив строк формата "HH:mm-HH:mm", представляющих свободные временные интервалы.

Объяснение работы алгоритма

Валидация входных данных:

В начале метода проверяется, что длины массивов startTimes и durations совпадают. Если это условие не выполняется, выбрасывается исключение ArgumentException.

Преобразование занятых интервалов:

Занятые временные интервалы преобразуются в список кортежей (Start, End), где Start – начало интервала, а End – его окончание.

Сортировка интервалов:

Список занятых интервалов сортируется по времени начала для удобства дальнейшей обработки.

Определение свободных интервалов:

Начало текущего свободного интервала устанавливается равным beginWorkingTime.

Для каждого занятого интервала проверяется, есть ли промежуток между текущим началом свободного интервала и началом занятого интервала. Если такой промежуток существует, он добавляется в список свободных интервалов.

После обработки всех занятых интервалов проверяется оставшийся промежуток между концом последнего занятого интервала и endWorkingTime.

Фильтрация по минимальной длительности:

Свободные интервалы фильтруются по минимальной длительности (consultationTime). Для каждого подходящего интервала генерируются подинтервалы длиной consultationTime и добавляются в результат.

Форматирование времени:

Для форматирования времени используется вспомогательный метод FormatTime, который преобразует объект TimeSpan в строку формата "HH:mm".

Проверка корректности работы метода

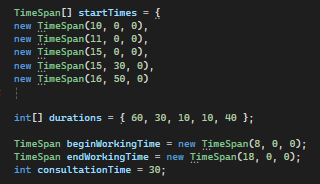


Рисунок 29. Входные параметры для оценки корректной работы метода

Реализованная программа была протестирована в консольном приложении. На рисунке 30 показан пример вызова метода с указанными выше входными данными и полученными результатами.

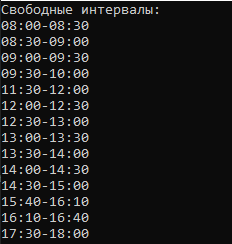


Рисунок 30. Проверка корректной работы реализованной функции расчета свободных интервалов

### 2.3.2 Разработка модульных тестов к библиотеке

Для разработанной ранее функции AvailablePeriods был создан проект тестов MSTest, содержащий 10 модульных тестов (они представлены в репозитории проекта – Приложение Б Часть 2). Эти тесты направлены на проверку работоспособности метода в различных сценариях использования.

Типы проверок, выполненных в рамках тестирования

Разработанные тесты охватывают следующие типы проверок:

Проверка соответствия ожидаемых результатов полученным:

Все тесты содержат сравнение результата выполнения метода с заранее определенными ожидаемыми значениями. Это обеспечивает корректность работы алгоритма в различных сценариях.

Пример: Тест Test\_AvailablePeriods\_ReturnsNoBusyIntervals проверяет, что при отсутствии занятых интервалов все временные промежутки рабочего дня доступны.

Проверка возвращения корректного списка интервалов при наличии занятых промежутков:

Тесты проверяют, что метод корректно обрабатывает случаи с одним или несколькими занятыми интервалами, а также пересекающимися интервалами.

Пример: Тест Test\_AvailablePeriods\_ReturnsSingleBusyInterval проверяет, что свободные интервалы находятся до и после одного занятого времени.

Проверка возвращения пустого списка при полностью занятом рабочем дне:

Тест Test\_AvailablePeriods\_ReturnsNoFreeIntervals проверяет, что метод возвращает пустой массив строк, если весь рабочий день занят.

Проверка обработки граничных условий:

Тесты проверяют корректность работы метода на границах рабочего дня (начало и конец).

Пример: Тест Test\_AvailablePeriods\_ReturnsFreeIntervalsWithEdgeCases проверяет, что метод корректно обрабатывает случаи, когда занятые интервалы начинаются или заканчиваются на границах рабочего дня.

Проверка фильтрации интервалов по минимальной длительности:

Тесты проверяют, что метод корректно фильтрует свободные интервалы по заданной минимальной длительности (consultationTime).

Пример: Тест Test\_AvailablePeriods\_ReturnsFreeIntervalsWithDifferentConsultationTime проверяет, что метод возвращает интервалы только той длины, которая соответствует минимальному времени консультации.

Проверка обработки пересекающихся интервалов:

Тест Test\_AvailablePeriods\_ReturnsFreeIntervalsWithOverlappingBusyIntervals проверяет, что метод корректно обрабатывает пересечения между занятыми интервалами.

Проверка вызова исключений при некорректных входных данных:

Тесты проверяют, что метод выбрасывает исключение ArgumentException, если длины массивов startTimes и durations не совпадают.

Проверка возвращения ненулевого результата:

Тесты проверяют, что метод возвращает ненулевой результат при наличии свободных интервалов.

Проверка формата выходных данных:

Тесты проверяют, что выходные данные представлены в формате "HH:mm-HH:mm".

Проверка работы с малыми значениями минимального времени консультации:

Тест Test\_AvailablePeriods\_ReturnsFreeIntervalsWithShortConsultationTime проверяет, что метод корректно обрабатывает случаи с очень маленьким значением consultationTime.

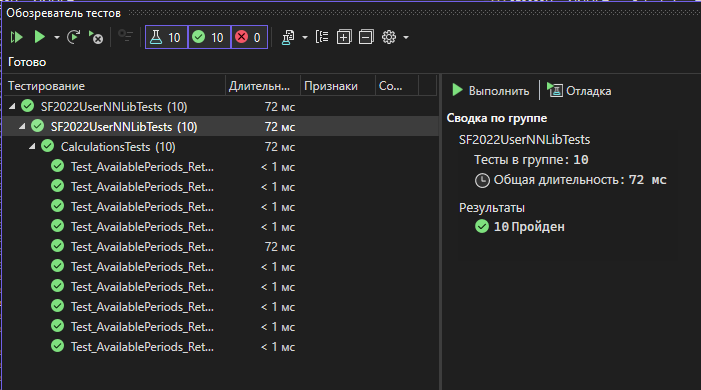


Рисунок 31. Сводка о прохождении тестов

### 2.3.3 Тестирование добавления товара администратором

Для выполнения процедуры тестирования добавления товара администратором были разработаны пять тестовых сценариев. Эти сценарии охватывают как успешное добавление товара, так и случаи, когда добавление отклоняется согласно требованиям предметной области. Каждый тестовый сценарий демонстрирует различные исходы работы алгоритма.

Общие требования к тестированию

1. Предварительные условия:

* Администратор авторизован в системе.
* База данных доступна для записи.

1. Постусловия:

* Если товар успешно добавлен, он сохраняется в базе данных.
* Если добавление отклонено, товар не сохраняется, и пользователь получает соответствующее сообщение об ошибке.

1. Тестовые данные:

* Все обязательные поля должны быть заполнены корректно.
* Некорректные данные (например, отрицательная стоимость или пустое название) должны вызывать ошибку.

Реализовано 5 тест кейсов, проверяющих функционал добавления товара администратором, один из которых представлен на рисунке 32.

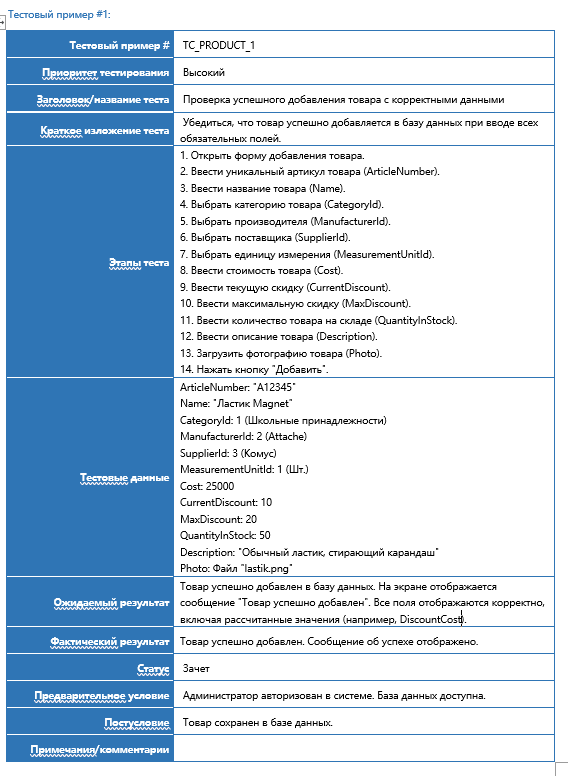


Рисунок 32. Тест кейс для проверки добавления товара с корректными данными

На примере представленного тест кейса можно увидеть, что каждый из них содержит уникальный идентификатор, приоритет тестирования, название теста, его краткое изложение, этапы, тестовые данные, ожидаемый и фактический результаты, статус, предварительное условие и постусловие.

Подробнее с разработанными тестами можно ознакомиться в репозитории, ссылка на который предоставлена в приложении Б (часть 2).

По результатам написания тест-кейсов и проверки описанной в них функциональности каждый проверенный аспект отработал корректно, что может судить о безотказной работе тестируемого функционала.

# 3 РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

В рамках учебной практики выполнена разработка мобильного приложения на языке программирования Kotlin с использованием технологии Jetpack Compose для создания пользовательского интерфейса. Приложение взаимодействует с сервером через платформу Supabase, обеспечивая функциональность регистрации, авторизации, а также работы с каталогом товаров, избранными товарами и профилем пользователя.

Разработанное приложение соответствует предоставленным макетам и требованиям, указанным в задании, соблюдает принципы модульности, чистой архитектуры и минимальной поддерживаемой версии Android 11.0.

Ознакомиться с кодом приложения можно в репозитории, ссылка на который находится в приложении Б (часть 3).

## 3.1 Верстка приложения

Согласно макетам, предоставленным по ссылке https://www.figma.com/design/t7yNd853ZuGihgKLNexZSr/UP-01.03?node-id=1-2&t=7U5UF6Zit4S5xnSy-1, разработаны экраны приложения:

Splash: Заставка приложения (Рисунок 33). На нем осуществляется вывод лого изображения и в течении 1 секунды осуществляется перенаправление пользователя на страницу авторизации.



Рисунок 33. Заставочный экран приложения

Sign In (Рисунок 34): Экран входа. На нем присутствуют поля ввода данных пользователя: email и пароля, и кнопка войти. Доступно сокрытие пароля по нажатию на иконку глаза. На странице присутствует валидация вводимых данных: email проверяется с помощью регулярного выражения, а пароль на заполненность с помощью функций, предоставляемых языком. Со страницы можно перейти на экран сброса пароля или регистрации.

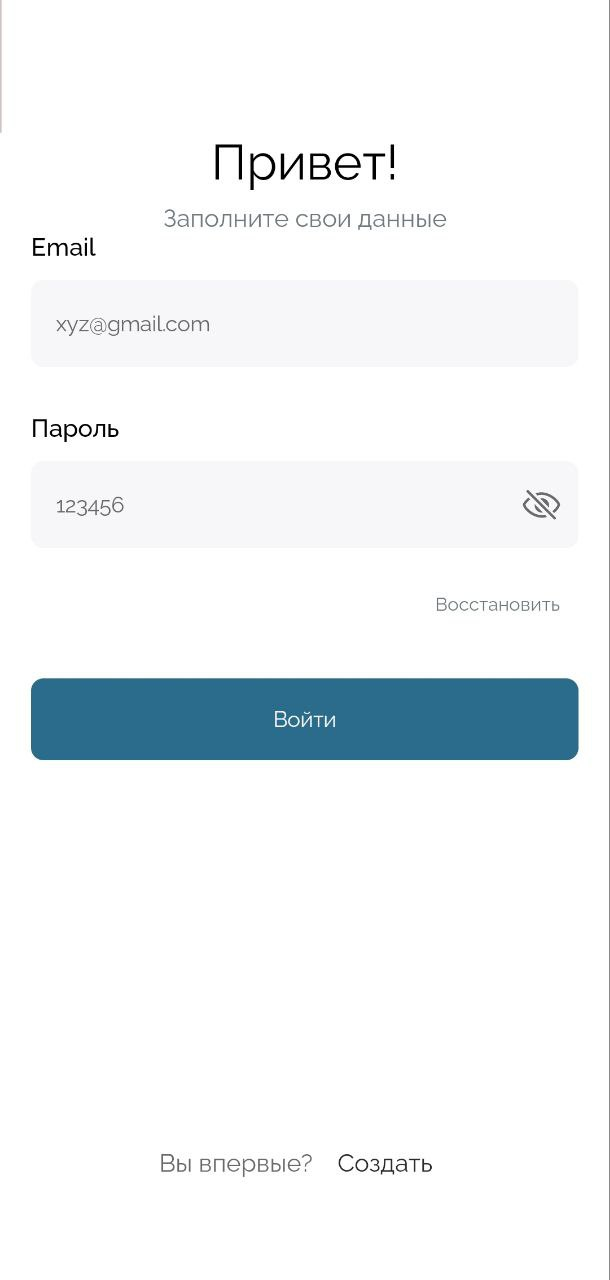


Рисунок 34. Экран авторизации

Register Account (Рисунок 35): Экран регистрации. На нем присутствуют поля ввода данных пользователя: email, пароля, имени пользователя, и кнопка войти. Доступно сокрытие пароля по нажатию на иконку глаза. На странице присутствует валидация вводимых данных: email проверяется с помощью регулярного выражения, а пароль и имя пользователя на заполненность с помощью функций, предоставляемых языком.

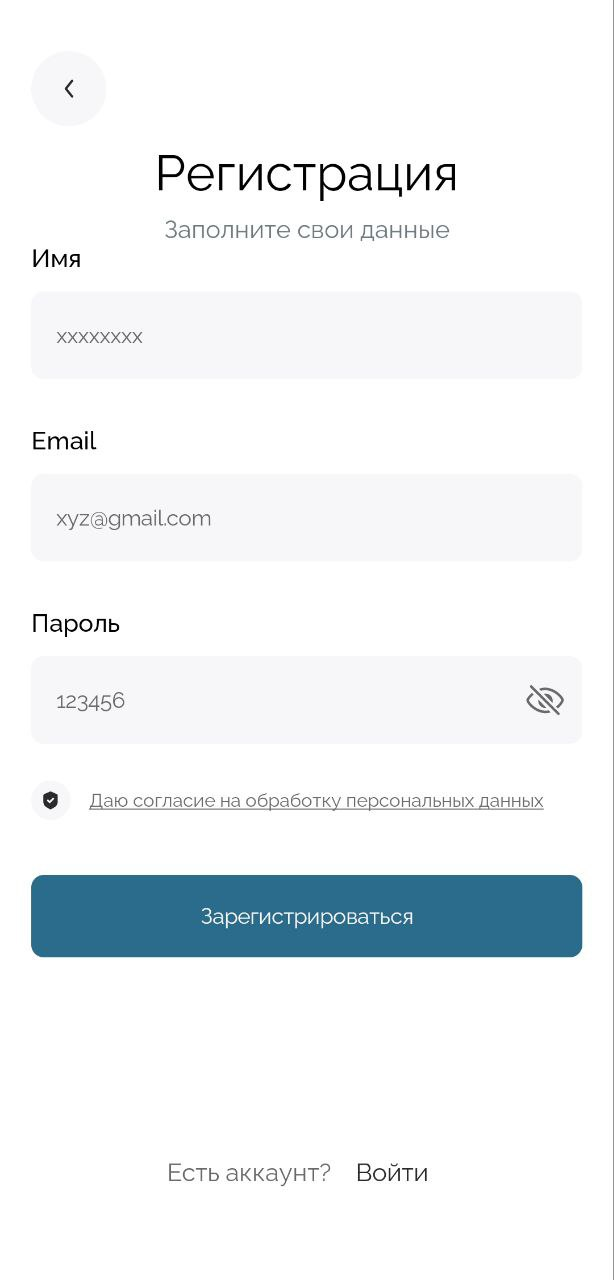


Рисунок 35. Экран регистрации

Forgot Password (Рисунок 36): Экран восстановления пароля с полем ввода email и кнопкой отправить. Используется для получения данных электронной почты пользователя и отправки на этот адрес кода восстановления учетной записи. При вводе корректных данных на экране появляется диалоговое окно (Рисунок 37), информирующее об отправке кода со стороны сервера. По нажатии на данное окно пользователь перенаправляется на экран верификации кода.

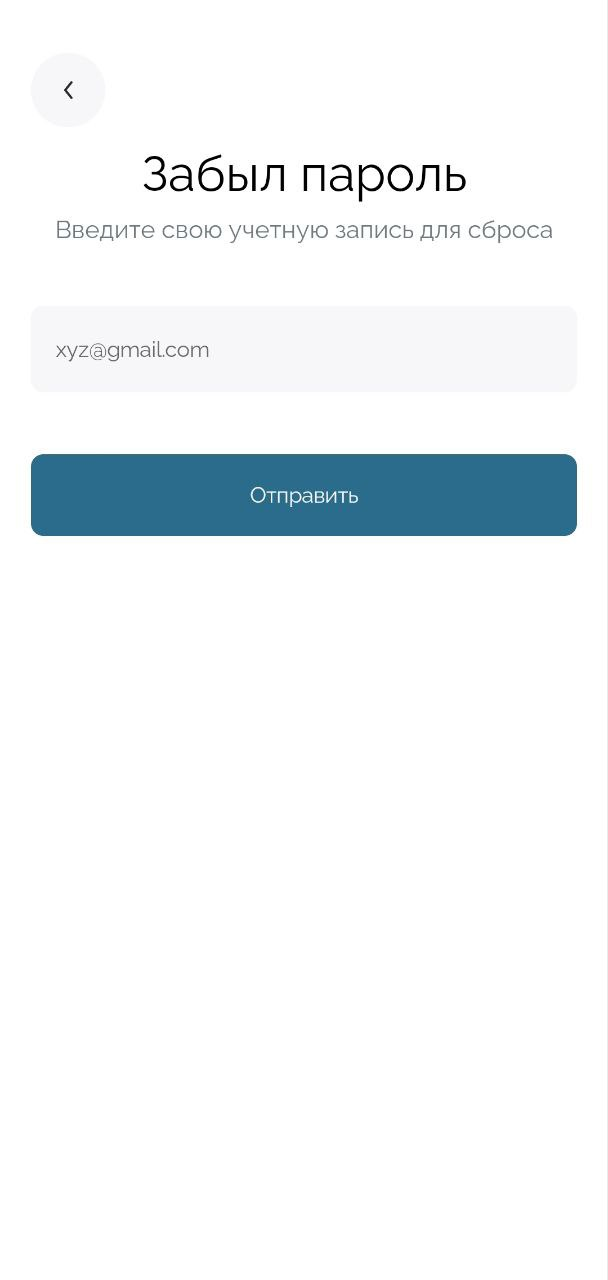


Рисунок 36. Экран восстановления пароля

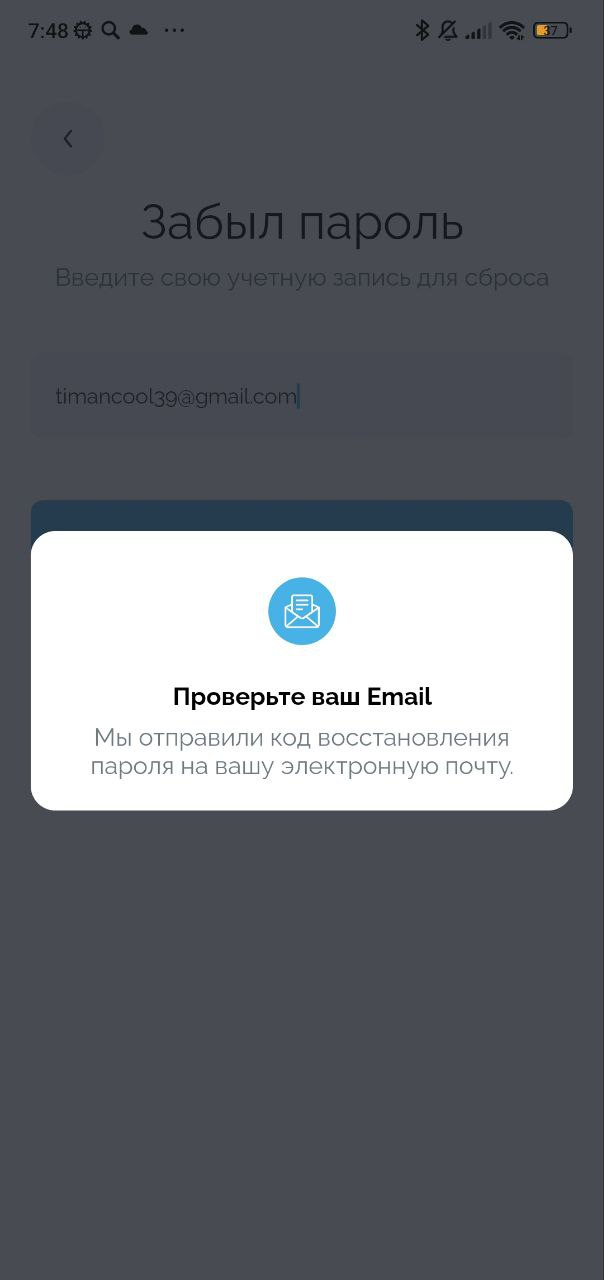


Рисунок 37. Диалоговое окно, информирующее о присланном коде

Verification (Рисунок 38): Экран верификации кода с таймером повторной отправки кода и индикацией ошибок. Для повторной отправки кода пользователю нужно подождать 60 секунд.

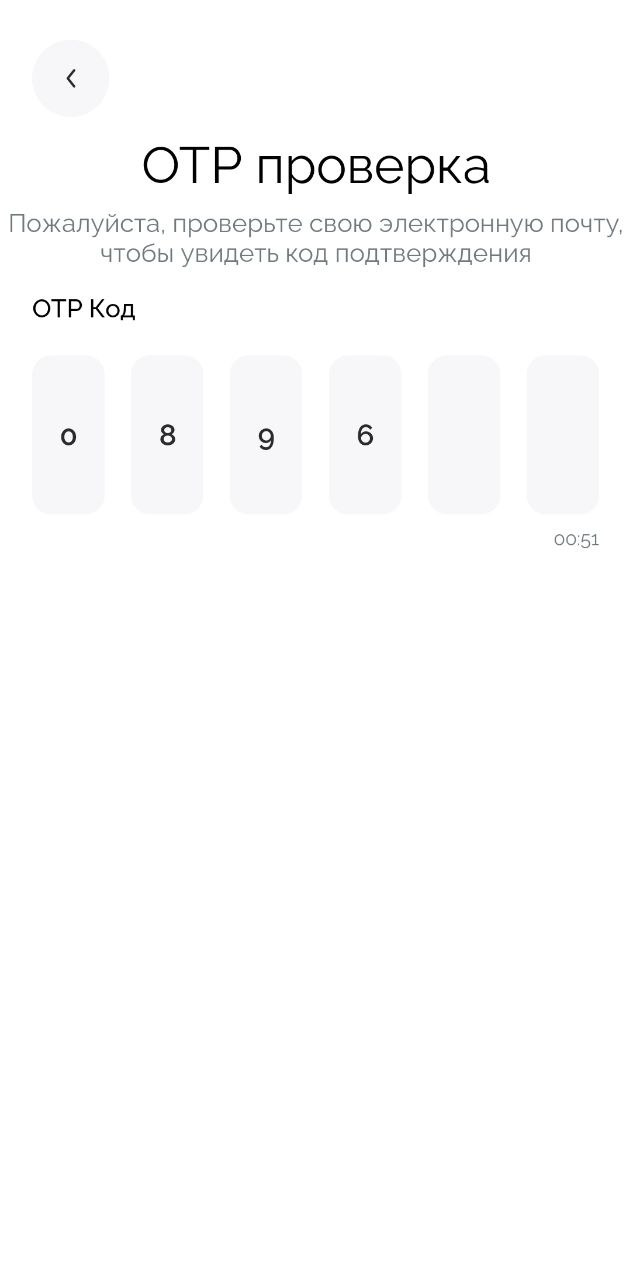


Рисунок 38. Экран верификации кода

Create New Password (Рисунок 39): Экран создания нового пароля с проверкой совпадения паролей. Сохранение нового пароля сопровождается всплыванием информативных сообщений об успешности операции.

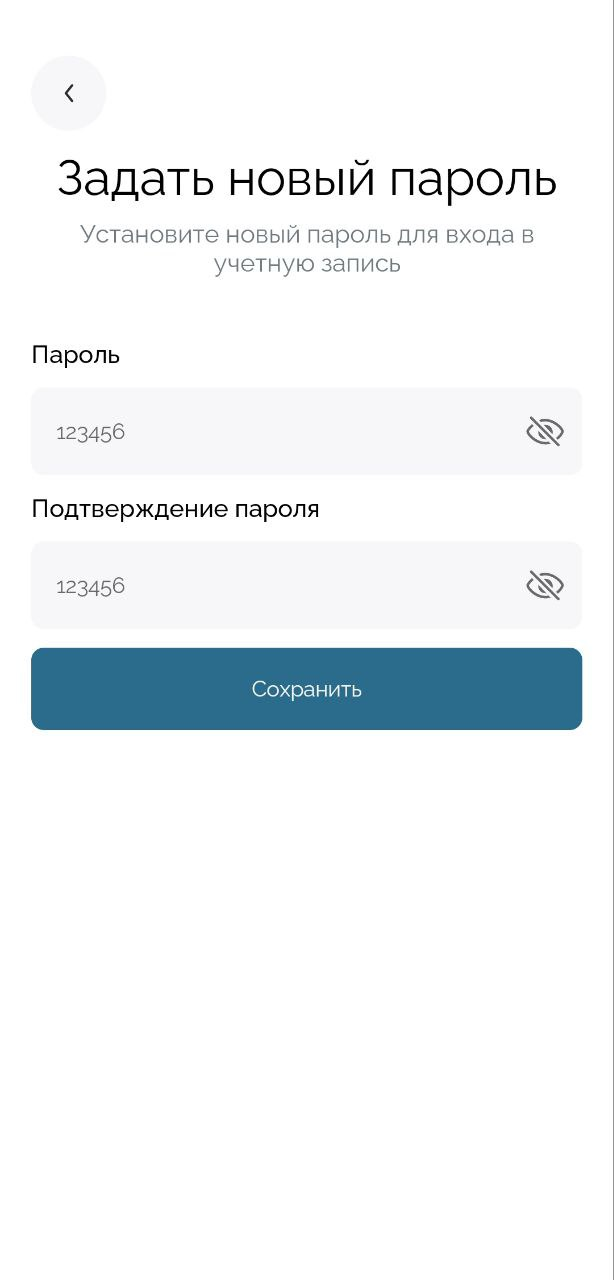


Рисунок 39. Экран создания нового пароля

Home (Рисунок 40): Главный экран с карточками товаров, акциями, фильтрами категорий товаров, нижним и боковым меню (Рисунок 41).

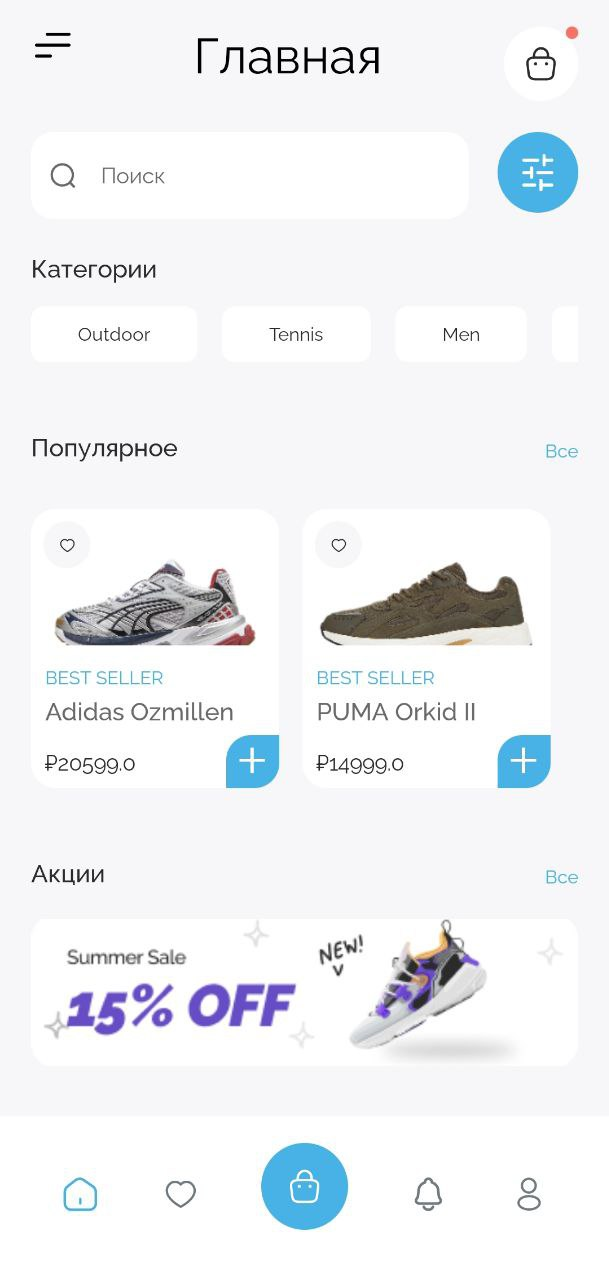


Рисунок 40. Главный экран

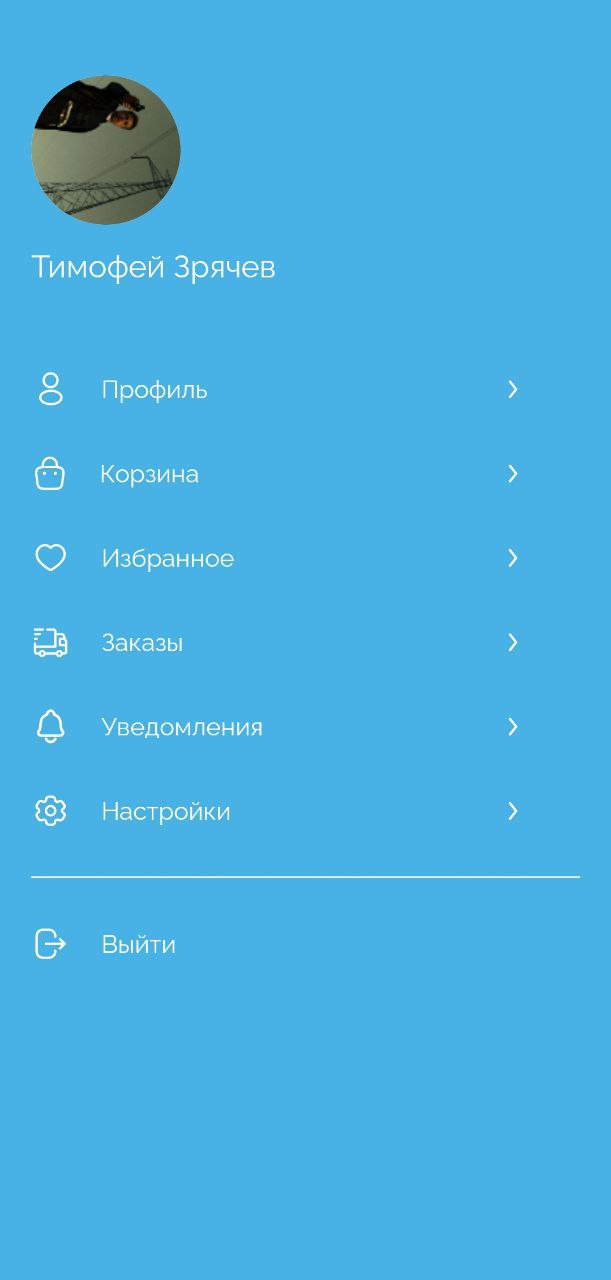


Рисунок 41. Боковое меню

Catalog (Рисунок 42): Экран каталога товаров с выводом данных по выбранному фильтру. Пользователь может переключаться между различными категориями, или вернуться к главной странице.

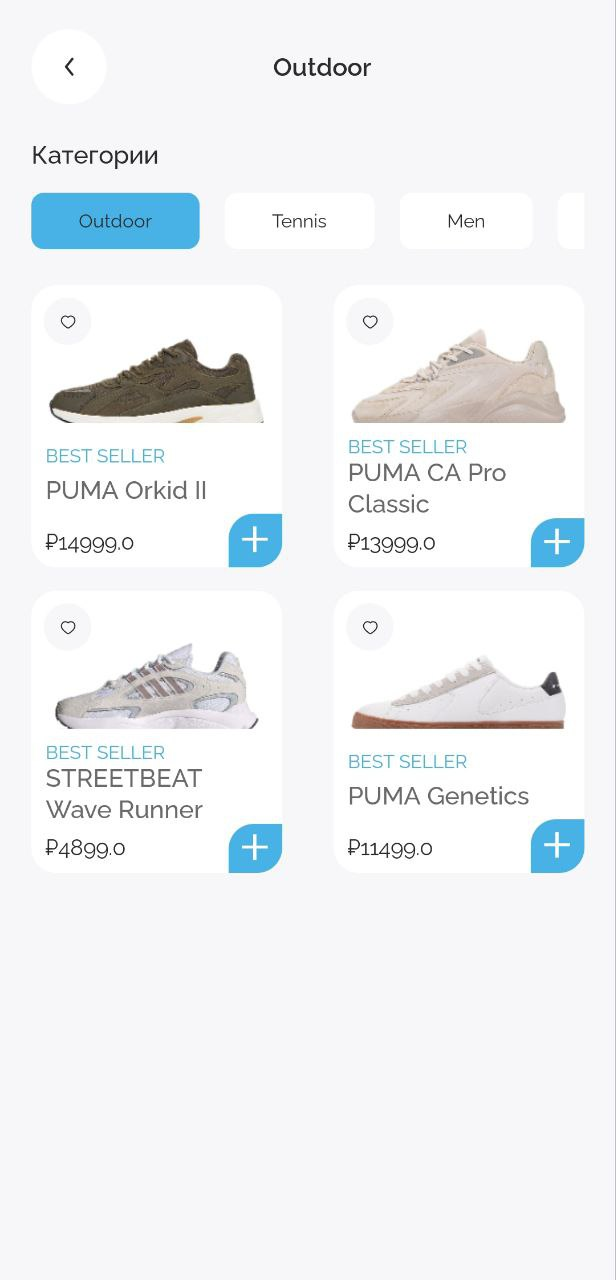


Рисунок 42. Каталог товаров

Profile (Рисунок 43): Экран профиля с возможностью редактирования данных, загрузки фото профиля из галереи, а также генерации штрих-кода из ID пользователя.

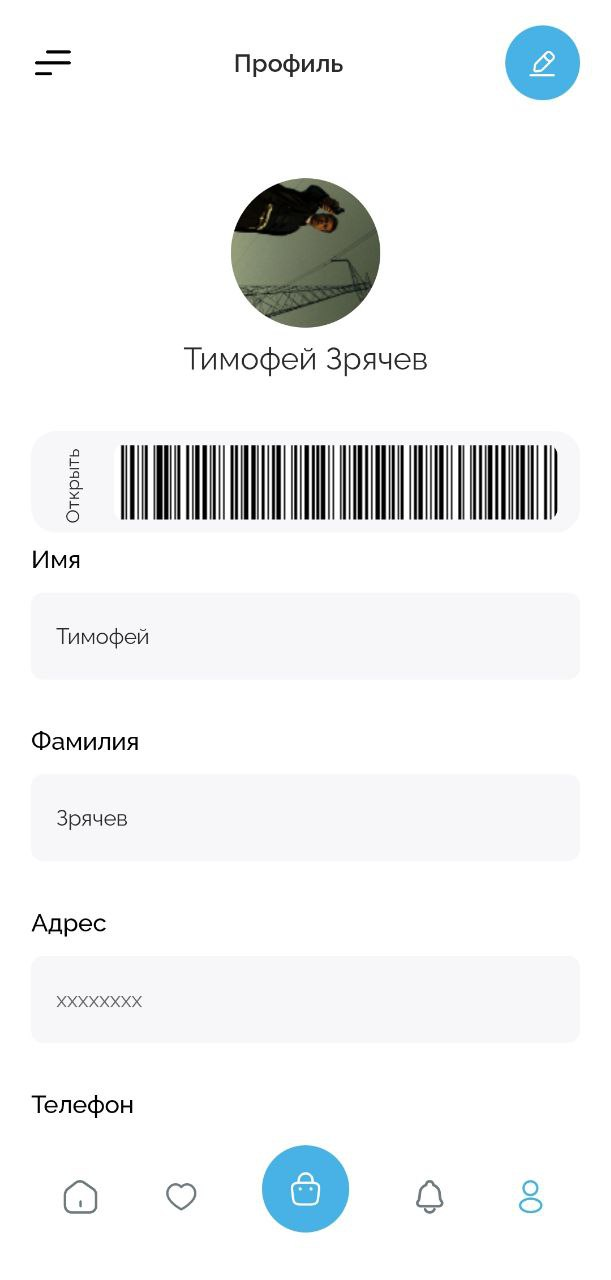


Рисунок 43. Профиль пользователя

Favorite (Рисунок 44): Экран с отображением всех товаров, добавленных пользователем в избранное.

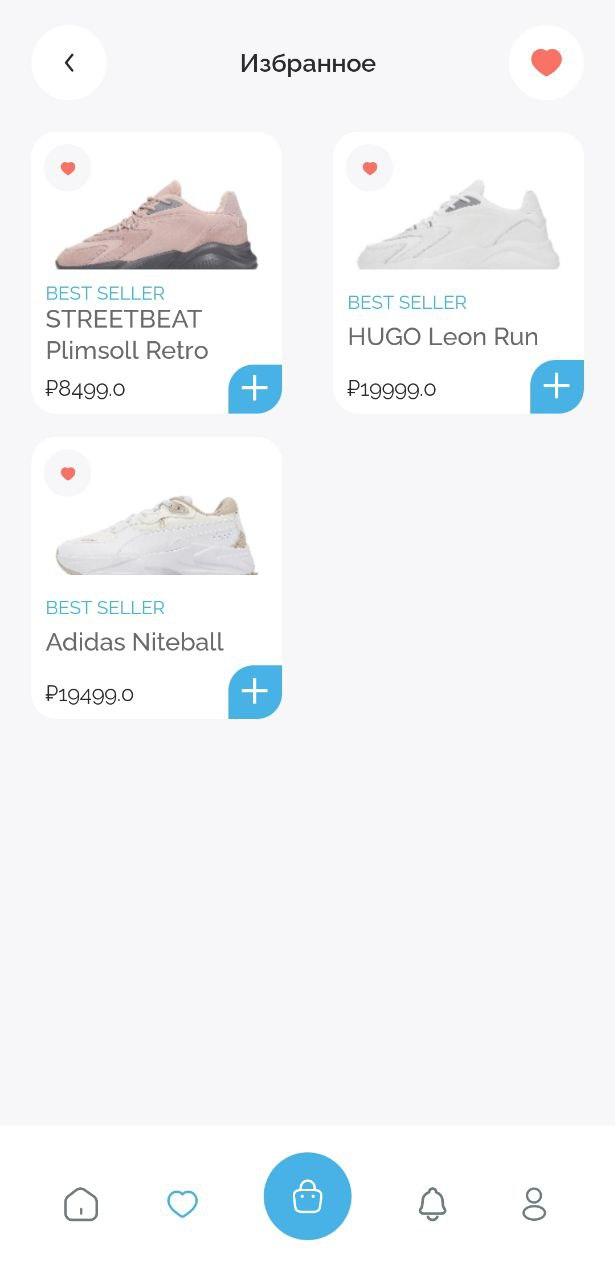


Рисунок 44. Избранное

Loyalty Card (Рисунок 45): Экран карты лояльности с отображением штрих-кода из ID пользователя.

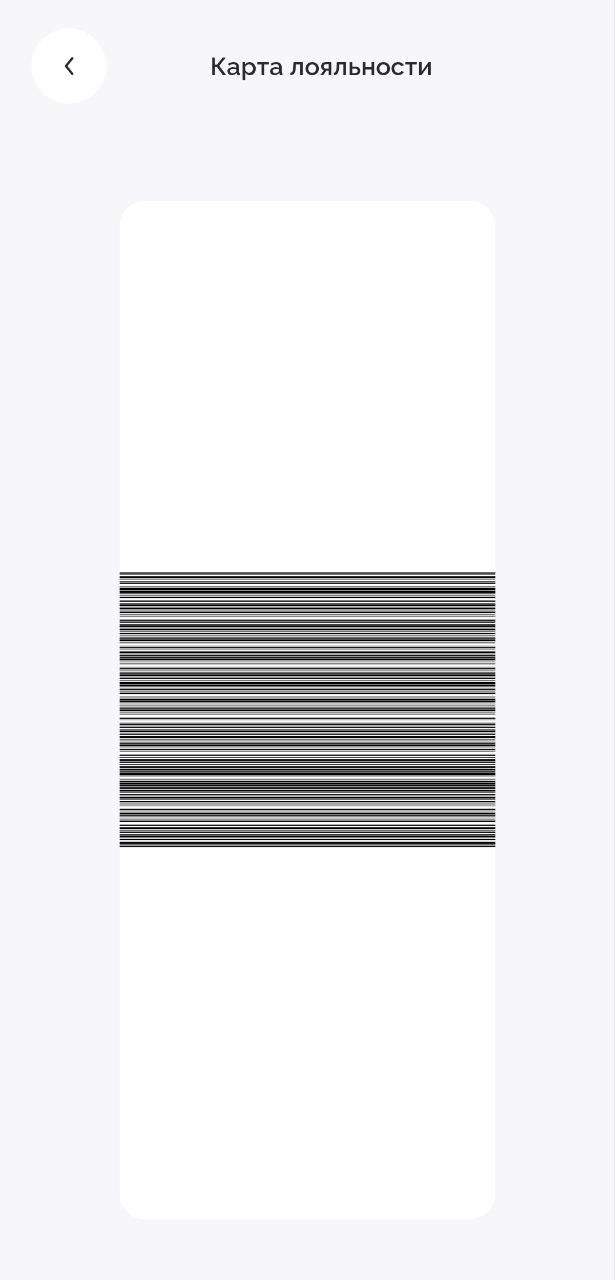


Рисунок 45. Карта лояльности

## 3.2 Клиент-серверное взаимодействие

В качестве серверной части выступает платформа Supabase, база данных в которой была восстановлена из предоставленного скрипта. Увидеть ее ER диаграмму можно на рисунке 46.

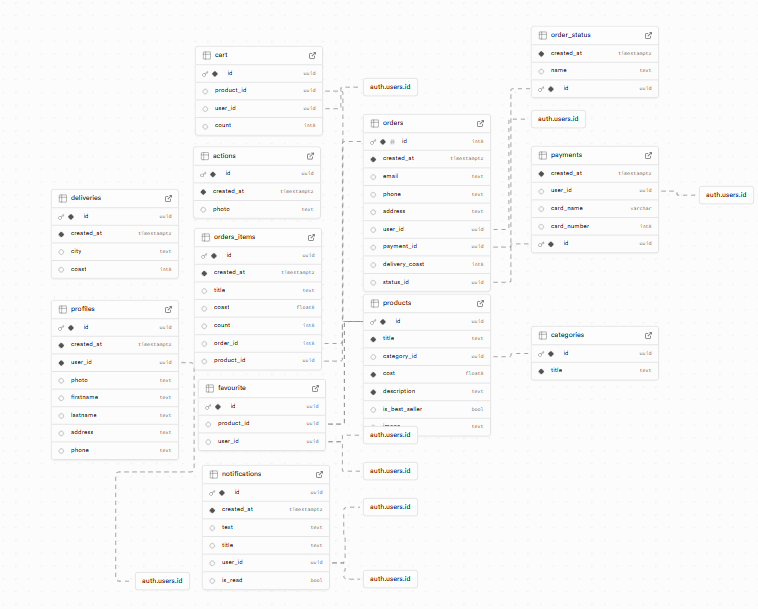


Рисунок 46. ER-диаграмма базы Supabase

### 3.2.1 Авторизация и регистрация

Реализованы с помощью предоставляемых supaBase функций. При авторизации клиент передает данные об email и пароле и получает статус авторизации. В случае успешной авторизации пользователь попадает на главную страницу, в противном – получает сообщение о допущенной ошибке в переданных данных или выполнении запроса.

При регистрации, данные из заполненных полей направляются в запрос на создание пользователя, а после, id созданного пользователя используется для создания его профиля.

Также пользователь может покинуть аккаунт. В этом случае стираются данные access и refresh токенов, а сам пользователь перенаправляется на страницу авторизации.

## 3.2.2 Восстановление пароля

Если пользователь забыл пароль от учетной записи, он может его восстановить, нажав на соответствующую кнопку на экране авторизации. Он будет переброшен на страницу, где введет email, после чего в api будет отправлен запрос на отправку кода по email. Далее, на экране верификации пользователь вводит полученный код, а на следующем экране – новый пароль. Эти данные отправляются в supabase совместно и клиент получает информацию об успешности осуществленной операции. При этом, он также будет проинформирован, в случае если код некорректный или в качестве нового пароля введен старый.

## 3.2.3 Вывод продуктов

Для вывода продуктов используются запросы на получение продуктов и категорий. На главной странице отображаются только 2 первых продукта, а при переходе на страницу каталога отображаются данные продуктов, относящихся к переданной категории.

## 3.2.4 Избранное

На каждой карточке товара присутствует кнопка добавления в избранное. При этом реализован отдельный запрос, выдающий, находится ли продукт в избранном или нет. При добавлении в избранное в таблице favourite формируется запись для переданного товара и текущего пользователя, при удалении – убирается.

## 3.2.5 Профиль

Доступна возможность изменения профиля. При осуществлении этого действия передаются измененные данные, и они обновляют уже существующую запись в базе. При этом есть отдельный запрос для изменения фото пользователя, подгружающий новое фото в качестве потока.

# ЗАКЛЮЧЕНИЕ

Практическое освоение модуля проводилось на базе Государственного бюджетного профессионального образовательного учреждения «Нижегородский Губернский колледж». Основной целью работы являлось формирование и закрепление практических навыков разработки модулей программного обеспечения, полученных в процессе изучения профессионального модуля ПМ.01.

По результатам выполнения модуля были успешно решены следующие задачи:

− реализованы практические навыки разработки программных модулей с применением современных технологий и инструментов;

− получены углубленные знания о принципах проектирования и архитектуры программного обеспечения;

− приобретен опыт работы с реальными задачами разработки программных решений;

− освоены методы оптимизации и тестирования программного кода.

Разработанные в ходе выполнения модуля программные решения демонстрируют практическую применимость полученных знаний для создания эффективных программных продуктов. Реализованные модули могут быть использованы для решения широкого спектра задач автоматизации различных процессов.

Выполнение практических заданий способствовало лучшему пониманию теоретического материала и формированию профессиональных компетенций, необходимых для работы в сфере разработки программного обеспечения. Полученный опыт позволит более успешно применять теоретические знания в реальных проектах.

Таким образом, все поставленные цели и задачи модуля "ПМ.01 Разработка модулей программного обеспечения для компьютерных систем" были успешно достигнуты в полном объеме и в соответствии с установленным графиком работы.

# ПРИЛОЖЕНИЯ

## Приложение А

Таблица 1. Скрипт для создания базы данных

|  |
| --- |
| create table product\_type(  id serial primary key not null,  title varchar(50) not null,  coeff float  );  create table partner\_type(  id serial primary key not null,  title varchar(50) not null  );  create table material\_type(  id serial primary key not null,  title varchar(50) not null,  defect\_coeff float  );  create table partner(  id serial primary key not null,  partner\_type\_id int not null references partner\_type(id) on delete cascade,  title varchar(50) not null,  address varchar(200),  INN varchar(30),  director varchar(150),  phone varchar(30),  email varchar(50),  logo text,  rating float  );  create table product(  id serial primary key,  article varchar(20) not null unique,  product\_type\_id int not null references product\_type(id) on delete cascade ,  description text,  image text,  title varchar(100) not null,  min\_cost decimal(9,2),  length float,  width float,  height float  );  create table partner\_products(  id serial primary key,  product\_id int not null references product(id) on delete cascade,  partner\_id int not null references partner(id) on delete cascade,  quantity int not null,  sale\_date date default now()  );  create table measurement\_unit(  id serial primary key,  title varchar(50)  );  create table material(  id serial primary key,  material\_type\_id int not null references material\_type(id) on delete cascade,  title varchar(50) not null,  pack\_count int,  measurement\_unit\_id int not null references measurement\_unit(id) on delete cascade,  description text,  image text,  price decimal(9,2),  quantity\_in\_stock int,  min\_quantity int  ); |

## Приложение Б

Ссылка на репозиторий проекта (часть 1): <https://github.com/timanitsme/UP_01.git>

Ссылка на репозиторий проекта (часть 2): <https://github.com/timanitsme/UP_01_02.git>

Ссылка на репозиторий проекта (часть 3): https://github.com/timanitsme/UP\_01\_03.git

# Приложение В

Таблица 2. Скрипт базы данных для компании Пиши-Стирай.

| Скрипт базы |
| --- |
| CREATE TABLE role(  id SERIAL PRIMARY KEY,  name VARCHAR(100) NOT NULL  );  CREATE TABLE "user" (  id SERIAL PRIMARY KEY,  surname VARCHAR(100) NOT NULL,  name VARCHAR(100) NOT NULL,  patronymic VARCHAR(100) NOT NULL,  login TEXT NOT NULL,  password TEXT NOT NULL,  role\_id INT NOT NULL REFERENCES role(id) ON DELETE CASCADE  );  CREATE TABLE manufacturer(  id SERIAL PRIMARY KEY,  name VARCHAR(100) NOT NULL  );  CREATE TABLE supplier(  id SERIAL PRIMARY KEY,  name VARCHAR(100) NOT NULL  );  CREATE TABLE category(  id SERIAL PRIMARY KEY,  name VARCHAR(100) NOT NULL  );  CREATE TABLE measurement\_unit(  id SERIAL PRIMARY KEY,  name VARCHAR(100) NOT NULL  );  CREATE TABLE product(  article\_number VARCHAR(100) PRIMARY KEY,  name TEXT NOT NULL,  category\_id INT NOT NULL REFERENCES category(id) ON DELETE CASCADE,  manufacturer\_id INT NOT NULL REFERENCES manufacturer(id) ON DELETE CASCADE,  supplier\_id INT NOT NULL REFERENCES supplier(id) ON DELETE CASCADE,  measurement\_unit\_id INT NOT NULL REFERENCES measurement\_unit(id) ON DELETE CASCADE,  cost DECIMAL(9,2) NOT NULL,  current\_discount DECIMAL(4,2),  max\_discount DECIMAL(4,2),  quantity\_in\_stock INT NOT NULL,  description TEXT NOT NULL,  photo BYTEA  );  CREATE TABLE status(  id SERIAL PRIMARY KEY,  name VARCHAR(100)  );  CREATE TABLE pickup\_point(  id SERIAL PRIMARY KEY,  address TEXT NOT NULL  );  CREATE TABLE "order" (  id SERIAL PRIMARY KEY,  status\_id INT NOT NULL REFERENCES status(id) ON DELETE CASCADE,  date TIMESTAMP NOT NULL DEFAULT now(),  delivery\_date TIMESTAMP NOT NULL,  pickup\_point INT NOT NULL REFERENCES pickup\_point(id),  user\_id INT NOT NULL REFERENCES "user"(id) ON DELETE CASCADE,  pickup\_code INT NOT NULL  );  CREATE TABLE order\_products(  order\_id INT NOT NULL REFERENCES "order"(id) ON DELETE CASCADE,  product\_article\_number VARCHAR(100) NOT NULL REFERENCES product(article\_number) ON DELETE CASCADE,  product\_quantity INT NOT NULL DEFAULT 1,  PRIMARY KEY (order\_id, product\_article\_number)  ); |

# Приложение Г

Предобработка данных и заполнение базы

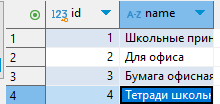


Рисунок 47. Таблица category

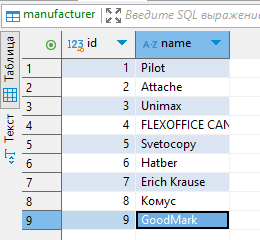


Рисунок 48. Таблица manufacturer

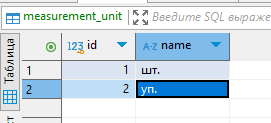


Рисунок 49. Таблица measurement\_unit

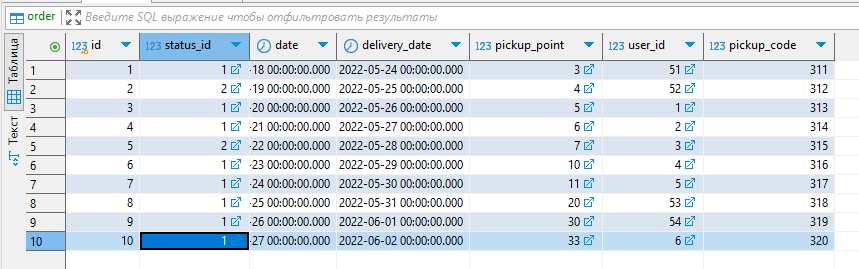


Рисунок 50. Таблица order

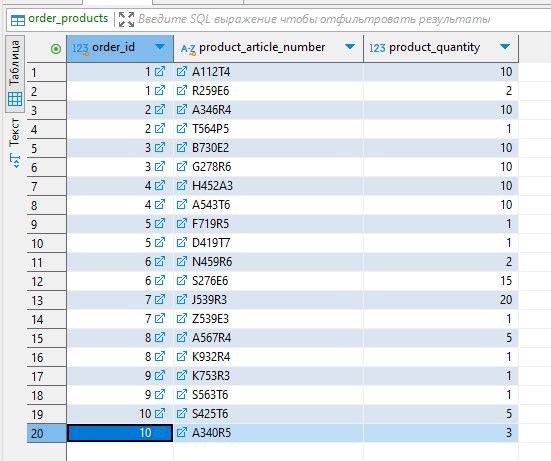


Рисунок 51. Таблица order\_products

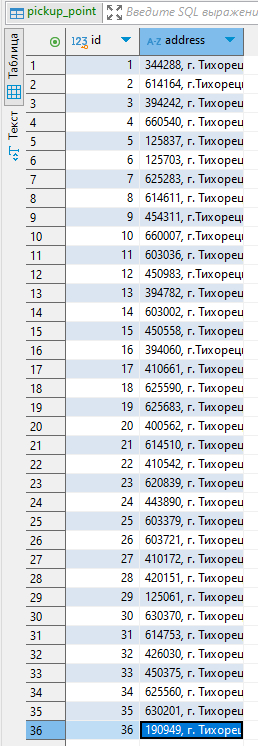


Рисунок 52. Таблица pickup\_point

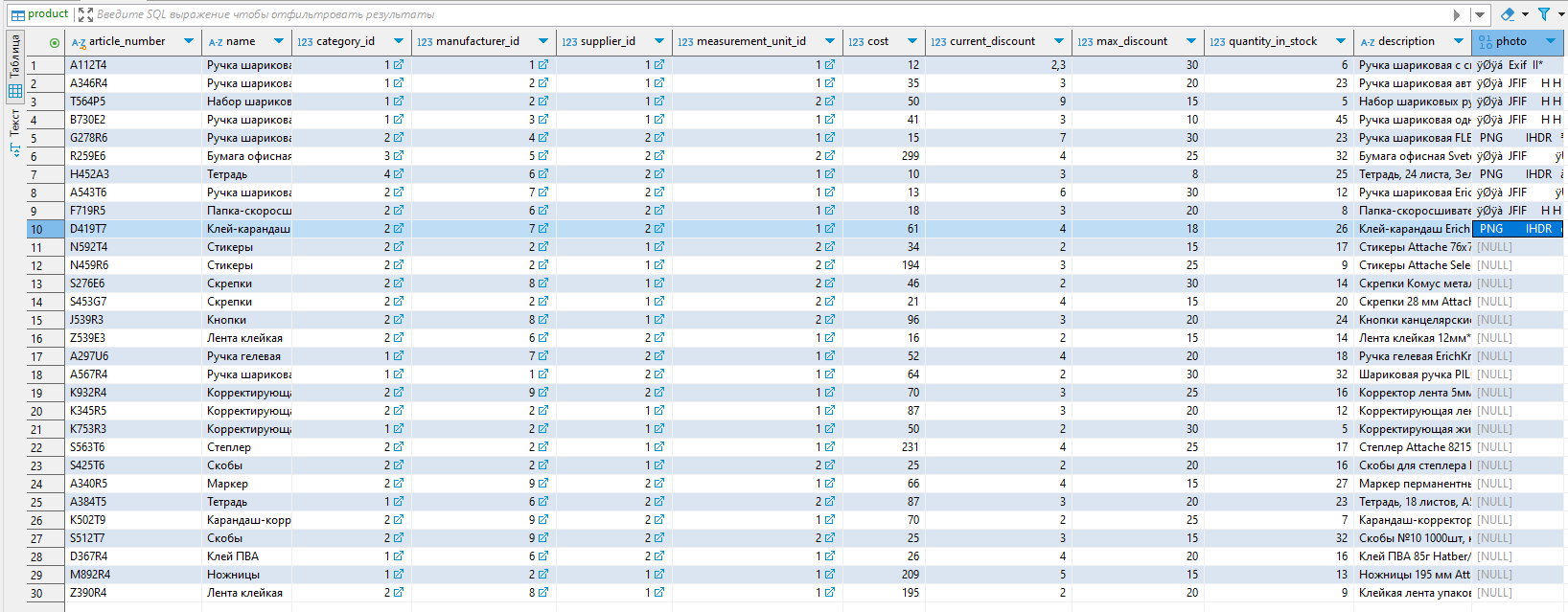


Рисунок 53. Таблица product

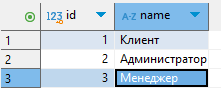


Рисунок 54. Таблица role

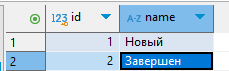


Рисунок 55. Таблица status

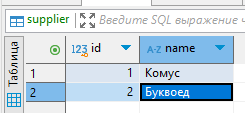


Рисунок 56. Таблица supplier

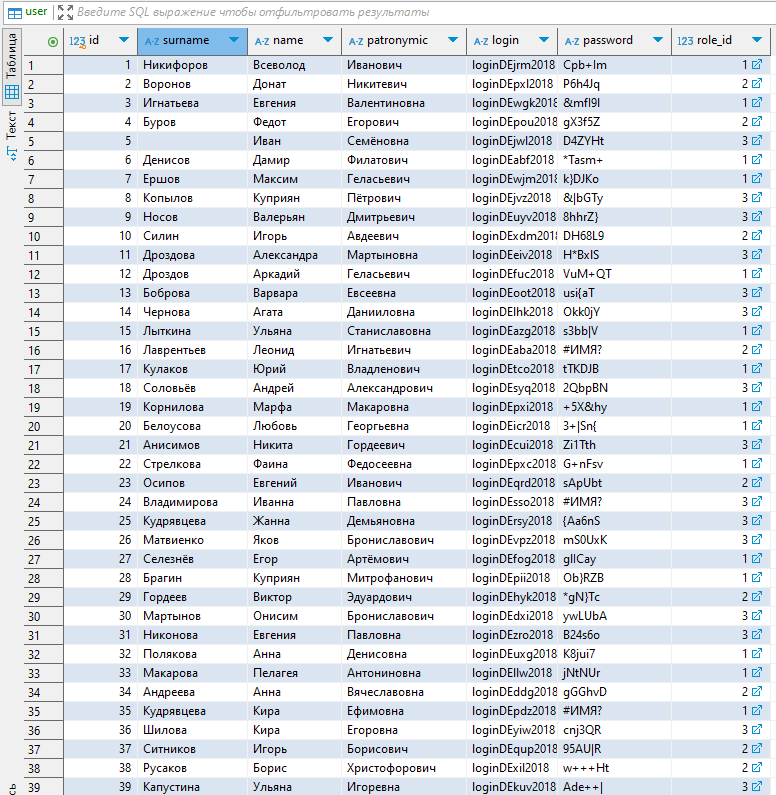


Рисунок 57. Таблица user