

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Вычислительной техники  
(полное название кафедры)

Утверждаю

Зав. кафедрой Якименко А.А.

\_\_\_\_\_  
(подпись, инициалы, фамилия)

«\_\_» \_\_\_\_\_ 2024 г.

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА**

Великжанин Сергей Игоревич

(фамилия, имя, отчество студента – автора работы)

Приложение для учета инвентаря бара в отеле “Grand Autograph”

(тема работы)

Факультет автоматики и вычислительной техники

(полное название факультета)

Направление подготовки 09.03.01 Информатика и вычислительная техника

(код и наименование направления подготовки бакалавра)

**Руководитель  
от НГТУ**

Борин Владислав Михайлович

(фамилия, имя, отчество)

К.ф.-м.н.

(ученая степень, ученое звание)

\_\_\_\_\_  
(подпись, дата)

**Автор выпускной  
квалификационной работы**

Великжанин Сергей Игоревич

(фамилия, имя, отчество)

АВТФ, АВТ-010

(факультет, группа)

\_\_\_\_\_  
(подпись, дата)

Новосибирск 2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Вычислительной техники  
(полное название кафедры)

УТВЕРЖДАЮ

Зав. кафедрой Якименко А.А.  
(фамилия, имя, отчество)

\_\_\_\_\_  
(подпись, дата)

**ЗАДАНИЕ  
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА**

студенту Великжанину Сергею Игоревичу  
(фамилия, имя, отчество)

Направление подготовки 09.03.01 Информатика и вычислительная техника  
(код и наименование направления подготовки бакалавра)

Факультет автоматики и вычислительной техники  
(полное название факультета)

Тема Приложение для учета инвентаря бара в отеле "Grand Autograph"  
(полное название темы выпускной квалификационной работы бакалавра)

Исходные данные (или цель работы) Разработать приложение для учета инвентаря бара с учетом особенностей работы отеля "Grand Autograph" с возможностью составления и редактирования заявок, управления персоналом и составлением статистики покупок и продаж.

Структурные части работы \_\_\_\_\_

Введение

1. Изучение рабочих процессов в баре отеля "Grand Autograph"

2. Проектирование архитектуры приложения

3. Анализ программного стека

4. Реализация приложения

4.1 Создание базы данных

4.2 Настройка серверного окружения (бэк-энд)

4.3 Разработка пользовательского интерфейса (фронт-энд)

4.4 Настройка взаимодействия компонентов приложения

5. Тестирование

Заключение

Список источников

Приложение

Задание согласовано и принято к исполнению.

**Руководитель  
от НГТУ**

Борин Владислав Михайлович

(фамилия, имя, отчество)

К.ф.-м.н.

(ученая степень, ученое звание)

(подпись, дата)

**Студент**

Великжанин Сергей Игоревич

(фамилия, имя, отчество)

АВТФ, АВТ-010

(факультет, группа)

(подпись, дата)

Тема утверждена приказом по НГТУ № \_\_\_\_\_ от «\_\_\_» \_\_\_\_\_ 202\_\_ г.

изменена приказом по НГТУ № \_\_\_\_\_ от «\_\_\_» \_\_\_\_\_ 202\_\_ г.

ВКР сдана в ГЭК № \_\_\_\_\_, тема сверена с данными приказа

(подпись секретаря государственной экзаменационной комиссии по защите ВКР, дата)

(фамилия, имя, отчество секретаря государственной  
экзаменационной комиссии по защите ВКР)

## **Аннотация**

Выпускная квалификационная работа бакалавра выполнена студентом Великжаниным С.И.

Руководитель проекта: к.ф.-м.н. Борин В.М.

Тема: “Приложение для учета инвентаря бара в отеле “Grand Autograph”.

Пояснительная записка содержит листов – 63, рисунков – 58.

Ключевые слова: WEB, приложение, MVT, Django, Python, SQLite, Bootstrap, сервер.

Цель выпускной квалификационной работы – разработка приложения для учета инвентаря бара в отеле “Grand Autograph”. Приложение решает проблему работников бара в управлении логистикой и администрированием рабочих процессов. В частности, отслеживание наличия продукции в баре, составление заявок на пополнение инвентаря, ведение учета продаж и координирование персонала.

Достигнуты поставленные цели:

- Изучены рабочие процессы в баре отеля “Grand Autograph”;
- Разработана структура базы данных;
- Разработана клиентская часть приложения;
- Разработана серверная часть приложения;
- Протестирована работа приложения.

## Содержание

Введение.....	7
1 Изучение рабочих процессов в баре отеля “Grand Autograph” .....	8
1.1 Цели и задачи ВКР .....	8
1.2 Описание рабочих процессов.....	9
1.3 Выявленные недостатки .....	10
2 Проектирование архитектуры приложения.....	11
2.1 Описание приложения .....	11
2.2 Архитектура приложения.....	12
3 Анализ программного стека.....	14
3.1 Язык гипертекстовой разметки HTML .....	15
3.2 Каскадные таблицы стилей CSS .....	16
3.3 Язык программирования JavaScript.....	17
3.4 Фреймворк Bootstrap.....	17
3.5 Язык программирования Python .....	18
3.6 Фреймворк Django.....	19
3.7 Система управления базами данных SQLite .....	20
3.8 Среда разработки PyCharm .....	21
4 Реализация приложения .....	23
4.1 Создание базы данных.....	24
4.1.1 Схема базы данных .....	24
4.1.2 Работа с базой данных .....	28
4.2 Настройка серверного окружения (бэк-энд) .....	29
4.2.1 Параметры приложения.....	29
4.2.2 Обработка запросов .....	30
4.2.3 Панель администратора.....	32
4.3 Разработка пользовательского интерфейса (фронт-энд).....	33
4.3.1 Формирование HTML-страниц.....	33
4.3.2 Взаимодействие с сервером через HTTP-запросы.....	35
4.4 Настройка взаимодействия компонентов приложения .....	37
4.4.1 Файловая структура компонентов.....	37
4.4.2 Руководство пользователя – панель сотрудника .....	39
4.4.3 Руководство пользователя – панель администратора .....	50
5 Тестирование .....	55

5.1 Тестирование приложения в различных браузерах.....	55
Заключение .....	60
Список источников .....	61
Приложение .....	63

## **Введение**

В сфере обслуживания и общественного питания главное – это качество обслуживания гостя. Для обеспечения наивысшего качества предоставления услуг большинство заведений внедряют информационные системы для отслеживания мест посадки гостя и их заказов, а также для упрощения процесса оплаты и возможности реализации программ лояльности. Однако не менее важную часть играет эффективность работы самих сотрудников заведения.

Веб-приложение для учета инвентаря бара позволит перевести большую часть обязанностей персонала в электронный вид и повысит эффективность работников. Замена процесса ежемесячной инвентаризации на отслеживание продукции в реальном времени оптимизирует рабочее время бармена на смене, в то время как бар-менеджер получит возможность более эффективно вести финансовые операции и управлять персоналом.

Актуальность выпускной квалификационной работы заключается в возникшей необходимости улучшения рабочих условий персонала в ходе работы барменом в отеле “Grand Autograph”. В процессе выполнения выпускной квалификационной работы будет создано веб-приложения для учета инвентаря бара и управления персоналом.

# **1 Изучение рабочих процессов в баре отеля “Grand Autograph”**

## **1.1 Цели и задачи ВКР**

Целью выпускной квалификационной работы является разработка веб-приложения для учета инвентаря бара для сотрудников отеля “Grand Autograph”. Приложение решает проблему неэффективности рабочих процессов, связанных с инвентаризацией. В частности, дает возможность отслеживать, управлять и редактировать информацию о продукции, находящейся на баре в реальном времени, управлять и отслеживать статус сотрудников, а также просматривать статистику доставок и продаж.

В приложении реализованы панель администратора и панель сотрудника. Пользователями приложения являются бармены, бар-менеджер, менеджер отдела “Еда и Напитки” отеля, бухгалтер, системный администратор. Доступ к функционалу приложения предоставляется в зависимости от уровня прав, выданных сотруднику.

Панель сотрудника доступна всем пользователям, зарегистрированным в системе. В панели находится информация о сотруднике, инвентаре бара, списке продаж. Функционал панели сотрудника отличается в зависимости от уровня прав.

Панель администратора доступна пользователям с ролью системного администратора. С помощью панели можно управлять записями о продукции, продажах, заявках на склад и списком персонала, а также управлять данными о пользователях приложения.

Этапы разработки веб-приложения можно выделить в следующие задачи:

- Разработка архитектуры базы данных;
- Разработка серверной части приложения;
- Разработка клиентской части приложения;
- Тестирование работоспособности, поиск и исправление ошибок.



## 1.2 Описание рабочих процессов

“Grand Autograph” – пятизвездочный отель, находящийся по адресу г. Новосибирск, ул. Орджоникидзе, 31. На территории отеля находятся два ресторана, в каждом из которых есть бар. С точки зрения клиента заведения не связаны между собой, однако в обоих барах работает одна и та же команда сотрудников.

В обязанности рядового бармена входит:

- Обслуживание гостей за барной стойкой;
- Приготовление напитков согласно заказу;
- Учет израсходованной продукции;
- Актуализация данных об использовании алкогольной продукции с использованием системы ЕГАИС (Единая государственная автоматизированная информационная система учета объема производства и оборота этилового спирта, алкогольной и спиртосодержащей продукции);
- Проведение ежемесячной инвентаризации для координирования данных бухгалтерии и фактических результатов работы бара.

Отдельно стоит выделить процесс проведения пополнения бара. Один из баров отеля работает круглосуточно, поэтому на бармена ночной смены ложится дополнительная обязанность по составлению заявок для пополнения бара.

Данный процесс можно разделить на несколько этапов:

- 1) Бармен ночной смены заполняет бланк заявки на склад и передает его бар-менеджеру в начале дневной смены;
- 2) Бар-менеджер проверяет соответствие позиций заявки и меню, затем передает заявку складской службе отеля;
- 3) Сотрудники складской службы обрабатывают заявку и выгружают запрошенную продукцию;
- 4) Один из двух барменов дневной смены перевозит продукцию со склада в бар в течении рабочего дня.

### **1.3 Выявленные недостатки**

Высокие стандарты обслуживания клиентов в баре отеля “Grand Autograph” влекут за собой проблемы с рабочими условиями, среди которых можно выделить:

- Большое количество клиентов сказывается на качестве учета в баре – бармены не всегда передают полную информацию об израсходованной продукции;
- Составление заявки происходит один раз на бумажном бланке – бармену необходимо запоминать все позиции меню, по которым возник недостаток в течении смены;
- В течении рабочего месяца мелкие ошибки учета накапливаются, что становится большой проблемой при проведении инвентаризации – сильная разница в данных между отчетом бар-менеджера и бухгалтерии приводит к вычету из зарплаты барменов.

Перечисленные недостатки сказываются на физическом здоровье работников и их мотивации, что приводит высоким уровням стресса, напряженности отношений внутри коллектива и высокой текучести кадров.

## **2 Проектирование архитектуры приложения**

В данном разделе описаны функционал приложения и выбранная архитектура для реализации приложения.

### **2.1 Описание приложения**

Приложение разработано для учета инвентаря бара и управления персоналом. В функционал приложения входит составление и редактирование заявок, отслеживание и изменение статуса персонала, статистика доставок и продаж.

Приложение состоит из двух частей: панель администратора и панель сотрудника. Функционал может отличаться в соответствии с уровнем прав, который соответствует роли авторизованного пользователя.

Панель сотрудника доступна всем авторизованным пользователям. В функционал панели входит:

- Просмотр статистики по ожидаемым доставкам, продажам, последних пользователей сайта;
- Операции добавления, редактирования, удаления записи из списка продукции бара, списка продаж, списка заявок на склад, списка доставок, списка персонала, выгрузка отдельных списков в таблицу Excel;
- Создание и редактирование профиля сотрудника.

Панель администратора доступна пользователям с уровнем доступа системного администратора. В функционал панели входит:

- Весь функционал панели сотрудника;
- Добавление, редактирование, удаление записей списка поставщиков;
- Добавление, редактирование, удаление категорий списка продукции бара;
- Управление данными пользователей системы;
- Управление правами доступа сотрудников.

## 2.2 Архитектура приложения

Для реализации приложения использован клиент-серверный архитектурный паттерн MVT (Model – View – Template), который позволяет создать расширяемую информационную систему с большим количеством компонентов. Такой вариант реализации дает возможность разделить приложение на клиентскую и серверную части. Разделение упрощает процесс разработки проекта, потому что изменения в программном коде интерфейса пользователя не касаются программного кода логики приложения, и наоборот – изменения в логике никак не влияют на внешний вид.

Данные приложения и пользователей хранятся на сервере в базе данных в защищенном виде, что повышает уровень безопасности приложения. На рисунке 2.1 приведена схема работы архитектурного паттерна MVT.

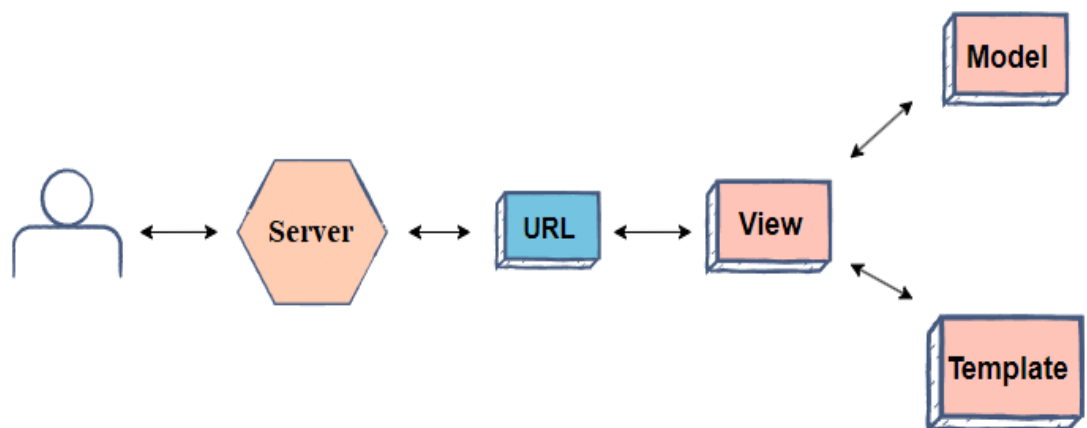


Рисунок 2.1 – Схема работы архитектурного паттерна MVT.

Формат MVT разделяет данные, представление данных и обработку запросов пользователя на следующие части:

Модель (Model) – объектная модель, которая отвечает за данные внутри приложения. Она определяет структуру данных и методы для добавления, редактирования, чтения и удаления записей в базе данных. В модели не содержится информация о визуальном представлении данных, пользователь не взаимодействует с ней напрямую;

Представление (View) – функция, которая принимает запросы пользователя, обрабатывает данные и возвращает ответ пользователю. Она запрашивает

необходимые данные из Модели и отображает результат в интерфейсе используя Шаблоны;

Шаблон (Template) – текстовый файл, который отвечает за структуру и расположение элементов интерфейса. Шаблон определяет, как данные, полученные от Представлений, будут отображаться пользователю [1].

Работа архитектурного паттерна MVT в общем виде сводится к пяти шагам:

- 1) Запрос пользователя поступает на сервер и интерпретируется URL-маршрутизатором;
- 2) Маршрутизатор направляет запрос к соответствующему представлению;
- 3) Представление обрабатывает запрос, взаимодействует с моделью для получения запрошенных данных или изменения записи в базе данных;
- 4) После обработки данных представление выбирает шаблон для генерации веб-страницы и передает в шаблон нужные данные;
- 5) Шаблон обрабатывается в полноценную веб-страницу, которая отображается пользователю в качестве ответа приложения.

### 3 Анализ программного стека

В данном разделе описаны программные средства, которые были выбраны для реализации приложения.

Для проектирования приложения взята клиент-серверная архитектура, которая подразумевает разделение на клиентскую и серверную части. Такой вариант реализации означает, что вся логика приложения находится на сервере, а пользователь взаимодействует только с интерфейсом клиентской части.

Интерфейс клиентской части приложения реализован с помощью веб-технологий, что облегчает процесс разработки и дает возможность разворачивать приложение как на локальном сервере, так и на удаленном, предоставляя доступ из любой точки мира через сеть Интернет. Для структурирования и представления содержимого интерфейса выбран язык разметки гипертекста HTML и каскадные таблицы стилей CSS. Данные технологии поддерживаются большинством современных браузеров, поэтому пользователю не придется устанавливать сторонние приложения, кроме самого браузера.

Для взаимодействия с клиентской частью приложения и реализации динамического функционала выбран язык JavaScript, потому что он может работать с содержимым HTML-страницы. Для упрощения процесса разработки был выбран фреймворк Bootstrap, который предоставляет готовые классы и компоненты.

Для серверной части выбран языка программирования Python, который имеет простой синтаксис и большой выбор библиотек для веб-разработки. Одна из основных библиотек приложения – высокоуровневый фреймворк Django, который включает в себя настройку веб-сервера, механизмы для авторизации пользователей, административный интерфейс. Для хранения данных приложения используется БД SQLite, потому что она имеет реляционную структуру хранения данных, встраивается напрямую в приложение и использует минимальные ресурсы системы.

В качестве среды разработки выбрана PyCharm, которая имеет глубокую интеграцию с фреймворком Django в виде инструментов анализа, автодополнения и рефакторинга программного кода в реальном времени, что позволяет предотвратить ошибки и повышает скорость написания кода.

### **3.1 Язык гипертекстовой разметки HTML**

HTML (Hypertext Markup Language) – это язык, используемый для структурирования и представления содержимого веб-страницы. Так, содержимое может быть организовано в виде блоков текста, маркированных списков, а также с использованием изображений и таблиц с данными. HTML не является языком программирования – используя его невозможно задать саму логику приложения. Язык разметки указывает браузеру, как отображать веб-страницу, которую запрашивает пользователь. Сложность разметки зависит от требований к дизайну приложения.

HTML состоит из различных комбинаций тегов и текста, которые используются для обозначения разных частей содержимого, чтобы задать их отображение или поведение. Теги могут сделать слово или изображение ссылкой на другую страницу, задать курсивное начертание слов, увеличить или уменьшить размер шрифта и так далее.

Преимущества:

- Файлы HTML совместимы с любым современным браузером. Эти файлы являются системно-независимыми и могут отображаться любым компьютером, способным поддерживать HTML;
- Файлы HTML имеют небольшой размер. Расходы системных ресурсов при использовании тегов с большим количеством параметров небольшие;
- Синтаксис HTML лёгок для изучения. В открытом доступе находится большое количество документаций и реальных примеров разметки, что позволяет вести обучение на эффективных решениях. К тому же, существует много редакторов HTML, облегчающих создание веб-страниц [2].

### 3.2 Каскадные таблицы стилей CSS

CSS (Cascading Style Sheets) – это язык, используемый для оформления web-страниц. Он определяет внешний вид и стиль элементов, расположенных на странице, которые уже размечены с помощью HTML. CSS отвечает за такие параметры, как размер, цвет, фоновое изображение, прозрачность, положение элементов относительно других элементов, реакция на наведение указателя мыши, визуальное изменение кнопок при нажатии и т.д.

Язык CSS является стандартом в веб-разработке, поскольку он позволяет легко изменять визуальное оформление веб-страницы без необходимости использования сложных языков программирования.

Преимущества:

- Простота синтаксиса языка, которая позволяет быстро изучить его и сократить время разработки благодаря разделению разметки веб страницы и её внешнего вида;
- Снижение времени загрузки страницы за счет выноса правил стилей в отдельный CSS-файл. Браузер загружает только разметку страницы в HTML и данные, хранящиеся на странице. Представление этих данных загружается один раз для всех страниц и может быть получено из кэша браузера при переходе между страницами. Это ускоряет загрузку страницы для пользователей и снижает нагрузку на сервер;
- Легкость изменения дизайна. Изменения в дизайн приложения можно вносить, не затрагивая каждую страницу. Достаточно изменить значения свойств CSS в соответствующем файле [3].



### 3.3 Язык программирования JavaScript

JavaScript (сокращенно JS) - язык программирования высокого уровня, который может быть встроен в сторонние приложения. Все функциональные модули создаются в виде сценариев (скриптов). По синтаксису он схож с языком Java, но имеет только частичное сходство в названии. JavaScript регулярно обновляется, и его код становится все более оптимизированным.

Использование JS-скриптов обусловлено необходимостью добавления функциональности, отсутствующей в HTML/CSS. С помощью скриптов можно добавить в веб-приложение обработку данных, поступающих от пользователя, анимацию интерфейса.

JavaScript применяется в клиентской части веб-приложений. Это интерфейс страницы, который включает контент, кнопки, формы обратной связи, меню. С помощью JavaScript интерфейс реагирует на действия пользователя (нажатия кнопок мыши, клавиш), а также отвечает за сохранение данных и автоматическое заполнение форм [4].

### 3.4 Фреймворк Bootstrap

Bootstrap – фреймворк, содержащий в себе богатый выбор HTML, CSS и JavaScript компонентов. Фреймворк очень просто внедряется в веб-приложение – достаточно указать ссылку на его компоненты в файлах разметки страниц. Классы Bootstrap позволяют разработчику быстро и без повторения кода создавать сайты и веб-приложения с современным и эффективным дизайном. Популярность фреймворка привела к формированию большого сообщества веб-разработчиков, которые создают библиотеки для интеграции Bootstrap с другими популярными фреймворками.

Классы Bootstrap можно разделяют на 3 общие группы:

- Классы для создания адаптивного макета страницы;
- Классы для стилизации текста, кода, изображений, таблиц и другой информации;

- Служебные классы, используемые для решения наиболее часто встречающихся вспомогательных задач, таких как выравнивание, управление отображением, разбиение на равномерные столбцы и др..

Использование Bootstrap при создании веб-приложений дает много преимуществ для разработчика:

- Быстрая верстка адаптирующегося дизайна веб-страниц (достигается благодаря использованию хорошо продуманных и протестированных большим количеством веб-разработчиков классов и готовых компонентов);

- Современный дизайн. Оформление HTML элементов и компонентов Bootstrap выполнено в едином стиле в последних тенденциях веб-дизайна, что подтверждается статистикой – около 20% сайтов сети Интернет используют этот фреймворк [5];

- Является кроссбраузерным и кроссплатформенным (адаптирован для всех популярных операционных систем и браузеров (Mozilla Firefox, Google Chrome, Safari, Internet Explorer и Opera и др.), работающих в этих системах);

- Является открытым и бесплатным. Фреймворк Bootstrap – это проект с открытым исходным кодом, который распространяется по лицензии, разрешающей использовать фреймворк в разработке как частных проектов, так и коммерческих;

- Богатая документация, переведенная на многие языки мира, упрощает изучение фреймворка [6].

### **3.5 Язык программирования Python**

Python – это высокоуровневый язык программирования, который отличается понятным синтаксисом и большой библиотекой сторонних модулей и инструментов, подходящих для веб-разработки. Он имеет структуры данных высокого уровня и простой, но эффективный подход к объектно-ориентированному программированию.

Преимущества языка:

- Python использует понятный синтаксис, что помогает на всех этапах использования языка – как в начале обучения, так и при написании больших веб-приложений;
- Разработчики со всего мира взаимодействуют с сообществом и создают библиотеки и модули для широкого круга задач, такие как веб-разработка, обучение нейросетей, работа с базами данных и др.;
- Язык является кроссплатформенным. Приложения, созданные с использованием Python можно запускать на всех популярных ОС без многочисленных конфликтов при сборке проекта;
- Python позволяет легко интегрировать код, написанный с использованием других языков, например, C и JavaScript. Это позволяет использовать уже написанные код и библиотеки этих языков, чтобы расширять возможности языка Python [7].

### 3.6 Фреймворк Django

Django — это высокоуровневый веб-фреймворк на Python, с помощью которого можно быстро создавать и эффективно поддерживать безопасные веб-приложения. Django предоставляет богатый набор библиотек, классов, методов и другого функционала, который позволяет разработчику сфокусироваться на уникальной части своего кода. Благодаря бесплатной форме распространения и открытому исходному коду вокруг фреймворка продолжает расти активное сообщество, которое создает подробную документацию, новые компоненты и которое всегда готово помочь новичкам в веб-разработке.

Преимущества фреймворка Django:

- Django предоставляет почти всё, что нужно для разработки веб-приложений, в своей базовой комплектации. Компоненты, являясь частью полноценного фреймворка, безотказно работают друг с другом и следуют одинаковым принципам проектирования;
- Фреймворк может быть использован для создания практически любого типа веб-сайтов и приложений — от систем управления контентом до

социальных сетей. Он может работать с любой клиентской средой и обрабатывать данные практически в любом формате (включая HTML, RSS-каналы, JSON, XML и т. д.);

- В Django “вшиты” методы обеспечения безопасности данных, которые нацелены на предотвращение наиболее частых ошибок в защите данных. В частности, при разработке фреймворка очень серьезно отнеслись к защите паролей пользователей – при регистрации данные передаются в зашифрованном виде, и даже системный администратор не знает, какой пароль используется в учетной записи. Однако при этом реализована автоматизированная форма для смены пароля, тем самым обеспечивая возможность восстановления доступа к веб-приложению;

- Django написан на Python, который является кроссплатформенным. Приложение, созданное с использованием фреймворка на устройстве с ОС Windows можно разместить на сервере с ОС Linux, а затем редактировать код приложения на устройстве с ОС Mac, при этом сохраняя логику приложения, архитектуру и синтаксис программного кода [8].

### **3.7 Система управления базами данных SQLite**

SQLite - это библиотека Python, изначально написанная на языке C, которая обеспечивает легковесную базу данных на диске, не требующую отдельного серверного процесса и позволяющую обращаться к базе данных с помощью языка запросов SQL. Благодаря тому, что фреймворк Django использует базу данных SQLite в качестве БД по умолчанию, приложения, созданные с использованием этого фреймворка, могут использовать SQLite для внутреннего хранения данных [9].

Помимо малого размера и простой интеграции, база данных SQLite имеет и другие преимущества:

- ACID-свойства - набор требований к транзакционной системе, обеспечивающий наиболее надёжную и предсказуемую её работу — атомарность, согласованность, изоляцию, устойчивость;

- SQLite поддерживается на множестве платформ, включая Windows, macOS и Linux;
- Использует минимальные ресурсы системы.

### 3.8 Среда разработки PyCharm

PyCharm – одна из самых популярных интегрированных сред разработки (IDE) для языка Python. PyCharm поддерживает версии языка Python от 2.7 и до самой современной 3.12. В данную среду разработки включен широкий выбор модулей, библиотек и инструментов, которые ускоряют процесс разработки веб-приложений и при этом требуют меньше однотипных действий от разработчика.

PyCharm также можно тонко настраивать и интегрировать с популярными фреймворками и сторонними платформами, в частности среда разработки поддерживает облачную платформу контроля версий GitHub, которая использовалась в процессе разработки веб-приложения.

Преимущества PyCharm:

- PyCharm имеет удобный редактор кода со всеми полезными функциями: подсветкой синтаксиса, автоматическим форматированием, дополнением и отступами. PyCharm позволяет проверять версии интерпретатора языка на совместимость, а также использовать шаблоны кода;
- PyCharm позволяет быстро производить рефакторинг кода – например, можно одновременно изменить название одного класса, используемого в разных файлах приложения. Также возможно проводить поиск кода по всей директории проекта;
- Среда поддерживает большое количество фреймворков для веб-разработки, в частности Django, интеграция с которым позволяет автоматически дополнять программный код, использовать шаблоны, генерировать диаграммы зависимостей моделей;
- Для отладки и тестирования программ в IDE имеется встроенный графический отладчик, который дает возможность проводить модульное тестирование с построчным покрытием кода;

- PyCharm - кросс-платформенная среда разработки: можно использовать на Linux, Windows и Mac OS.

## 4 Реализация приложения

В данном разделе описана программная реализация клиентской и серверной частей приложения.

Использование языка Python и фреймворка Django подразумевает использование методологии объектно-ориентированного программирования, поэтому логика приложения описана в виде классов. Файловая структура разделена на папки static, InventoryMS, accounts, store, transactions (см. рисунок 4.1). В папке static находятся файлы JavaScript, CSS, изображения, шрифты, необходимые для работы приложения. В папке InventoryMS содержатся файлы настроек серверной части приложения. В папках accounts, store, transactions содержатся модели, представления и шаблоны соответствующих компонентов приложения. На рисунке 4.1 приведена файловая структура приложения.

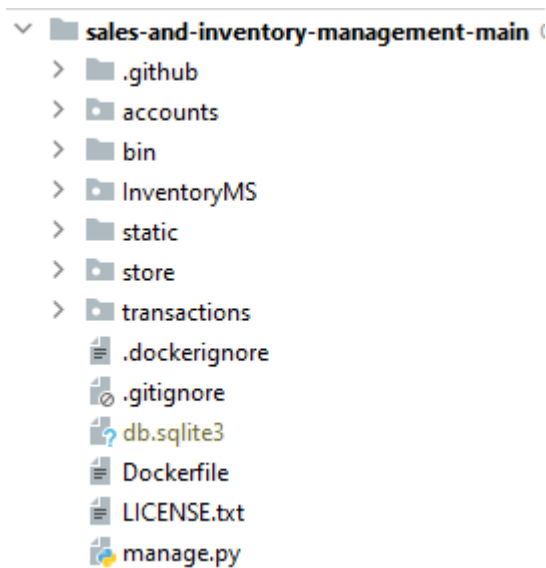


Рисунок 4.1 – Файловая структура приложения

Реализован архитектурный паттерн MVT, в котором описывается логика приложения – взаимодействие с базой данных, обработка запросов пользователя, динамическое формирование веб-страниц, контроль доступа к веб-страницам.

## **4.1 Создание базы данных**

### **4.1.1 Схема базы данных**

Реализованы таблицы `accounts_profile` и `auth_user` для разделения для взаимодействия пользователей с приложением и разделения доступа к частям приложения в зависимости от уровня доступа сотрудника отеля. Для учета продукции в инвентаре бара используется таблица `store_item`, с которой связаны остальные таблицы, отвечающие за данные о поставках, продажах, заявках и поставщиках продукции. База данных состоит из восьми таблиц.

Таблица `accounts_profile` содержит информацию о сотрудниках отеля, которые пользуются приложением: ФИО, номер телефона, электронная почта, статус сотрудника и его роль.

Таблица `auth_user` содержит аккаунты пользователей для входа в приложение, а также дату регистрации в системе. В таблице содержится пароль в зашифрованном виде, код для расшифровки которого хранится на сервере в настройках приложения, получить доступ к этому коду извне невозможно.

Таблица `store_delivery` содержит информацию о доставке продукции со склада в бар, имя сотрудника, ответственного за доставку и дату доставки.

Таблицы `transactions_sale` и `transactions_purchase` содержат информацию о продажах и оформленных заявках на склад соответственно. Таблица `accounts_vendor` содержит информацию о поставщиках продукции, с которыми работает отель.

Таблица `store_category` содержит информацию о видах продукции, доступных в баре. Доступ к этой таблице имеют пользователи с ролью системного администратора.

На рисунке 4.2 приведена структура базы данных.



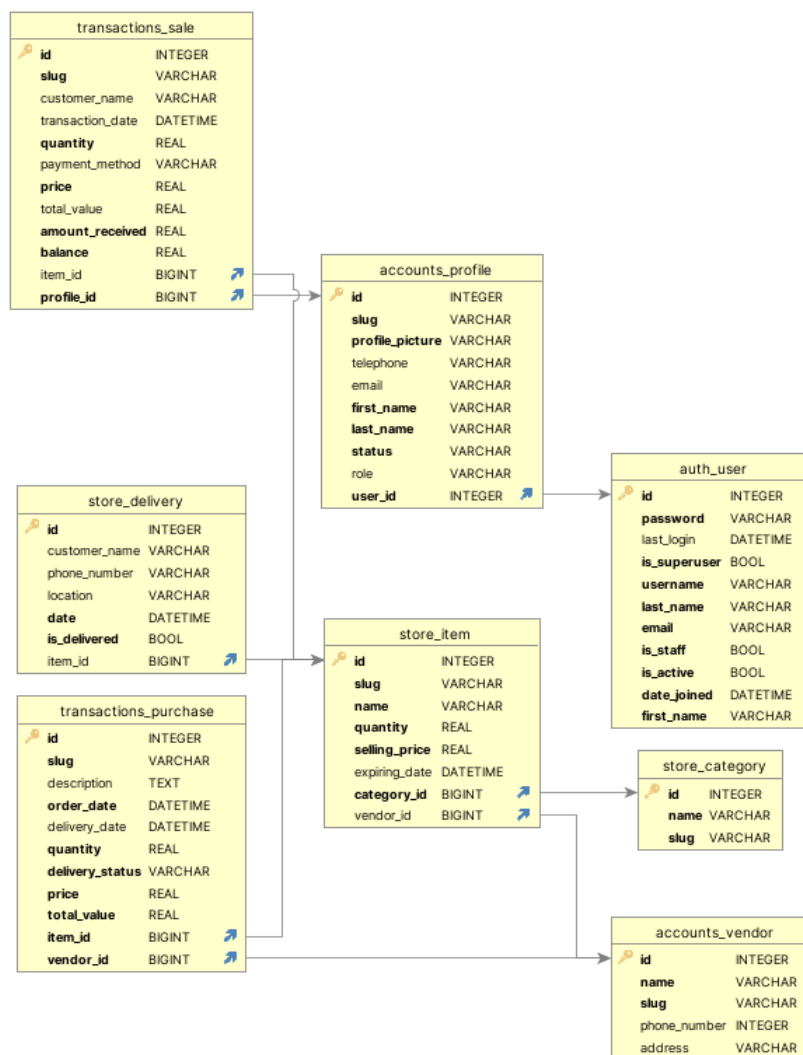


Рисунок 4.2 – Структура базы данных

Таблица store\_item - список инвентаря бара:

- Id – номер записи;
- Slug – уникальный фрагмент URL - адреса, уменьшает размер адреса в браузере и защищает от SQL – инъекций. Описание применимо ко всем таблицам;
- Name – название товара;
- Quantity – количество на баре;
- Selling\_price – розничная цена;
- Expiring\_date – срок годности;
- Category\_id – категория товара (алкоголь, сироп и т.п.);
- Vendor\_id – номер записи о поставщике товара.

Таблица store\_category – справочник видов продукции:

- Id – номер записи;
- Name – вид продукции.

Таблица store\_delivery – список поставок со склада:

- Id – номер записи;
- Customer\_name – имя сотрудника, собравшего заявку на складе;
- Phone\_number – номер телефона сотрудника;
- Location – где находится поставка в данный момент (ждет разгрузки, склад, бросили на полпути);
- Date – дата поставки;
- Is\_delivered – статус доставки;
- Item\_id – ключ товара.

Таблица transactions\_purchase – список заявок на склад:

- Id – номер записи;
- Description – свободное описание заявки;
- Order\_date – дата составления заявки;
- Quantity – количество запрашиваемого товара;
- Delivery\_status – статус доставки;
- Price – цена за штуку;
- Total\_value – общая стоимость;
- Item\_id – ключ товара;
- Vendor\_id – ключ поставщика товара.

Таблица transactions\_sale – список продаж в баре:

- Id – номер записи;
- Customer\_name – имя покупателя;
- Transaction\_date – дата операции;
- Quantity – количество проданного товара;
- Payment\_method – способ оплаты;

- Price – цена за штуку;
- Total\_value – общая стоимость;
- Amount\_recieved – общая розничная стоимость;
- Item\_id – ключ товара;
- Profile\_id – ключ сотрудника, участвовавшего в операции.

Таблица accounts\_profile – список сотрудников:

- Id – номер записи;
- Profile\_picture – аватар сотрудника в системе;
- Telephone – номер телефона;
- Email – электронная почта;
- Status – статус сотрудника (активен, в отпуске);
- Role – роль сотрудника;
- User\_id – ключ пользователя внутри системы.

Таблица auth\_user – список пользователей внутри системы:

- Id – номер записи;
- Password – пароль в зашифрованном виде;
- Last\_login – последний вход в систему;
- Username – имя пользователя;
- Is\_staff – является ли сотрудником;
- Is\_active – активен ли сотрудник;
- Date\_joined – дата регистрации.

Таблица accounts\_vendor – справочник поставщиков:

- Id – номер записи;
- Name – название поставщика;
- Phone\_number – номер телефона;
- Address – адрес поставщика.

### 4.1.2 Работа с базой данных

Согласно архитектуре паттерна Model-View-Template, для взаимодействия с базой данных разработаны модели соответствующих объектов Item, Category, Delivery, Vendor, Sale, Purchase, Profile. Каждая модель является классом языка Python, который наследует класс фреймворка Django `django.db.models.Model` [10]. Каждый атрибут модели соответствует полю базы данных. На рисунке 4.3 приведен пример реализации модели Item, которая представляет запись о продукции в баре.

```
class Item(models.Model):
    slug = AutoSlugField(unique=True, populate_from='name')
    name = models.CharField(max_length=50, blank=False, null=False)
    category = models.ForeignKey(Category, on_delete=models.CASCADE)
    quantity = models.FloatField(default=0.00)
    selling_price = models.FloatField(default=0)
    expiring_date = models.DateTimeField(null=True, blank=True)
    vendor = models.ForeignKey(Vendor, on_delete=models.SET_NULL, null=True)
```

Рисунок 4.3 – Модель Item

Для каждой модели описан метод `__str__`, который возвращает информацию о модели в виде строки. Такая реализация упрощает отображение данных в интерфейсе в формате таблиц. На рисунке 4.4 приведен пример метода `__str__` для модели Item.

```
def __str__(self):
    return f"{self.name} - Category: {self.category}, Quantity: {self.quantity}"
```

Рисунок 4.4 – Метод `__str__` модели Item

Для получения данных от пользователя и внесения изменений в базу данных используются встроенный класс фреймворка Form. Форма сопоставляет данные, полученные от пользователя, с нужной моделью и проверяет правильность введенных данных. Форма наследует атрибуты модели и задает условия проверки. На рисунке 4.5 приведен пример формы ProductForm для модели Item.

```

class ProductForm(forms.ModelForm):
    class Meta:
        model = Item
        fields = '__all__'
        widgets = {
            'expiring_date': forms.DateTimeInput(attrs={
                'class': 'form-control datetimepicker-input',
                'data-target': '#datetimepicker1',
                'placeholder': 'mm/dd/yyyy'
            }),
        }

```

Рисунок 4.5 – Форма ProductForm для модели Item

## 4.2 Настройка серверного окружения (бэк-энд)

### 4.2.1 Параметры приложения

Параметры для работы приложения прописаны в файле settings.py. Для защиты паролей пользователей используется параметр SECRET\_KEY, которым фреймворк Django хэширует пароль, вводимый пользователем при регистрации и повторном входе в приложение, и передает уже зашифрованный пароль в базу данных. На рисунке 4.6 представлен вид хранения паролей в базе данных.

	* id	password	
1	1	pbkdf2_sha256\$6000000\$ca8F05nxEEjhKyS...	2
2	2	pbkdf2_sha256\$6000000\$6NyGB6FMNa1lrAk...	2

Рисунок 4.6 – Вид хранения паролей в базе данных

Для подключения базы данных используется параметр ENGINE, который указывает тип используемой базы данных, и параметр NAME, который указывает путь до самого файла базы данных. На рисунке 4.7 представлены настройки для используемой базы данных SQLite.

```

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}

```

Рисунок 4.7 – Параметры подключения базы данных

Также в параметрах приложения указывается адрес локального сервера, путь до файлов с настройками компонентов приложения, путь до директории, содержащей статические файлы (JS-скрипты, CSS, медиа-файлы, шрифты веб-страниц) и настройки для проверки введенных паролей пользователей при регистрации, в частности проверка на минимальную длину, наличие цифр и специальных символов, соответствие часто используемым и простым паролям, проверка повторно введенного пароля.

#### 4.2.2 Обработка запросов

Для обработки запросов пользователей и взаимодействия с моделями используются представления, которые могут быть определены как функция, так и классы, которые наследуют атрибуты и методы соответствующих моделей [11].

Метод представления `register` отвечает за внесение данных о новом пользователе в базу данных. В метод передается заполненная пользователем форма `CreateUserForm`, при прохождении проверки на правильность введенных данных они сохраняются методом `save`. Далее встроенный метод фреймворка Django отправляет соответствующий SQL-запрос `INSERT`, который вносит новую запись в базу данных [12]. Метод `save` также применяется для внесения изменений в существующую запись. На рисунке 4.8 приведен метод `register`.

```
def register(request):
    if request.method == 'POST':
        form = CreateUserForm(request.POST)
        if form.is_valid():
            user = form.save()
            return redirect('user-login')
    else:
        form = CreateUserForm()
    context = {
        'form': form
    }
    return render(request, template_name='accounts/register.html', context)
```

Рисунок 4.8 – Метод `register`

Класс представления `ProductCreateView` отвечает за внесение записи о новой продукции в базу данных. Атрибут `model` указывает название модели, свойства которой наследуются представлением, `template_name` – название шаблона, используемого для отображения веб-страницы, `form_class` соответствует форме, используемой для получения введенных данных, `success_url` указывает на какую веб-страницу нужно вернуть пользователя после успешного внесения данных. Метод `test_func` проверяет, чтобы после изменения информации о количестве продукции значение не стало отрицательным. На рисунке 4.9 приведен класс представления `ProductCreateView`.

```
class ProductCreateView(LoginRequiredMixin, CreateView):

    model = Item
    template_name = 'store/productcreate.html'
    form_class = ProductForm
    success_url = '/products'

    def test_func(self):
        if self.request.POST.get("quantity") < 0:
            return False
        else:
            return True
```

Рисунок 4.9 – Класс представления `ProductCreateView`

Класс представления `ProductUpdateView` отвечает за внесение изменений в существующую запись базы данных. Атрибут `model` указывает название модели, свойства которой наследуются представлением, `template_name` – название шаблона, используемого для отображения веб-страницы, `fields` соответствует изменяемым полям таблицы базы данных, `success_url` указывает на какую веб-страницу нужно вернуть пользователя после успешного внесения данных. Метод `test_user` проверяет наличие нужного уровня прав пользователя для выполнения изменений. Этот метод прописан во всех классах представления, которые требуют ограничения по уровню доступа. На рисунке 4.10 приведен класс представления `ProductUpdateView`.

```

class ProductUpdateView(LoginRequiredMixin, UserPassesTestMixin, UpdateView):

    model = Item
    template_name = 'store/productupdate.html'
    fields = ['name', 'category', 'quantity', 'selling_price', 'expiring_date', 'vendor']
    success_url = '/products'

    def test_user(self):
        if self.request.user.is_superuser:
            return True
        else:
            return False

```

Рисунок 4.10 – Класс представления ProductUpdateView

### 4.2.3 Панель администратора

Для реализации панели системного администратора используется библиотека фреймворка Django.contrib.admin. Эта библиотека позволяет использовать уже написанный программный код моделей и представлений для автоматической верстки интерфейса [13]. Доступ к панели по умолчанию имеет только пользователь, созданный в терминале среды разработки при первом запуске приложения.

В директории каждого компонента приложения находится файл admin.py, в котором указывается, какие модели нужно наследовать. Для настройки интерфейса панели используются классы-наследники от ModelAdmin. Атрибут list\_display указывает какие поля модели отображать в интерфейсе, search\_fields дает возможность поиска по указанным полям. На рисунке 4.11 приведен фрагмент реализации панели администратора, отвечающий за работу с моделями Profile и Vendor.

```

@admin.register(Profile)
class ProfileAdmin(admin.ModelAdmin):
    list_display = ('user', 'telephone', 'email', 'role', 'status')

@admin.register(Vendor)
class VendorAdmin(admin.ModelAdmin):
    list_display = ('name', 'phone_number', 'address')
    search_fields = ['name', 'phone_number', 'address']

```

Рисунок 4.11 – Фрагмент реализации панели администратора для работы с моделями Profile и Vendor



## 4.3 Разработка пользовательского интерфейса (фронт-энд)

### 4.3.1 Формирование HTML-страниц

HTML-страницы формируются динамически, чтобы избежать повторения кода разметки и облегчить процесс верстки веб-страниц приложения. Архитектура MVT позволяет динамически создавать целую HTML-страницу путем разбиения ее на отдельные шаблоны. В приложении это реализовано через три шаблона – основной шаблон `base.html`, который используется для заднего фона всех веб-страниц, шаблон `sidebar.html`, который отвечает за постоянное отображение меню на левой части интерфейса, и соответствующие шаблоны компонентов приложения, которые используются по одному в зависимости от того, на какую веб-страницу хочет перейти пользователь. В начале каждого шаблона прописаны наследования файлов главного шаблона `base`, файлов стилей и название шаблона, который подставляется в главный. На рисунке 4.12 приведен фрагмент шаблона добавления записи доставки.

```
{% extends "store/base.html" %}
{% load static %}
{% load crispy_forms_tags %}
{% block title %}Add delivery record{%endblock title%}{% block content %}
```

Рисунок 4.12 – Фрагмент шаблона добавления записи доставки

В главном шаблоне `base` используются HTML-элементы фреймворка Bootstrap, в частности элемент `nav`, который позволяет создать адаптирующийся к размеру окна браузера интерфейс [14]. Класс `navbar-expand-md` контролирует масштаб интерфейса, класс `navbar-brand` является ссылкой на домашнюю страницу. На рисунке 4.13 приведен фрагмент шаблона `base`.

```
<section id="wrapper">
  <nav class="navbar navbar-expand-md" style="background-color: #8c0822">
    <div class="container-fluid mx-2">
      <div class="navbar-header">
        <a class="navbar-brand" style="font-size: 20px;" href="{% url 'dashboard' %}">GA Inventory</a>
      </div>
```

Рисунок 4.13 – Фрагмент шаблона `base`

В шаблоне sidebar описана разметка бокового меню. Элемент `aside` и его параметры отвечают за фиксирование меню в левом верхнем углу веб-страницы. Класс `sidebar-header` центрирует фрагменты бокового меню. Элемент `img` запрашивает ссылку на изображение профиля, обрезает его до формы круга, и делает ссылкой на профиль сотрудника. На рисунке 4.14 приведен фрагмент шаблона sidebar.

```
<aside class="sidebar position-fixed top-0 left-0 overflow-auto h-100 float-left" id="show-side-navigation1">
  <i class="fa fa-bars close-aside d-md-none d-lg-none" data-close="show-side-navigation1"></i>
  <div class="sidebar-header d-flex justify-content-center align-items-center px-3 py-4">
    <a href="{% url 'user-profile' %}">
      
    </a>
  </div>
</aside>
```

Рисунок 4.14 – фрагмент шаблона sidebar

Для отображения информации о продукции, доставках, заявках, продажах и персонале в виде таблиц используются методы `render_table` и `querystring` библиотеки `Django_tables2`. Элемент `thead` отвечает за создание заголовка таблицы с названиями столбцов, элемент `tbody` заполняет поля данными в соответствии со строкой, которую возвращает метод соответствующей модели. На рисунке 4.15 приведен пример реализации таблицы для отображения списка продукции в баре.

```
<tbody>
{% for item in items %}
  <tr>
    <th scope="row"><a> {{item.id}}</a></th>
    <td>{{item.name}}</td>
    <td>{{item.category}}</td>
    <td>{{item.quantity}}</td>
    <td>{{item.selling_price}}</td>
    <th scope="col">{{item.expiring_date}}</th>
    <td>{{item.vendor}}</td>
    <td>
```

Рисунок 4.15 – Реализация таблицы для отображения списка продукции в баре

### 4.3.2 Взаимодействие с сервером через HTTP-запросы

Передача данных между клиентской и серверной частями происходит посредством отправки HTTP-запросов.

Для поиска продукции в базе данных используется элемент `form` и класс `input-group`, который создает на странице строку поиска. После нажатия кнопки “Поиск” элемент вызывает метод представления `item_search_list_view`. Для составления запроса используется HTTP-метод `GET`, потому что при поиске продукции пользователь не вносит изменений в базу данных. На рисунке 4.16 приведен пример реализации запроса на поиск продукции в базе данных.

```
<form class="input-group" role="search" id="searchform" action="{% url 'item_search_list_view' %}"
      method="get" accept-charset="utf-8">
  <div class="form-group">
    <div class="input-group">
      <input id="searchbox" name="q" type="text" class="form-control" placeholder="Найти продукцию">
      <span class="input-group-btn">
        <button class="btn btn-outline-success" type="submit">Поиск</i></button>
      </span>
    </div>
  </div>
</form>
```

Рисунок 4.16 – Запрос на поиск продукции в базе данных.

Метод `get_queryset` класса представлений `ItemSearchListView` обрабатывает запрос пользователя и возвращает результат поиска по базе данных. На рисунке 4.17 приведен метод `get_queryset` для поиска продукции в базе данных.

```
def get_queryset(self):
    result = super(ItemSearchListView, self).get_queryset()

    query = self.request.GET.get('q')
    if query:
        query_list = query.split()
        result = result.filter(
            reduce(operator.and_,
                (Q(name__icontains=q) for q in query_list))
        )
    return result
```

Рисунок 4.17 – Метод `get_queryset` для поиска продукции

Для внесения изменений в базу данных (например, создание новой записи продукции) используется элемент `form` и класс `form-group`, который дает пользователю форму для заполнения. После нажатия кнопки “Добавить” элемент

отправляет данные используя HTTP-метод POST. Для повышения уровня безопасности в заполненной форме автоматически создается дополнительное поле со случайно сгенерированным токеном csrf\_token для защиты от подделки межсайтовых запросов [15]. На рисунке 4.18 приведена заполняемая форма для добавления продукции в базу данных.

```
<form method="POST" enctype="multipart/form-data">
  {% csrf_token %}
  <section class="section-bg" id="plans">
    <fieldset class="form-group">
      <header class="section-header">
        <h1 class="text" style="color:black">Добавить продукцию</h1>
        <hr>
      </header>
      <label>Название</label>
      <span class="text-danger">{{form.name.errors }}</span>
      {{form.name}}
      <label> Категория </label>
      <span class="text-danger">{{form.category.errors }}</span>
      {{form.category}}
      <label> Количество </label>
      <span class="text-danger">{{form.quantity.errors }}</span>
      {{form.quantity}}
      <label> Цена продажи</label>
      <span class="text-danger">{{form.selling_price.errors }}</span>
      {{form.selling_price}}
      <label> Срок годности</label>
      <span class="text-danger">{{form.expiring_date.errors }}</span>
      {{form.expiring_date}}
      <label>Поставщик</label>
      {{form.vendor}}
    </fieldset>
  </section>
  <div class="form-group mt-4 text-center">
    <button class="btn btn-success" type="submit">Добавить</button>
```

Рисунок 4.18 – Форма для добавления продукции в базу данных

В приложении для расшифровки токена используются методы родительского класса фреймворка Django, которые наследуются всеми классами представлений. Однако для представлений регистрации и входа в приложение используется метод render с атрибутами request, context для расшифровки [16]. На рисунке 4.19 приведен пример метода render.

```

def register(request):
    if request.method == 'POST':
        form = CreateUserForm(request.POST)
        if form.is_valid():
            user = form.save()
            return redirect('user-login')
    else:
        form = CreateUserForm()
    context = {
        'form': form
    }
    return render(request, template_name: 'accounts/register.html', context)

```

Рисунок 4.19 – Пример метода render

## 4.4 Настройка взаимодействия компонентов приложения

### 4.4.1 Файловая структура компонентов

Файловая структура приложения разделена на три каталога accounts, store, transactions, в которых содержатся файлы с описанием моделей, классов и методов представлений и шаблоны веб-страниц, и отдельный каталог InventoryMS, в котором содержатся файлы настроек серверного окружения приложения. На рисунке 4.20 приведена файловая структура компонента приложения store.

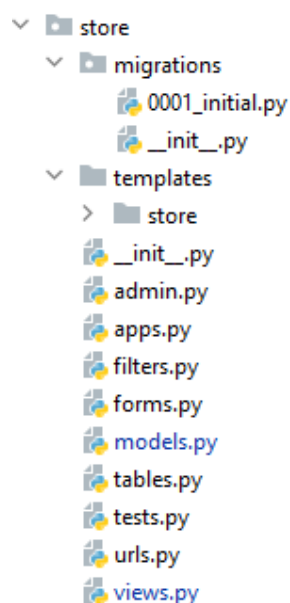


Рисунок 4.20 – Файловая структура компонента приложения store

Файл 0001\_initial.py содержит класс фреймворка Django Migration, в котором прописаны зависимости и модели компонента приложения. Данный файл используется для первого запуска приложения на новом сервере и генерации базы данных с соответствующими таблицами.

В файле urls.py прописаны отображения адреса веб-страниц в браузере и пути до файлов шаблона, которые используются для создания полной страницы. На рисунке 4.21 приведен фрагмент файла urls.py, в котором описаны пути до шаблонов компонента приложения store.

```
urlpatterns = [
    path('', views.dashboard, name='dashboard'),
    path('products/', ProductListView.as_view(), name="productslist"),
    path('product/<slug:slug>/', ProductDetailView.as_view(), name='product-detail'),
    path('new-product/', ProductCreateView.as_view(), name='product-create'),
    path('product/<slug:slug>/update/', ProductUpdateView.as_view(), name='product-update'),
    path('product/<slug:slug>/delete/', ProductDeleteView.as_view(), name='product-delete'),
    path('search/', ItemSearchListView.as_view(), name="item_search_list_view"),
    path('deliveries/', DeliveryListView.as_view(), name="deliveries"),
    path('delivery/<slug:slug>/', DeliveryDetailView.as_view(), name='delivery-detail'),
    path('new-delivery/', DeliveryCreateView.as_view(), name='delivery-create'),
    path('delivery/<int:pk>/update/', DeliveryUpdateView.as_view(), name='delivery-update'),
    path('delivery/<int:pk>/delete/', DeliveryDeleteView.as_view(), name='delivery-delete'),
```

Рисунок 4.21 – Фрагмент файла urls.py

В корневом каталоге приложения находится файл manage.py, который проверяет целостность приложения и инициализирует командную строку фреймворка Django. На рисунке 4.22 приведен программный код файла manage.py.

```

import os
import sys

def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'InventoryMS.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Не найден Django. Вы уверены что фреймворк установлен и "
            "путь до него прописан в PYTHONPATH виртуальной среды?"
        ) from exc
    execute_from_command_line(sys.argv)

if __name__ == '__main__':
    main()

```

Рисунок 4.22 – программный код файла manage.py

#### 4.4.2 Руководство пользователя – панель сотрудника

При входе в приложение пользователя встречает страница ввода логина и пароля. Для регистрации в приложении необходимо нажать на кнопку “Создать аккаунт”. На рисунке 4.23 приведена страница входа в приложение.

Рисунок 4.23 – Страница входа в приложение.

На странице регистрации требуется ввести имя пользователя, электронную почту и пароль. Пароль проверяется на обязательное наличие специальных символов и цифр, минимальную длину, на соответствие простым и часто используемым паролям. При нажатии кнопки “Регистрация” пользователя возвращает на страницу входа в приложение. На рисунке 4.24 приведена страница регистрации в приложении.

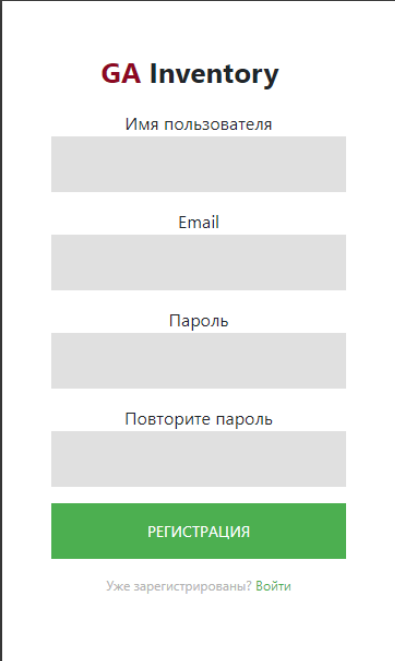


Рисунок 4.24 – Страница регистрации в приложении

После входа в приложение пользователя перенаправляет на домашнюю страницу “Дэшборд”, на которой содержится статистика по бару, персоналу, доставкам, продажам. Для навигации в приложении используется боковое меню. На рисунке 4.25 приведена домашняя страница приложения “Дэшборд”.



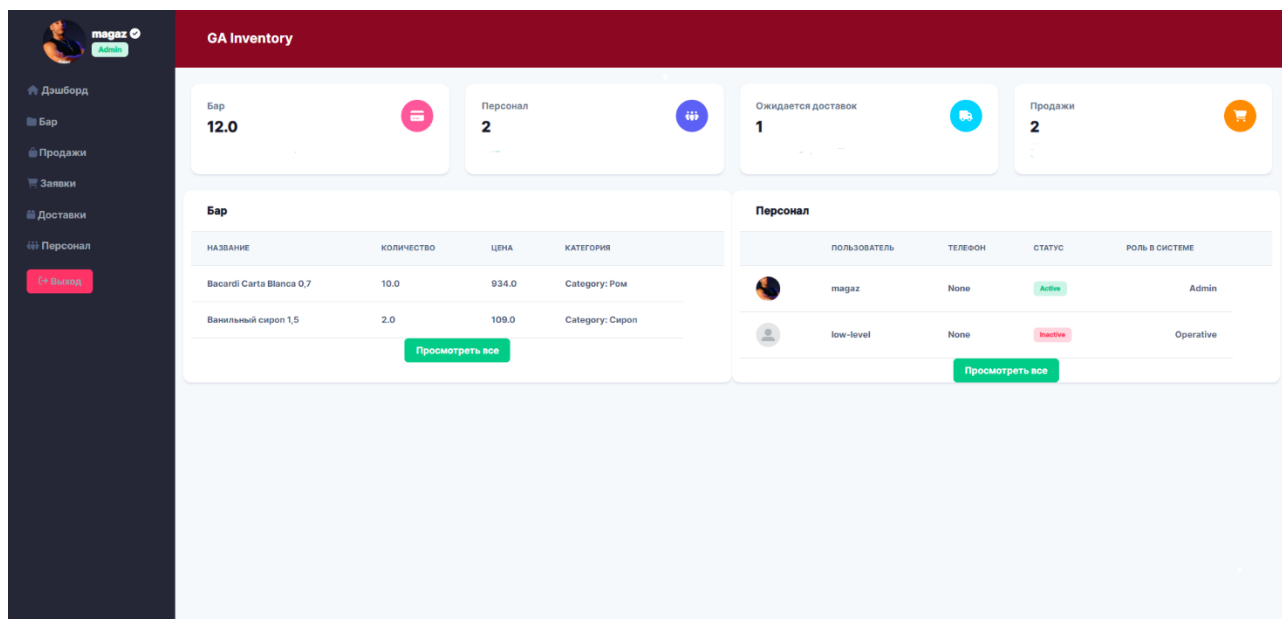


Рисунок 4.25 – Домашняя страница “Дэшборд”

На странице “Бар” приведена информация о продукции, доступной в баре в формате таблицы. Доступный функционал: добавление записи, поиск по названию, экспорт данных в таблицу Excel, редактирование и удаление записи. На рисунке 4.26 приведена страница “Бар”.

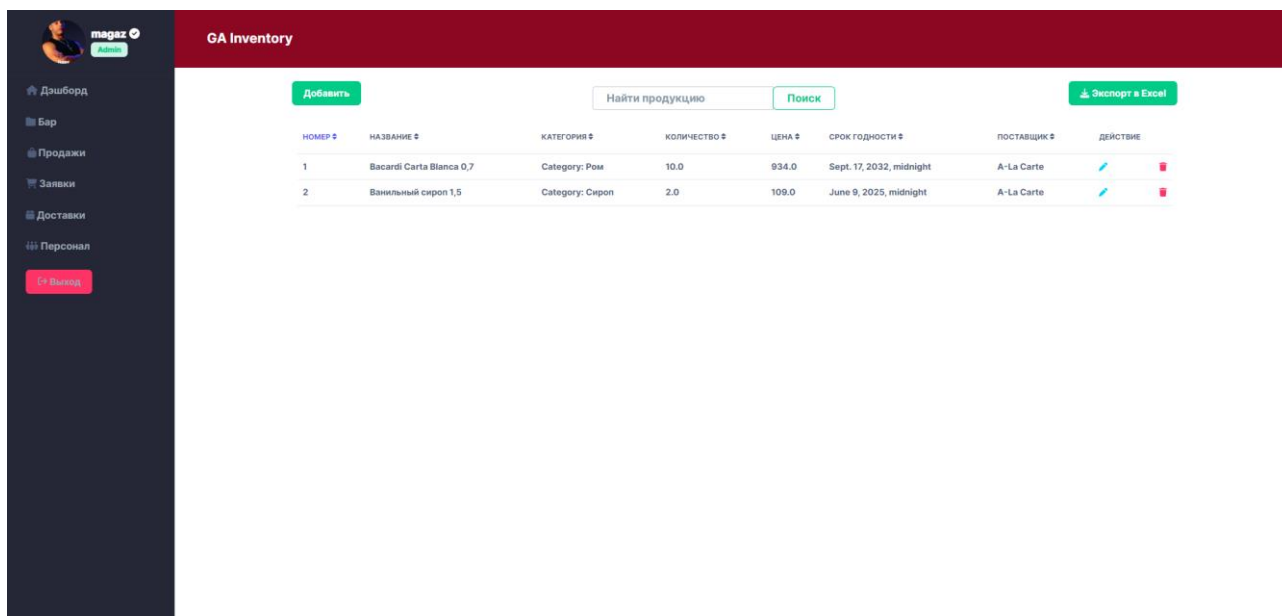


Рисунок 4.26 – Страница “Бар”

При нажатии на кнопку “Добавить” пользователя переносит на страницу заполнения формы “Добавить продукцию”. Категория и поставщик выбираются

из выпадающих меню. Кнопка “Добавить” внесет новую запись в базу данных. На рисунке 4.27 приведена страница заполнения формы “Добавить продукцию”.

The screenshot shows the 'GA Inventory' application interface. On the left is a dark sidebar with a user profile 'magaz Admin' and a menu with items: 'Дашборд', 'Бар', 'Продажи', 'Заявки', 'Доставки', 'Персонал', and a red 'Выход' button. The main area has a dark red header 'GA Inventory' and a white form titled 'Добавить продукцию'. The form contains the following fields: 'Название' (text input), 'Категория' (dropdown menu), 'Количество' (text input with value '0,0'), 'Цена продажи' (text input with value '0'), 'Срок годности' (date input with format 'dd/mm/yyyy'), and 'Поставщик' (dropdown menu). A green 'Добавить' button is at the bottom right of the form.

Рисунок 4.27 – Страница заполнения формы “Добавить продукцию”

При нажатии на кнопку в форме карандаша открывается страница редактирования записи продукции. Кнопка “Изменить” внесет изменения в базу данных. На рисунке 4.28 приведена страница редактирования записи продукции.

The screenshot shows the 'GA Inventory' application interface. On the left is a dark sidebar with a user profile 'magaz Admin' and a menu with items: 'Дашборд', 'Бар', 'Продажи', 'Заявки', 'Доставки', 'Персонал', and a red 'Выход' button. The main area has a dark red header 'GA Inventory' and a white form titled 'Изменить запись о товаре'. The form contains the following fields: 'Название' (text input with value 'Bacardi Carta Blanca 0,7'), 'Категория' (dropdown menu with value 'Category: Ром'), 'Количество' (text input with value '10,0'), 'Цена продажи' (text input with value '934,0'), 'Срок годности' (date input with value '2032-09-17 00:00:00'), and 'Поставщик' (dropdown menu with value 'A-La Carte'). A green 'Изменить' button is at the bottom right of the form.

Рисунок 4.28 – Страница редактирования записи продукции

При нажатии на кнопку в форме корзины открывается страница удаления записи продукции. Кнопка “Да, удалить” внесет изменения в базу данных, кнопка “Отмена” вернет пользователя на страницу “Бар”. На рисунке 4.29 приведена страница удаления записи продукции.

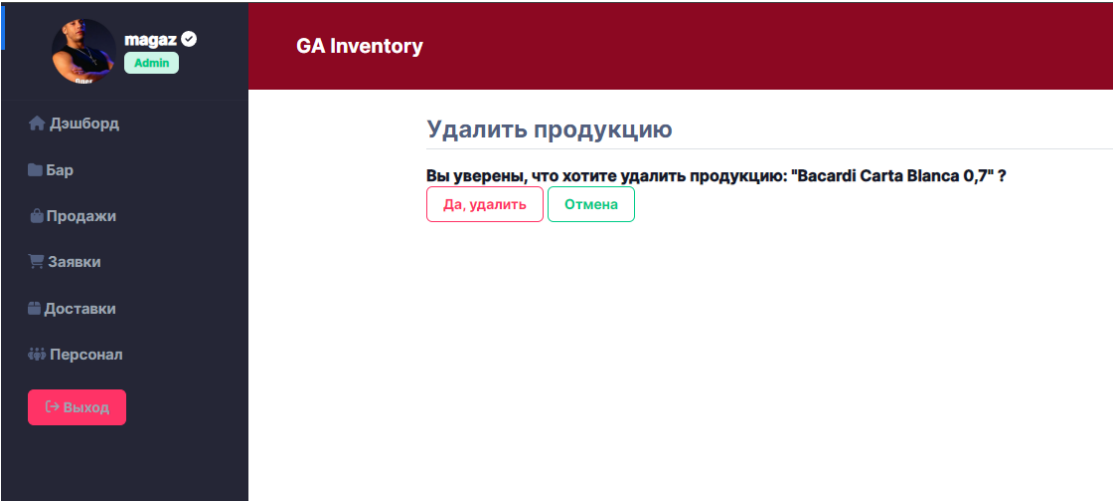


Рисунок 4.29 – Страница удаления записи продукции

На странице “Продажи” приведена информация о продажах, совершенных в баре в формате таблицы. Доступный функционал: добавление записи, экспорт данных в таблицу Excel, редактирование и удаление записи. На рисунке 4.30 приведена страница “Продажи”.





номер	товар	количество	цена за единицу	общая цена	способ оплаты	имя покупателя	дата	продавец	действие
1	Bacardi Carta Blanca 0,7	1,0	934,0	934,0	БАНК	Сегрей	2024/05/07 - 04:05:07	magaz	 
2	Bacardi Carta Blanca 0,7	3,0	934,0	2802,0	МКБ	Сегрей	2024/06/11 - 07:06:28	magaz	 

Рисунок 4.30 – Страница “Продажи”

При нажатии на кнопку “Добавить продажу” пользователя переносит на страницу заполнения формы “Добавить продажу”. Название продукции и способ оплаты выбираются из выпадающих меню. Кнопка “Добавить” внесет новую запись в базу данных. На рисунке 4.31 приведена страница заполнения формы “Добавить продажу”.

The screenshot shows the 'Add Sale' form in the GA Inventory system. The interface has a dark sidebar on the left with navigation links: 'Дашборд', 'Бар', 'Продажи', 'Заказы', 'Доставки', 'Персонал', and a 'Выход' button. The main header is red with 'GA Inventory'. The form title is 'Добавить продажу'. It contains the following fields: 'Название' (a dropdown menu), 'Покупатель' (a text input), 'Способ оплаты' (a dropdown menu), 'Количество\*' (a text input with '0,0'), and 'Получено\*' (a text input). A green 'Добавить' button is at the bottom.

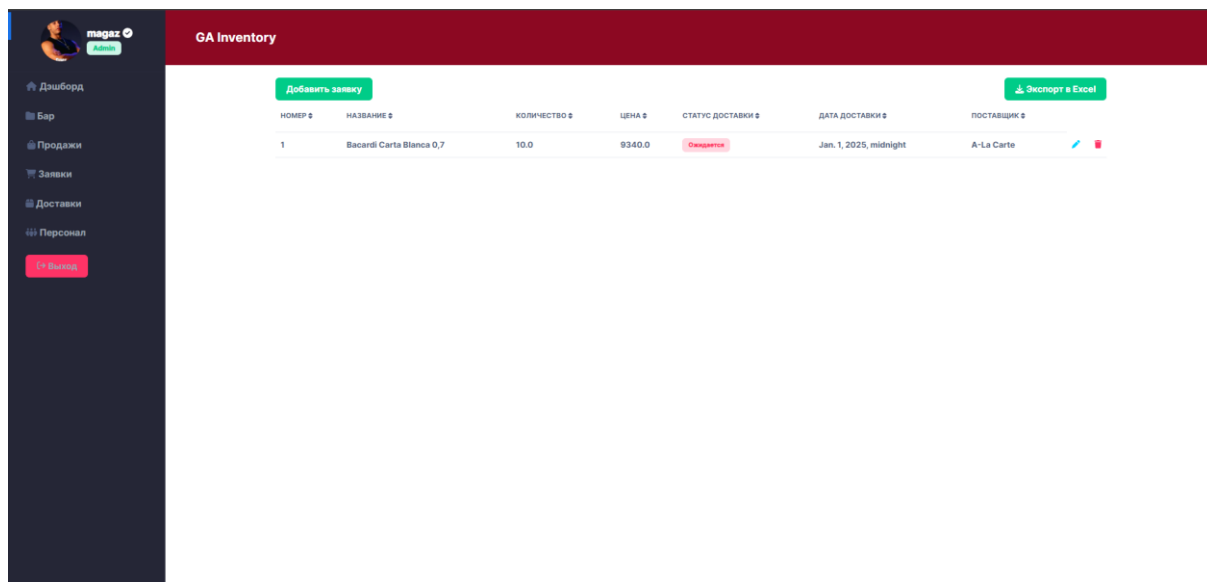
Рисунок 4.31 – Страница заполнения формы “Добавить продажу”

При нажатии на кнопку в форме карандаша открывается страница редактирования записи о продаже. Кнопка “Изменить” внесет изменения в базу данных. На рисунке 4.32 приведена страница редактирования записи о продаже.

The screenshot shows the 'Edit Sale Record' form in the GA Inventory system. The interface is identical to the previous one, with the same sidebar and header. The form title is 'Изменить запись о продаже'. The fields are pre-filled with data: 'Название' is 'Bacardi Carta Blanca 0,7 - Category: Category: Pom, Quantity: 10.0', 'Покупатель' is 'Сергей', 'Способ оплаты' is 'MIR', 'Количество\*' is '1,0', 'Цена\*' is '934,0', and 'Получено\*' is '1090,0'. A green 'Изменить' button is at the bottom.

Рисунок 4.32 – Страница редактирования записи о продаже

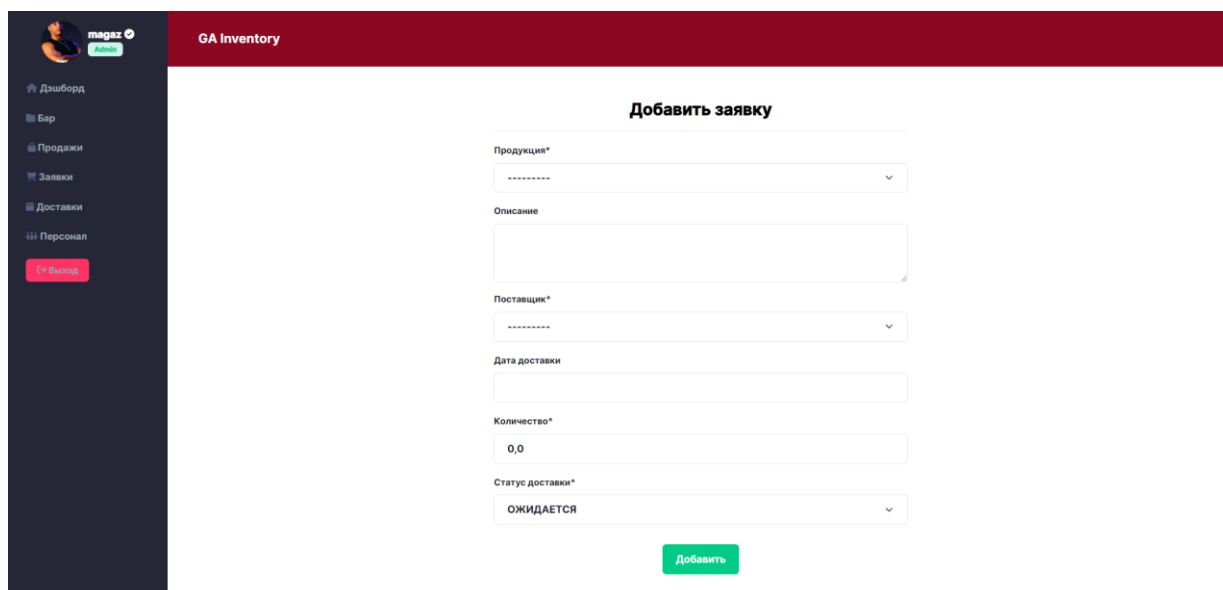
На странице “Заявки” приведена информация о заявках на склад в формате таблицы. Доступный функционал: добавление записи, экспорт данных в таблицу Excel, редактирование и удаление записи. На рисунке 4.33 приведена страница “Заявки”.



NOМЕР	НАЗВАНИЕ	КОЛИЧЕСТВО	ЦЕНА	СТАТУС ДОСТАВКИ	ДАТА ДОСТАВКИ	ПОСТАВЩИК
1	Bacardi Carta Blanca 0,7	10.0	9340.0	Ожидается	Jan. 1, 2025, midnight	A-La Carte

Рисунок 4.33 – Страница “Заявки”

При нажатии на кнопку “Добавить заявку” пользователя переносит на страницу заполнения формы “Добавить заявку”. Название продукции, поставщик и статус доставки выбираются из выпадающих меню. Кнопка “Добавить” внесет новую запись в базу данных. На рисунке 4.34 приведена страница заполнения формы “Добавить заявку”.



**Добавить заявку**

Продукция\*  
.....

Описание  
.....

Поставщик\*  
.....

Дата доставки  
.....

Количество\*  
0,0

Статус доставки\*  
ОЖИДАЕТСЯ

Добавить

Рисунок 4.34 – Страница заполнения формы “Добавить заявку”

При нажатии на кнопку в форме карандаша открывается страница редактирования заявки. Кнопка “Изменить” внесет изменения в базу данных. На рисунке 4.35 приведена страница редактирования заявки.

**GA Inventory**

**Изменить заявку**

Продукция\*  
Bacardi Carta Blanca 0,7 - Category: Category: Ром, Quantity: 10.0

Описание  
срочно, ром на баре кончился!

Поставщик\*  
A-La Carte

Дата доставки  
2025-01-01 00:00:00

Количество\*  
10,0

Цена за единицу\*  
934,0

Статус доставки\*  
ОЖИДАЕТСЯ

**Изменить**

Рисунок 4.35 – Страница редактирования заявки

На странице “Доставки” приведена информация о заявках, которые обработал склад. Доступный функционал: добавление записи, экспорт данных в таблицу Excel, редактирование и удаление записи. На рисунке 4.36 приведена страница “Доставки”.

**GA Inventory**

**Добавить доставку** **Экспорт в Excel**

ID	НАЗВАНИЕ	ЗАПРОС ОТ	ТЕЛЕФОН	ЛОКАЦИЯ	ДАТА ДОСТАВКИ	СТАТУС ДОСТАВКИ	ДЕЙСТВИЕ
1	Ванильный сироп 1,5	Сергей	+12125552368	Бар	Jan. 1, 2024, midnight	Доставлено	

Рисунок 4.36 – Страница “Доставки”

При нажатии на кнопку “Добавить доставку” пользователя переносит на страницу заполнения формы “Добавить доставку”. Название продукции выбирается из выпадающего меню, устанавливается отметка о выполнении. Кнопка “Добавить” внесет новую запись в базу данных. На рисунке 4.37 приведена страница заполнения формы “Добавить доставку”.

The screenshot shows the 'GA Inventory' web application interface. On the left is a dark sidebar with navigation links: 'Дашборд', 'Бар', 'Продажи', 'Заявки', 'Доставки', and 'Персонал'. A red 'Выход' button is at the bottom of the sidebar. The main content area has a red header with 'GA Inventory'. Below the header, the title 'Добавить доставку' is centered. The form contains the following fields: 'Продукция' (a dropdown menu), 'Обработал' (a text input field), 'Номер телефона' (a text input field), 'Локация' (a text input field), and 'Дата\*' (a date and time input field). At the bottom of the form is a checkbox labeled 'Доставлено' and a green 'Добавить' button.

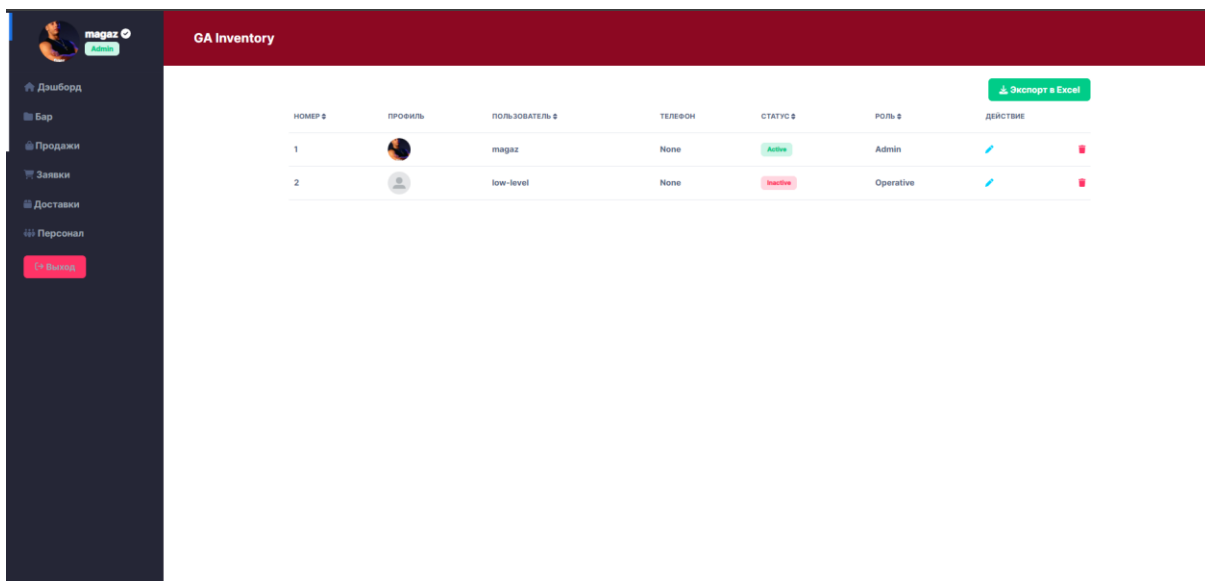
Рисунок 4.37 – Страница заполнения формы “Добавить доставку”

При нажатии на кнопку в форме карандаша открывается страница редактирования доставки. Кнопка “Изменить” внесет изменения в базу данных. На рисунке 4.38 приведена страница редактирования заявки.

The screenshot shows the 'GA Inventory' web application interface for editing a delivery. The sidebar and header are identical to the previous screenshot. The main content area has a red header with 'GA Inventory'. Below the header, the title 'Изменить запись о доставке' is centered. The form contains the following fields: 'Продукция' (a dropdown menu showing 'Ванильный сыр 1,5 - Category: Category: Сыр, Quantity: 2.0'), 'Обработал' (a text input field with 'Сергей'), 'Номер телефона' (a text input field with '+12125552368'), 'Локация' (a text input field with 'Бар'), and 'Дата\*' (a date and time input field with '2024-01-01 00:00:00'). At the bottom of the form is a checked checkbox labeled 'Доставлено' and a green 'Изменить' button.

Рисунок 4.38 – Страница редактирования доставки

На странице “Персонал” приведена информация о сотрудниках. Доступный функционал: экспорт данных в таблицу Excel, редактирование и удаление записи. На рисунке 4.39 приведена страница “Персонал”.

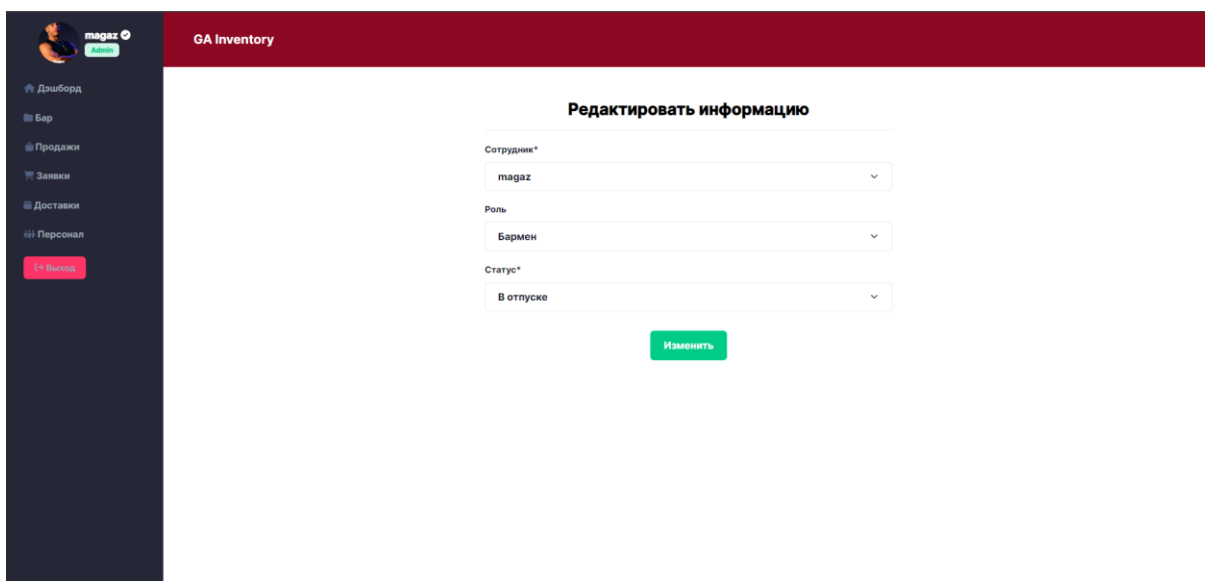


The screenshot shows the 'GA Inventory' application interface. On the left is a dark sidebar with a user profile 'magaz Admin' and navigation links: 'Дашборд', 'Бар', 'Продажи', 'Заказы', 'Доставки', and 'Персонал'. A red 'Выход' button is at the bottom of the sidebar. The main content area has a red header 'GA Inventory' and a green 'Экспорт в Excel' button. Below is a table with employee data.

НОМЕР #	ПРОФИЛЬ	ПОЛЬЗОВАТЕЛЬ #	ТЕЛЕФОН	СТАТУС #	РОЛЬ #	ДЕЙСТВИЕ
1		magaz	None	Active	Admin	
2		low-level	None	Inactive	Operative	

Рисунок 4.39 – Страница “Персонал”

При нажатии на кнопку в форме карандаша открывается страница редактирования информации о сотруднике. Кнопка “Изменить” внесет изменения в базу данных. Стоит отметить, что редактирование информации о сотруднике не сказывается на данных пользователя внутри приложения. На рисунке 4.40 приведена страница редактирования информации о сотруднике.



The screenshot shows the 'GA Inventory' application interface for editing employee information. The sidebar is identical to the previous screenshot. The main content area has a red header 'GA Inventory' and a title 'Редактировать информацию'. Below the title is a form with three dropdown menus: 'Сотрудник\*' (set to 'magaz'), 'Роль' (set to 'Бармен'), and 'Статус\*' (set to 'В отпуске'). A green 'Изменить' button is at the bottom of the form.

Рисунок 4.40 – Страница редактирования информации о сотруднике



При нажатии на изображение профиля сотрудника в левом верхнем углу интерфейса, отображается страница “Ваш профиль”. В ней содержится полная информация о сотруднике. На рисунке 4.41 приведена страница “Ваш профиль”.

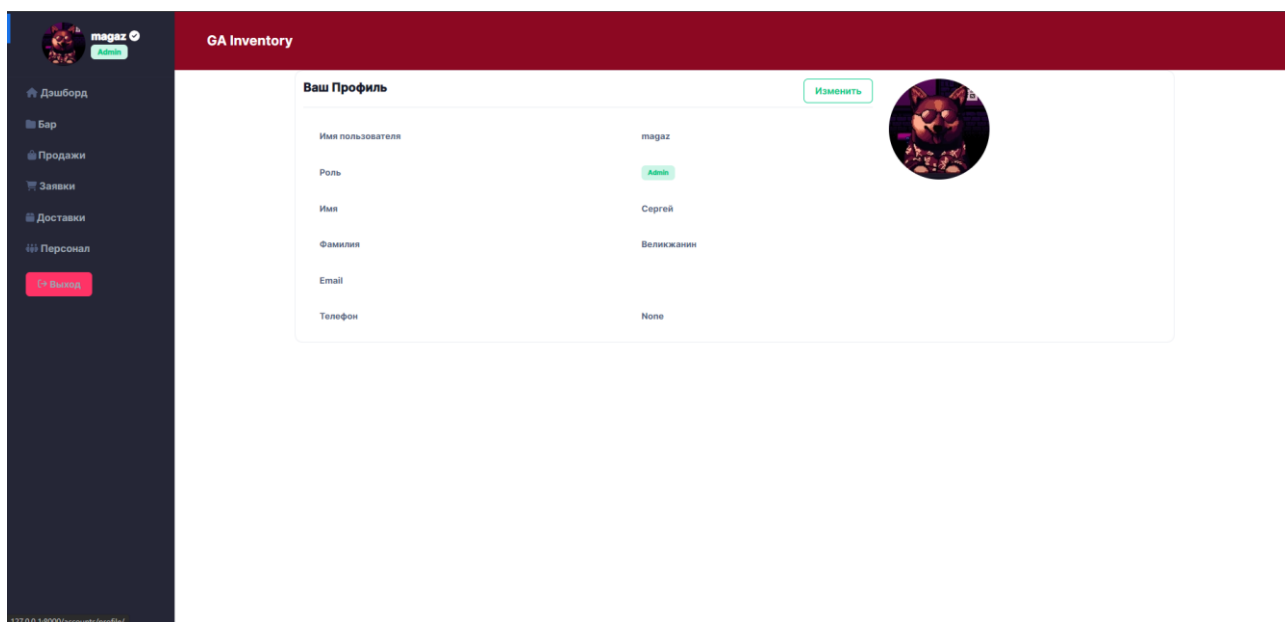


Рисунок 4.41 – Страница “Ваш профиль”

При нажатии на кнопку “Изменить” пользователя переносит на страницу редактирования профиля. В профиль можно загрузить любой файл формата jpg в качестве изображения. На рисунке 4.42 приведена страница “Изменение профиля”.

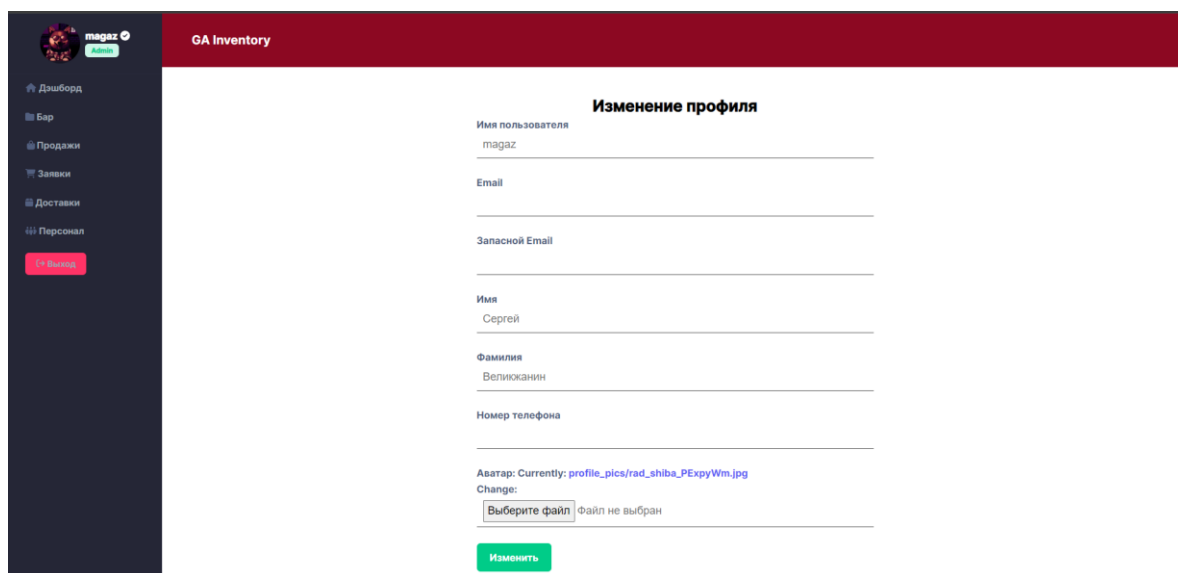


Рисунок 4.42 – Страница “Изменения профиля”

### 4.4.3 Руководство пользователя – панель администратора

Доступ к панели администратора выдается вручную системным администратором. Для перехода к панели необходимо в адресной строке браузера вручную дописать к адресу домашней страницы “/admin”. При условии, что у пользователя достаточный уровень прав, его направит на страницу “Django administration”. Панель администратора включает в себя весь функционал панели сотрудника, а также добавляет возможность создания и редактирования записей о поставщиках и работу с учетными записями пользователей приложения. На рисунке 4.43 приведена домашняя страница “Django administration”.

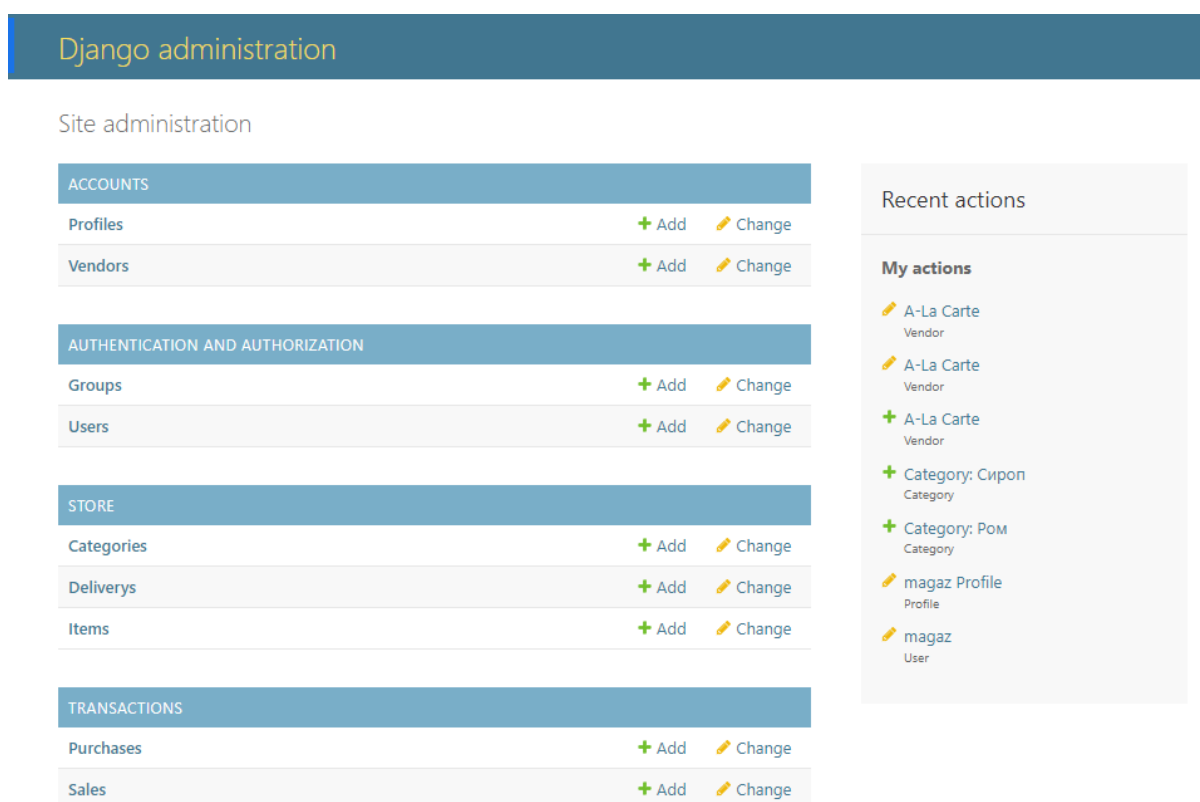


Рисунок 4.43 – Домашняя страница “Django administration”

В разделе “Profiles” можно добавлять и редактировать записи о сотрудниках. Сохраняется история изменений. На рисунке 4.44 приведена страница раздела “Profiles”.

Django administration

Home > Accounts > Profiles > magaz Profile

Start typing to filter...

**ACCOUNTS**

Profiles + Add

Vendors + Add

**AUTHENTICATION AND AUTHORIZATION**

Groups + Add

Users + Add

**STORE**

Categories + Add

Deliverys + Add

Items + Add

**TRANSACTIONS**

Purchases + Add

Sales + Add

**Change profile**

**magaz Profile**

User: magaz

Profile picture: Currently: profile\_pics/rad\_shiba\_PExpyWm.jpg  
Change: Выберите файл Файл не выбран

Telephone:

Email:

First name: Сергей

Last name: Великжанин

Status: Активен

Role: Администратор

SAVE Save and add another Save and continue editing

Рисунок 4.44 – Страница раздела Profiles

В разделе “Vendors” возможно добавлять и редактировать записи о поставщиках. Данный функционал доступен только пользователям с ролью администратора и только через эту панель. В качестве информации указывается название, номер телефона и адрес поставщика. На рисунке 4.45 приведена страница раздела “Vendors”.

Django administration

Home > Accounts > Vendors > Add vendor

Start typing to filter...

**ACCOUNTS**

Profiles + Add

Vendors + Add

**AUTHENTICATION AND AUTHORIZATION**

Groups + Add

Users + Add

**STORE**

Categories + Add

Deliverys + Add

Items + Add

**TRANSACTIONS**

Purchases + Add

Sales + Add

**Add vendor**

Name:

Phone number:

Address:

SAVE Save and add another Save and continue editing

Рисунок 4.45 – Страница раздела Vendors

В разделе “Users” возможно редактировать данные зарегистрированных пользователей. В частности, изменять имя пользователя, контактные данные, выдавать доступ к функционалу приложения в панели сотрудника и доступ к панели администратора в целом. Фреймворк Django предусматривает шифрование пароля пользователя, поэтому вместо простого текста в панели администратора отображается алгоритм шифрования пароля и фрагмент зашифрованных данных. Также через этот раздел можно изменить пароль. На рисунке 4.46 приведена страница раздела “Users”.

The screenshot displays the Django administration interface. The top navigation bar shows the breadcrumb path: Home > Authentication and Authorization > Users > magaz. On the left, a sidebar menu lists various sections: ACCOUNTS (Profiles, Vendors), AUTHENTICATION AND AUTHORIZATION (Groups, Users), STORE (Categories, Deliverys, Items), and TRANSACTIONS (Purchases, Sales). The 'Users' link is highlighted. The main content area is titled 'Change user' and shows details for the user 'magaz'. The 'Username' field is set to 'magaz'. The 'Password' field displays the hashing algorithm (pbkdf2\_sha256), iterations (600000), salt (ca8f05), and hash (bm1fxg). Below this, the 'Personal info' section contains fields for 'First name', 'Last name', and 'Email address'. The 'Permissions' section at the bottom has three checked checkboxes: 'Active', 'Staff status', and 'Superuser status', each with a descriptive note.

Рисунок 4.46 – Страница раздела Users

В разделе “Deliveries” можно добавлять и редактировать записи о доставках. Сохраняется история изменений. На рисунке 4.47 приведена страница раздела “Deliveries”.

Django administration

Home > Store > Deliverys > Delivery of Ванильный сыр 1,5 - Category: Category: Сырон, Quantity: 2.0 to Сепрей at Бар on 2024-01-01 00:00:00+00:00

Start typing to filter...

ACCOUNTS

Profiles + Add

Vendors + Add

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

STORE

Categories + Add

Deliverys + Add

Items + Add

TRANSACTIONS

Purchases + Add

Sales + Add

Change delivery

Delivery of Ванильный сыр 1,5 - Category: Category: Сырон, Quantity: 2.0 to Сепрей at Бар on 2024-01-01

Item:

Ванильный сыр 1,5 - Category: Category: Сырон, Quantity: 2.0

Customer name:

Сепрей

Phone number:

+12125552368

Location:

Бар

Date:

Date:

2024-01-01

Today

Time:

00:00:00

Now

Note: You are 7 hours ahead of server time.

☒ Is-delivered

SAVE

Save and add another

Save and continue editing

Рисунок 4.47 – Страница раздела Deliveries

В разделе “Items” можно добавлять и редактировать записи о продукции в баре. Сохраняется история изменений. На рисунке 4.48 приведена страница раздела “Items”.

Django administration

Home > Store > Items > Bacardi Carta Blanca 0,7 - Category: Category: Пом, Quantity: 10.0

Start typing to filter...

ACCOUNTS

Profiles + Add

Vendors + Add

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

STORE

Categories + Add

Deliverys + Add

Items + Add

TRANSACTIONS

Purchases + Add

Sales + Add

Change item

Bacardi Carta Blanca 0,7 - Category: Category: Пом, Quantity: 10.0

Name:

Bacardi Carta Blanca 0,7

Category:

Category: Пом

Quantity:

10,0

Selling price:

934,0

Expiring date:

Date:

2032-09-17

Today

Time:

00:00:00

Now

Note: You are 7 hours ahead of server time.

Vendor:

A-La Carte

SAVE

Save and add another

Save and continue editing

Рисунок 4.48 – Страница раздела Items

В разделе “Sales” можно добавлять и редактировать записи о продажах в баре. Сохраняется история изменений. На рисунке 4.49 приведена страница раздела “Sales”.

The screenshot displays the Django administration interface. The top navigation bar shows the path: Home > Transactions > Sales > Bacardi Carta Blanca 0,7. On the left, a sidebar menu lists various sections: ACCOUNTS (Profiles, Vendors), AUTHENTICATION AND AUTHORIZATION (Groups, Users), STORE (Categories, Deliverys, Items), and TRANSACTIONS (Purchases, Sales). The 'Sales' item is highlighted. The main content area is titled 'Change sale' and shows the details for a specific sale: 'Bacardi Carta Blanca 0,7'. The form includes fields for 'Название' (Name), 'Покупатель' (Buyer), 'Способ оплаты' (Payment method), 'Количество' (Quantity), 'Цена' (Price), and 'Получено' (Received). The 'Название' field is pre-filled with 'Bacardi Carta Blanca 0,7 - Category: Category: Пом, Quantity: 10.0'. The 'Покупатель' field contains 'Сергей'. The 'Способ оплаты' field is set to 'MIR'. The 'Количество' field is '3,0', the 'Цена' field is '934,0', and the 'Получено' field is '100500,0'. At the bottom, there are three buttons: 'SAVE', 'Save and add another', and 'Save and continue editing'.

Section	Item	Action
ACCOUNTS	Profiles	+ Add
	Vendors	+ Add
AUTHENTICATION AND AUTHORIZATION	Groups	+ Add
	Users	+ Add
STORE	Categories	+ Add
	Deliverys	+ Add
	Items	+ Add
TRANSACTIONS	Purchases	+ Add
	Sales	+ Add

Change sale

**Bacardi Carta Blanca 0,7**

Название: Bacardi Carta Blanca 0,7 - Category: Category: Пом, Quantity: 10.0

Покупатель: Сергей

Способ оплаты: MIR

Количество: 3,0

Цена: 934,0

Получено: 100500,0

SAVE Save and add another Save and continue editing

Рисунок 4.49 – Страница раздела Sales

## 5 Тестирование

### 5.1 Тестирование приложения в различных браузерах

Работоспособность приложения протестирована в трех браузерах: Google Chrome, Yandex Browser, Mozilla Firefox. Также протестирована мобильная версия приложения в браузере Google Chrome.

Для тестирования выбрана страница “Бар”, т.к. с этой страницей на постоянной основе взаимодействует большая часть пользователей.

Тестирование во всех браузерах показало, что все стили загружаются, выполняются все JS-скрипты. Запросы работают корректно во всех браузерах и мобильной версии. На рисунках 5.1, 5.3, 5.5 приведена страница “Бар” в разных браузерах. На рисунках 5.2, 5.4, 5.6 приведена страница заполнения формы для добавления записи о продукции.

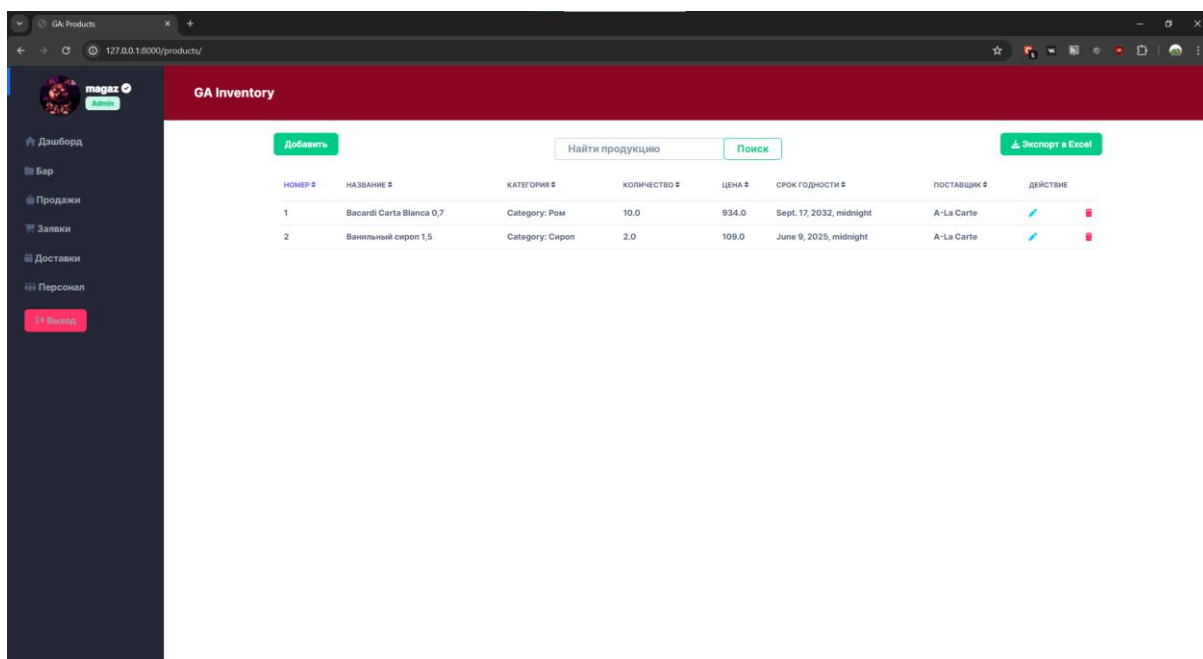


Рисунок 5.1 – Страница “Бар” в браузере Google Chrome

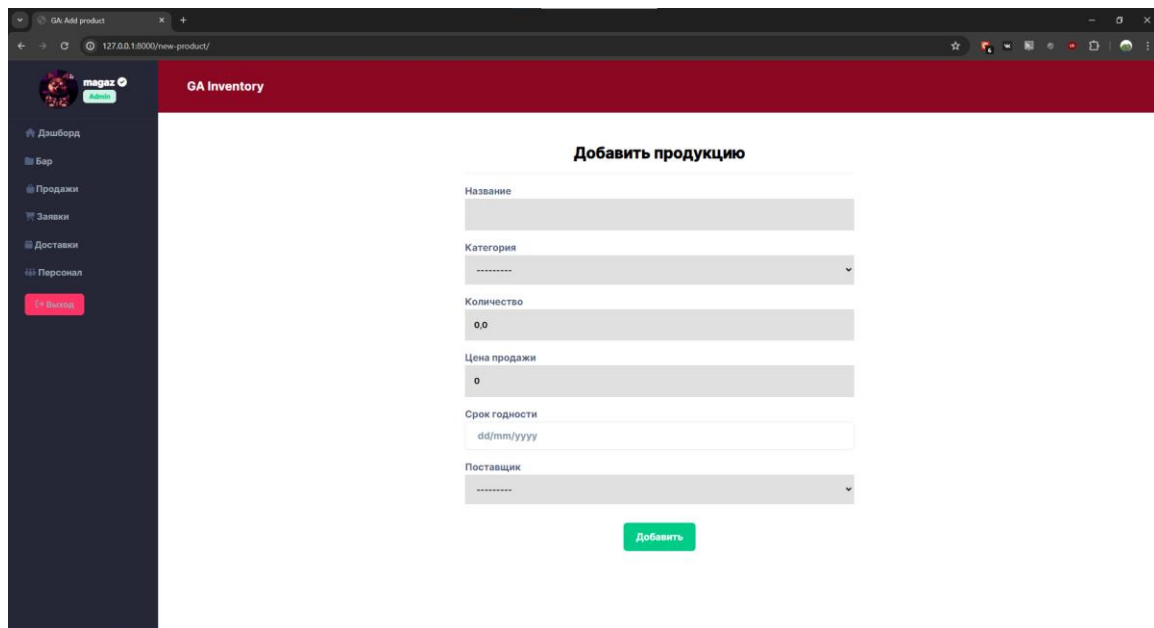


Рисунок 5.2 – Страница заполнения формы в браузере Google Chrome

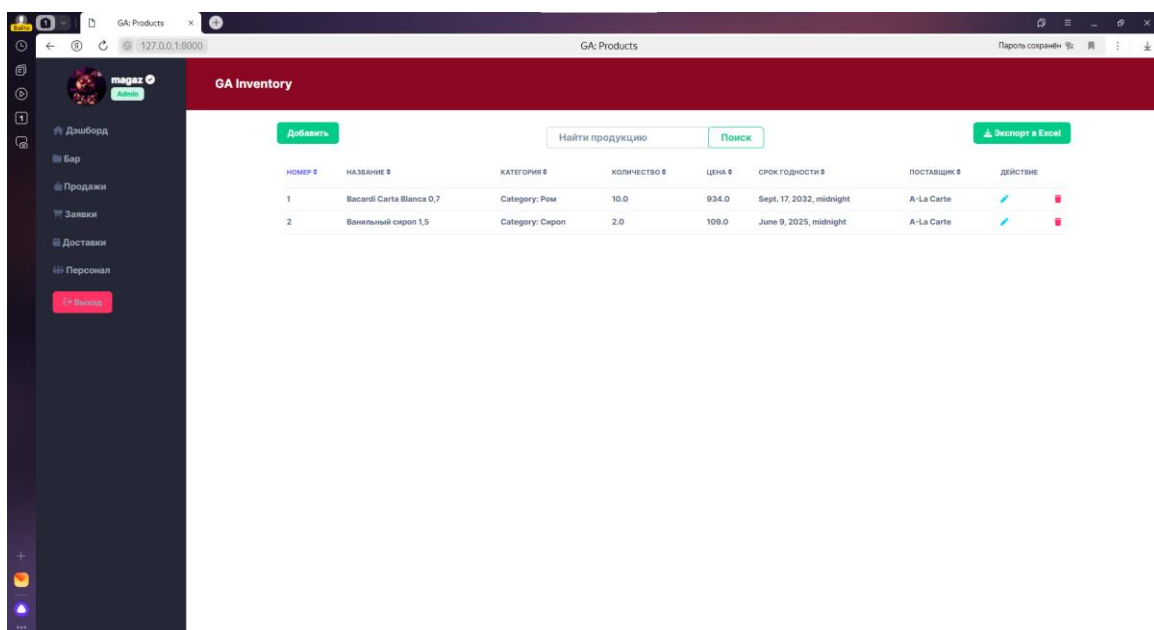


Рисунок 5.3 – Страница “Бар” в браузере Yandex Browser



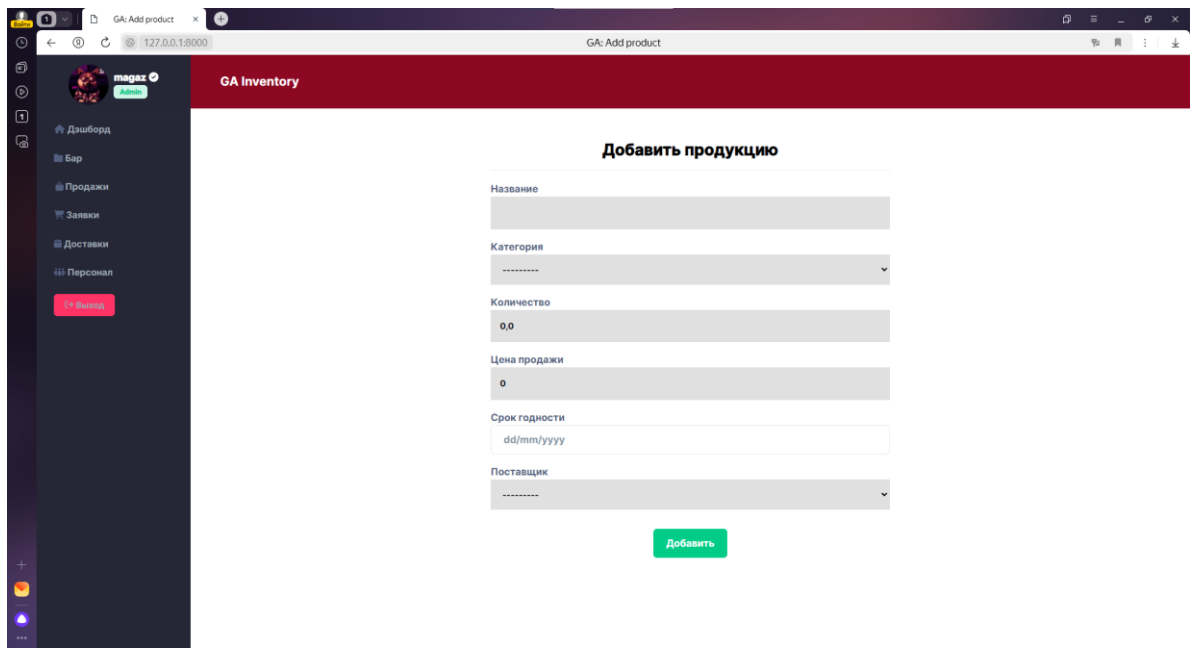


Рисунок 5.4 – Страница заполнения формы в браузере Yandex Browser

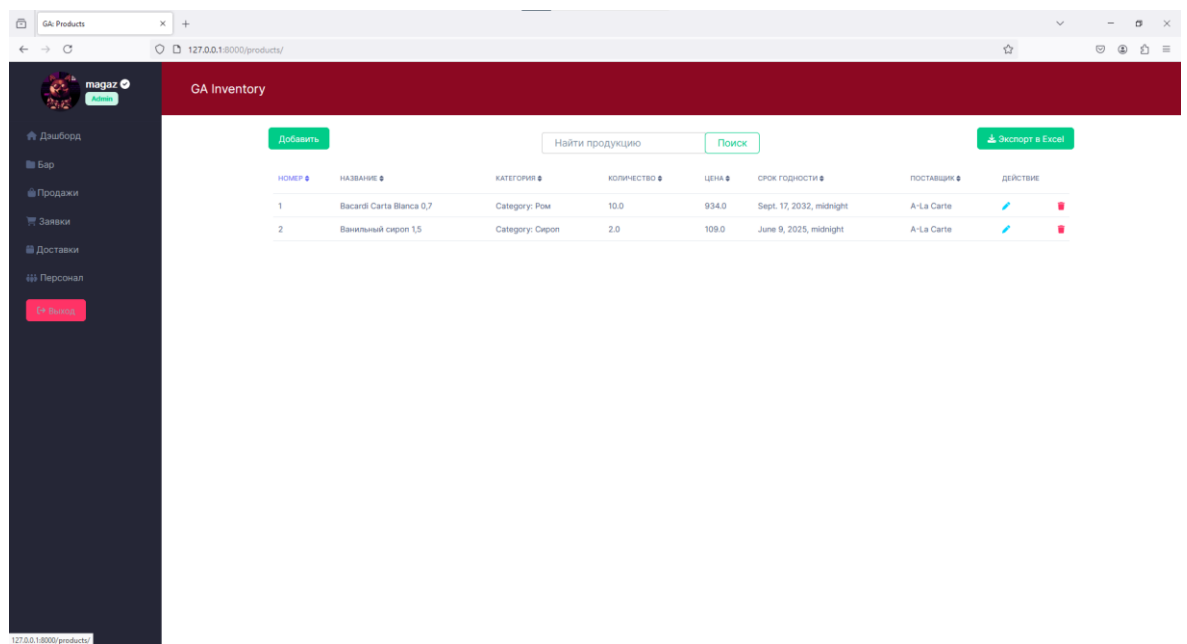


Рисунок 5.5 – Страница “Бар” в браузере Mozilla Firefox

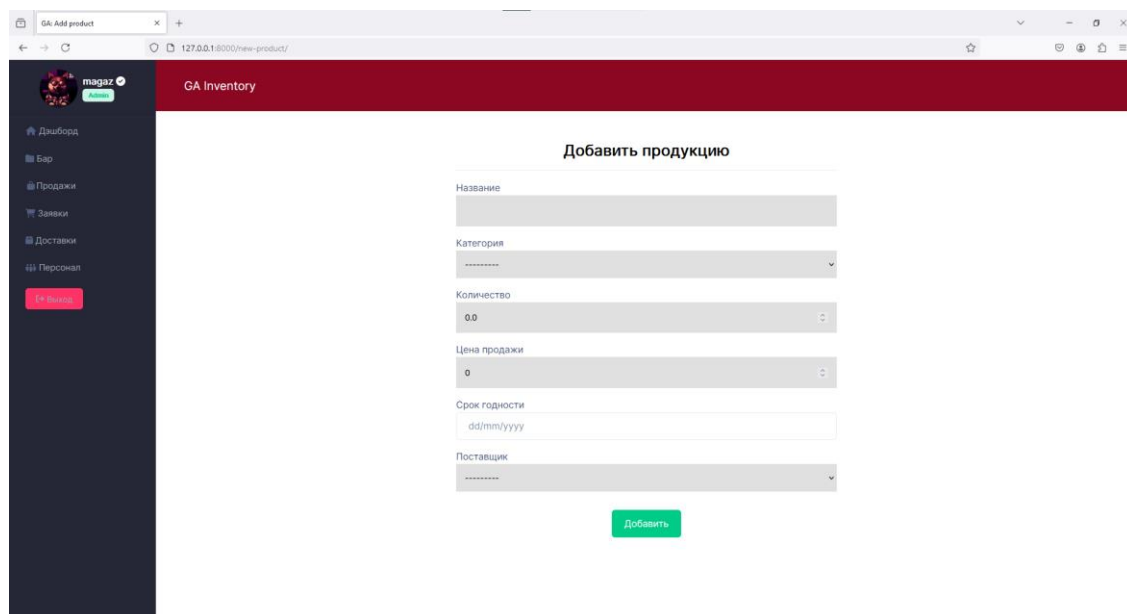


Рисунок 5.6 – Страница заполнения формы в браузере Mozilla Firefox

В мобильной версии найден недостаток. На данный момент верстка веб-страниц плохо адаптируется к форме экрана современных смартфонов, поэтому при просмотре страницы “Бар” ее необходимо приближать. Но стоит отметить, что все файлы стилей и JS-скрипты загрузились корректно. Страница заполнения формы отображается корректно и без видимых ошибок. На рисунках 5.7, 5.8 приведена мобильная версия страницы “Бар” и страницы заполнения формы в браузере Google Chrome.

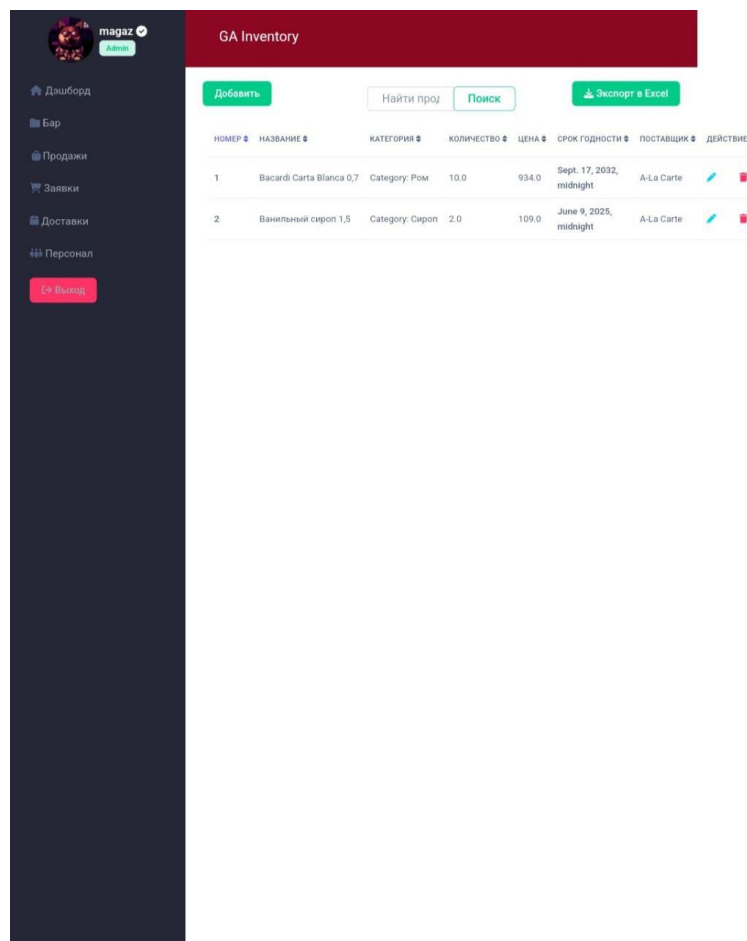


Рисунок 5.7 – Мобильная версия страницы “Бар” в браузере Google Chrome

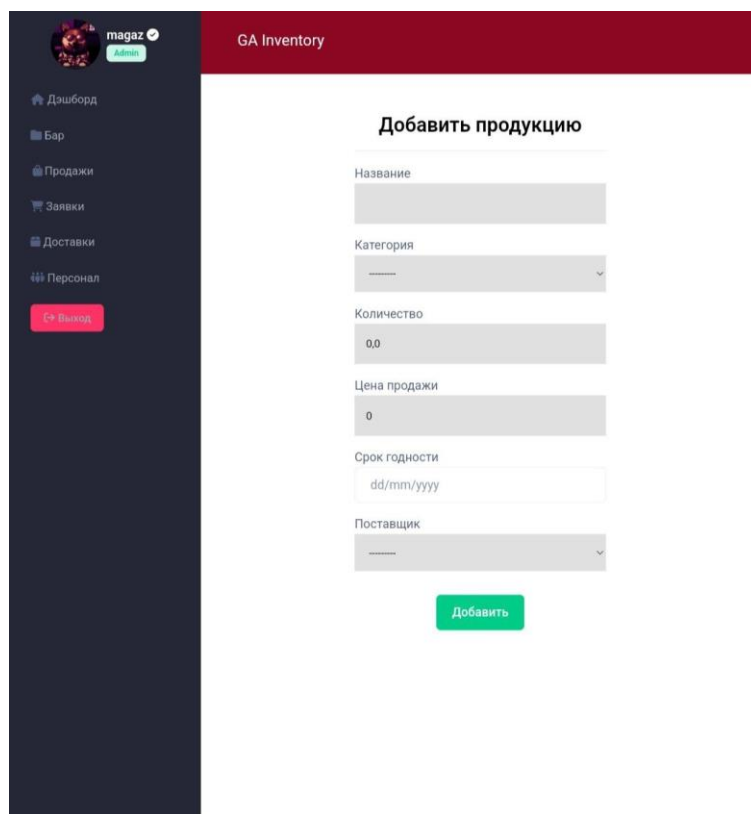


Рисунок 5.8 – Мобильная версия страницы заполнения формы

## **Заключение**

В выпускной квалификационной работе бакалавра “Приложение для учета инвентаря бара в отеле “Grand Autograph” было разработано веб-приложение, которое решает проблему неэффективности рабочих процессов, связанных с инвентаризацией.

В частности, дает возможность отслеживать, управлять и редактировать информацию о продукции, находящейся на баре в реальном времени, управлять и отслеживать статус сотрудников, а также просматривать статистику доставок и продаж.

Были выбраны необходимые технологии для успешного выполнения поставленной задачи, в том числе среда разработки, язык программирования и проч.

Выпускная работа на тему “Приложение для учета инвентаря бара в отеле “Grand Autograph” выполнена в соответствии с поставленным заданием.

В работе было выполнено следующее:

- Изучены рабочие процессы в баре отеля “Grand Autograph”;
- Разработана структура базы данных;
- Разработана серверная часть приложения;
- Разработана клиентская часть приложения;
- Протестирована работа приложения.

Итогом выпускной квалификационной работы является веб-приложения, готовое к работе и выполняющее функции отслеживания информации о продукции, сотрудниках, доставках и продажах. Реализованы панель сотрудника и администратора, регистрация пользователей, управление инвентарем бара.

## Список источников

1. What is MVT structure in Django? [Электронный ресурс] – Электрон. текстовые дан. – <https://www.educative.io/answers/what-is-mvt-structure-in-django> Режим доступа: свободный (дата обращения 27.04.2024).
2. Основы HTML [Электронный ресурс] – Электрон. текстовые дан. – [https://developer.mozilla.org/ru/docs/Learn/Getting\\_started\\_with\\_the\\_web/HTML\\_basics](https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/HTML_basics) Режим доступа: свободный (дата обращения 27.04.2024).
3. Что такое CSS и зачем нужны каскадные таблицы стилей [Электронный ресурс] – Электрон. текстовые дан. – <https://timeweb.com/ru/community/articles/chto-takoe-css> Режим доступа: свободный (дата обращения 27.04.2024).
4. Что такое JavaScript: назначение, особенности и сферы применения языка [Электронный ресурс] – Электрон. текстовые дан. – <https://timeweb.com/ru/community/articles/chto-takoe-javascript> Режим доступа: свободный (дата обращения 27.04.2024).
5. Comparison of the usage statistics of Bootstrap vs. React for websites [Электронный ресурс] – Электрон. текстовые дан. – <https://w3techs.com/technologies/comparison/js-bootstrap.js-react#:~:text=Usage%20and%20market%20share&text=Bootstrap%20is%20used%20by%2020.0,library%20market%20share%20of%2024.6%25> Режим доступа: свободный (дата обращения 19.05.2024).
6. Что такое Bootstrap [Электронный ресурс] - Электрон. текстовые дан. - <http://web.spt42.ru/index.php/chto-takoe-bootstrap> Режим доступа: свободный (дата обращения 03.05.2024).
7. Основы языка программирования Python [Электронный ресурс] – Электрон. текстовые дан. – [https://www.nic.ru/help/osnovy-yazyka-programmirovaniya-python\\_11662.html](https://www.nic.ru/help/osnovy-yazyka-programmirovaniya-python_11662.html) Режим доступа: свободный (дата обращения 03.05.2024).

8. Django введение [Электронный ресурс] – Электрон. текстовые дан. – <https://developer.mozilla.org/ru/docs/Learn/Server-side/Django/Introduction>. Режим доступа: свободный (дата обращения 03.05.2024).

9. sqlite3 — DB-API 2.0 интерфейс для баз данных SQLite [Электронный ресурс] – Электрон. текстовые дан. – <https://django.fun/docs/python/3.10/library/sqlite3/> Режим доступа: свободный (дата обращения 03.05.2024).

10. Models [Электронный ресурс] – Электрон. текстовые дан. – <https://docs.djangoproject.com/en/5.0/topics/db/models/> Режим доступа: свободный (дата обращения 07.05.2024).

11. Class-based views [Электронный ресурс] – Электрон. текстовые дан. – <https://docs.djangoproject.com/en/5.0/topics/class-based-views/> Режим доступа: свободный (дата обращения 07.05.2024).

12. Making queries [Электронный ресурс] – Электрон. текстовые дан. – <https://docs.djangoproject.com/en/5.0/topics/db/queries/> Режим доступа: свободный (дата обращения 07.05.2024).

13. The Django admin site [Электронный ресурс] – Электрон. текстовые дан. – <https://docs.djangoproject.com/en/5.0/ref/contrib/admin/> Режим доступа: свободный (дата обращения 07.05.2024).

14. Navbar [Электронный ресурс] – Электрон. текстовые дан. – <https://getbootstrap.com/docs/5.0/components/navbar/> Режим доступа: свободный (дата обращения 07.05.2024).

15. Cross Site Request Forgery protection [Электронный ресурс] – Электрон. текстовые дан. – <https://docs.djangoproject.com/en/5.0/ref/csrf/> Режим доступа: свободный (дата обращения 07.05.2024).

16. How to use Django's CSRF protection [Электронный ресурс] – Электрон. текстовые дан. – <https://docs.djangoproject.com/en/5.0/howto/csrf/#using-csrf> Режим доступа: свободный (дата обращения 07.05.2024).

## **Приложение**

Исходные коды приложения, разработанного в рамках выпускной квалификационной работы бакалавра, размещены в GitHub, облачной платформе для хостинга IT-проектов, контроля версий и совместной разработки, и доступны по ссылке: <https://github.com/realtuatar/sales-and-inventory-management-main>