СОДЕРЖАНИЕ

[1 Исследование информационной системы предприятия 3](#_Toc195653047)

[2 Анализ предметной области 4](#_Toc195653048)

[3 Проектирование программного модуля 7](#_Toc195653049)

[3.1 Проектирование диаграмм 7](#_Toc195653050)

[3.2 Разработка макетов интерфейса программного модуля 12](#_Toc195653051)

[4 Разработка программного модуля 17](#_Toc195653052)

[4.1 Разработка интерфейсов программного модуля 17](#_Toc195653053)

[4.2 Разработка кода программного модуля 29](#_Toc195653054)

[5 Тестирование программного модуля 41](#_Toc195653055)

[5.1 Разработка тест-кейсов 41](#_Toc195653056)

[5.2 Проведение модульного тестирования 43](#_Toc195653057)

[Приложение А – Спецификация требований 46](#_Toc195653058)

[Приложение Б – Ссылка на систему контроля версий 54](#_Toc195653059)

[Приложение В – ER-диаграмма 55](#_Toc195653060)

1. ИССЛЕДОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ ПРЕДПРИЯТИЯ

В салоне цветов «Привет, я букет!» имеется персональный компьютер ASUS VivoPC VM42 с операционной системой Windows 10 Pro, процессором Intel Core i5-10400, оперативной памятью 8 ГБ DDR4. Принтер представлен моделью HP LaserJet Pro M203dw, которая обеспечивает черно-белую лазерную печать со скоростью до 30 страниц в минуту, разрешением печати до 1200x1200 dpi, автоматической двусторонней печатью, функцией автоподачи документов на 50 листов, встроенным дуплексным устройством, а также оборудована интерфейсами для подключения к сети. Сканер штрих-кода Symbol LS2208 от Zebra Technologies – беспроводной лазерный сканер с диапазоном считывания от 1 до 76 см в зависимости от типа кода, способный распознавать все стандартные линейные штрих-коды. Кассовый терминал CHERRY T25 с сенсорным экраном диагональю 15.6 дюйма, разрешением 1366x768 точек, процессором Intel Celeron J1900, 2 ГБ оперативной памяти, жестким диском на 32 ГБ eMMC, встроенным фискальным регистратором.

Программы, которыми пользуются директор и продавец-флорист:

* Microsoft Excel.
* Microsoft Word.

А также сайт салона, <http://www.привет-я-букет.рф/index.php>, основной функционал которого представлен на рисунке 1.

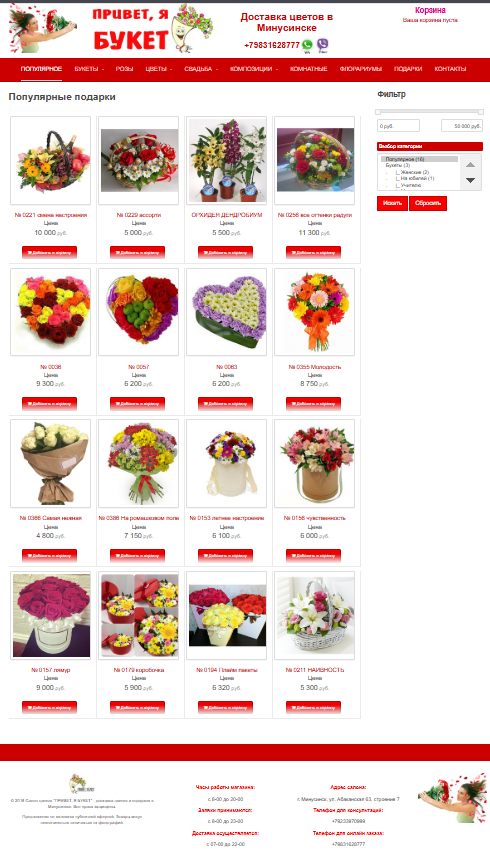


Рисунок 1 – Сайт «Привет, я букет!»

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Перечень документов и лиц.

Документы:

Бланк заказа («OrderBlank») сообщает информацию о поставщике и покупателе, а также отображет таблицу с полями: «№ п/п», «Товар», «Количество», «Цена», «Сумма», «Возможная цена», «Условия получения». Данный документ составляется, когда покупатель заказывает у поставщика товар, документ формируется на определенное число, которое показывает, когда был сделан заказ, часть документа представлена на рисунке 2.

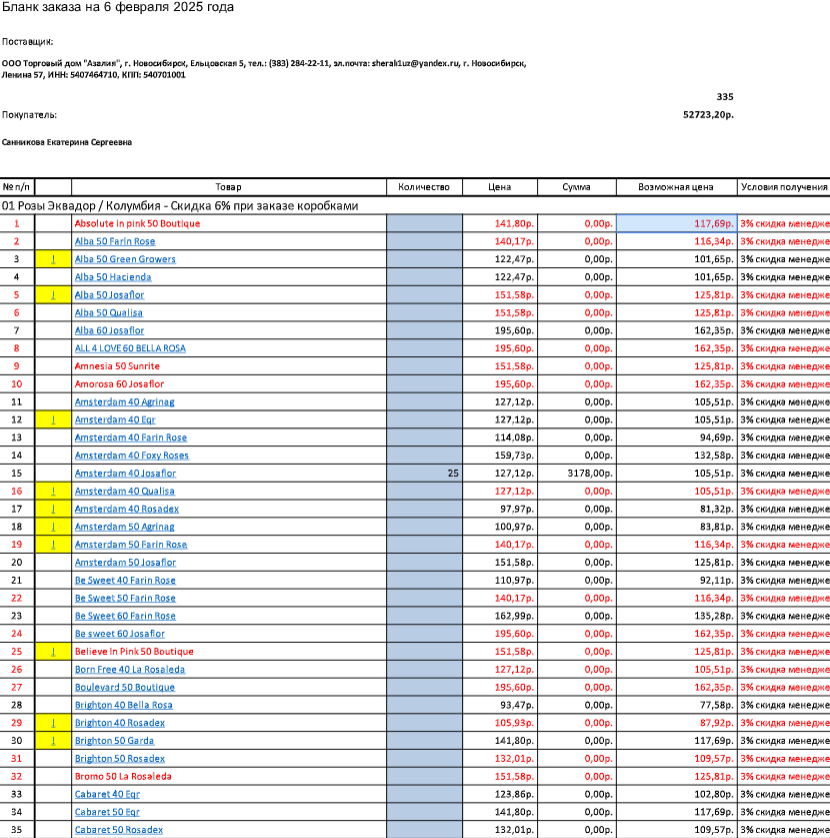


Рисунок 2 – Бланк заказа

Документ «Общий учет!» представляет собой таблицу, описывающую расходы и выручку по датам определенного месяца, что позволяет сравнить выручку и расходы за месяц, составляется документ вначале месяца и показывает выручку и расходы за прошлый месяц. Документ представлен на рисунке 3.

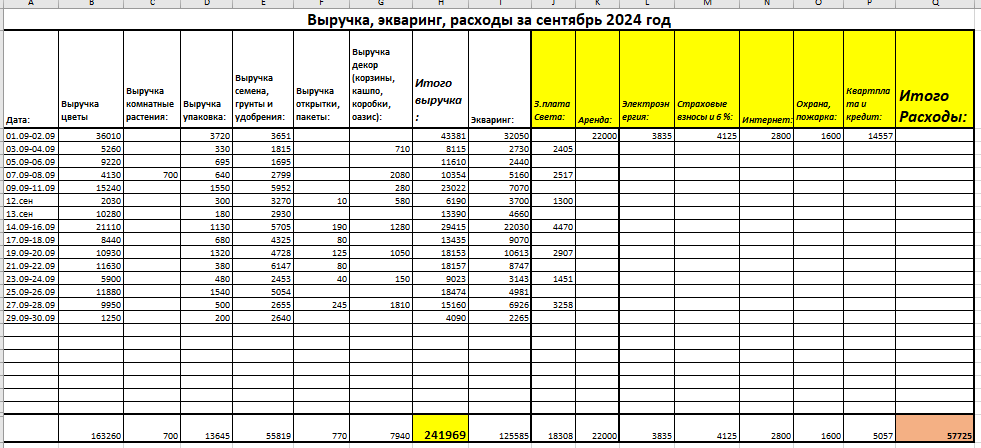


Рисунок 3 – Общий учет

Документ «Счет-фактура» представляет собой документ, показывающий, что продукция отгружена продавцом покупателю, составляется после продажи товара поставщиком покупателю, в этот же день или позже, в этом документе можно увидеть код товара, №, наименование товара, код вида товара, единицу измерения, количество, цену, стоимость, и т.д. документ показан на рисунке 4.

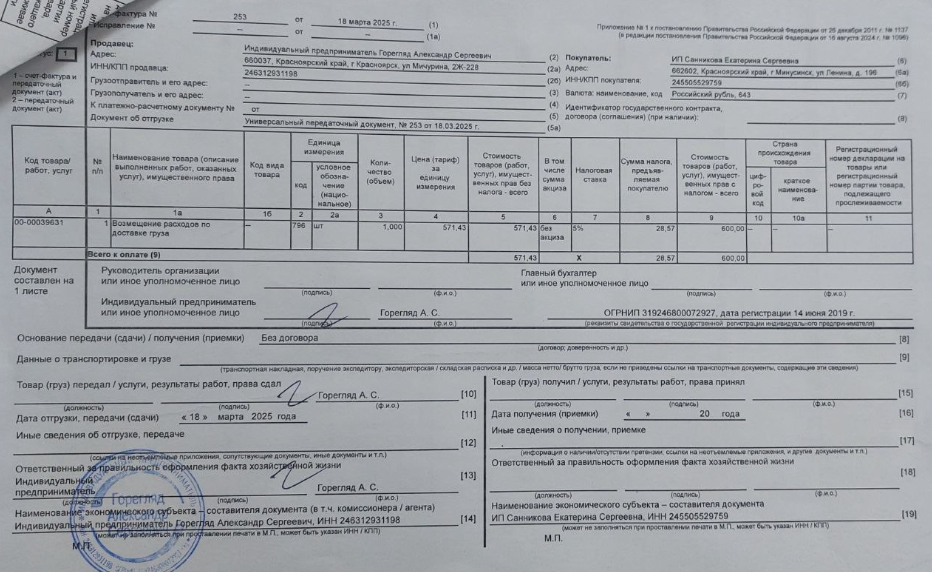


Рисунок 4 – Документ «Счет-фактура»

Документ «Спецификация» представляет собой документ, устанавливающий требования к поставляемому товару: номер, товар, количество, цена, возможная цена, сумма, а также данные о поставщике и о покупателе. Составляется для точности того, какой товар и на какую сумму будет куплен покупателем, формируется до покупки товара, представлено на рисунке 5.

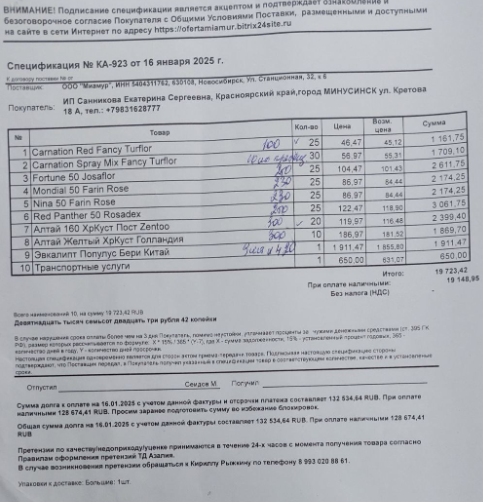


Рисунок 5 – Документ «Спецификация»

Лица и их задачи в салоне цветов «Привет, я букет!»:

* Директор: задача контролировать поступление товаров, обновлять данные о запасах с помощью документов «Бланк заказа» и «Общий учет», оформлять заказы у поставщиков с помощью документов «Спецификация» и «Счет-фактура», вести учет финансовых операций также с помощью документов «Бланк заказа» и «Общий учет» и контролировать финансовые потоки.
* Продавец-консультант: задача принимать заказы от клиентов офлайн, а также онлайн с помощью сайта «Привет, я букет!», предоставлять информацию о товарах и оформлять продажи, а также помогать директору в составлении документов «Бланк заказа», «Общий учет», «Спецификация» и «Счет-фактура».

1. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО МОДУЛЯ
   1. Проектирование диаграмм

Проектируемая информационная система предназначена для автоматизации и оптимизации ключевых бизнес-процессов цветочного магазина. Ниже представлены основные функциональные задачи, которые будут решаться с помощью данной системы:

Учет поступления товаров:

* Регистрация новых поступлений: автоматизация процесса ввода информации о новых товарах, поступивших в магазин, включает возможность добавления данных о товаре.

Управление ассортиментом:

* Добавление товаров: возможность ввода новых товаров в базу данных с указанием всех необходимых характеристик (название, описание, категория и т.д.).
* Редактирование товаров: обновление информации о существующих товарах, что позволит поддерживать актуальность данных в системе.
* Удаление товаров: удаление устаревших или неактуальных товаров из базы данных.
* Классификация товаров: организация товаров по категориям, что облегчит поиск и управление ассортиментом.

Отслеживание остатков:

* Мониторинг текущих остатков: постоянный контроль за количеством товаров в магазине, что позволит избежать дефицита или избытка запасов.

Отчетность:

* Генерация отчетов по остаткам товаров: создание отчетов, содержащих информацию о текущих запасах товаров, что позволит быстро оценивать ситуацию в магазине.
* Генерация отчетов по поступлению товаров: формирование отчетов о поступлениях товаров и их количестве.
* Генерация отчетов по убытию товаров: формирование отчетов по выбытию товаров и их количестве.
* Генерация документа «Бланк заказа».
* Генерация документа «Общий учет».
* Генерация документа «Спецификация».
* Генерация документа «Счет фактура».

Проектируемая информационная система по учету товаров в цветочном магазине направлена на повышение эффективности управления товарами и улучшение качества обслуживания клиентов. Автоматизация вышеуказанных процессов позволит сократить время на выполнение рутинных задач и снизить вероятность ошибок, связанных с ручным вводом данных.

Функциональная диаграмма IDEF0 нулевого уровня представлена на рисунке 6.

Описание:

Данная диаграмма представляет собой:

В качестве управления: документ «Бланк заказа», сайт магазина «Привет, я букет!», а также документы: «Общий учет», «Счет-фактура», «Спецификация».

В качестве входных данных: данные о товарах, данные о поставщиках, данные о сотрудниках.

Процессом в данной диаграмме является учет товаров.

Механизмами в данной системе являются: программа, директор и оператор-флорист.

В качестве выходных данных представлены отчеты: «Бланк заказа», «Общий учет», «Спецификация», «Счет-фактура», «Отчет по остаткам», «Отчет по поступлениям», «Отчет по убытию».



Рисунок 6 – IDEF0 нулевого уровня

Функциональная диаграмма IDEF0 первого уровня представлена на рисунке 7.

Данная диаграмма представляет собой:

В качестве управления: документ «Бланк заказа», сайт магазина «Привет, я букет!», а также документы: «Общий учет», «Счет-фактура», «Спецификация».

В качестве входных данных: данные о товарах, данные о поставщиках, данные о сотрудниках.

Процессами в данной диаграмме являются: добавление данных, изменение данных, удаление данных, а также генерация отчетов.

Механизмами в данной системе являются: программа, директор и оператор-флорист.

В качестве выходных данных представлена генерация отчетов.

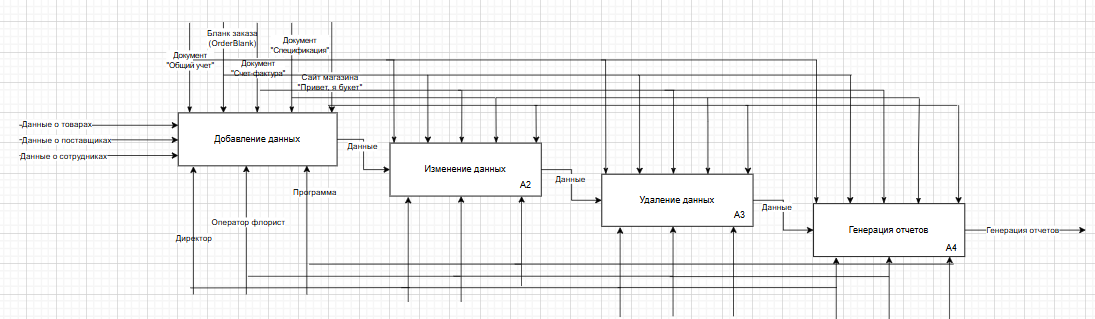


Рисунок 7 – IDEF0 первого уровня

Описание диаграммы ВИ:

На данной диаграмме пользователи приложения могут иметь две роли:

* Директор: высшее должностное лицо, отвечающее за общее управление и принятие ключевых решений.
* Оператор-флорист: сотрудник, ответственный за работу с поставщиками и управление товарными запасами.

Функции:

* Авторизоваться: функция входа в систему для аутентификации директора или флориста.
* Просмотреть список поставщиков: позволяет просматривать информацию о поставщиках.
* Добавить поставщика: добавление нового поставщика в систему и в БД.
* Редактировать поставщика: изменение данных существующего поставщика в системе и в БД.
* Удалить поставщика: удаление поставщика из системы и БД.
* Просмотреть список товаров: функция просмотра списка товаров.
* Редактировать товар: изменение данных о товаре.
* Добавить товар: добавление нового товара в систему и БД.
* Удалить товар: удаление товара из системы и БД.
* Просмотреть список сотрудников: позволяет просматривать информацию о сотрудниках.
* Редактировать сотрудника: изменение данных о сотруднике.
* Добавить сотрудника: добавление нового сотрудника в систему.
* Удалить сотрудника: удаление сотрудника из системы.
* Сформировать отчет по остаткам товаров: генерация отчета об остатках товаров.
* Сформировать отчет по поступлению товаров: генерация отчета о поступлении товаров.
* Сформировать отчет по убытию товаров: генерация отчета об убытии товаров.
* Сформировать «Бланк заказа».
* Выйти из учетной записи: функция для перехода на окно авторизации.
* Выйти из программы: дополнительная функция выхода из программы.

Стрелки на диаграмме показывают связи между ролями и функциями, демонстрируя, какие роли могут выполнять те или иные действия в системе.

Диаграмма вариантов использования представлена на рисунке 8.

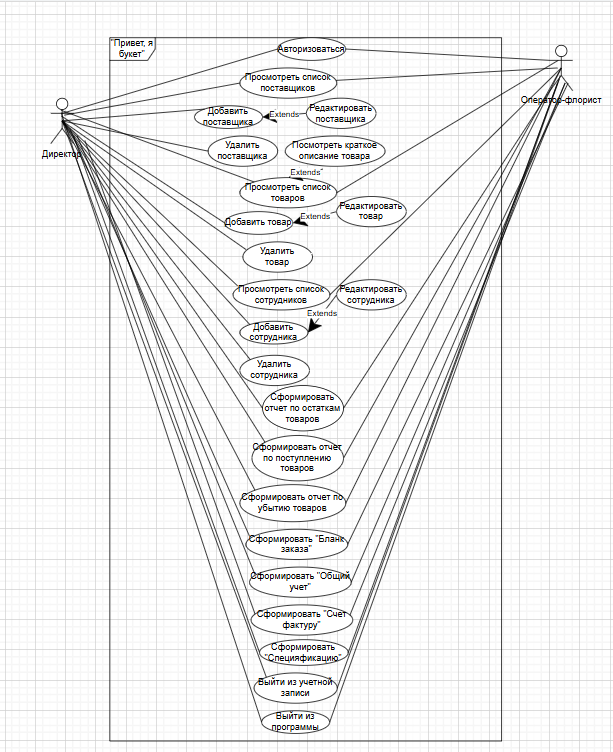


Рисунок 8 – Диаграмма ВИ

ER-диаграмма представлена на рисунке В1 в приложении В. На ней изображена структура базы данных для разрабатываемой системы.

Таблица «Область» содержит поля: УИД (serial), наименование (varchar(100)) и город (integer).

Таблица «Город» содержит поля: УИД (serial), наименование (varchar(100)), сокращенное\_наименование (varchar(100)) и улица (integer).

Таблица «Улица» содержит поля: УИД (serial), название (varchar(100)), дом\_строение (integer).

Таблица «Дом\_строение» содержит поля: УИД (serial), номер (integer).

Таблица «Поставщик» содержит поля: УИД (первичный ключ, serial), название\_поставщика (varchar(100), обязательное поле), тип\_поставщика (перечисление tip\_postavshika\_organizacii, обязательное поле), контактный\_телефон (varchar(100)), электронная\_почта (varchar(100)), юридический\_адрес (внешний ключ, ссылается на таблицу «Дом\_строение»).

Таблица «Документ» содержит поля: УИД (первичный ключ, serial), серия (integer, обязательное поле), номер (integer, обязательное поле), дата\_выдачи (date), кем\_выдан (varchar(100)).

Таблица «Подробная\_информация\_о\_поставщике» содержит поля: УИД (первичный ключ, serial), поставщик\_id (внешний ключ, ссылается на таблицу «Поставщик»), документ\_id (внешний ключ, ссылается на таблицу «Документ»).

Таблица «Товар» содержит поля: УИД (первичный ключ, serial), название (перечисление tip\_tovara, обязательное поле), номер (integer, обязательное поле), описание (varchar(100)), цена (numeric(10,2)), картинка (bytea), поставщик\_id (внешний ключ, ссылается на таблицу «Поставщик»).

Таблица «Сотрудник» содержит поля: УИД (первичный ключ, serial), фамилия (varchar(100), обязательное поле), имя (varchar(100), обязательное поле), отчество (varchar(100)), дата\_рождения (date), ИНН (integer), СНИЛС (integer), пол (перечисление pol, обязательное поле).

Таблица «Подробная\_информация\_о\_сотруднике» содержит поля: УИД (первичный ключ, serial), сотрудник\_id (внешний ключ, ссылается на таблицу «Сотрудник»), документ\_id (внешний ключ, ссылается на таблицу «Документ»).

Таблица «Формирование\_отчета» содержит поля: УИД (первичный ключ, serial), название (varchar(100), обязательное поле), краткое\_описание (varchar(100)), полное\_описание (varchar(250)).

Таблица «Должность» содержит поля: УИД (первичный ключ, serial), название (перечисление nazvanie\_dolzhnosti, обязательное поле), краткое\_описание (varchar(100)), полное\_описание (varchar(250)), дата\_приема (date), дата\_договора (date), формирование\_отчета\_id (внешний ключ, ссылается на таблицу «Формирование\_отчета»).

Таблица «Отчеты\_по\_поступлению\_товаров» содержит поля: УИД (первичный ключ, serial), дата\_поступления (date, обязательное поле), количество\_поступивших\_товаров (integer, обязательное поле), дата\_формирования\_отчета (date, обязательное поле), товар\_id (внешний ключ, ссылается на таблицу «Товар»), поставщик\_id (внешний ключ, ссылается на таблицу «Поставщик»).

Таблица «Отчеты\_по\_остаткам\_товаров» содержит поля: УИД (первичный ключ, serial), количество\_товара (integer, обязательное поле), дата\_формирования\_отчета (date, обязательное поле), товар\_id (внешний ключ, ссылается на таблицу «Товар»).

Таблица «Отчеты\_по\_убытому\_товару» содержит поля: УИД (первичный ключ, serial), дата\_убытия (date, обязательное поле), количество\_убытого\_товара (integer, обязательное поле), дата\_формирования\_отчета (date, обязательное поле), товар\_id (внешний ключ, ссылается на таблицу «Товар»).

* 1. Разработка макетов интерфейса программного модуля

Макет окна авторизации представлен на рисунке 9.

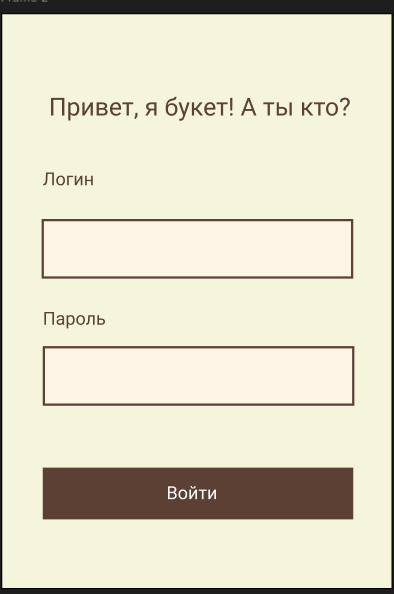


Рисунок 9 – Макет окна авторизации

Макет раздела «Поставщики» для директора представлен на рисунке 10.

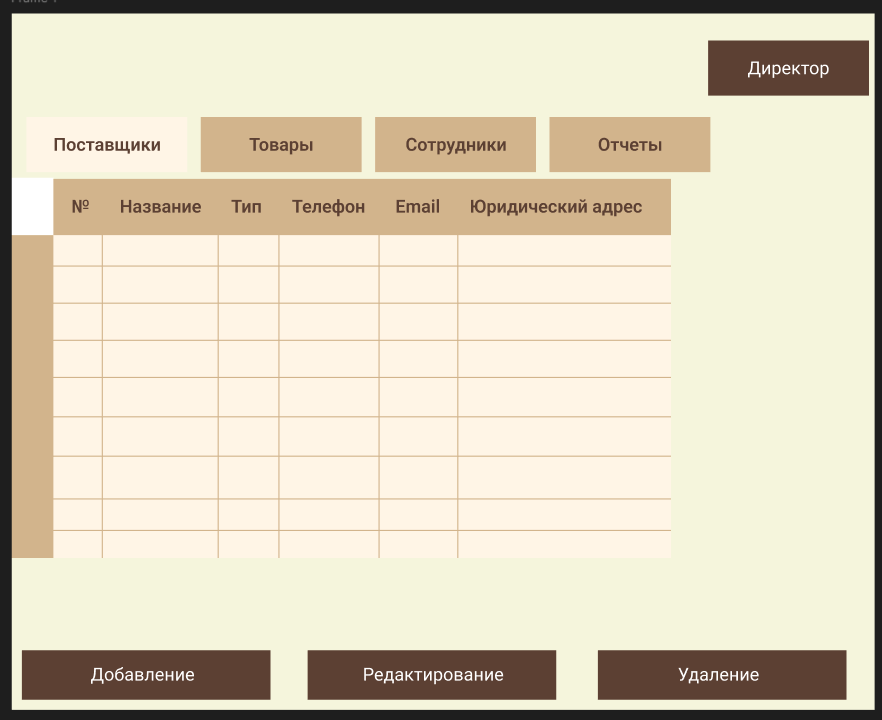


Рисунок 10 – Макет раздела «Поставщики»

Макет раздела «Товары» для директора представлен на рисунке 11.



Рисунок 11 – Макет раздела «Товары»

Макет раздела «Сотрудники» для директора представлен на рисунке 12.

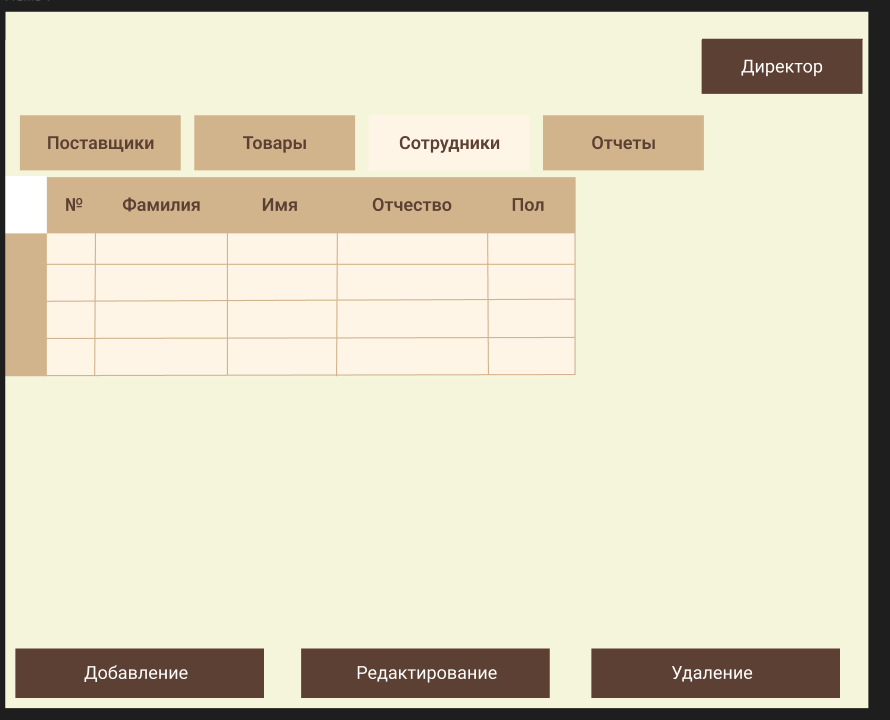


Рисунок 12 – Макет раздела «Сотрудники»

Макет раздела «Отчеты» для директора и оператора флориста представлен на рисунке 13.



Рисунок 13 – Макет раздела «Отчеты»

Макет отчета для директора и оператора-флориста представлен на рисунке 14.



Рисунок 14 – Макет отчета «Бланк заказа»

Макет раздела «Поставщики» для оператора-флориста представлен на рисунке 15.

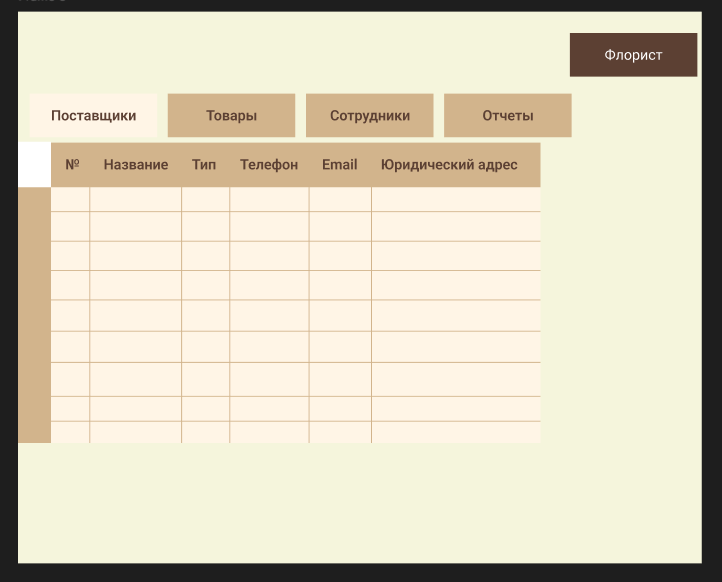


Рисунок 15 – Макет раздела «Поставщики»

Макет раздела «Товары» для оператора-флориста представлен на рисунке 16.



Рисунок 16 – Макет раздела «Товары»

Макет раздела «Сотрудники» для оператора-флориста представлен на рисунке 17.



Рисунок 17 – Макет раздела «Сотрудники»

1. РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ
   1. Разработка интерфейсов программного модуля

Программный код для реализации интерфейсов представлен в листинге 1.

Листинг 1 – Программный код файла styles.qss

/\* Главное окно (MainWindow) \*/

QMainWindow, QDialog {

background-color: #F5F5DC; /\* Светло-бежевый фон \*/

}

/\* Окно авторизации (AuthWindow) \*/

AuthWindow {

background-color: #F5F5DC; /\* Светло-бежевый фон \*/

}

/\* Сообщения об ошибках (QMessageBox) \*/

QMessageBox {

background-color: #F5F5DC;

border: 1px solid #D2B48C;

border-radius: 10px;

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.15);

}

QMessageBox QLabel {

font-family: "Roboto", sans-serif;

font-size: 12pt;

color: #5C4033;

background-color: transparent;

}

QMessageBox QPushButton {

font-family: "Roboto", sans-serif;

font-size: 12pt;

font-weight: 500;

background-color: #5C4033;

color: white;

border: none;

border-radius: 7px;

padding: 8px 15px;

box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);

}

QMessageBox QPushButton:hover {

background-color: #7A5A47;

}

QMessageBox QPushButton:pressed {

background-color: #4A3228;

}

/\* Приветственный текст \*/

#greetingLabel {

font-family: "Roboto", sans-serif;

font-size: 17pt;

font-weight: 300;

color: #5C4033;

background-color: transparent;

padding: 3px 0;

text-align: center;

}

/\* Метка учетной записи \*/

#accountLabel {

font-family: "Roboto", sans-serif;

font-size: 12pt;

font-weight: 500;

color: white;

background-color: #5C4033;

padding: 5px 10px;

border-radius: 5px;

margin: 5px;

box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);

}

#accountLabel:hover {

background-color: #7A5A47;

cursor: pointer;

}

#accountLabel:pressed {

background-color: #4A3228;

}

/\* Метки для полей ввода \*/

#inputLabel, QLabel {

font-family: "Roboto", sans-serif;

font-size: 12pt;

color: #5C4033;

background-color: transparent;

padding: 5px 0;

}

/\* Поля ввода \*/

#inputField, QLineEdit, QSpinBox, QComboBox {

font-family: "Roboto", sans-serif;

font-size: 12pt;

background-color: #F5F5DC;

color: #5C4033;

border: 1px solid #D2B48C;

border-radius: 8px;

padding: 8px;

margin-bottom: 10px;

box-shadow: 0 1px 3px rgba(0, 0, 0, 0.1);

min-width: 200px;

}

#inputField:focus, QLineEdit:focus, QSpinBox:focus, QComboBox:focus {

border: 2px solid #5C4033;

background-color: #FFF5E6;

}

/\* Кнопки \*/

#loginButton, QPushButton {

font-family: "Roboto", sans-serif;

font-size: 12pt;

font-weight: 500;

background-color: #5C4033;

color: white;

border: none;

border-radius: 7px;

padding: 8px 15px;

margin-top: 8px;

box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);

}

#loginButton:hover, QPushButton:hover {

background-color: #7A5A47;

}

#loginButton:pressed, QPushButton:pressed {

background-color: #4A3228;

}

/\* Таблицы \*/

QTableWidget {

background-color: #FFF5E6;

border: 1px solid #D2B48C;

gridline-color: #D2B48C;

font-family: "Times New Roman";

font-size: 12px;

/\* background-color: #FFF5E6;

border: 1px solid #D2B48C;

border-radius: 5px;

box-shadow: 0 1px 3px rgba(0, 0, 0, 0.1); \*/

}

/\* Заголовки таблиц \*/

QHeaderView::section {

background-color: #D2B48C;

color: #5C4033;

padding: 5px;

border: 1px solid #5C4033;

font-family: "Times New Roman";

font-size: 12px;

}

/\* Стили для ячеек таблицы \*/

QTableWidget::item {

padding: 5px;

color: #5C4033;

}

/\* Вкладки \*/

QTabWidget::pane {

border: 1px solid #D2B48C;

background-color: #F5F5DC;

border-radius: 5px;

}

/\* Стили для редактора ячеек (QLineEdit, который появляется при редактировании) \*/

QTableWidget QLineEdit {

min-width: 150px; /\* Минимальная ширина редактора \*/

min-height: 25px; /\* Минимальная высота редактора \*/

padding: 2px; /\* Внутренние отступы \*/

font-family: "Times New Roman";

font-size: 12px; /\* Размер шрифта \*/

color: #5C4033; /\* Цвет текста \*/

background-color: #FFFFFF; /\* Цвет фона редактора \*/

border: 1px solid #D2B48C; /\* Граница редактора \*/

}

QTabBar::tab {

background-color: #D2B48C;

color: #5C4033;

padding: 8px 20px;

border-top-left-radius: 5px;

border-top-right-radius: 5px;

margin-right: 2px;

font-family: "Roboto", sans-serif;

font-size: 12pt;

}

QTabBar::tab:selected {

background-color: #FFF5E6;

color: #5C4033;

font-weight: bold;

}

QTabBar::tab:hover {

background-color: #E6D5B8;

color: #4A3228;

box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);

}

/\* Стили для выпадающего меню (отчеты и учетная запись) \*/

QMenu {

background-color: #FFF5E6;

border: 1px solid #D2B48C;

border-radius: 5px;

font-family: "Roboto", sans-serif;

font-size: 12pt;

color: #5C4033;

}

QMenu::item {

padding: 8px 20px;

background-color: transparent;

color: #5C4033;

}

QMenu::item:selected {

background-color: #D2B48C;

color: #FFF5E6;

}

/\* Боковая панель \*/

QWidget#sidebarWidget {

background-color: #FFF5E6;

border-left: 5px solid #D2B48C;

padding: 10px;

}

/\* Форма (SupplierForm, ProductForm, EmployeeForm) \*/

QWidget#formWidget {

background-color: #FFF5E6;

border: 3px solid #5C4033;

border-radius: 10px;

padding: 20px;

margin-top: 50px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);

}

/\* Заголовок формы \*/

QLabel#titleLabel {

font-family: "Roboto", sans-serif;

font-size: 14pt;

font-weight: 500;

color: #5C4033;

background-color: transparent;

padding: 5px 0;

margin-bottom: 10px;

border-bottom: 2px solid #D2B48C;

}

/\* Форма деталей товара (ProductDetailsForm) \*/

QWidget#detailsFormWidget {

background-color: #FFF5E6;

border: 3px solid #5C4033;

border-radius: 10px;

padding: 20px;

margin-top: 50px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);

}

/\* Заголовок формы деталей "Краткое описание" \*/

QLabel#detailsFormTitleLabel {

font-family: "Roboto", sans-serif;

font-size: 16pt;

font-weight: 600;

color: #5C4033;

background-color: transparent;

padding: 5px 0;

margin-bottom: 10px;

border-bottom: 2px solid #D2B48C;

text-align: center;

}

/\* Название категории товара \*/

QLabel#detailsCategoryLabel {

font-family: "Roboto", sans-serif;

font-size: 14pt;

font-weight: 500;

color: #5C4033;

background-color: transparent;

padding: 5px 0;

text-align: center;

}

/\* Метка "Описание" \*/

QLabel#detailsLabel {

font-family: "Roboto", sans-serif;

font-size: 14pt;

font-weight: 500;

color: #5C4033;

background-color: transparent;

padding: 5px 0;

}

/\* Текст описания \*/

QLabel#detailsTextLabel {

font-family: "Roboto", sans-serif;

font-size: 12pt;

color: #5C4033;

background-color: #F5F5DC;

padding: 10px;

border: 1px solid #D2B48C;

border-radius: 5px;

}

QLineEdit[placeholderText] {

color: #7A5A47;

}

Цветовая схема построена на теплой и гармоничной палитре, где ключевую роль играют оттенки бежевого и коричневого, такие как

* #F5F5DC (светло-бежевый, «Beige»).
* #5C4033 (темно-коричневый, «Dark Brown»).
* #D2B48C (мягкий коричневый, «Tan»).
* #FFF5E6 (легкий кремовый, «Light Cream»).
* #7A5A47 (средний коричневый, «Medium Brown»).
* #4A3228 (очень темный коричневый, «Very Dark Brown»).
* #E6D5B8 (светлый песочный, «Light Sand»)

Основной фон для главного окна, окна авторизации, диалоговых окон и полей ввода выполнен в светло-бежевом #F5F5DC, что создает спокойную и ненавязчивую атмосферу, не утомляющую глаза при длительной работе, а также придает интерфейсу чистый и минималистичный вид, подходящий для профессиональных приложений. Этот цвет хорошо контрастирует с темными элементами, такими как текст или кнопки, обеспечивая отличную читаемость в разных условиях освещения.

Темно-коричневый #5C4033 используется для текста, а также для кнопок, добавляя элегантный и строгий акцент, который воспринимается как теплый и надежный, что важно для делового интерфейса, а в сочетании с белым текстом на кнопках он выглядит стильно и четко.

Мягкий коричневый #D2B48C применяется для рамок диалоговых окон, таблиц, вкладок и меню, а также для фона заголовков таблиц, выступая связующим звеном между светлым фоном и темными акцентами, смягчая переходы и добавляя глубину без перегрузки дизайна.

Легкий кремовый #FFF5E6 используется для фона таблиц, активных вкладок, выпадающих меню и полей ввода, выделяя активные или интерактивные элементы, усиливая визуальную иерархию и помогая пользователю быстро ориентироваться, при этом сохраняя мягкость и чистоту.

Средний коричневый #7A5A47 появляется при наведении на кнопки, сигнализируя о взаимодействии и делая интерфейс более отзывчивым, а очень темный коричневый #4A3228 используется при нажатии на те же элементы, показывая активацию и обеспечивая визуальную обратную связь.

Светлый песочный #E6D5B8 применяется для эффекта наведения на вкладки, добавляя динамики и подчеркивая их интерактивность.

Такая схема выбрана из-за ее гармоничности и минимализма, ведь все цвета находятся в одной теплой палитре, создавая цельный и приятный глазу дизайн без резких контрастов, что делает интерфейс спокойным и профессиональным.

Окно авторизации пользователя (директора или оператора-флориста). Представляет собой небольшое окно с заголовком. В этом окне есть два поля: логин и пароль. Пользователь (директор или флорист) вводит свой логин и пароль для входа в свою учетную запись. После этого он нажимает на кнопку «Войти». Окно авторизации представлено на рисунке 18.

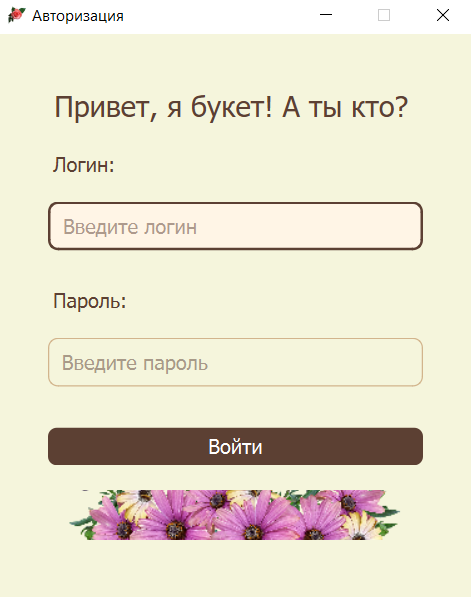


Рисунок 18 – Авторизация

Интерфейс пользователя (директора). После входа в программу представляет собой главное окно программы «Привет, я букет!», на котором отображаются поставщики товаров. Тут же находятся кнопки «Добавить поставщика», «Редактировать поставщика», «Удалить поставщика».

Интерфейс пользователя (флориста) после входа также представляет собой раздел «Поставщики», только без возможности добавления, редактирования и удаления данных.

Программа имеет 4 раздела:

* Поставщики.
* Товары.
* Сотрудники.
* Отчеты.

Автоматически открывается раздел «Поставщики». В данном разделе директор видит таблицу со всеми поставщиками и информацию о них: название, тип поставщика, телефон и email, что и показано на рисунке 19.



Рисунок 19 – Отображение раздела «Поставщики»

При нажатии на кнопку «Добавить поставщика» открывается форма добавления поставщика. Здесь директор вводит название поставщика, выбирает тип поставщика, вводит номер телефона, Email и юридический адрес. Аналогичным образом выполнены формы добавления в разделах «Товары» и «Сотрудники».

Форма добавления представлена на рисунке 20.

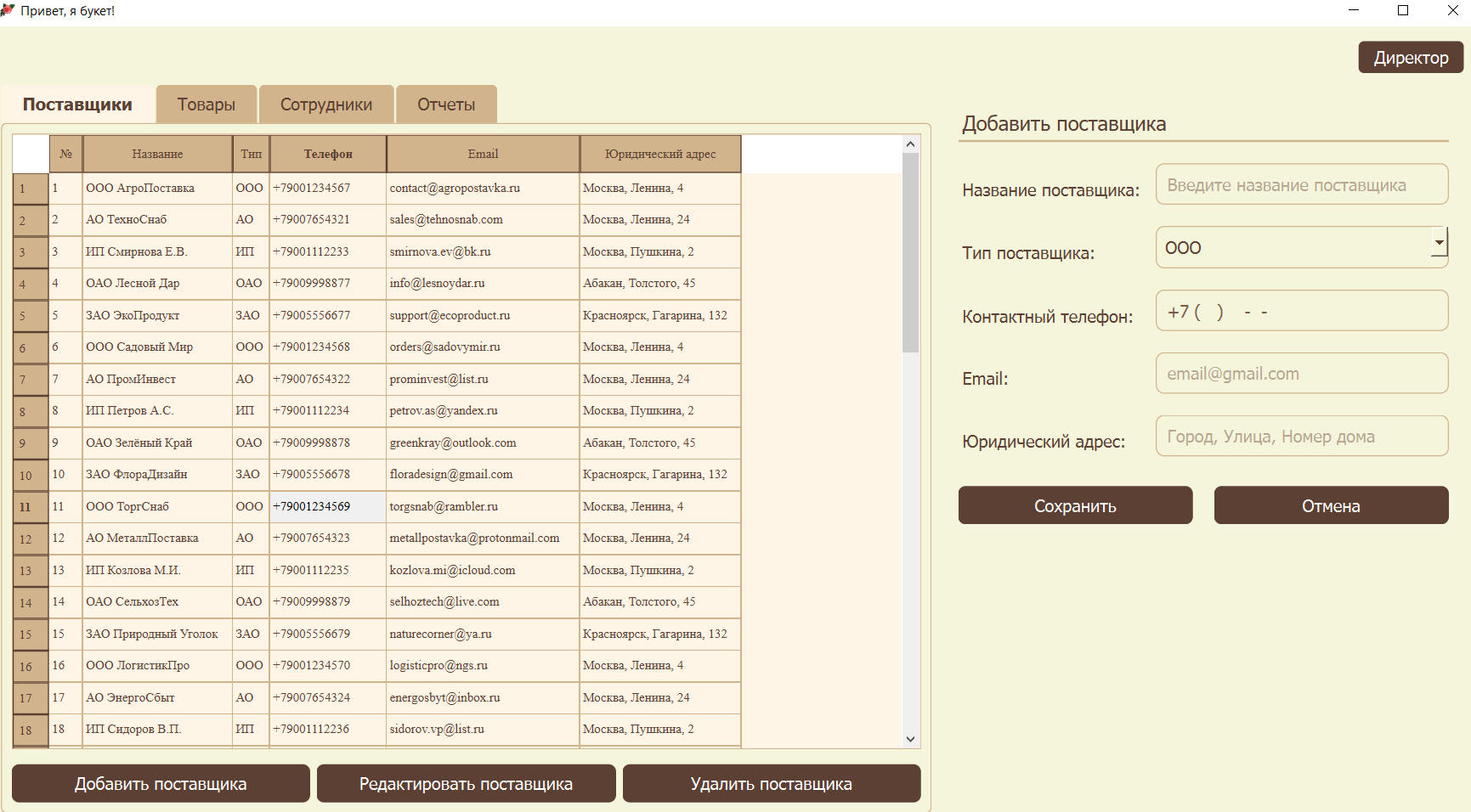


Рисунок 20 – Отображение формы добавления поставщика

Форма редактирования аналогична форме добавления и представлена на рисунке 21. Формы редактирования для разделов «Товары» и «Сотрудники» выполнены аналогичным образом.

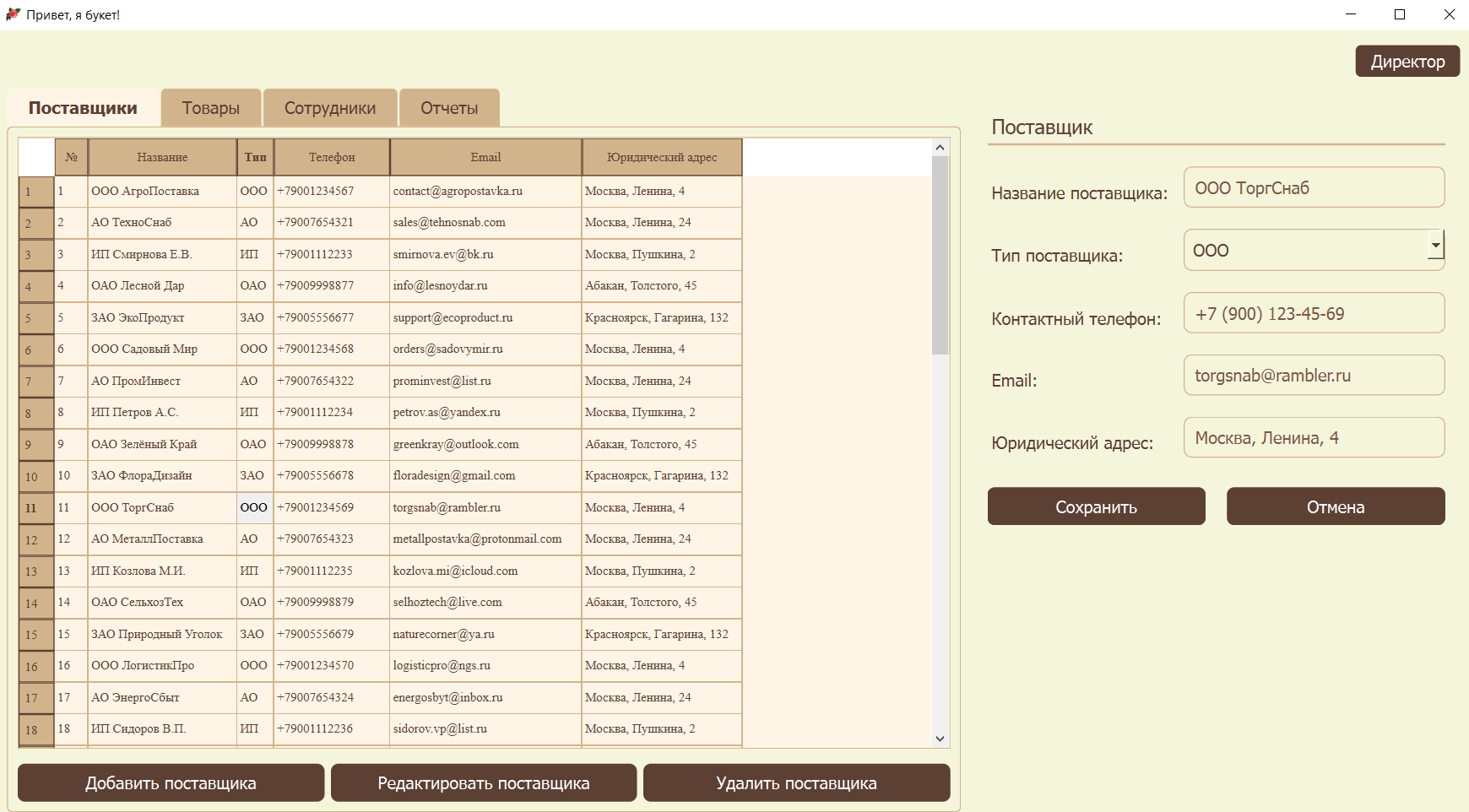


Рисунок 21 – Отображение формы редактирования поставщика

Диалоговое окно, представленное на рисунке 22, является подтверждением удаления поставщика. Аналогичным образом выполнены все диалоговые окна для удаления, закрытия чего-либо, подтверждения изменений.

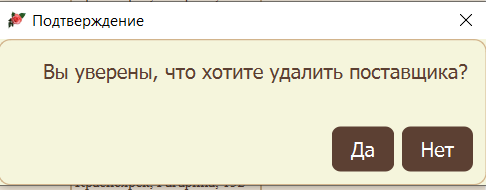


Рисунок 22 – Отображение диалогового окна

После нажатии пользователем (директором) на раздел «Товары» система отображает таблицу с товарами, в которой хранятся данные о типе товара, его номере, описании, цене и поставщике. Также тут присутствуют кнопки для добавления, редактирования и удаления данных из этой таблицы.

Для флориста разница только в том, что у него нет кнопок добавления, редактирования и удаления данных из таблиц.

Раздел «Товары» представлен на рисунке 23.



Рисунок 23 – Отображение раздела «Товары»

При двойном нажатии (и директором и флористом) на любой товар открывается его краткое описание, которое состоит из наименования типа товара, его описания и его картинки. Краткое описание представлено на рисунке 24.

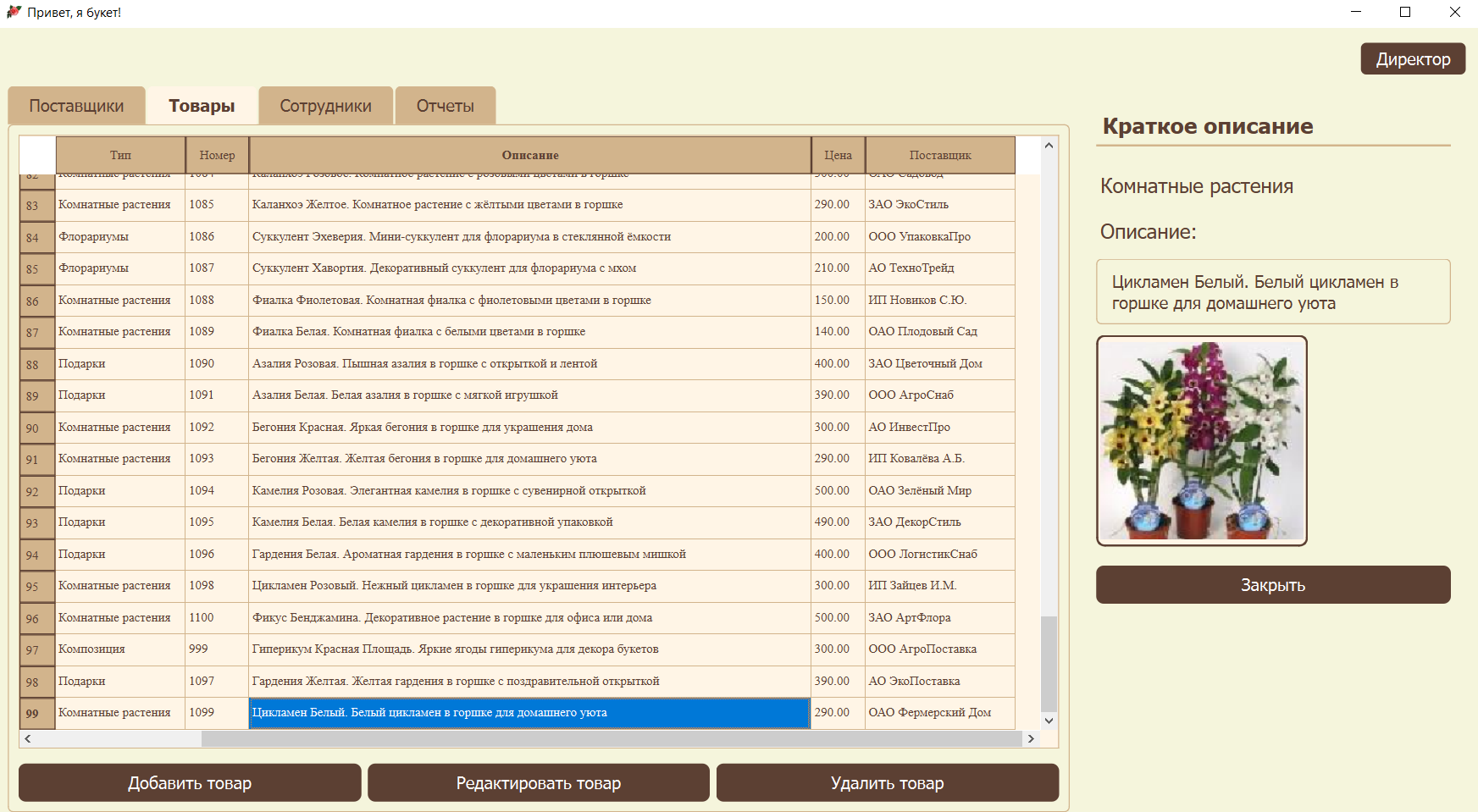


Рисунок 24 – Краткое описание товара

При переходе на раздел «Сотрудники» директор видит таблицу сотрудников, а также кнопки, позволяющие добавить, редактировать или удалить сотрудника.

Для флориста разница только в том, что у него нет кнопок добавления, редактирования и удаления данных из таблиц.

Раздел «Сотрудники» представлен на рисунке 25.

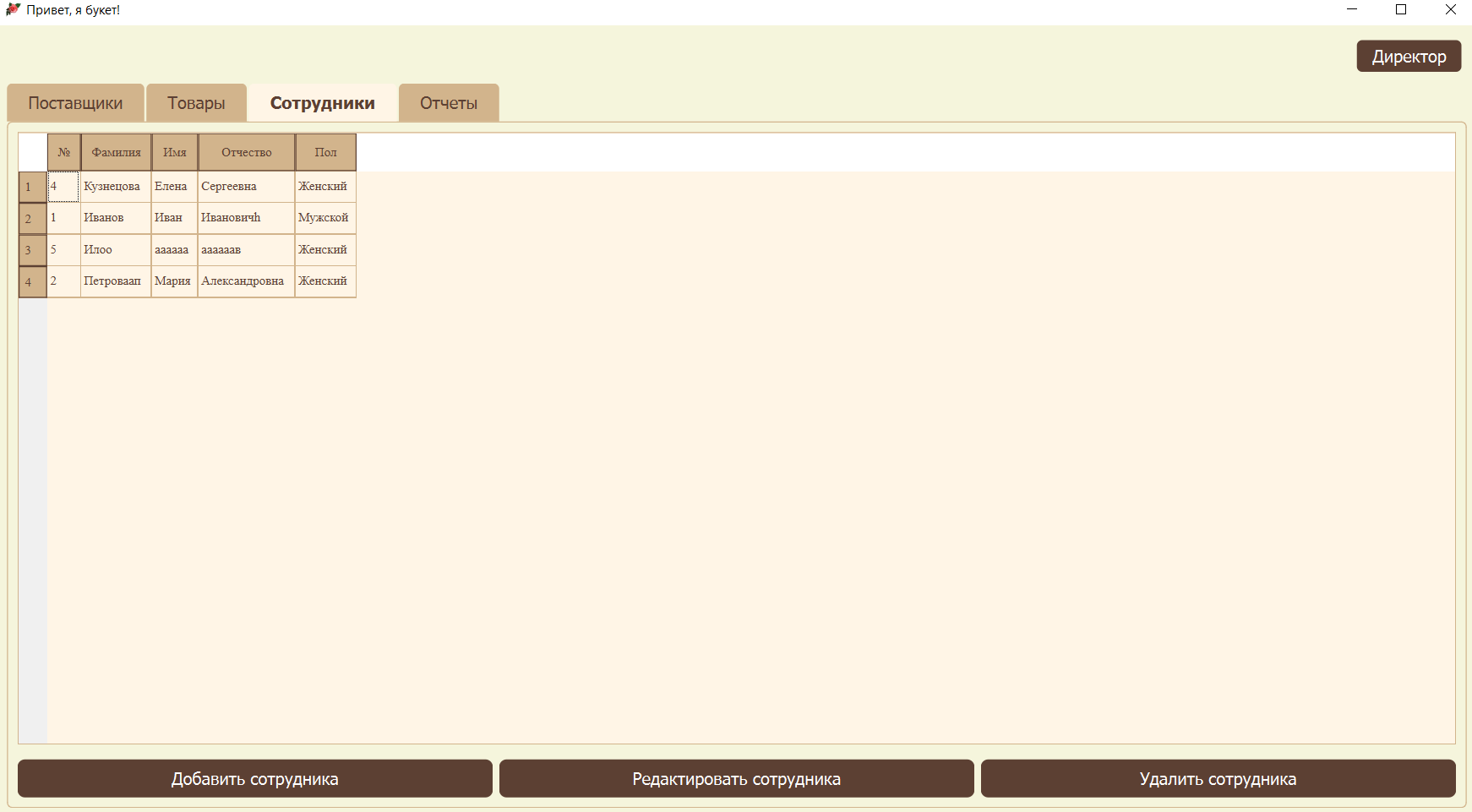


Рисунок 25 – Отображение раздела «Сотрудники»

При переходе на раздел «Отчеты» пользователь (и директор и флорист) изначально видит пустое окно с возможностью выбора необходимого отчета. Раздел «Отчеты» представлен на рисунке 26.



Рисунок 26 – Отображение раздела «Отчеты»

При выборе любого из отчетов и у директора, и у флориста появляется диалоговое окно с возможностью выбора даты формирования отчета.

Диалоговое окно представлено на рисунке 27.

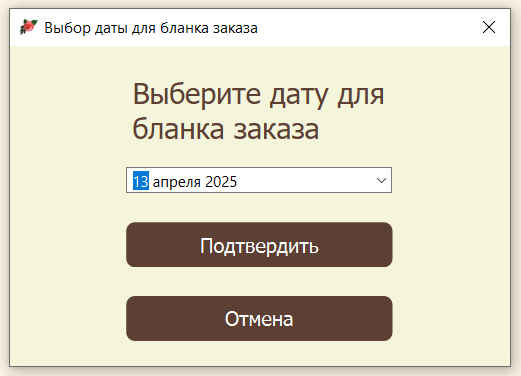


Рисунок 27 – Отображение раздела «Поставщики»

После нажатия на кнопку «Подтвердить» и у директора, и у флориста отображается выбранный отчет. Отчет представлен на рисунке 28.

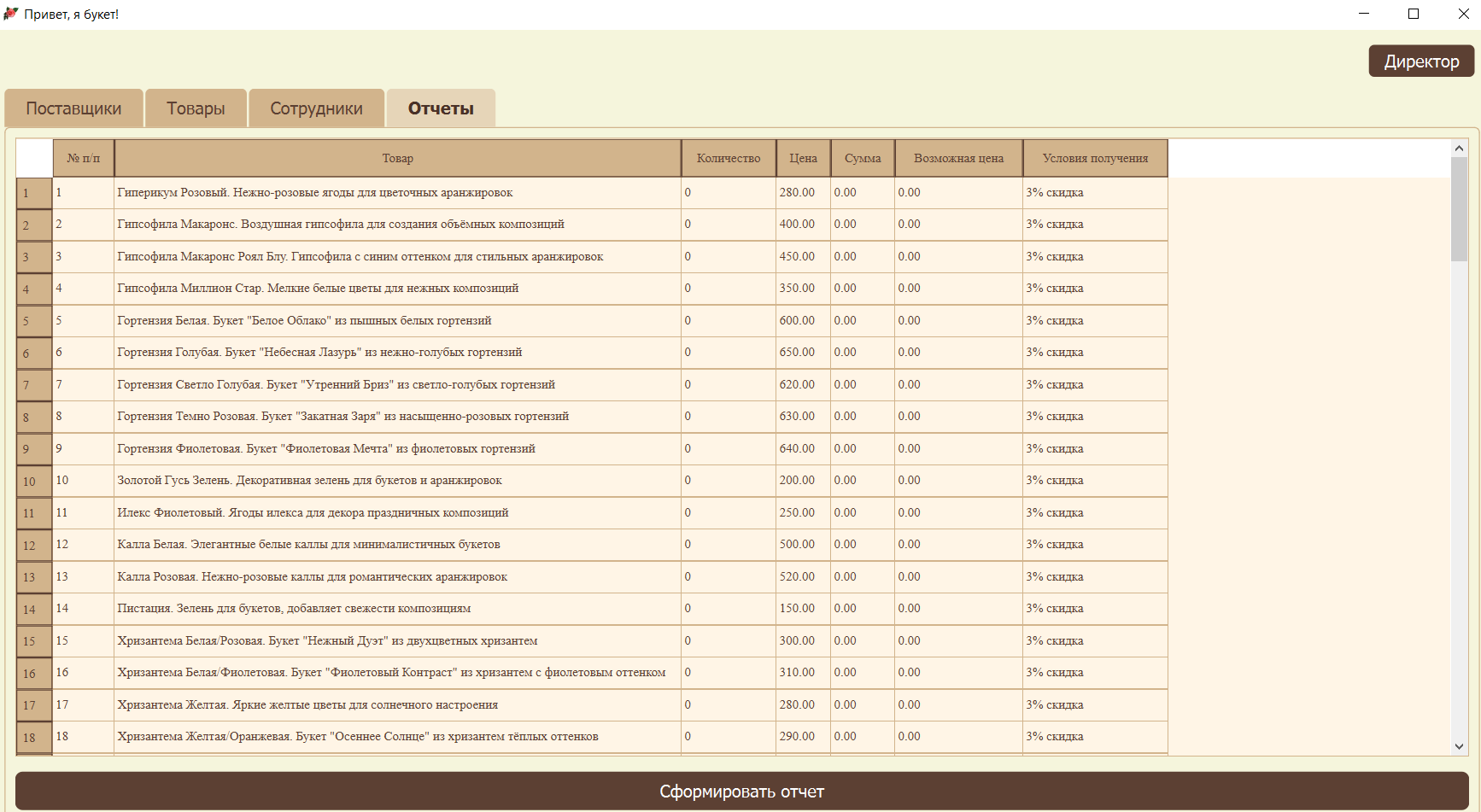


Рисунок 28 – Отображение отчета

При нажатии на кнопку в правом верхнем углу, если это директор, то кнопка «Директор», если это флорист, то кнопка «Флорист», у пользователя появляется 2 варианта: выход из учетной записи (в этом случае пользователя перебрасывает на окно авторизации, чтобы он сменил учетную запись) или выход из программы (в этом случае, при подтверждении, программа закрывается). Это представлено на рисунке 29.

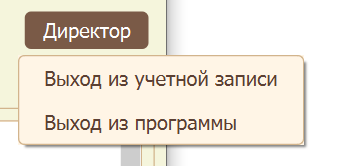


Рисунок 29 – Выход из учетной записи/Выход из программы

* 1. Разработка кода программного модуля

Пароли в программе хэшированные, в листинге 2 представлены пароли для директора и флориста.

Листинг 2 – Хэширование паролей

USERS = {

"direktor": hashlib.sha256("123456".encode()).hexdigest(),

"florist": hashlib.sha256("123456".encode()).hexdigest()

Метод authenticate в классе AuthWindow для авторизации представлен на листинге 3.

Листинг 3 – Метод authenticate

    def authenticate(self):

        login = self.login\_entry.text().strip()

        password = self.password\_entry.text().strip()

        hashed\_password = hashlib.sha256(password.encode()).hexdigest()

        if login in USERS and USERS[login] == hashed\_password:

            print(f"Авторизация успешна для пользователя: {login}")

            self.close()

            try:

                self.launch\_main\_window(login)

            except Exception as e:

                msg = QMessageBox(self)

                msg.setWindowTitle("Ошибка")

                msg.setText(f"Ошибка при запуске главного окна: {str(e)}")

                msg.setStandardButtons(QMessageBox.Ok)

                msg.button(QMessageBox.Ok).setText("Хорошо")

                msg.exec\_()

                raise

        else:

            msg = QMessageBox(self)

            msg.setWindowTitle("Ошибка")

            msg.setText("Неверный логин или пароль!")

            msg.setStandardButtons(QMessageBox.Ok)

            msg.button(QMessageBox.Ok).setText("Хорошо")

            msg.exec\_()

При вводе неверного пароля, как в случае, представленном на рисунке 30, система выводит сообщение о том, что логин или пароль неверный и не дает зайти в программу.

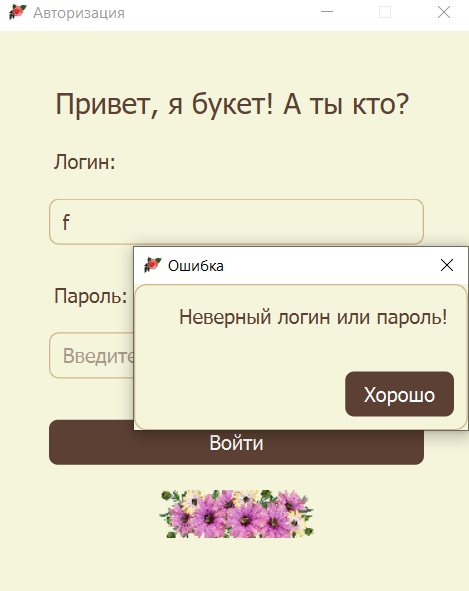


Рисунок 30 – Авторизация

Загрузка поставщиков из БД представлена на листинге 4.

Листинг 4 – Загрузка поставщиков из БД

    def load\_suppliers\_data(self):

        suppliers = self.session.query(Postavshik).all()

        self.suppliers\_table.setRowCount(len(suppliers))

        self.suppliers\_table.setColumnCount(6)

        self.suppliers\_table.setHorizontalHeaderLabels([

            "№", "Название", "Тип", "Телефон", "Email", "Юридический адрес"

        ])

        for row, supplier in enumerate(suppliers):

            self.suppliers\_table.setItem(row, 0, QTableWidgetItem(str(supplier.id)))

            self.suppliers\_table.setItem(row, 1, QTableWidgetItem(supplier.nazvanie\_postavshika))

            self.suppliers\_table.setItem(row, 2, QTableWidgetItem(supplier.tip\_postavshika.value))

            self.suppliers\_table.setItem(row, 3, QTableWidgetItem(supplier.kontakt\_tel or ""))

            self.suppliers\_table.setItem(row, 4, QTableWidgetItem(supplier.email or ""))

            address\_str = ""

            if supplier.juridicheskij\_adres\_rel:

                address\_str = (

                    f"{supplier.juridicheskij\_adres\_rel.ulica.gorod.nazvanie}, "

                    f"{supplier.juridicheskij\_adres\_rel.ulica.nazvanie}, "

                    f"{supplier.juridicheskij\_adres\_rel.nomer}"

                )

            self.suppliers\_table.setItem(row, 5, QTableWidgetItem(address\_str))

        self.suppliers\_table.resizeColumnsToContents()

Загруженные поставщики представлены на рисунке 31. Аналогичным образом представлена загрузка данных о товарах и о сотрудниках.



Рисунок 31 – Авторизация

Пример добавления товара представлен на листинге 5. Аналогичным образом работает добавление поставщика и сотрудника.

Листинг 5 – Добавление товара

    def load\_products\_data(self):

        products = self.session.query(Tovar).all()

        self.products\_table.setRowCount(len(products))

        self.products\_table.setColumnCount(6)

        self.products\_table.setHorizontalHeaderLabels([

            "№", "Тип", "Номер", "Описание", "Цена", "Поставщик"

        ])

        for row, product in enumerate(products):

            self.products\_table.setItem(row, 0, QTableWidgetItem(str(product.id)))

            self.products\_table.setItem(row, 1, QTableWidgetItem(product.nazvanie.value))

            self.products\_table.setItem(row, 2, QTableWidgetItem(str(product.nomer)))

            self.products\_table.setItem(row, 3, QTableWidgetItem(product.opisanie or ""))

            self.products\_table.setItem(row, 4, QTableWidgetItem(str(product.cena) if product.cena else ""))

            self.products\_table.setItem(row, 5, QTableWidgetItem(product.postavshik.nazvanie\_postavshika if product.postavshik else ""))

        self.products\_table.resizeColumnsToContents()

        # Подключаем обработчик двойного нажатия

        self.products\_table.doubleClicked.connect(self.show\_product\_details)

    def add\_product(self):

        print("Открытие формы добавления товара...")

        try:

            suppliers = self.session.query(Postavshik).all()

            if not suppliers:

                msg = QMessageBox(self)

                msg.setWindowTitle("Ошибка")

                msg.setText("Сначала добавьте поставщика!")

                msg.setStandardButtons(QMessageBox.Ok)

                msg.button(QMessageBox.Ok).setText("Хорошо")

                msg.exec\_()

                return

            self.clear\_sidebar()

            self.show\_sidebar()

            form = ProductForm(self, product=None, suppliers=suppliers)

            self.sidebar\_layout.addWidget(form)

            form.submit\_button.clicked.connect(lambda: self.save\_product(form, suppliers))

            form.cancel\_button.clicked.connect(self.clear\_sidebar)

            print("Форма добавления товара успешно открыта.")

        except Exception as e:

            print(f"Ошибка при открытии формы добавления товара: {str(e)}")

            msg = QMessageBox(self)

            msg.setWindowTitle("Ошибка")

            msg.setText(f"Произошла ошибка при открытии формы добавления товара: {str(e)}")

            msg.setStandardButtons(QMessageBox.Ok)

            msg.button(QMessageBox.Ok).setText("Хорошо")

            msg.exec\_()

Добавление товара показано на рисунке 32.

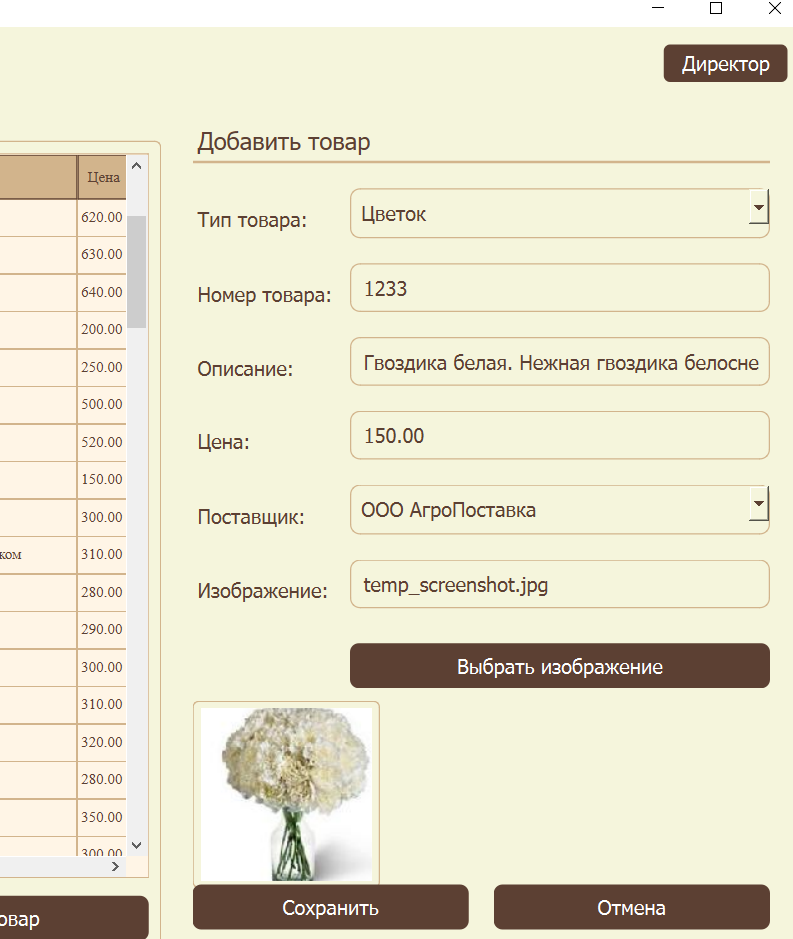


Рисунок 32 – Добавление товара

Форма для детального просмотра товара представлена на листинге 6.

Листинг 6 – Краткое описание товара

    def show\_product\_details(self):

        print("Двойное нажатие на таблицу товаров...")

        selected = self.products\_table.currentRow()

        if selected < 0:

            msg = QMessageBox(self)

            msg.setWindowTitle("Ошибка")

            msg.setText("Выберите товар для просмотра деталей!")

            msg.setStandardButtons(QMessageBox.Ok)

            msg.button(QMessageBox.Ok).setText("Хорошо")

            msg.exec\_()

            return

        product\_id = int(self.products\_table.item(selected, 0).text())

        product = self.session.query(Tovar).filter\_by(id=product\_id).first()

        if not product:

            msg = QMessageBox(self)

            msg.setWindowTitle("Ошибка")

            msg.setText("Товар не найден!")

            msg.setStandardButtons(QMessageBox.Ok)

            msg.button(QMessageBox.Ok).setText("Хорошо")

            msg.exec\_()

            return

        self.clear\_sidebar()

        self.show\_sidebar()

        form = ProductDetailsForm(self, product=product, main\_window=self)  # Передаём self как main\_window

        self.sidebar\_layout.addWidget(form)

        print(f"Открыта форма деталей для товара ID: {product\_id}")

На рисунке 33 представлено краткое описание товара.



Рисунок 33 – Краткое описание товара

Форма редактирования на примере поставщика представлена на листинге 7. Аналогичным образом выполнено редактирование товаров и сотрудников.

Листинг 7 – Редактирование поставщика

    def edit\_supplier(self):

        print("Открытие формы редактирования поставщика...")

        try:

            selected = self.suppliers\_table.currentRow()

            if selected < 0:

                msg = QMessageBox(self)

                msg.setWindowTitle("Ошибка")

                msg.setText("Выберите поставщика для редактирования!")

                msg.setStandardButtons(QMessageBox.Ok)

                msg.button(QMessageBox.Ok).setText("Хорошо")

                msg.exec\_()

                return

            supplier\_id = int(self.suppliers\_table.item(selected, 0).text())

            supplier = self.session.query(Postavshik).filter\_by(id=supplier\_id).first()

            if not supplier:

                msg = QMessageBox(self)

                msg.setWindowTitle("Ошибка")

                msg.setText("Поставщик не найден!")

                msg.setStandardButtons(QMessageBox.Ok)

                msg.button(QMessageBox.Ok).setText("Хорошо")

                msg.exec\_()

                return

            self.clear\_sidebar()

            self.show\_sidebar()

            form = SupplierForm(self, supplier=supplier)

            self.sidebar\_layout.addWidget(form)

            form.submit\_button.clicked.connect(lambda: self.save\_supplier(form, supplier))

            form.cancel\_button.clicked.connect(self.clear\_sidebar)

            print("Форма редактирования поставщика успешно открыта.")

        except Exception as e:

            print(f"Ошибка при открытии формы редактирования поставщика: {str(e)}")

            msg = QMessageBox(self)

            msg.setWindowTitle("Ошибка")

            msg.setText(f"Произошла ошибка при открытии формы редактирования поставщика: {str(e)}")

            msg.setStandardButtons(QMessageBox.Ok)

            msg.button(QMessageBox.Ok).setText("Хорошо")

            msg.exec\_()

На рисунке 34 представлена форма редактирования поставщика.

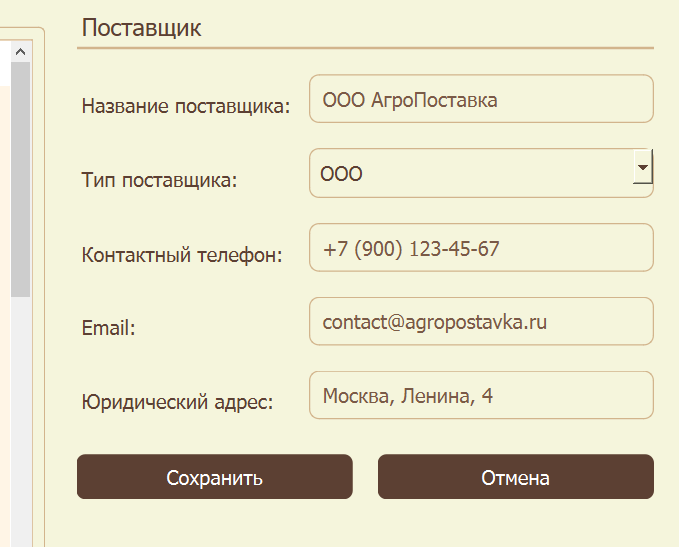


Рисунок 34 – Форма редактирования поставщика

Функция удаления показана на примере сотрудника, что и представлено на листинге 8. Аналогичным образом реализовано удаление поставщика и товара.

Листинг 8 – Удаление сотрудника

    def delete\_employee(self):

        selected = self.employees\_table.currentRow()

        if selected < 0:

            msg = QMessageBox(self)

            msg.setWindowTitle("Ошибка")

            msg.setText("Выберите сотрудника для удаления!")

            msg.setStandardButtons(QMessageBox.Ok)

            msg.button(QMessageBox.Ok).setText("Хорошо")

            msg.exec\_()

            return

        employee\_id = int(self.employees\_table.item(selected, 0).text())

        employee = self.session.query(Sotrudnik).filter\_by(id=employee\_id).first()

        if not employee:

            msg = QMessageBox(self)

            msg.setWindowTitle("Ошибка")

            msg.setText("Сотрудник не найден!")

            msg.setStandardButtons(QMessageBox.Ok)

            msg.button(QMessageBox.Ok).setText("Хорошо")

            msg.exec\_()

            return

        msg = QMessageBox(self)

        msg.setWindowTitle("Подтверждение")

        msg.setText("Вы уверены, что хотите удалить сотрудника?")

        msg.setStandardButtons(QMessageBox.Yes | QMessageBox.No)

        msg.setDefaultButton(QMessageBox.No)

        msg.button(QMessageBox.Yes).setText("Да")

        msg.button(QMessageBox.No).setText("Нет")

        reply = msg.exec\_()

        if reply == QMessageBox.Yes:

            self.session.delete(employee)

            self.session.commit()

            self.load\_employees\_data()

            msg\_success = QMessageBox(self)

            msg\_success.setWindowTitle("Успех")

            msg\_success.setText("Сотрудник удален!")

            msg\_success.setStandardButtons(QMessageBox.Ok)

            msg\_success.button(QMessageBox.Ok).setText("Хорошо")

            msg\_success.exec\_()

Удаление сотрудника показано на рисунках 35 – 36.

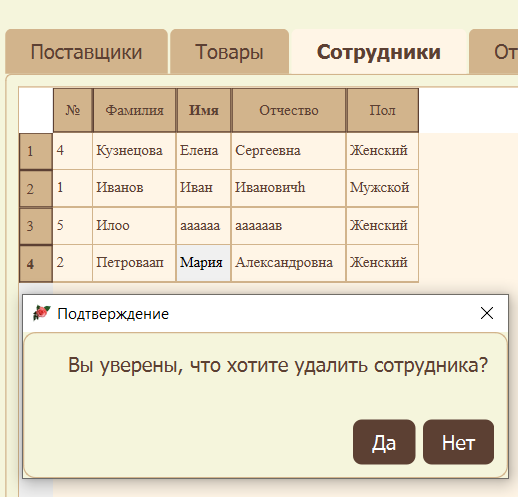


Рисунок 35 – Удаление сотрудника

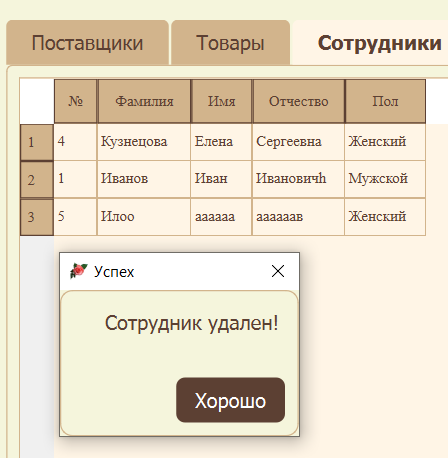


Рисунок 36 – Удаление сотрудника

Формирование отчета на примере документа «Бланк заказа» представлено на листинге 9. Остальные отчеты выполнены аналогичным образом.

Листинг 9 – Формирование «Бланка заказа»

    def load\_order\_conversion\_reports(self):

        self.current\_report\_type = "order\_conversion"

        date\_dialog = DateSelectionDialog(self)

        date\_dialog.setWindowTitle("Выбор даты для бланка заказа")

        date\_dialog.title\_label.setText("Выберите дату для бланка заказа")

        if date\_dialog.exec\_() == QDialog.Accepted:

            selected\_qdate = date\_dialog.get\_selected\_date()

            self.selected\_order\_date\_obj = selected\_qdate.toPython()  # Сохраняем как datetime.date

            # Задаем локализацию на русский язык

            locale = QLocale(QLocale.Russian, QLocale.Russia)

            self.selected\_order\_date = locale.toString(selected\_qdate, "d MMMM yyyy") + "г"  # Строковое представление для PDF

            # Загружаем все товары из таблицы Tovar

            products = self.session.query(Tovar).all()

            # Сохраняем список товаров для дальнейшего использования (например, в PDF)

            self.order\_products = products

            # Формируем таблицу

            self.reports\_table.setRowCount(len(products))

            self.reports\_table.setColumnCount(7)  # Добавляем колонку "Условия получения"

            self.reports\_table.setHorizontalHeaderLabels([

                "№ п/п", "Товар", "Количество", "Цена", "Сумма", "Возможная цена", "Условия получения"

            ])

            for row, product in enumerate(products):

                # № п/п (порядковый номер строки, начиная с 1)

                self.reports\_table.setItem(row, 0, QTableWidgetItem(str(row + 1)))

                self.reports\_table.item(row, 0).setFlags(Qt.ItemIsSelectable | Qt.ItemIsEnabled)  # Запрещаем редактирование

                # Товар: берем из Tovar.opisanie

                self.reports\_table.setItem(row, 1, QTableWidgetItem(product.opisanie if product.opisanie else ""))

                self.reports\_table.item(row, 1).setFlags(Qt.ItemIsSelectable | Qt.ItemIsEnabled)  # Запрещаем редактирование

                # Количество: изначально 0, редактируемое

                quantity\_item = QTableWidgetItem("0")

                self.reports\_table.setItem(row, 2, quantity\_item)

                # Цена: берем из Tovar.cena

                price = float(product.cena) if product.cena else 0.00

                self.reports\_table.setItem(row, 3, QTableWidgetItem(f"{price:.2f}"))

                self.reports\_table.item(row, 3).setFlags(Qt.ItemIsSelectable | Qt.ItemIsEnabled)  # Запрещаем редактирование

                # Сумма: изначально 0 (Количество \* Цена)

                self.reports\_table.setItem(row, 4, QTableWidgetItem("0.00"))

                self.reports\_table.item(row, 4).setFlags(Qt.ItemIsSelectable | Qt.ItemIsEnabled)  # Запрещаем редактирование

                # Возможная цена: изначально 0 (Сумма \* 0.03)

                self.reports\_table.setItem(row, 5, QTableWidgetItem("0.00"))

                self.reports\_table.item(row, 5).setFlags(Qt.ItemIsSelectable | Qt.ItemIsEnabled)  # Запрещаем редактирование

                # Условия получения: фиксированное значение "3% скидка"

                self.reports\_table.setItem(row, 6, QTableWidgetItem("3% скидка"))

                self.reports\_table.item(row, 6).setFlags(Qt.ItemIsSelectable | Qt.ItemIsEnabled)  # Запрещаем редактирование

            self.reports\_table.resizeColumnsToContents()

            # Подключаем обработчик изменения количества

            self.reports\_table.itemChanged.connect(self.update\_order\_totals)

            # Добавляем контекстное меню для удаления строк

            self.reports\_table.setContextMenuPolicy(Qt.CustomContextMenu)

            self.reports\_table.customContextMenuRequested.connect(self.show\_context\_menu)

        else:

            msg = QMessageBox(self)

            msg.setWindowTitle("Отмена")

            msg.setText("Формирование бланка заказа отменено.")

            msg.setStandardButtons(QMessageBox.Ok)

            msg.button(QMessageBox.Ok).setText("Хорошо")

            msg.exec\_()

Раздел «Отчеты» представлен на рисунке 37.



Рисунок 37 – Раздел «Отчеты»

Выбор даты для формирования отчета представлен на рисунке 38.

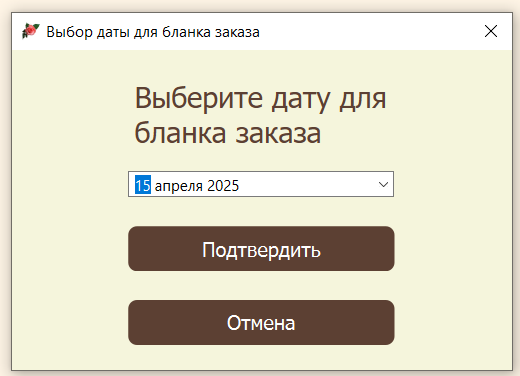


Рисунок 38 – Выбор даты

Готовый отчет в программе представлен на рисунке 39.



Рисунок 39 – Отчет «Бланк заказа» в программе

Готовый отчет в PDF представлен на рисунке 40 – 41.

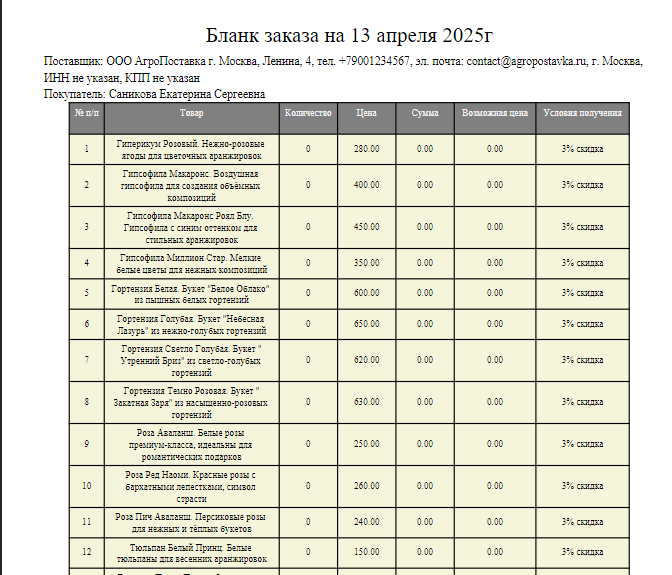
****

Рисунок 40 – Отчет «Бланк заказа»

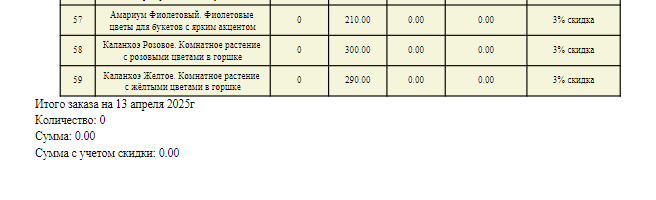


Рисунок 41 – Отчет «Бланк заказа»

1. ТЕСТИРОВАНИЕ ПРОГРАММНОГО МОДУЛЯ
   1. Разработка тест-кейсов

Название: Загрузка изображения товара при добавлении.

УИД: 1.

Предусловие: Пользователь вошел под учетной записью директора, программа автоматически загрузила данные из БД, раздел «Товары» открыт.

Шаги:

* Нажать на кнопку «Добавить товар».
* В форме добавления товара выбрать тип товара «Композиция» из выпадающего списка.
* Ввести номер товара «15».
* Ввести описание «Композиция «Летний сад» из роз и лилий».
* Ввести цену «2500.00».
* Выбрать поставщика «ИП Сидоров Ф.Ю.» из выпадающего списка.
* Нажать на кнопку «Выбрать изображение» и выбрать файл «summer\_garden.jpg» (размер 500x500 пикселей).
* Нажать на кнопку «Сохранить».

Ожидаемый результат: Появляется сообщение «Товар добавлен!», товар «Композиция «Летний сад»» отображается в списке товаров, изображение корректно отображается в кратком описании товаров (масштабировано до 150x150 пикселей).

Фактический результат: Появляется сообщение «Товар добавлен!», товар «Композиция «Летний сад»» отображается в списке товаров, изображение корректно отображается в кратком описании товаров (масштабировано до 150x150 пикселей).

Название: Проверка ошибки при добавлении поставщика с некорректным email.

УИД: 2.

Предусловие: Пользователь вошел под учетной записью директора, программа автоматически загрузила данные из БД, раздел «Поставщики» открыт.

Шаги:

* Нажать на кнопку «Добавить поставщика».
* Ввести название поставщика «ООО Ромашка».
* Выбрать тип поставщика «ООО» из выпадающего списка.
* Ввести контактный телефон «+79991234567».
* Ввести некорректный email «romashka@».
* Ввести юридический адрес «Москва, ул. Ленина, 10».
* Нажать на кнопку «Сохранить».

Ожидаемый результат: Появляется сообщение «Введите корректный email!». Поставщик не добавляется в список.

Фактический результат: Появляется сообщение «Введите корректный email!». Поставщик не добавляется в список.

Название: Формирование отчета по поступлению товаров с указанием даты.

УИД: 3.

Предусловие: Пользователь вошел под учетной записью флориста, программа автоматически загрузила данные из БД, в базе есть товары и поставщики.

Шаги:

* Нажать на раздел «Отчеты».
* Навести курсор на вкладку «Отчеты» и выбрать «Отчет по поступлению товаров».
* В появившемся окне выбора даты выбрать «12 апреля 2025».
* Нажать на кнопку «Подтвердить».
* В таблице отчета для товара «Букет «Фиолетовая Мечта»» ввести количество «10».
* Нажать на кнопку «Сформировать отчет».

Ожидаемый результат: Отчет в формате PDF формируется с названием «arrival\_report\_[дата\_время].pdf», содержит данные о поступлении 10 единиц товара «Букет «Фиолетовая Мечта»» за 12 апреля 2025, сохраняется в папке программы.

Фактический результат: Отчет в формате PDF формируется с названием «arrival\_report\_[дата\_время].pdf», содержит данные о поступлении 10 единиц товара «Букет «Фиолетовая Мечта»» за 12 апреля 2025, сохраняется в папке программы.

Название: Просмотр деталей товара с изображением.

УИД: 4.

Предусловие: Пользователь вошел под учетной записью директора, программа автоматически загрузила данные из БД, в базе есть товар «Подарок «Плюшевый мишка»» с изображением.

Шаги:

* Нажать на раздел «Товары».
* Дважды нажать на строку с товаром «Подарок «Плюшевый мишка»» в таблице товаров.

Ожидаемый результат: На боковой панели открывается форма «Краткое описание», отображаются: категория «Подарки», описание «Подарок «Плюшевый мишка»», изображение товара (масштабировано до 200x200 пикселей), кнопка «Закрыть».

Фактический результат: На боковой панели открывается форма «Краткое описание», отображаются: категория «Подарки», описание «Подарок «Плюшевый мишка»», изображение товара (масштабировано до 200x200 пикселей), кнопка «Закрыть».

Название: Удаление строки из отчета по заказу с пересчетом номеров.

УИД: 5.

Предусловие: Пользователь вошел под учетной записью флориста, программа автоматически загрузила данные из БД, открыт раздел «Отчеты», выбран «Бланк заказа» за 15 апреля 2025.

Шаги:

* В таблице бланка заказа для товара «Букет «Фиолетовая Мечта»» ввести количество «5».
* Выделить строку с товаром «Подарок «Плюшевый мишка»».
* Щелкнуть правой кнопкой мыши и выбрать «Удалить строку» в контекстном меню.
* Нажать на кнопку «Сформировать отчет».

Ожидаемый результат: Строка с товаром «Подарок «Плюшевый мишка»» удалена, номера строк в столбце «№ п/п» пересчитаны (начинаются с 1), отчет в формате PDF формируется с данными о 5 единицах «Букет «Фиолетовая Мечта»», сохраняется в папке программы.

Фактический результат: Строка с товаром «Подарок «Плюшевый мишка»» удалена, номера строк в столбце «№ п/п» пересчитаны (начинаются с 1), отчет в формате PDF формируется с данными о 5 единицах «Букет «Фиолетовая Мечта»», сохраняется в папке программы.

* 1. Проведение модульного тестирования

Модульное тестирование – это метод тестирования программного обеспечения, который направлен на проверку отдельных модулей или компонентов системы на корректность их работы. Основной целью модульного тестирования является выявление ошибок на ранних этапах разработки, что позволяет существенно снизить затраты на исправление дефектов в дальнейшем.

В процессе модульного тестирования был создан небольшой тестовый модуль для приложения «Привет, я букет!». Данный тестовый модуль автоматизирует тест-кейс «ошибка входа при введении неверного логина или пароля». Для создания модуля использовалась библиотека pytest.

Листинг 10 – Тестовый модуль

import pytest

import pyautogui

import subprocess

import time

import os

import sys

# Настройки PyAutoGUI

pyautogui.FAILSAFE = True # Перемещение мыши в верхний левый угол останавливает выполнение

pyautogui.PAUSE = 0.5 # Пауза между действиями (в секундах)

# Путь к исполняемому файлу программы (замените на актуальный путь)

APP\_PATH = os.path.abspath("main.py") # Предполагается, что main.py запускает приложение

@pytest.fixture(scope="module")

def setup\_app():

"""Запускает приложение и закрывает его после тестов."""

# Запускаем приложение

process = subprocess.Popen([sys.executable, APP\_PATH])

time.sleep(3) # Даем приложению время на запуск

yield

# Закрываем приложение

process.terminate()

time.sleep(1) # Ждем завершения процесса

def test\_invalid\_login(setup\_app):

"""Тест-кейс: ошибка входа при вводе неверного логина или пароля (УИД: 1)."""

# Предполагаем, что окно авторизации активно

# Фокусируемся на окне приложения (можно уточнить координаты, если нужно)

pyautogui.click(100, 100) # Клик в область окна для активации

# Шаг 1: Вводим правильный логин "direktor"

pyautogui.write("direktor")

pyautogui.press("tab") # Переход к полю пароля

# Шаг 2: Вводим неверный пароль "1"

pyautogui.write("1")

pyautogui.press("tab") # Переход к кнопке "Войти"

# Шаг 3: Нажимаем кнопку "Войти" (предполагаем, что кнопка активна после табуляции)

pyautogui.press("enter")

# Ждем появления сообщения об ошибке (окно QMessageBox)

time.sleep(1)

# Проверяем наличие сообщения об ошибке

# PyAutoGUI не может напрямую читать текст окна, поэтому используем эвристику:

# Проверяем, что фокус перешел на кнопку "Хорошо" в диалоговом окне

pyautogui.press("enter") # Нажимаем "Хорошо" для закрытия сообщения

# Проверяем, что окно авторизации осталось открытым (косвенная проверка)

# Если окно авторизации активно, можем попробовать снова ввести текст

pyautogui.click(100, 100) # Снова фокусируемся на окне

pyautogui.write("test") # Пробуем ввести текст в поле логина

pyautogui.hotkey("ctrl", "a") # Выделяем введенный текст

pyautogui.press("backspace") # Удаляем его

# Если мы можем ввести текст, предполагаем, что окно авторизации осталось открытым

assert True, "Окно авторизации осталось открытым после ошибки входа"

if \_\_name\_\_ == "\_\_main\_\_":

pytest.main(["-v", \_\_file\_\_])

Результаты тестирования представлены на рисунке 42.

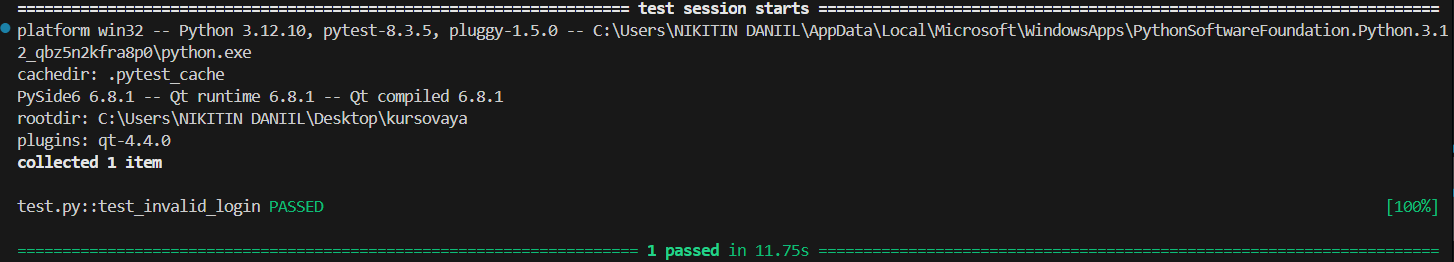


Рисунок 42 – Результат модульного тестирования

Приложение А – Спецификация требований

Спецификация требований для приложения «Привет, я букет!».

Системные характеристики. Системные характеристики определяют технические и аппаратно-программные требования, необходимые для работы приложения «Привет, я букет!».

Операционные системы:

Приложение должно быть совместимо с операционными системами Windows 10/11, macOS 12+ и Linux (Ubuntu 20.04+). Требуется поддержка 64-битных архитектур.

Аппаратные требования:

Минимальные требования:

* Процессор: 1 ГГц или выше.
* Оперативная память: 2 ГБ.
* Свободное место на диске: 500 МБ для установки приложения и зависимостей.

Рекомендуемые требования:

* Процессор: 2 ГГц или выше.
* Оперативная память: 4 ГБ и более.
* Свободное место на диске: 1 ГБ.

Программные зависимости:

* Язык программирования: Python версии 3.9 или выше.
* PySide6 для создания графического интерфейса.
* SQLAlchemy для работы с базой данных.
* ReportLab для генерации PDF-отчетов.
* psycopg2 для взаимодействия с PostgreSQL.
* СУБД: PostgreSQL версии 12 или выше.
* Шрифт TimesNewRoman.ttf для корректного отображения кириллицы в PDF-отчетах.

Сетевые требования:

Приложение работает в однопользовательском режиме и не требует подключения к интернету. Требуется локальное подключение к базе данных PostgreSQL (по умолчанию: postgresql://postgres:1@localhost:5433/postgres).

Производительность:

* Время загрузки списка данных (до 1000 записей) в любой таблице (поставщики, товары, сотрудники, отчеты) не должно превышать 2 секунд.
* Генерация PDF-отчета для 1000 записей должна занимать не более 5 секунд.

Пользовательские требования:

Пользовательские требования описывают функциональные возможности, которые приложение должно предоставлять конечным пользователям (директору и оператору-флористу).

Авторизация:

* Пользователь должен иметь возможность авторизоваться в системе, используя логин и пароль.
* Поддерживаемые роли: «Директор» (логин: direktor) и «Оператор-флорист» (логин: florist).
* При успешной авторизации пользователь переходит в главное окно приложения, где отображается его роль в правом верхнем углу.

Управление поставщиками (доступно только для роли «Директор»):

* Пользователь должен иметь возможность добавлять, удалять и редактировать поставщиков.
* Пользователь должен иметь возможность просматривать список поставщиков в виде таблицы с колонками: «Номер», «Название», «Тип», «Телефон», «Email».

Управление товарами (доступно только для роли «Директор»):

* Пользователь должен иметь возможность добавлять, удалять и редактировать товары.
* Пользователь должен иметь возможность просматривать список товаров в виде таблицы с колонками: «Номер», «Тип», «Номер товара», «Описание», «Цена», «Поставщик».
* При добавлении или редактировании товара пользователь должен иметь возможность загрузить изображение.

Управление сотрудниками (доступно только для роли «Директор»):

* Пользователь должен иметь возможность добавлять, удалять и редактировать сотрудников.
* Пользователь должен иметь возможность просматривать список сотрудников в виде таблицы с колонками: «Номер», «Фамилия», «Имя», «Отчество», «Пол».

Управление отчетами (доступно для обеих ролей):

* Пользователь должен иметь возможность просматривать три типа отчетов:
* Отчет по поступлению товаров (колонки: «Номер», «Дата поступления», «Количество», «Товар»).
* Отчет по остаткам товаров (колонки: «Номер», «Количество товара», «Товар», «Срок годности»).
* Отчет по убытию товаров (колонки: «Номер», «Дата убытия», «Количество убытого товара», «Товар»).
* Пользователь должен иметь возможность выбрать тип отчета через выпадающее меню на вкладке «Отчеты».
* Пользователь должен иметь возможность сгенерировать PDF-отчет для текущего типа отчета и сохранить его в файл с именем {тип\_отчета}\_report\_{временная\_метка}.pdf.

Управление учетной записью (доступно для обеих ролей):

* Пользователь должен видеть свою роль в правом верхнем углу окна («Директор» или «Оператор-флорист»).
* При нажатии на роль должно отображаться выпадающее меню с двумя пунктами: «Выход из учетной записи» и «Выход из программы».
* «Выход из учетной записи» возвращает пользователя в окно авторизации.
* «Выход из программы» показывает диалоговое окно подтверждения, и при согласии закрывает приложение.

Интерфейс пользователя:

* Интерфейс должен быть интуитивно понятным и удобным для пользователей с базовыми навыками работы на компьютере.
* Все элементы управления (кнопки, поля ввода, таблицы) должны быть четко обозначены.
* Текст в интерфейсе, включая выпадающие списки, должен быть темно-коричневым (#5C4033) для лучшей читаемости.
* Фон приложения – светло-бежевый (#F5F5DC).
* Кнопки и метка учетной записи – темно-коричневые (#5C4033) с белым текстом, при наведении – более светлый коричневый (#7A5A47), при нажатии – более темный коричневый (#4A3228).
* Вкладки – мягкий коричневый (#D2B48C), при выборе – легкий кремовый (#FFF5E6).

Навигация:

* Пользователь должен иметь возможность переключаться между вкладками: «Поставщики», «Товары», «Сотрудники», «Отчеты».
* Для перехода к окну авторизации используется пункт «Выход из учетной записи».

Бизнес-правила. Бизнес-правила определяют логику работы приложения и ограничения, которые должны быть соблюдены для обеспечения корректного функционирования системы.

Авторизация:

* Логин и пароль должны быть предопределены в коде приложения:
* Логин: direktor, пароль: 123456 (роль «Директор»).
* Логин: florist, пароль: 123456 (роль «Оператор-флорист»).
* Пароли хранятся в хэшированном виде с использованием алгоритма SHA-256.

Управление поставщиками:

* Название поставщика должно быть уникальным в пределах системы.
* Тип поставщика выбирается из предопределенного списка: «ООО», «АО», «ОАО», «ИП», «ЗАО».
* При удалении поставщика все связанные товары должны быть удалены.

Управление товарами:

* Тип товара выбирается из предопределенного списка: «Букет», «Цветок», «Композиция», «На свадьбу», «Комнатные растения», «Флорариумы», «Подарки», «Популярное».
* Номер товара должен быть уникальным в пределах системы.
* Каждый товар должен быть связан с одним поставщиком.
* При добавлении товара пользователь должен указать цену (число с плавающей точкой, больше 0).

Управление сотрудниками:

* Фамилия и имя сотрудника – обязательные поля.
* Пол сотрудника выбирается из предопределенного списка: «Мужской», «Женский».

Управление отчетами:

* PDF-отчет должен включать данные, соответствующие текущему типу отчета.
* Текст в отчете должен быть черным, шрифт – TimesNewRoman с поддержкой кириллицы.
* Заголовки таблиц в отчете должны иметь серый фон (colors.grey).

Целостность данных:

* Все операции с базой данных должны быть транзакционными: если операция не завершена, изменения должны быть откатаны.
* Приложение должно корректно обрабатывать ошибки подключения к базе данных и отображать пользователю понятное сообщение об ошибке.

Детальные спецификации. Детальные спецификации описывают конкретные функциональные и нефункциональные требования к приложению, включая описание интерфейса, поведения системы и ограничений.

Авторизация.

Вход в систему:

* Пользователь вводит логин и пароль в окне авторизации.
* Требуемые поля:
* Логин (строка, обязательное поле).
* Пароль (строка, обязательное поле).
* После успешной авторизации открывается главное окно приложения.
* При неверных данных отображается сообщение об ошибке: «Неверный логин или пароль!»

Управление поставщиками. Добавление поставщика:

* Пользователь нажимает кнопку «Добавить поставщика», открывается диалоговое окно.

Требуемые поля:

* Название поставщика (строка, обязательное поле, максимальная длина – 255 символов).
* Тип поставщика (выпадающий список: «ООО», «АО», «ОАО», «ИП», «ЗАО»).
* Контактный телефон (строка, необязательное поле).
* Email (строка, необязательное поле).
* Юридический адрес (город, улица, дом).
* После добавления поставщик отображается в таблице поставщиков.

Удаление поставщика:

* Пользователь выбирает поставщика в таблице и нажимает кнопку «Удалить поставщика».
* Отображается диалоговое окно подтверждения: «Вы уверены, что хотите удалить поставщика?».
* При удалении поставщика все связанные товары удаляются.

Редактирование поставщика:

* Пользователь выбирает поставщика в таблице и нажимает кнопку «Редактировать поставщика».
* Открывается диалоговое окно с заполненными текущими данными поставщика.
* После изменения данные сохраняются в базе данных.

Управление товарами. Добавление товара:

* Пользователь нажимает кнопку «Добавить товар», открывается диалоговое окно.

Требуемые поля:

* Тип товара (выпадающий список: «Букет», «Цветок», «Композиция», «На свадьбу», «Комнатные растения», «Флорариумы», «Подарки», «Популярное»).
* Номер товара (число, от 1 до 999999, уникальное).
* Описание (строка, необязательное поле).
* Цена (число с плавающей точкой, больше 0).
* Поставщик (выпадающий список с доступными поставщиками).
* Изображение (поле для загрузки файла, необязательное).
* После добавления товар отображается в таблице товаров.

Удаление товара:

* Пользователь выбирает товар в таблице и нажимает кнопку «Удалить товар».
* Отображается диалоговое окно подтверждения: «Вы уверены, что хотите удалить товар?»

Редактирование товара:

* Пользователь выбирает товар в таблице и нажимает кнопку «Редактировать товар».
* Открывается диалоговое окно с заполненными текущими данными товара.
* После изменения данные сохраняются в базе данных.

Управление сотрудниками. Добавление сотрудника:

* Пользователь нажимает кнопку «Добавить сотрудника», открывается диалоговое окно.

Требуемые поля:

* Фамилия (строка, обязательное поле, максимальная длина – 255 символов).
* Имя (строка, обязательное поле, максимальная длина – 255 символов).
* Отчество (строка, необязательное поле).
* Пол (выпадающий список: «Мужской», «Женский»).
* После добавления сотрудник отображается в таблице сотрудников.

Удаление сотрудника:

* Пользователь выбирает сотрудника в таблице и нажимает кнопку «Удалить сотрудника».
* Отображается диалоговое окно подтверждения: «Вы уверены, что хотите удалить сотрудника?»

Редактирование сотрудника:

* Пользователь выбирает сотрудника в таблице и нажимает кнопку «Редактировать сотрудника».
* Открывается диалоговое окно с заполненными текущими данными сотрудника.
* После изменения данные сохраняются в базе данных.

Управление отчетами:

Выбор типа отчета:

Пользователь нажимает на вкладку «Отчеты», открывается выпадающее меню с типами отчетов:

* «Отчет по поступлению товаров».
* «Отчет по остаткам товаров».
* «Отчет по убытию товаров».
* После выбора типа отчета таблица обновляется с соответствующими данными.

Создание PDF-отчета:

* Пользователь нажимает кнопку «Сформировать отчет».
* Отчет сохраняется в файл {тип\_отчета}\_report\_{временная\_метка}.pdf в текущей директории.

Содержание отчета:

* Для отчета по поступлению товаров: «Номер», «Дата поступления», «Количество», «Товар».
* Для отчета по остаткам товаров: «Номер», «Количество товара», «Товар».
* Для отчета по убытию товаров: «Номер», «Дата убытия», «Количество убытого товара», «Дата отчета», «Товар».
* Текст в отчете: черный, шрифт – TimesNewRoman с поддержкой кириллицы.
* Заголовки таблиц: серый фон (colors.grey).

Управление учетной записью. Отображение роли:

* В правом верхнем углу окна отображается метка с текущей ролью: «Директор» или «Оператор-флорист».

Выход из учетной записи:

* Пользователь нажимает на метку роли, открывается выпадающее меню.
* Пользователь выбирает «Выход из учетной записи».
* Текущее окно закрывается, открывается окно авторизации.

Выход из программы:

* Пользователь нажимает на метку роли, открывается выпадающее меню.
* Пользователь выбирает «Выход из программы».
* Отображается диалоговое окно подтверждения: «Вы уверены, что хотите выйти из программы?»
* При подтверждении приложение закрывается.

Нефункциональные требования:

Производительность:

* Время загрузки данных в таблицы: не более 2 секунд для 1000 записей.
* Время генерации PDF-отчета: не более 5 секунд для 1000 записей.

Надежность:

* Приложение должно корректно обрабатывать ошибки подключения к базе данных.
* Все операции с базой данных должны быть транзакционными.

Удобство использования:

* Интерфейс должен быть интуитивно понятным.
* Текст в интерфейсе и выпадающих списках – темно-коричневый (#5C4033).
* Фон приложения – светло-бежевый (#F5F5DC).
* Кнопки и метка учетной записи – темно-коричневые (#5C4033), при наведении – более светлый коричневый (#7A5A47), при нажатии – более темный коричневый (#4A3228).

Локализация:

* Приложение поддерживает русский язык.
* PDF-отчеты корректно отображают кириллицу.

Приложение Б – Ссылка на систему контроля версий

<https://github.com/nikadobrovskaa807/PP>

Приложение В – ER-диаграмма

На рисунке В1 представлена ER-диаграмма.

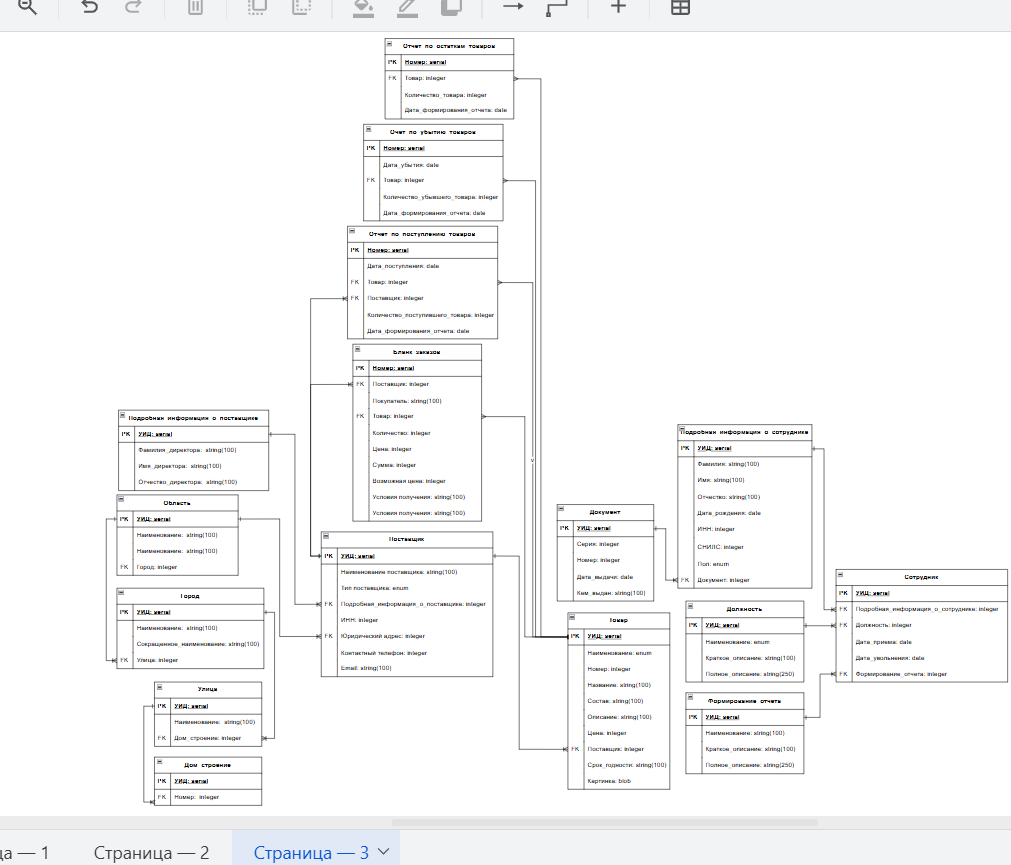


Рисунок В1 – ER-диаграмма