1 ОБЗОР СУЩЕСТВУЮЩИХ АНАЛОГОВ

На рынке существует множество готовых решений для автоматизации кофеен и ресторанов, таких как iiko, R-Keeper, OMS, а также облачные сервисы типа "Яндекс.Еда", "Delivery Club" и другие. Однако каждый из них имеет свои плюсы и минусы.

**1.1. Готовые POS-системы (iiko, R-Keeper)**

**iiko**

Это облачная ERP-система, специализирующаяся на автоматизации предприятий общественного питания (кафе, рестораны, кофейни, доставка). Предоставляет широкий функционал: управление заказами и столами, контроль склада, аналитика продаж, интеграция с онлайн-кассами, курьерской службой, маркетплейсами и системами лояльности. Имеет интуитивно понятный интерфейс для персонала и мощные инструменты для администраторов/владельцев. Работает по модели подписки. (см. рисунок 1)



Рисунок 1 – Интерфейс системы «iiko»

**R-Keeper**

Это классическая, локально-устанавливаемая POS-система (Point of Sale) для автоматизации общепита. Включает в себя модули для работы с заказами, складом, персоналом, аналитикой. Поддерживает интеграцию с различным торговым оборудованием (кассы, весы, принтеры). Известна своей надежностью и гибкостью настроек. Может быть как локальной, так и облачной (R-Keeper Cloud). Требует первоначальной покупки лицензий и, возможно, дополнительных затрат на настройку. (см. рисунок 2)



Рисунок 2 – Интерфейс системы «R-Keeper»

Преимущества:

* Полностью готовое решение с широким функционалом.
* Поддержка различных типов оборудования (терминалы, принтеры, сканеры).
* Надежность и стабильность работы.
* Часто включают функции аналитики и отчетности.

Недостатки:

* Высокая стоимость лицензий и подписок. Особенно для небольших кофеен это может быть неподъемной нагрузкой.
* Сложность настройки. Требуют специалиста для установки и первоначальной настройки.
* Низкая гибкость. Изменить функционал или внешний вид под свои нужды крайне сложно или невозможно.
* Зависимость от поставщика. Все обновления и исправления зависят от разработчика.
* Часто не имеют встроенного удобного онлайн-магазина. Требуется интеграция с отдельными сервисами доставки.

**1.2. Онлайн-сервисы доставки (Яндекс.Еда, Delivery Club)**

**Яндекс.Еда** и **Delivery Club** — это популярные онлайн-платформы, объединяющие пользователей и рестораны/кофейни для заказа и доставки еды. Обе системы предоставляют удобные мобильные приложения и веб-интерфейсы, где клиенты могут просматривать меню, выбирать заведения, оформлять заказы и отслеживать их доставку. Для заведений они служат каналом привлечения новой аудитории и увеличения продаж за счёт онлайн-заказов. Однако, несмотря на схожую основу, каждая платформа имеет свои уникальные особенности.

**Яндекс.Еда**

Уникальной особенностью Яндекс.Еды является тесная интеграция с экосистемой Яндекса (например, карты, такси), что может быть удобно пользователям, уже пользующимся другими сервисами Яндекса. Также Яндекс.Еда часто акцентирует внимание на собственных акциях и бонусах для пользователей, а также может предлагать более гибкие условия для доставки в собственном формате (например, "Яндекс Лавка"). (см. рисунки 3-4)

**Delivery Club**

Delivery Club, как один из старейших игроков на рынке, может похвастаться обширной географией доставки и большим количеством партнерских заведений в различных городах. Уникальной чертой Delivery Club часто считается его акцент на работе с независимыми курьерами и локальными ресторанами, а также гибкая система промо-акций и скидок, включая подписку "Delivery Club Плюс".

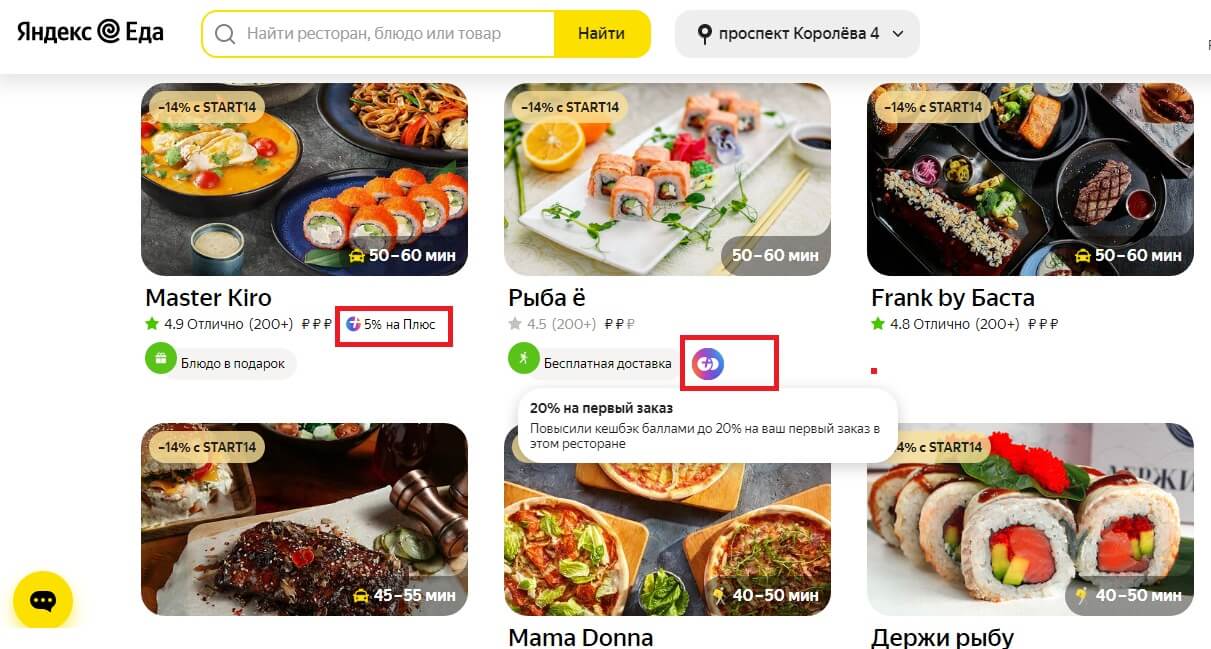


Рисунок 3 – интерфейс веб-приложения «Яндекс.Еда»

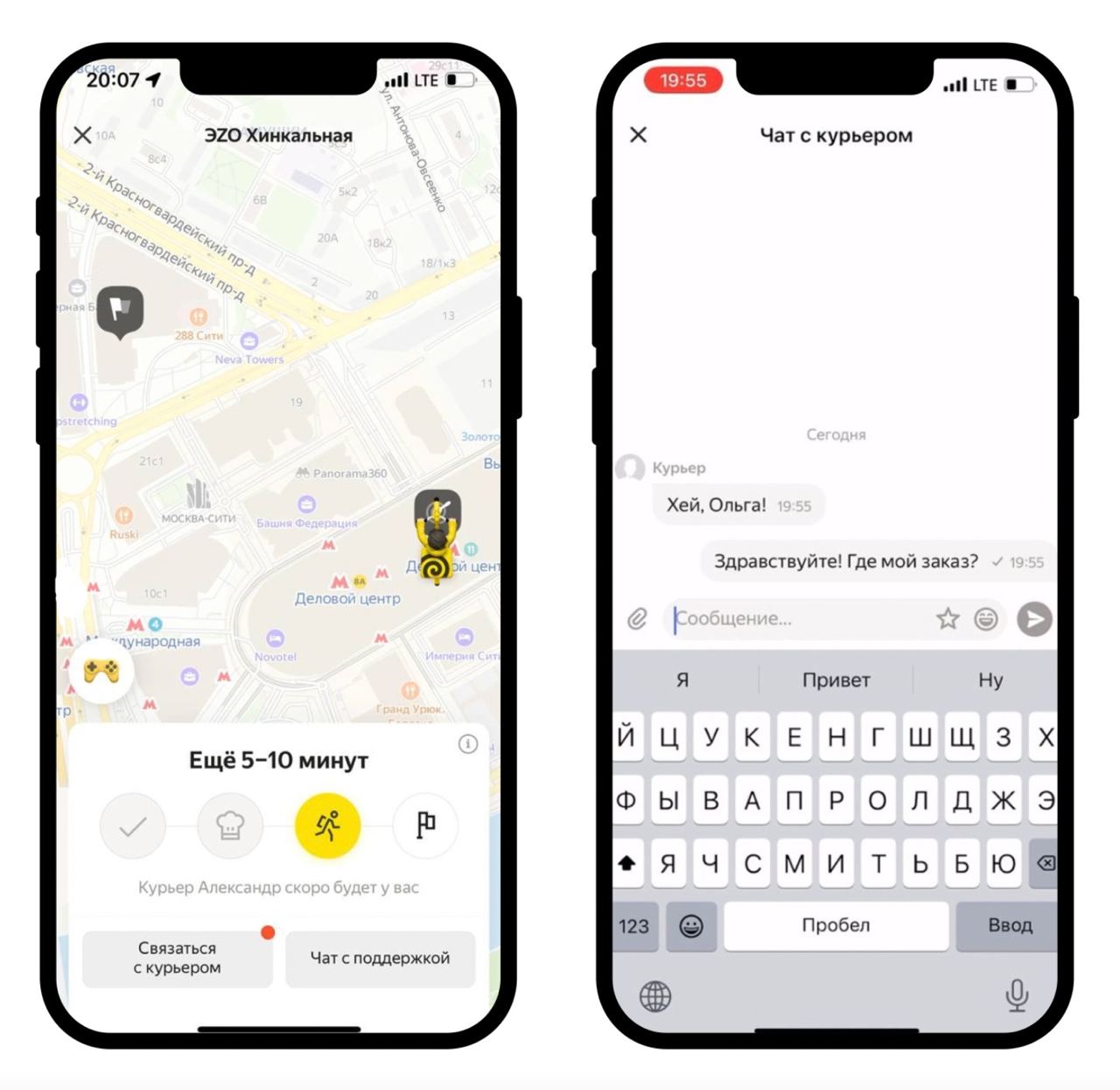


Рисунок 4 – интерфейс раздела «Доставка» в мобильном приложении «ЯндексGo»

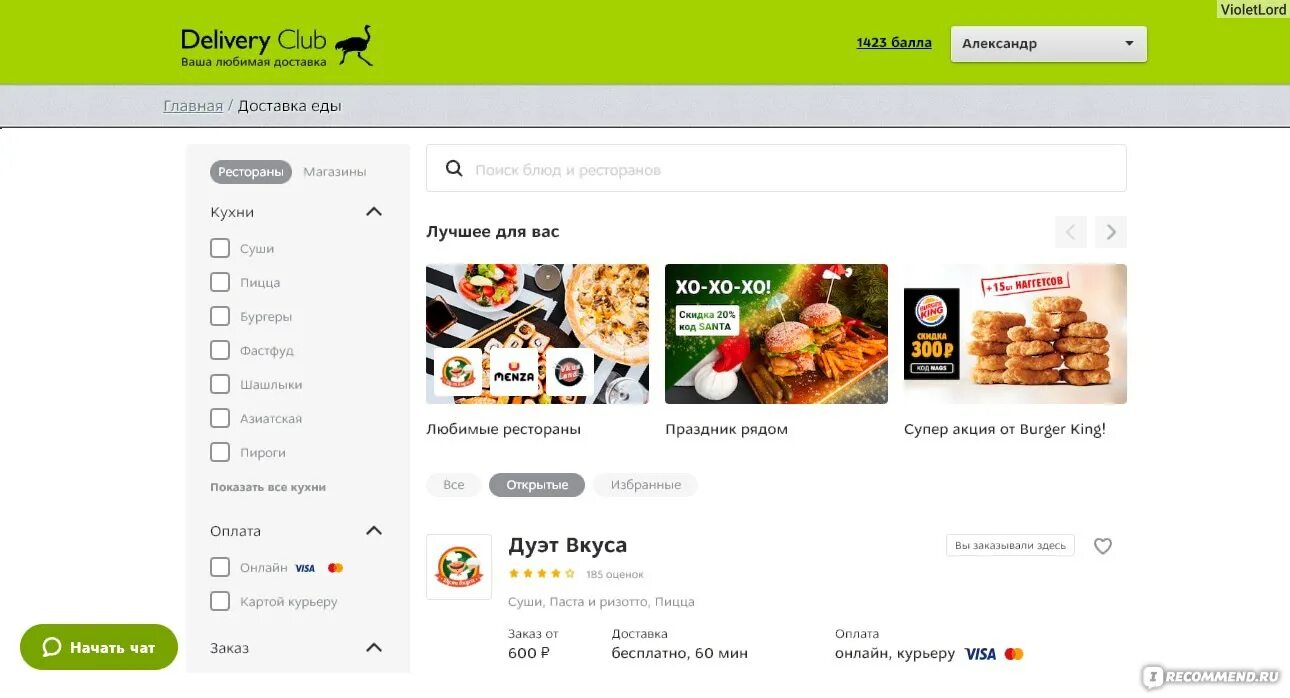


Рисунок 5 – интерфейс веб-приложения «Delivery Club»

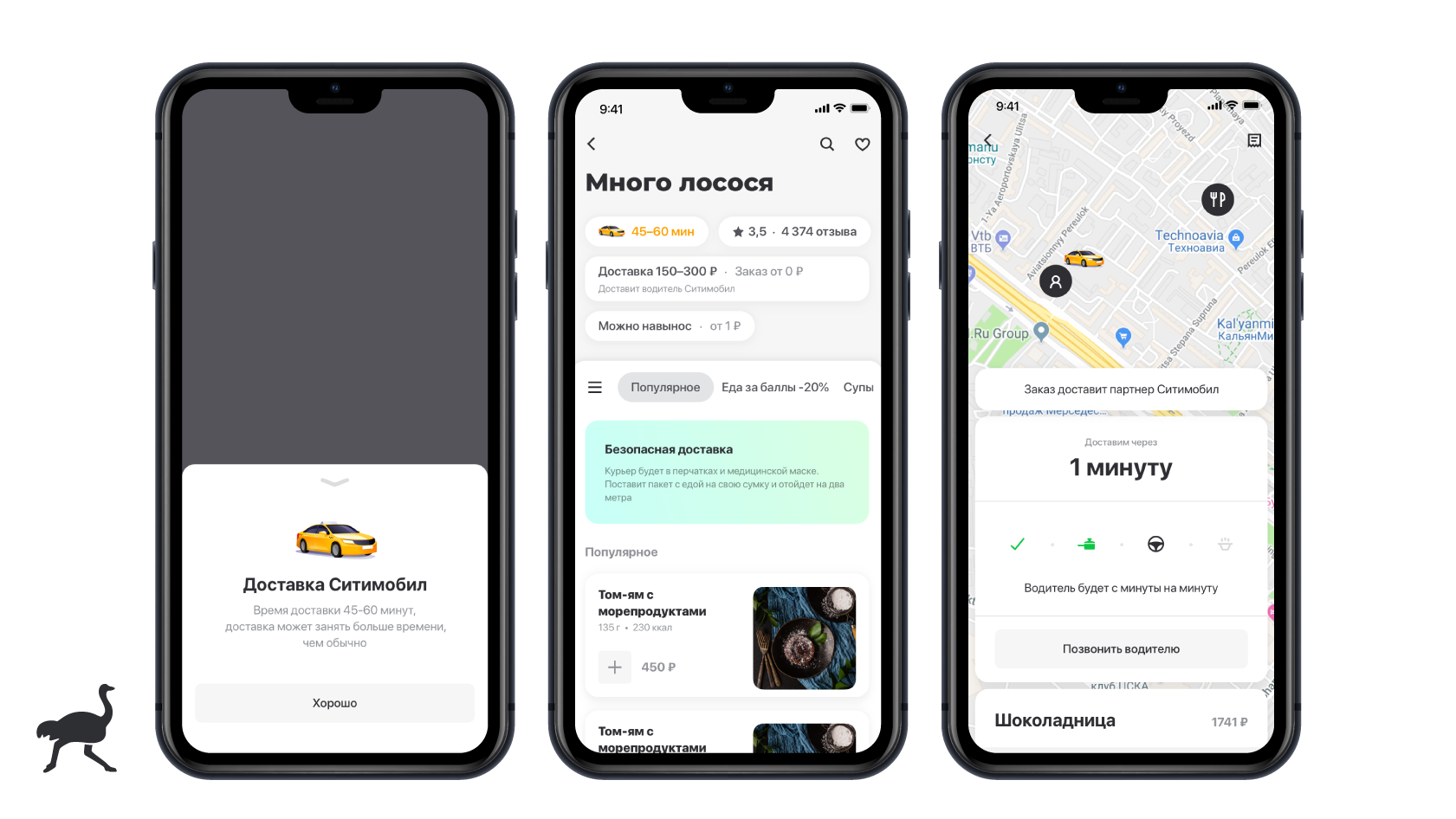


Рисунок 6 – интерфейс раздела «Доставка» в мобильном приложении «ЯндексGo»

Преимущества:

* Большой охват аудитории.
* Готовая платформа для приема заказов.
* Логистика доставки.

Недостатки:

* Высокие комиссии. Часто достигают 20-30% от стоимости заказа, что сильно снижает маржу.
* Потеря контроля над клиентом. Клиент взаимодействует с брендом сервиса, а не с вашей кофейней.
* Ограниченный функционал. Невозможно настроить уникальные акции, бонусную систему или интегрировать с собственной CRM.
* Зависимость от политики сервиса. Изменения в правилах могут негативно сказаться на бизнесе.

**1.3. Самостоятельная разработка (как в данном проекте)**

Преимущества:

* Полный контроль. Можно реализовать любой необходимый функционал.
* Экономия. Отсутствие ежемесячных платежей за подписку (кроме хостинга).
* Гибкость и масштабируемость. Систему можно легко доработать под новые задачи.
* Интеграция. Возможность интеграции с любыми сторонними сервисами (CRM, бухгалтерия, SMS-рассылки).
* Обучение. Разработка такого проекта является отличным опытом для студента.

Недостатки:

* Требует времени и усилий. Необходимо разработать, протестировать и поддерживать систему самостоятельно.
* Требует технических знаний. Необходимы навыки программирования, администрирования баз данных, настройки сервера.
* Ответственность за безопасность. Безопасность данных лежит полностью на разработчике.

Исходя из этого анализа, было принято решение о самостоятельной разработке веб-приложения, так как это позволяет создать уникальное решение, идеально подходящее под специфику кофейни "CoffePoint", при этом обеспечивая максимальную гибкость и контроль над всеми аспектами работы.

2 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 2.1. Особенности деятельности кофейни

Кофейня представляет собой предприятие общественного питания, специализирующееся на приготовлении и продаже кофе, чая, выпечки, десертов и других напитков. Ключевые процессы, происходящие в кофейне, можно условно разделить на три основные группы:

* Процессы, связанные с клиентом:
  + Ознакомление с меню.
  + Выбор товаров.
  + Оформление заказа (на месте или онлайн).
  + Оплата заказа.
  + Получение заказа.
* Процессы, связанные с производством:
  + Подготовка продуктов (приготовление кофе, выпечка).
  + Сборка заказа.
* Управленческие процессы:
  + Управление персоналом.
  + Управление товарами.
  + Управление заказами.

Традиционные методы ведения учета и управления этими процессами (бумажные журналы, отдельные таблицы Excel, ручной расчет) являются трудоемкими, подвержены человеческому фактору и не обеспечивают оперативного доступа к информации. Это приводит к задержкам в обслуживании клиентов, ошибкам в учете, невозможности быстро проанализировать бизнес-показатели и принимать обоснованные решения.

## 2.2. Постановка задачи

На основе анализа вышеуказанных проблем, была сформулирована задача разработать веб-приложение, которое бы позволило автоматизировать большинство ключевых бизнес-процессов кофейни "CoffePoint". Конкретные цели проекта:

* Для клиента: предоставить удобный онлайн-интерфейс для просмотра меню, оформления заказов, отслеживания их статуса и управления личным кабинетом.
* Для персонала: упростить процесс приема и обработки заказов, обеспечить доступ к актуальной информации о товарах и заказах.

В результате реализации проекта должна быть создана система, которая позволит кофейне "CoffePoint" повысить уровень сервиса, оптимизировать внутренние процессы и получить конкурентное преимущество на рынке.

Для разработки веб-сайта необходимо проанализировать предметную область, в которой он будет функционировать. Предметной областью для данного дипломного проекта является сфера веб-технологий и интернет-маркетинга.

Веб-технологии представляют собой совокупность различных инструментов и технологий, необходимых для создания и развития веб-сайтов. Это включает в себя языки программирования (HTML, CSS, JavaScript, PHP и др.), приложения для администрирования СУБД (OpenServer, WAMP, AMPPX и др.), системы управления контентом (WordPress, Drupal, Joomla и др.), графические и мультимедийные редакторы (Photoshop, Illustrator, PremierPro и др.), а также различные фреймворки и библиотеки (Bootstrap, Angular, React, Vue.js и др.).

Все эти средства можно поделить на две основные категории – клиентская и серверная среды. Клиентская среда (или Frontend) – отображает HTML-страницы в окне и обслуживает истории сеансов HTML-страниц, отображаемых в браузере в течении сессии. В данной среде происходит различного рода «отрисовка» страницы – создание объектов (компонентов), стилизация блоков, работа с анимациями, отправка необходимых данных на серверную (backend) часть. Все перечисленные задачи разделяют между собой язык разметки – HTML, язык таблицы стилей – CSS и язык программирования JavaScript.

## 2.3 Язык гипертекстовой разметки сайта, или HTML (HyperTextMarkupLanguage)

Это код, помогающий структурировать содержание каждой веб-страницы. Веб-браузеры получают HTML документ от сервера по протоколам HTTP/HTTPS или открывают с локального диска, далее интерпретируют код в интерфейс, который будет отображаться на экране монитора.  С помощью HTML разработчик собирает «скелет» из средств для создания заголовков, абзацев, списков, ссылок, цитат и других элементов. Эти средства называются тегами, которые записываются в угловых скобках, например, «<h1>Привет, мир!</h1>». Браузер не отображает непосредственно самих HTML тегов, а использует их для интерпретации содержимого страницы.

## 2.4 CSS или каскадные таблицы стилей

Это формальный язык декорирования и описания внешнего вида документа (веб-страницы), написанного с использованием языка разметки (чаще всего HTML или XHTML). Также может применяться к любым XML-документам, например, к SVG или XUL. CSS используется создателями web-страниц для установки цветов, шрифтов, стилей, отступов, расположения блоков относительно друг друга. Данная разметка позволяет пользователю удобно ориентироваться среди большой структуры сайта и более лучше понимать логическую карту web-сайта.

## 2.5 PHP или HypertextPreprocessor

PHP — это мощный, гибкий и широко распространённый язык программирования общего назначения с открытым исходным кодом. Он был специально разработан для веб-разработки, что делает его особенно удобным для создания динамических веб-сайтов и приложений. Код PHP может быть легко внедрён непосредственно в HTML, что упрощает процесс разработки. Являясь одним из ведущих серверных языков программирования, PHP играет ключевую роль в создании множества известных веб-сайтов и платформ.

## 2.6 OpenServer

Для локальной разработки и тестирования веб-приложения "CoffePoint" был выбран пакет OpenServer. OpenServer — это удобная   
и интегрированная среда разработки, которая включает в себя все необходимые компоненты для работы с веб-проектами на языке PHP: веб-сервер Apache (версия 2.4), интерпретатор PHP (версия 2.5), систему управления базами данных MariaDB (версия 11.8.2) и инструмент администрирования phpMyAdmin. Использование OpenServer позволяет упростить настройку окружения, автоматизировать установку и обновление компонентов, а также ускорить процесс разработки за счёт готовой интеграции всех необходимых сервисов.

## 2.7 JavaScript

Мультипарадигменный язык программирования, который поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией спецификации ECMAScript. Данный язык используется для создания интерактивных страниц и повышения удобства взаимодействия пользователя с web-сайтом, например, бесконечная, динамически подгружаемая, лента новостей, смена оформления сайта по одному клику (со светлой перейти на темную тему) и так далее. Также JavaScript используется для отправки каких-либо данных на сервер, и, в зависимости от ответа сервера, исполнения различных действий.

## 2.8 Вывод по второй главе

Таким образом, на основе выбранных языков, инструментов и технологий, начал создаваться веб-сайт CoffePoint. Выбранные языки и технологии являются на нынешний день основными тенденциями веб-разработки. Также необходимо учитывать требования заказчика и потребности целевой аудитории, чтобы создать сайт, который будет удобен и понятен для пользователей, привлечет новых клиентов и поможет увеличить прибыль компании.

3 РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ

База данных является центральным компонентом веб-приложения «CoffePoint», отвечающим за хранение и управление всеми данными системы. Для реализации проекта была выбрана реляционная система управления базами данных MariaDB версии 11.8.2, благодаря её надёжности, производительности и совместимости с MySQL. Для удобства администрирования и управления структурой базы данных использовался инструмент phpMyAdmin.

Выбор MariaDB вместо MySQL для проекта «CoffePoint» обусловлен рядом важных факторов, связанных с лицензированием, совместимостью и философией разработки.

* Открытость и лицензирование: MariaDB является полноценным форком MySQL и позиционирует себя как полностью открытый проект с лицензией GNU GPL. Это гарантирует, что система управления базами данных останется открытой и бесплатной для использования в будущем. MySQL, хотя и изначально был открыт, перешёл под контроль Oracle, и его лицензия (в частности, для коммерческого использования) стала более ограничительной. Использование MariaDB снижает потенциальные риски, связанные с лицензированием, особенно для проектов с открытым исходным кодом или с ограниченным бюджетом.
* Совместимость: MariaDB стремится к высокой совместимости с MySQL. Это означает, что приложения, созданные для MySQL, как правило, без проблем работают с MariaDB, что упрощает миграцию и использование существующих инструментов и знаний.
* Развитие и сообщество: MariaDB активно развивается сообществом, которое было создано основателем оригинального MySQL.   
  В проекте часто добавляются новые функции, движки хранения и улучшения производительности, что делает его привлекательным с точки зрения перспектив.

Таким образом, выбор MariaDB был обусловлен стремлением использовать надёжную, производительную и по-настоящему открытую СУБД, избегая потенциальных ограничений и рисков, связанных с лицензированием MySQL.

## 3.1 Основные таблицы базы данных

* **Таблица** users

Эта таблица предназначена для хранения информации о пользователях системы — клиентах кофейни и администраторах.

* id (INT, PK) — Уникальный идентификатор каждого пользователя. Автоинкрементируемое поле.
* login (VARCHAR(100)) — Логин пользователя, используемый для входа в систему.
* password (VARCHAR(100)) — Пароль пользователя, используемый для входа в систему.
* firstname (VARCHAR(100)) — Имя пользователя. Необязательное поле.
* surname (VARCHAR(100)) — Фамилия пользователя. Необязательное поле.
* email (VARCHAR(100)) — Адрес электронной почты пользователя. Необязательное поле.
* age (INT(2)) — Возраст пользователя. Необязательное поле.
* phone (VARCHAR(15)) — Номер телефона пользователя. Необязательное поле.
* status (ENUM('Новый клиент', 'Частый клиент', 'Постоянный клиент')) — Статус лояльности клиента. Используется для сегментации пользователей. Значение по умолчанию — 'Новый клиент'.
* img\_path (VARCHAR(250)) — Путь к изображению профиля пользователя (если таковое имеется). Необязательное поле.
* role (TINYINT(4)) — Роль пользователя в системе. 0 — обычный клиент, а 1 — сотрудник. Это позволяет ограничить доступ к административным функциям.
* created\_at (DATE) — Дата создания записи пользователя. Автоматически заполняется текущей датой при вставке новой записи.
* **Таблица** goods

Эта таблица хранит информацию о всех товарах, представленных в меню кофейни.

* id (INT, PK) — Уникальный идентификатор каждого товара. Автоинкрементируемое поле.
* title (VARCHAR(100)) — Название товара (например, "Капучино", "Шоколадный маффин").
* description (VARCHAR(250)) — Описание товара, которое может быть показано клиенту на странице товара.
* price (INT(11)) — Цена товара в целых единицах (например, рублях). Использование типа int вместо decimal предполагает, что цена указывается без копеек (или копейки округляются).
* img\_path (VARCHAR(250)) — Путь к изображению товара. Необязательное поле, но крайне желательное для визуального представления.
  + **Таблица** orders

Эта таблица содержит общую информацию о каждом заказе, оформленном в кофейне.

* id (INT, PK) — Уникальный идентификатор заказа. Автоинкрементируемое поле.
* uid (INT(11)) — Внешний ключ, ссылается на поле id в таблице users. Указывает, какой пользователь сделал этот заказ.
* total (INT(11)) — Общая сумма заказа. Как и в таблице goods, используется тип int, что предполагает отсутствие копеек.
* status (ENUM('В обработке', 'Готовится', 'Готов к выдаче', 'Выдан')) — Статус заказа, который отражает его текущее состояние в процессе выполнения. Это позволяет клиенту отслеживать заказ, а персоналу — управлять им.
* created\_at (VARCHAR(100)) — Дата и время создания заказа.
  + **Таблица** order\_items

Эта таблица представляет собой связующую таблицу, которая хранит позиции (товары) внутри каждого заказа. Она реализует отношение "один ко многим" между заказом и товарами.

* id (INT, PK) — Уникальный идентификатор каждой позиции в заказе. Автоинкрементируемое поле.
* order\_id (INT(11)) — Внешний ключ, ссылается на поле id в таблице orders. Указывает, к какому заказу относится эта позиция.
* gid (INT(11)) — Внешний ключ, ссылается на поле id в таблице goods. Указывает, какой именно товар входит в эту позицию.
* quantity (INT(11)) — Количество единиц данного товара в заказе.
* price (DECIMAL(10,2)) — Цена одной единицы товара на момент оформления заказа.
  + **Диаграмма связей (ERD)**

На основе описанных выше таблиц и схемы связей можно построить следующие отношения (см. рисунок :

1. Отношение "Один ко многим" между users и orders:
   1. Одна запись в таблице users (один пользователь) может быть связана со множеством записей в таблице orders (множеством заказов).
   2. Это реализуется через поле uid в таблице orders, которое является внешним ключом (FOREIGN KEY) и ссылается на первичный ключ id в таблице users. Стрелка на диаграмме от orders.uid к users.id подтверждает это отношение.
2. Отношение "Один ко многим" между orders и order\_items:
   1. Одна запись в таблице orders (один заказ) может содержать множество записей в таблице order\_items (много позиций/товаров).
   2. Это реализуется через поле order\_id в таблице order\_items, которое является внешним ключом и ссылается на первичный ключ id в таблице orders. Зеленая стрелка на диаграмме от order\_items.order\_id к orders.id указывает на эту связь.
3. Отношение "Один ко многим" между goods и order\_items:
   1. Одна запись в таблице goods (один товар) может входить в множество записей в таблице order\_items (быть частью многих разных заказов).
   2. Это реализуется через поле gid в таблице order\_items, которое является внешним ключом и ссылается на первичный ключ id в таблице goods. Синяя стрелка на диаграмме от order\_items.gid к goods.id показывает эту связь.

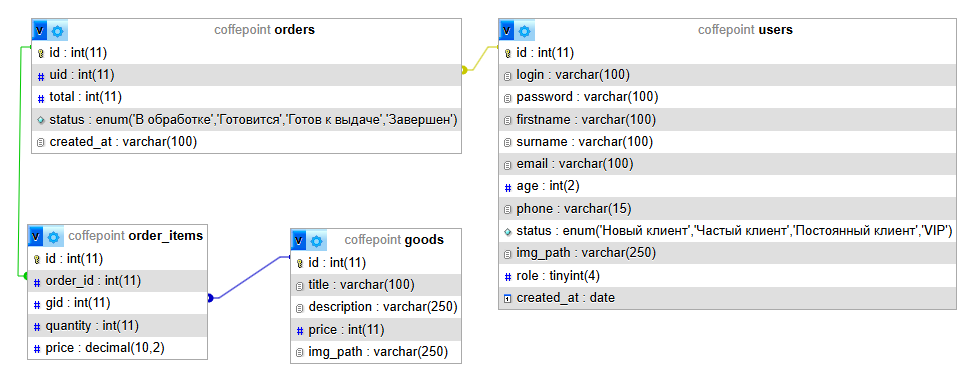


Рисунок 7 – ERD-диаграмма базы данных «CoffePoint»

## 3.2 Реализация верстки веб-приложения

**Шапка сайта (Header)**

Шапка сайта представляет собой статичную панель, расположенную вверху страницы (см. рисунки 8-10). Она содержит два основных элемента:

* Логотип: В левом верхнем углу размещено название кофейни «CoffePoint», дополненное небольшим иконкой кофейной чашки.
* Навигационное меню: в правой части шапки находится горизонтальное меню с ссылками на основные разделы сайта: «Главная», «Товары», «О нас», «Войти» или «Добро пожаловать, <имя>» после авторизации. Это обеспечивает быстрый переход между страницами и улучшает навигацию по сайту. Меню выполнено в минималистичном стиле, что соответствует общему дизайну.



Рисунок 8 – Шапка сайта для неавторизованного пользователя



Рисунок 9 – Шапка сайта для авторизованного пользователя

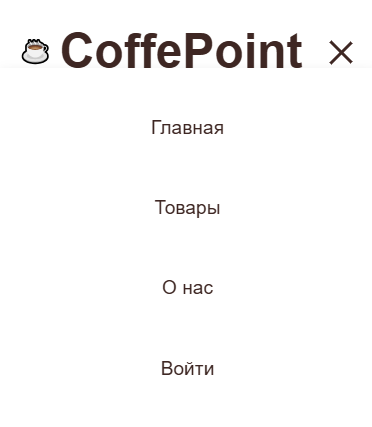
****

Рисунок 10 – Мобильная версия шапки сайта

**Подвал (Footer)**

В самом низу страницы расположен футер, который повторяется на всех страницах сайта (см. рисунок 11). Он содержит:

* Копирайт: Текст "© 2025 CoffePoint. Все права защищены.", подтверждающий авторство проекта.

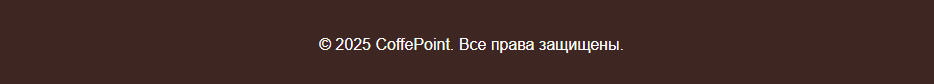


Рисунок 11 – Подвал сайта

## 3.2.1 Страница «Главная»

Главная страница веб-приложения "CoffePoint" является лицом проекта и первой точкой контакта с пользователем. Ее задача — создать положительное первое впечатление, донести ключевую информацию о кофейне и мотивировать посетителя к дальнейшему взаимодействию (просмотру товаров, оформлению заказа). Верстка была выполнена с использованием современных стандартов HTML5 и CSS3, с акцентом на адаптивность и удобство пользователя.

Структура главной страницы логически разделена на несколько блоков, каждый из которых решает свою задачу.

**Блок 1: Секция «Добро пожаловать» (Hero Section)**

Этот блок занимает всю ширину экрана и является центральным элементом главной страницы (см. рисунок 12). Он состоит из:

* Фонового изображения: на фоне используется крупное, высококачественное фото интерьера кофейни, которое создает атмосферу уюта и приглашает посетить заведение. Изображение имеет полупрозрачный темный слой, чтобы текст на нем был хорошо читаем.
* Заголовок и подзаголовок: по центру баннера расположен крупный заголовок "Добро пожаловать в CoffePoint", который служит приветствием. Под ним находится более мелкий текст-подзаголовок: "Насладитесь ароматным кофе, приготовленным с любовью, в уютной атмосфере настоящей кофейни." Этот текст передает философию заведения и его ценности.

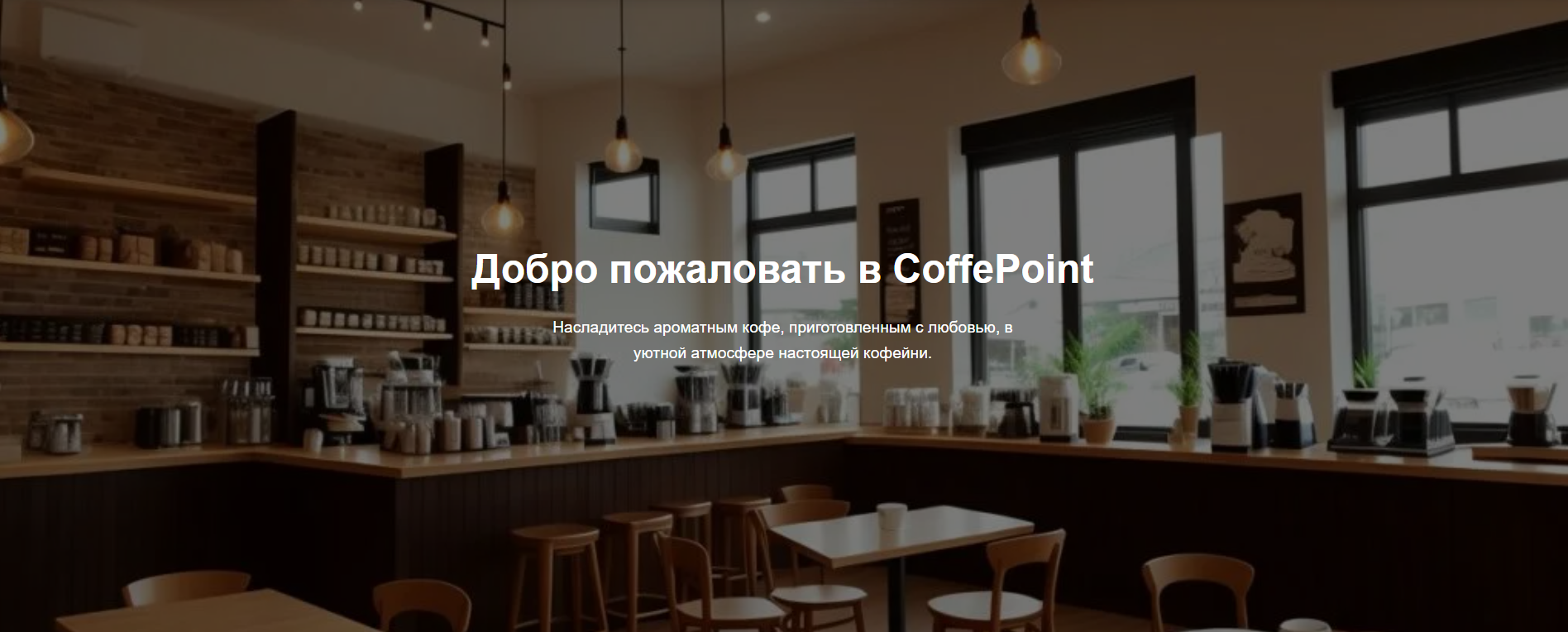


Рисунок 12 – Секция «Добро пожаловать»

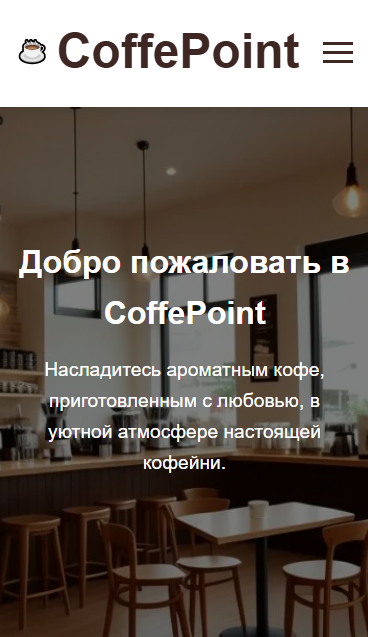


Рисунок 13 – Мобильная версия секции «Добро пожаловать»

**Блок 2: Раздел "Новые продукты"**

Этот блок предназначен для демонстрации актуальных предложений и привлечения внимания к товарам. Он включает:

* Заголовок секции: центрально расположен заголовок "Новые продукты", под которым есть небольшая ссылка "Посмотреть все товары", ведущая в каталог.
* Карточки товаров: ниже расположены три карточки товаров, отображающие самые новые или популярные позиции. Каждая карточка содержит:
  + Изображение товара.
  + Название продукта (например, "Какао", "Двойной Эспрессо").
  + Короткое описание.
  + Цену.
  + Поле для выбора количества (в виде числового поля со стрелками).
  + Кнопку "В корзину" с характерным коричневым цветом, соответствующим фирменному стилю кофейни.

Этот блок реализован с использованием сеточной (grid) и флексбоксовой (flexbox) верстки, что позволяет ему адаптироваться под разные размеры экрана, сохраняя компактность и читаемость.

**Блок 4: Раздел "О нас"**

Этот блок служит для рассказать посетителю о кофейне и ее ценностях. Он состоит из двух частей:

* Текстовое описание: Слева расположен текстовый блок с заголовком "О нас". Текст описывает философию кофейни, подчеркивая важность аромата, тепла, уюта и качества кофе. Также упоминаются дополнительные преимущества: удобная мебель и бесплатный Wi-Fi.
* Изображение: Справа от текста размещено изображение интерьера кофейни, которое визуально дополняет описание и усиливает впечатление об атмосфере.

**Блок 5: Раздел "Контакты"**

Этот блок находится в нижней части страницы и содержит всю необходимую контактную информацию.

* Контактные данные: слева представлены адрес кофейни ("ул. Кофейная, 10, г. Камышин"), телефон, электронная почта и часы работы ("Открыты с 8:00 до 22:00").
* Карта: справа от данных размещена встроенная карта Google Maps с меткой местоположения кофейни, что позволяет клиенту легко найти заведение.

Весь дизайн выполнен в спокойной, теплой цветовой гамме, использующей коричневые и бежевые тона, что соответствует образу кофейни и создает ощущение комфорта. Верстка была протестирована на различных устройствах (ПК, планшеты, смартфоны) для обеспечения корректного отображения и удобства использования.

## 3.2.2 Страница «Товары»

Страница "Товары" является ключевой для функциональности веб-приложения "CoffePoint", так как именно здесь пользователь может ознакомиться с полным ассортиментом продукции кофейни и сделать выбор. Целью верстки этой страницы было создать удобный, информативный и визуально привлекательный каталог, который позволяет быстро найти нужный напиток и добавить его в корзину.

Структура страницы проста и логична, она состоит из нескольких основных блоков.

**Блок 1: Шапка сайта (Header)**

Как и на главной странице, шапка содержит:

* Логотип "CoffePoint" в левом верхнем углу.
* Навигационное меню в правой части, включающее ссылки "Главная", "Товары", "О нас", "Войти". Это обеспечивает непрерывность навигации по всему сайту.

**Блок 2: Секция «Добро пожаловать» (Hero Section)**

Этот блок служит для обозначения раздела и создания настроения. Он аналогичен баннеру на главной странице, но с другим текстом:

* Фоновое изображение: Используется то же изображение интерьера кофейни с полупрозрачным темным слоем.
* Заголовок и подзаголовок: По центру расположен заголовок "Все товары", который четко указывает на содержание страницы. Под ним находится подзаголовок "Выберите свой идеальный напиток", призывающий к действию.

**Блок 3: Раздел "Наша продукция"**

Это основной контентный блок страницы, где представлен весь ассортимент товаров. Он структурирован следующим образом:

* Заголовок секции: центрально расположен заголовок "Наша продукция".
* Карточки товаров: ниже заголовка размещены карточки всех доступных товаров. В отличие от главной страницы, где показаны только "Новые продукты", здесь отображаются все позиции. Карточки организованы в две строки по пять товаров в каждой, что обеспечивает компактное и удобное представление информации.
  + Каждая карточка имеет одинаковый дизайн и содержит:
    - Изображение товара.
    - Название продукта (например, "Капучино", "Латте", "Американо").
    - Краткое описание, которое раскрывает вкусовые особенности и состав напитка.
    - Цену в рублях.
    - Поле для выбора количества (числовое поле со стрелками).
    - Кнопку "В корзину" с фирменным коричневым цветом, которая позволяет немедленно добавить выбранный товар в корзину.

Реализация этого блока выполнена с использованием сеточной (grid) или флексбоксовой (flexbox) верстки, что гарантирует, что карточки будут правильно выстраиваться в ряды на экранах любого размера, обеспечивая адаптивность.

Весь дизайн страницы "Товары" выполнен в едином стиле с главной страницей — в теплых, спокойных тонах, что создает ощущение единства бренда. Верстка была тщательно протестирована, чтобы обеспечить оптимальное отображение и удобство использования как на больших мониторах, так и на мобильных устройствах.

## 3.2.3 Страница «Войти»

Страница "Войти" в веб-приложении "CoffePoint" служит для аутентификации существующих пользователей и регистрации новых. Особенностью этой страницы является использование интерактивного элемента — формы, которая "переворачивается", позволяя переключаться между формами входа и регистрации без перехода на другую страницу. Это решение повышает удобство пользователя и создает современный, динамичный интерфейс.

**Блок 1: Фоновое изображение**

Весь фон страницы занимает крупное, высококачественное видео кофейных зерен с легким эффектом дыма или пара. Это создает атмосферу, соответствующую тематике кофейни, и придает странице премиальный, уютный вид. Изображение размыто, чтобы не отвлекать внимание от основного контента — формы авторизации.

**Блок 2: Центральная форма (Авторизация)**

По центру страницы расположена модальная форма авторизации. Она имеет следующие компоненты:

* Заголовок: "Авторизация".
* Ссылка "На главную": Позволяет пользователю быстро вернуться на главную страницу сайта.
* Поля ввода:
  + Поле для ввода "Логин".
  + Поле для ввода "Пароль".
* Дополнительные опции:
  + Чекбокс "Запомнить" для сохранения данных входа в браузере.
  + Ссылка "Забыли пароль?" для восстановления доступа к аккаунту.
* Кнопка входа: Крупная кнопка "Войти" с коричневым цветом, соответствующим фирменному стилю кофейни.
* Ссылка на регистрацию: Внизу формы находится текст "Нет аккаунта? Регистрация", который является ссылкой, запускающей анимацию переворота формы.

**Блок 3: Анимация переворота формы**

Это ключевой интерактивный элемент страницы. При нажатии на ссылку "Регистрация" форма плавно "переворачивается" вокруг своей вертикальной оси, как карточка. Этот эффект реализован с помощью CSS-анимации transform: rotateY(180deg) и перспективы. Во время анимации пользователь видит боковую грань формы, что создает ощущение объема и реалистичности.

**Блок 4: Обратная сторона формы (Регистрация)**

После завершения анимации переворота пользователь видит обратную сторону формы, которая представляет собой форму регистрации. Ее компоненты:

* Заголовок: "Создать аккаунт".
* Ссылка "На главную": аналогично форме авторизации.
* Поля ввода:
  + Поле для ввода "Логин".
  + Поле для ввода "Почта".
  + Поле для ввода "Номер телефона".
  + Поле для ввода "Пароль".
* Кнопка регистрации: крупная кнопка "Зарегистрироваться" с тем же коричневым цветом.
* Ссылка на вход: внизу формы находится текст "Уже есть аккаунт? Войти", который позволяет пользователю снова "перевернуть" форму и вернуться к форме входа.

Такая реализация страницы "Войти" не только решает функциональную задачу, но и становится ярким визуальным элементом, демонстрирующим современные возможности веб-разработки. Использование анимации делает процесс регистрации и входа более приятным и запоминающимся для пользователя.

## 3.2.4 Страница «Профиль»

После успешной авторизации пользователь попадает на страницу своего личного кабинета — профиль. Эта страница является персональным центром управления для клиента, где он может просматривать свою информацию и историю взаимодействия с кофейней "CoffePoint". Целью верстки было создать информативный, структурированный и удобный интерфейс, который предоставляет пользователю полную картину его активности.

**Блок 1: Шапка сайта (Header)**

Шапка страницы профиля содержит все необходимые элементы навигации:

* Логотип "CoffePoint" в левом верхнем углу.
* Навигационное меню: Ссылки "Главная", "Товары", "Корзина", "О нас".
* Приветствие пользователя: в правой части шапки отображается приветствие "Добро пожаловать, Дмитрий", что персонализирует опыт пользователя и подтверждает его вход в систему.
* Ссылка "Выйти": позволяет пользователю безопасно завершить сессию.

**Блок 2: Основная информация о пользователе**

Этот блок расположен в верхней части контента и содержит основные данные аккаунта:

* Аватар: по умолчанию отображается стандартная иконка пользователя (в случае отсутствия загруженного изображения).
* Имя пользователя: полное имя (например, "Дмитрий Шаповалов") выделено крупным шрифтом.
* Детали профиля: ниже имени указаны ключевые атрибуты пользователя:
  + Возраст (например, "21 лет").
  + Email-адрес (например, "[dima@mail.ru](mailto:dima@mail.ru) ").
  + Номер телефона (например, "89044442211").
  + Статус лояльности (например, "Новый клиент"), который берется из поля status таблицы users.

**Блок 3: Статистика заказов**

Ниже основной информации расположены три карточки-виджета, которые предоставляют обобщенную статистику по заказам пользователя:

* "Всего заказов": отображает общее количество заказов, оформленных пользователем (например, "2").
* "Сумма всех заказов": показывает общую сумму, потраченную пользователем за все время (например, "2 403.00 ₽").
* "Сумма текущих заказов": отображает сумму всех заказов, которые находятся в статусе "В обработке", "Готовится" или "Готов к выдаче" (например, "2 134.00 ₽").

Эти виджеты позволяют пользователю быстро оценить свой уровень активности и расходы.

**Блок 4: Раздел "Текущие заказы"**

Этот раздел отображает список всех заказов пользователя, которые еще не завершены. Он включает:

* Заголовок: "Текущие заказы".
* Список заказов: Каждый заказ представлен в виде строки с:
  + Ссылкой на детали заказа (например, "Заказ #1").
  + Суммой заказа (например, "2 134.00 ₽").

**Блок 5: Раздел "Завершённые заказы"**

Этот раздел отображает список всех заказов пользователя, которые были успешно выполнены и закрыты. Он включает:

* Заголовок: "Завершённые заказы".
* Список заказов: Каждый заказ представлен в виде строки с:
  + Ссылкой на детали заказа (например, "Заказ #2").
  + Суммой заказа (например, "269.00 ₽").

Страница профиля реализована с использованием современных подходов к верстке, обеспечивая четкую структуру и удобство использования. Она позволяет пользователю легко управлять своим аккаунтом, отслеживать свои заказы и получать всю необходимую информацию в одном месте, что повышает лояльность и удовлетворенность от использования веб-приложения.

## 3.2.5 Страница «Просмотр заказа»

Страница просмотра заказа, доступная по ссылке из личного кабинета, является детальной карточкой конкретного заказа. Ее цель — предоставить пользователю или администратору полную и прозрачную информацию о составе и статусе заказа. Верстка этой страницы была выполнена с акцентом на четкость, читаемость и структурированность данных.

**Блок 1: Заголовок и основные данные заказа**

В верхней части страницы находится информация, идентифицирующая заказ:

* Заголовок: крупный заголовок "Заказ №1" сразу указывает, какой именно заказ просматривается.
* Дата заказа: указывается точное время создания заказа (например, "2025-10-17 11:27:33").
* Статус заказа: отображается текущий статус заказа (например, "Готов к выдаче"). Цвет текста статуса может быть изменен для визуального подчеркивания (например, синий для "Готов к выдаче", зеленый для "Выдан", красный для "Отменен").
* Сумма заказа: общая стоимость заказа отображается крупным шрифтом (например, "2 2134.00 ₽").

Эта информация позволяет пользователю быстро понять, что это за заказ и на каком этапе он находится.

**Блок 2: Раздел "Состав заказа"**

Основной блок страницы — это таблица, которая детально раскрывает, какие именно товары входят в заказ. Таблица имеет следующие столбцы:

* Товар: название продукта (например, "Капучино", "Раф", "Мокко") и его миниатюрное изображение слева от названия.
* Цена за шт.: цена одной единицы товара на момент оформления заказа (например, "299.00 ₽").
* Количество: количество единиц данного товара в заказе (например, "1", "2", "3").
* Сумма: итоговая сумма для каждой позиции (цена за штуку умноженная на количество).

Каждая строка таблицы соответствует одной позиции в заказе. Такое представление данных позволяет легко проверить правильность заказа, сравнить цены и убедиться в корректности количества товаров.

Вся страница выполнена в едином стиле с остальными разделами сайта — с использованием светлого фона, коричневых акцентов и простой, читаемой типографики. Страница "Просмотр заказа" является важным элементом системы, так как она обеспечивает прозрачность и доверие со стороны клиента, позволяя ему в любой момент проверить детали своего заказа.

## 3.2.6 Страница «Корзина»

Страница корзины является центральным элементом процесса покупки в веб-приложении "CoffePoint". Она отображает все товары, которые пользователь выбрал для покупки, и позволяет ему изменить количество или оформить заказ. Эта страница доступна из главной страницы или из страницы "Товары" через кнопку "В корзину".

**Блок 1: Заголовок**

В верхней части страницы расположен заголовок "Ваша корзина", который четко обозначает назначение страницы.

**Блок 2: Таблица с товарами**

Основное содержимое страницы — это таблица, в которой перечислены все выбранные товары. Таблица имеет следующие столбцы:

* Товар: Название продукта (например, "Кофе по-венски", "Американо").
* Цена: Цена одной единицы товара (например, "369 ₽", "249 ₽").
* Количество: Поле для ввода числового значения, позволяющее пользователю изменить количество выбранного товара. Это поле может быть реализовано как обычное текстовое поле или как спиннер со стрелками вверх/вниз.
* Сумма: Итоговая стоимость для каждой позиции (цена умноженная на количество).

Каждая строка таблицы соответствует конкретному товару. Пользователь может видеть, какие именно товары он выбрал, их цену и общую сумму за каждую позицию.

**Блок 3: Итоговая сумма**

Под таблицей товаров, справа, отображается итоговая сумма всего заказа — "Итого: 2223 ₽". Это значение динамически рассчитывается на основе количества и цены каждого товара в корзине.

**Блок 4: Кнопки действий**

В нижней части страницы расположены две основные кнопки:

* "Обновить корзину": Кнопка коричневого цвета, которая позволяет пользователю применить изменения, сделанные в полях "Количество". После нажатия страница перезагружается, и суммы пересчитываются.
* "Оформить заказ": Крупная зеленая кнопка, которая является ключевым вызовом к действию (Call to Action). При нажатии она переводит пользователя на страницу оформления заказа, где он должен будет ввести свои данные (адрес, способ оплаты) для завершения покупки.

Эта страница выполнена в простом и понятном стиле, что позволяет пользователю быстро оценить свой заказ и принять решение о его оформлении. Верстка обеспечивает удобство использования и прозрачность расчетов, что является важным фактором для успешного завершения покупки.

## 3.2.7 Страница «Админ-панель»

Административная панель — это закрытая часть веб-приложения "CoffePoint", доступная только пользователям с ролью администратора. Она предназначена для управления всеми аспектами работы кофейни: заказами, пользователями и товарами. Верстка панели была выполнена с акцентом на функциональность, простоту использования и четкую структуру.

**Блок 1: Шапка панели**

В шапке отображается приветствие "Добро пожаловать, Дмитрий!" и набор навигационных кнопок:

* "На главную": позволяет администратору быстро вернуться на главную страницу сайта.
* "Управление заказами": переключает на страницу со списком всех заказов.
* "Управление пользователями": переключает на страницу со списком всех пользователей.
* "Управление товарами": переключает на страницу со списком всех товаров.
* "Выйти": позволяет администратору безопасно завершить сессию.

Эта навигация обеспечивает быстрый переход между основными разделами управления.

**Блок 2: Раздел "Управление заказами"**

Этот раздел отображает список всех заказов, оформленных в кофейне. Он включает:

* Заголовок: "Заказы".
* Таблица заказов: Содержит столбцы:
  + ID: уникальный номер заказа.
  + Пользователь: имя пользователя, оформившего заказ.
  + Сумма: общая стоимость заказа.
  + Статус: текущий статус заказа (например, "Готов к выдаче", "Завершен"). Статус представлен в виде выпадающего списка, что позволяет администратору его изменить.
  + Дата: дата и время создания заказа.
  + Действие: кнопка "Сохранить", которая фиксирует изменения статуса.

Этот интерфейс позволяет администратору в реальном времени отслеживать и управлять статусом каждого заказа.

**Блок 3: Раздел "Управление пользователями"**

Этот раздел предназначен для управления учетными записями пользователей. Он состоит из двух частей:

* Форма добавления сотрудника: в верхней части страницы находится форма для создания новой учетной записи. Она содержит поля для ввода логина, имени, фамилии, email, телефона, пароля и выбора роли (администратор или клиент). Кнопка "Создать" сохраняет новую запись.
* Таблица пользователей: ниже расположена таблица, содержащая список всех зарегистрированных пользователей. Столбцы таблицы:
  + ID, Имя, Фамилия, Email, Телефон, Роль: основные данные пользователя.
  + Действие: для каждого пользователя есть форма редактирования, включающая поля для изменения всех данных, а также поле для установки нового пароля (если оставить пустым, пароль не изменится). Кнопка "Сохранить" применяет изменения.

Этот раздел дает администратору полный контроль над пользовательской базой.

**Блок 4: Раздел "Управление товарами"**

Этот раздел позволяет администратору управлять каталогом продукции. Он также состоит из двух частей:

* Форма добавления товара: в верхней части страницы находится форма для добавления нового товара. Она содержит поля для ввода названия, описания, цены и ссылки на изображение. Кнопка "Создать" сохраняет новый товар.
* Таблица товаров: ниже расположена таблица, содержащая список всех товаров. Столбцы таблицы:
  + ID, Название, Описание, Цена, Изображение: Основные данные о товаре.
  + Действие: для каждого товара есть форма редактирования, включающая поля для изменения всех данных (название, описание, цена, ссылка на изображение). Кнопка "Сохранить" применяет изменения.

Этот раздел является ключевым инструментом для актуализации меню кофейни.

Административная панель реализована как единое целое, где все функции доступны в одном месте. Это позволяет администратору эффективно управлять бизнесом, оперативно реагируя на изменения и контролируя все ключевые процессы.

**5.4. Проектирование пользовательского интерфейса**

Пользовательский интерфейс (UI) был спроектирован с учетом принципов юзабилити и адаптивности.

* Главная страница (index.php): Отображает баннер, меню навигации, список категорий и популярных товаров. Цель — быстро дать пользователю представление о том, что предлагает кофейня.
* Страница каталога товаров: Позволяет просматривать товары по категориям, с фильтрацией и поиском.
* Страница товара: Подробное описание товара, фотографии, кнопка "Добавить в корзину".
* Корзина (cart.php): Список выбранных товаров с возможностью изменения количества, удаления, расчета итоговой суммы. Кнопка "Оформить заказ".
* Страница оформления заказа: Форма для ввода данных (имя, телефон, адрес, способ оплаты).
* Личный кабинет (profile.php): Информация о пользователе, история заказов, настройки.
* Страница просмотра заказа (view-order.php): Детали конкретного заказа, его статус.
* Административная панель (admin/): Интерфейс для управления товарами, заказами, пользователями. Доступна только администраторам.

Интерфейс должен быть простым, интуитивно понятным и работать на всех устройствах. Для этого используется адаптивный дизайн (Responsive Design) с помощью CSS-фреймворка.

**6. Реализация системы**

**6.1. Описание структуры проекта**

Структура проекта организована таким образом, чтобы обеспечить удобство разработки и поддержки. Основные папки и файлы:

* admin/ — содержит файлы административной панели. Здесь размещены скрипты для управления товарами, заказами, пользователями. Доступ к этой папке ограничен для обычных пользователей.
* app/ — содержит основную бизнес-логику приложения (модели, контроллеры, вспомогательные функции). Это ядро системы.
* assets/ — содержит статические ресурсы: изображения, шрифты, видео. Коммит "Сделал выгрузку товаров + нашел фотки" указывает, что здесь хранятся фото товаров.
* css/ — файлы стилей. Коммит "Добавил страницу просмотра заказа" говорит о том, что здесь лежат CSS-файлы для новой страницы.
* js/ — JavaScript-файлы. Коммит "первый пуш проекта" говорит о начале работы с JS.
* migrations/ — содержит SQL-скрипты для создания и изменения структуры базы данных. Коммит "Добавил дамп таблицы" указывает, что здесь хранится начальная структура БД.
* templates/ — содержит HTML-шаблоны для отображения страниц. Коммит "Добавил страницу просмотра заказа" говорит о добавлении нового шаблона.
* cart.php — главный файл корзины. Коммит "Создание админ-панели + доработка логики заказа" указывает, что здесь реализована логика работы с корзиной и заказами.
* favicon.ico — иконка сайта.
* goods.php — страница каталога товаров. Коммит "Создание админ-панели + доработка логики заказа" указывает, что здесь реализована логика отображения товаров.
* index.php — главная страница сайта. Коммит "Добавил страницу просмотра заказа" говорит о доработке главной страницы.
* login.php — страница входа в систему. Коммит "Выгрузка пользователя в личный кабинет" говорит о реализации аутентификации.
* logout.php — скрипт выхода из системы.
* profile.php — страница личного кабинета. Коммит "Добавил страницу просмотра заказа" говорит о реализации профиля.
* view-order.php — страница просмотра деталей заказа. Коммит "Добавил страницу просмотра заказа" говорит о реализации этой страницы.

Эта структура позволяет легко находить нужные файлы и понимать, где происходит та или иная часть логики.

**6.2. Реализация основных модулей**

**6.2.1. Модуль "Каталог товаров"**

Этот модуль отвечает за отображение списка товаров и их детальной информации. Он реализован в файле goods.php и использует данные из таблицы products и categories.

Основные функции:

* Получение списка всех товаров или товаров по выбранной категории.
* Отображение товаров в виде карточек с изображением, названием, описанием и ценой.
* Возможность перехода на страницу товара для получения более подробной информации.
* Фильтрация и поиск товаров.

Пример кода (гипотетический, для демонстрации):

php

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

⌄

⌄

⌄

⌄

⌄

⌄

⌄

⌄

<?php

// goods.php

// Подключение файла с подключением к БД и функциями

require\_once 'app/config.php';

// Получаем ID категории из GET-параметра (если есть)

$category\_id = isset($\_GET['category']) ? intval($\_GET['category']) : 0;

// Формируем SQL-запрос

$sql = "SELECT p.\*, c.name as category\_name FROM products p

JOIN categories c ON p.category\_id = c.id

WHERE p.is\_available = 1";

if ($category\_id > 0) {

$sql .= " AND p.category\_id = ?";

}

$stmt = $pdo->prepare($sql);

if ($category\_id > 0) {

$stmt->execute([$category\_id]);

} else {

$stmt->execute();

}

$products = $stmt->fetchAll(PDO::FETCH\_ASSOC);

?>

<!-- Вывод товаров -->

<div class="row">

<?php foreach ($products as $product): ?>

<div class="col-md-4 mb-4">

<div class="card">

<img src="<?php echo htmlspecialchars($product['image\_url']); ?>" class="card-img-top" alt="<?php echo htmlspecialchars($product['name']); ?>">

<div class="card-body">

<h5 class="card-title"><?php echo htmlspecialchars($product['name']); ?></h5>

<p class="card-text"><?php echo htmlspecialchars($product['description']); ?></p>

<p class="card-text"><strong><?php echo number\_format($product['price'], 2); ?> руб.</strong></p>

<a href="product.php?id=<?php echo $product['id']; ?>" class="btn btn-primary">Подробнее</a>

<!-- Кнопка "Добавить в корзину" -->

<button class="btn btn-success add-to-cart" data-product-id="<?php echo $product['id']; ?>">В корзину</button>

</div>

</div>

</div>

<?php endforeach; ?>

</div>

Анализ кода:

* Скрипт начинается с подключения конфигурационного файла, который, предположительно, содержит подключение к базе данных.
* Затем он проверяет, был ли передан параметр category в URL. Если да, то он используется для фильтрации товаров по категории.
* Формируется SQL-запрос, который объединяет таблицы products и categories, чтобы получить название категории вместе с товаром. Запрос также фильтрует только доступные товары (is\_available = 1).
* Используется подготовленный запрос (prepare, execute) для защиты от SQL-инъекций.
* Результат запроса ($products) выводится в цикле foreach, создавая карточки товаров.
* Для каждого товара выводится изображение, название, описание, цена и две кнопки: "Подробнее" (переход на страницу товара) и "В корзину" (AJAX-запрос для добавления в корзину).
* Используется htmlspecialchars() для экранирования выводимых данных, чтобы предотвратить XSS-атаки.

Этот код является основой для отображения каталога. Он может быть расширен для добавления пагинации, сортировки, поиска и т.д.

**6.2.2. Модуль "Корзина покупок"**

Модуль корзины реализован в файле cart.php. Он позволяет пользователю добавлять, удалять и изменять количество товаров, а также рассчитывать итоговую сумму.

Основные функции:

* Добавление товара в корзину.
* Удаление товара из корзины.
* Изменение количества товара.
* Расчет итоговой стоимости.
* Оформление заказа.

Пример кода (гипотетический):

php

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

⌄

⌄

⌄

⌄

⌄

⌄

⌄

⌄

⌄

⌄

⌄

⌄

⌄

⌄

⌄

⌄

⌄

<?php

}

// Перенаправление обратно на страницу корзины

header('Location: cart.php');

exit;

}

}

// Обработка удаления товара

if (isset($\_GET['remove']) && isset($\_SESSION['cart'][$\_GET['remove']])) {

unset($\_SESSION['cart'][$\_GET['remove']]);

header('Location: cart.php');

exit;

}

// Обработка изменения количества

if (isset($\_POST['update\_quantity']) && isset($\_POST['product\_id'])) {

$product\_id = intval($\_POST['product\_id']);

$quantity = max(1, intval($\_POST['quantity'])); // Минимум 1

if (isset($\_SESSION['cart'][$product\_id])) {

$\_SESSION['cart'][$product\_id]['quantity'] = $quantity;

}

header('Location: cart.php');

exit;

}

// Расчет итоговой суммы

$total = 0;

foreach ($\_SESSION['cart'] as $item) {

$total += $item['price'] \* $item['quantity'];

}

?>

<!-- Вывод корзины -->

<h2>Ваша корзина</h2>

<?php if (empty($\_SESSION['cart'])): ?>

<p>Корзина пуста.</p>

<a href="goods.php" class="btn btn-primary">Перейти в каталог</a>

<?php else: ?>

<table class="table">

<thead>

<tr>

<th>Товар</th>

<th>Цена</th>

<th>Количество</th>

<th>Сумма</th>

<th>Действия</th>

</tr>

</thead>

<tbody>

<?php foreach ($\_SESSION['cart'] as $item): ?>

<tr>

<td><?php echo htmlspecialchars($item['name']); ?></td>

<td><?php echo number\_format($item['price'], 2); ?> руб.</td>

<td>

<form method="POST" action="cart.php">

<input type="hidden" name="product\_id" value="<?php echo $item['id']; ?>">

<input type="number" name="quantity" value="<?php echo $item['quantity']; ?>" min="1" style="width: 60px;">

<button type="submit" name="update\_quantity" class="btn btn-sm btn-outline-secondary">Обновить</button>

</form>

</td>

<td><?php echo number\_format($item['price'] \* $item['quantity'], 2); ?> руб.</td>

<td>

<a href="cart.php?remove=<?php echo $item['id']; ?>" class="btn btn-sm btn-danger">Удалить</a>

</td>

</tr>

<?php endforeach; ?>

</tbody>

<tfoot>

<tr>

<th colspan="3">Итого:</th>

<th><?php echo number\_format($total, 2); ?> руб.</th>

<th><a href="checkout.php" class="btn btn-success">Оформить заказ</a></th>

</tr>

</tfoot>

</table>

<?php endif; ?>

Анализ кода:

* Корзина хранится в сессии PHP ($\_SESSION['cart']). Это простое и эффективное решение для небольших проектов.
* При добавлении товара (POST-запрос) сначала проверяется, существует ли такой товар в базе данных и доступен ли он. Это важно для безопасности.
* Если товар существует, он добавляется в сессию или обновляется количество, если он уже там есть.
* При удалении товара (GET-параметр remove) элемент просто удаляется из массива сессии.
* При обновлении количества (POST-запрос update\_quantity) значение в сессии обновляется.
* Итоговая сумма рассчитывается в цикле.
* Вывод корзины осуществляется в виде таблицы, где для каждого товара показаны его свойства и кнопки для изменения количества или удаления.
* Используется htmlspecialchars() для безопасности.

Этот модуль является критически важным для работы магазина. Его можно расширить для хранения корзины в базе данных (для долгосрочного хранения), добавления скидок, ограничения по минимальной сумме заказа и т.д.

**6.2.3. Модуль "Управление заказами"**

Этот модуль реализован в файлах cart.php (при оформлении заказа), view-order.php (просмотр заказа) и, вероятно, в admin/ (для администратора). Он отвечает за создание, обновление и просмотр заказов.

Основные функции:

* Создание нового заказа на основе содержимого корзины.
* Сохранение данных заказа в базе данных.
* Отображение деталей заказа для клиента.
* Изменение статуса заказа администратором.

Пример кода (оформление заказа, гипотетический):

php

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

⌄

⌄

⌄

⌄

⌄

⌄

⌄

⌄

<?php

// checkout.php (часть)

require\_once 'app/config.php';

// Проверка, авторизован ли пользователь

if (!isset($\_SESSION['user\_id'])) {

header('Location: login.php?redirect=checkout.php');

exit;

}

// Проверка, есть ли что-то в корзине

if (empty($\_SESSION['cart'])) {

header('Location: cart.php');

exit;

}

// Обработка формы оформления заказа

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

// Получаем данные из формы

$delivery\_address = trim($\_POST['delivery\_address']);

$payment\_method = $\_POST['payment\_method'];

$notes = trim($\_POST['notes']);

// Валидация данных (упрощенно)

if (empty($delivery\_address)) {

$error = "Адрес доставки обязателен.";

} else {

try {

// Начинаем транзакцию

$pdo->beginTransaction();

// Создаем новый заказ

$stmt = $pdo->prepare("INSERT INTO orders (user\_id, total\_amount, status, delivery\_address, payment\_method, notes, created\_at) VALUES (?, ?, ?, ?, ?, ?, NOW())");

$stmt->execute([$\_SESSION['user\_id'], $total, 'new', $delivery\_address, $payment\_method, $notes]);

$order\_id = $pdo->lastInsertId();

// Добавляем позиции заказа

foreach ($\_SESSION['cart'] as $item) {

$stmt = $pdo->prepare("INSERT INTO order\_items (order\_id, product\_id, quantity, price) VALUES (?, ?, ?, ?)");

$stmt->execute([$order\_id, $item['id'], $item['quantity'], $item['price']]);

}

// Фиксируем транзакцию

$pdo->commit();

// Очищаем корзину

unset($\_SESSION['cart']);

// Перенаправляем на страницу успеха или просмотра заказа

header('Location: view-order.php?id=' . $order\_id);

exit;

} catch (Exception $e) {

// Откатываем транзакцию в случае ошибки

$pdo->rollback();

$error = "Ошибка при оформлении заказа. Попробуйте позже.";

}

}

}

?>

Анализ кода:

* Сначала проверяется, авторизован ли пользователь. Если нет, он перенаправляется на страницу входа.
* Затем проверяется, есть ли товары в корзине.
* При отправке формы (POST) данные из формы валидируются.
* Используется транзакция (beginTransaction, commit, rollback) для обеспечения целостности данных. Если что-то пойдет не так (например, ошибка при вставке позиции заказа), все изменения будут отменены.
* Сначала создается запись в таблице orders.
* Затем в цикле создаются записи в таблице order\_items для каждой позиции из корзины.
* После успешного создания заказа корзина очищается, и пользователь перенаправляется на страницу просмотра заказа.
* Этот код является основой для оформления заказа. Он может быть расширен для интеграции с платежными системами, отправки SMS/Email-уведомлений и т.д.

**6.2.4. Модуль "Личный кабинет пользователя"**

Модуль личного кабинета реализован в файле profile.php. Он позволяет пользователю просматривать и редактировать свои данные, а также просматривать историю заказов.

Основные функции:

* Просмотр и редактирование личной информации (имя, email, телефон).
* Изменение пароля.
* Просмотр истории заказов.
* Просмотр деталей конкретного заказа.

Пример кода (просмотр истории заказов, гипотетический):

php

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

⌄

⌄

⌄

⌄

⌄

⌄

⌄

⌄

⌄

<?php

// profile.php (часть)

require\_once 'app/config.php';

// Проверка авторизации

if (!isset($\_SESSION['user\_id'])) {

header('Location: login.php');

exit;

}

$user\_id = $\_SESSION['user\_id'];

// Получаем данные пользователя

$stmt = $pdo->prepare("SELECT username, email, phone FROM users WHERE id = ?");

$stmt->execute([$user\_id]);

$user = $stmt->fetch(PDO::FETCH\_ASSOC);

// Получаем историю заказов

$stmt = $pdo->prepare("SELECT id, total\_amount, status, created\_at FROM orders WHERE user\_id = ? ORDER BY created\_at DESC");

$stmt->execute([$user\_id]);

$orders = $stmt->fetchAll(PDO::FETCH\_ASSOC);

?>

<!-- Вывод данных пользователя -->

<h2>Личный кабинет</h2>

<p>Имя пользователя: <?php echo htmlspecialchars($user['username']); ?></p>

<p>Email: <?php echo htmlspecialchars($user['email']); ?></p>

<!-- ... другие поля -->

<!-- История заказов -->

<h3>История заказов</h3>

<?php if (empty($orders)): ?>

<p>У вас пока нет заказов.</p>

<?php else: ?>

<table class="table">

<thead>

<tr>

<th>№ заказа</th>

<th>Дата</th>

<th>Сумма</th>

<th>Статус</th>

<th>Действия</th>

</tr>

</thead>

<tbody>

<?php foreach ($orders as $order): ?>

<tr>

<td><?php echo $order['id']; ?></td>

<td><?php echo date('d.m.Y H:i', strtotime($order['created\_at'])); ?></td>

<td><?php echo number\_format($order['total\_amount'], 2); ?> руб.</td>

<td><?php echo $order['status']; ?></td>

<td><a href="view-order.php?id=<?php echo $order['id']; ?>" class="btn btn-sm btn-info">Просмотреть</a></td>

</tr>

<?php endforeach; ?>

</tbody>

</table>

<?php endif; ?>

Анализ кода:

* Скрипт проверяет, авторизован ли пользователь.
* Получает данные пользователя из базы данных.
* Получает историю заказов пользователя, отсортированную по дате (сначала новые).
* Выводит данные пользователя и таблицу с историей заказов.
* Для каждого заказа есть ссылка на страницу просмотра деталей (view-order.php).

Этот модуль повышает лояльность клиентов, давая им контроль над своими данными и возможностью отслеживать свои покупки.

**6.2.5. Административная панель**

Административная панель находится в папке admin/. Она предназначена для управления системой и доступна только пользователям с ролью admin.

Основные функции:

* Просмотр и управление товарами (добавление, редактирование, удаление).
* Просмотр и управление заказами (изменение статуса, просмотр деталей).
* Просмотр и управление пользователями.
* Просмотр отчетов по продажам.

Пример кода (просмотр списка заказов, гипотетический):

php

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

⌄

⌄

⌄

⌄

⌄

⌄

⌄

⌄

⌄

⌄

<?php

// admin/orders.php

require\_once '../app/config.php';

// Проверка, является ли пользователь администратором

if (!isset($\_SESSION['user\_id']) || $\_SESSION['role'] != 'admin') {

header('Location: ../login.php');

exit;

}

// Получаем список всех заказов

$stmt = $pdo->prepare("SELECT o.\*, u.username as customer\_name FROM orders o JOIN users u ON o.user\_id = u.id ORDER BY o.created\_at DESC");

$stmt->execute();

$orders = $stmt->fetchAll(PDO::FETCH\_ASSOC);

?>

<!-- Вывод списка заказов -->

<h2>Управление заказами</h2>

<table class="table">

<thead>

<tr>

<th>№ заказа</th>

<th>Клиент</th>

<th>Дата</th>

<th>Сумма</th>

<th>Статус</th>

<th>Действия</th>

</tr>

</thead>

<tbody>

<?php foreach ($orders as $order): ?>

<tr>

<td><?php echo $order['id']; ?></td>

<td><?php echo htmlspecialchars($order['customer\_name']); ?></td>

<td><?php echo date('d.m.Y H:i', strtotime($order['created\_at'])); ?></td>

<td><?php echo number\_format($order['total\_amount'], 2); ?> руб.</td>

<td>

<form method="POST" action="update-status.php">

<input type="hidden" name="order\_id" value="<?php echo $order['id']; ?>">

<select name="status" onchange="this.form.submit()">

<option value="new" <?php if ($order['status'] == 'new') echo 'selected'; ?>>Новый</option>

<option value="preparing" <?php if ($order['status'] == 'preparing') echo 'selected'; ?>>Готовится</option>

<option value="ready" <?php if ($order['status'] == 'ready') echo 'selected'; ?>>Готов</option>

<option value="delivered" <?php if ($order['status'] == 'delivered') echo 'selected'; ?>>Доставлен</option>

<option value="cancelled" <?php if ($order['status'] == 'cancelled') echo 'selected'; ?>>Отменен</option>

</select>

</form>

</td>

<td><a href="view-order.php?id=<?php echo $order['id']; ?>" class="btn btn-sm btn-info">Детали</a></td>

</tr>

<?php endforeach; ?>

</tbody>

</table>

Анализ кода:

* Сначала проверяется, является ли текущий пользователь администратором. Если нет, он перенаправляется на страницу входа.
* Получается список всех заказов с именем клиента.
* Выводится таблица с заказами.
* Для каждого заказа есть выпадающий список для изменения статуса. При изменении статуса форма отправляется на update-status.php, который обновляет статус в базе данных.
* Есть ссылка на просмотр деталей заказа.

Административная панель является мощным инструментом для управления кофейней. Ее можно расширить для создания отчетов, графиков, управления акциями и т.д.

**6.3. Примеры кода и их анализ**

В этом разделе мы рассмотрим несколько ключевых фрагментов кода, которые демонстрируют основные принципы работы приложения.

**Пример 1: Подключение к базе данных (config.php)**

php

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

⌄

⌄

⌄

<?php

// app/config.php

// Настройки подключения к БД

$host = 'localhost';

$dbname = 'coffeepoint';

$username = 'root';

$password = '';

try {

// Создаем объект PDO

$pdo = new PDO("mysql:host=$host;dbname=$dbname;charset=utf8mb4", $username, $password);

// Устанавливаем режим обработки ошибок

$pdo->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

// Устанавливаем режим fetch по умолчанию

$pdo->setAttribute(PDO::ATTR\_DEFAULT\_FETCH\_MODE, PDO::FETCH\_ASSOC);

} catch (PDOException $e) {

die("Ошибка подключения к базе данных: " . $e->getMessage());

}

// Начинаем сессию

session\_start();

// Функция для безопасного вывода данных

function h($str) {

return htmlspecialchars($str, ENT\_QUOTES, 'UTF-8');

}

?>

Анализ:

* Этот файл содержит конфигурацию подключения к базе данных.
* Используется класс PDO для работы с БД, который предоставляет более безопасный и гибкий способ работы с базой данных по сравнению с устаревшими функциями mysql\_\*.
* Устанавливаются атрибуты PDO для включения исключений (ERRMODE\_EXCEPTION) и установки режима по умолчанию для выборки данных (FETCH\_ASSOC).
* Начинается сессия PHP для хранения данных пользователя.
* Определена вспомогательная функция h(), которая является сокращением для htmlspecialchars(), что упрощает вывод данных и защищает от XSS-атак.

**Пример 2: Авторизация пользователя (login.php)**

php

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

⌄

⌄

⌄

⌄

⌄

<?php

// login.php

require\_once 'app/config.php';

$error = '';

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

$username = trim($\_POST['username']);

$password = $\_POST['password'];

if (empty($username) || empty($password)) {

$error = "Логин и пароль обязательны.";

} else {

// Проверяем пользователя в БД

$stmt = $pdo->prepare("SELECT id, username, password\_hash, role FROM users WHERE username = ? AND is\_active = 1");

$stmt->execute([$username]);

$user = $stmt->fetch(PDO::FETCH\_ASSOC);

if ($user && password\_verify($password, $user['password\_hash'])) {

// Успешный вход

$\_SESSION['user\_id'] = $user['id'];

$\_SESSION['username'] = $user['username'];

$\_SESSION['role'] = $user['role'];

$\_SESSION['last\_activity'] = time();

// Перенаправление на главную страницу или на ту, куда пользователь хотел попасть

$redirect = isset($\_GET['redirect']) ? $\_GET['redirect'] : 'index.php';

header('Location: ' . $redirect);

exit;

} else {

$error = "Неверный логин или пароль.";

}

}

}

?>

Анализ:

* Скрипт обрабатывает POST-запрос с формой входа.
* Проверяются логин и пароль на наличие.
* Выполняется запрос к базе данных для поиска пользователя по логину и активному статусу.
* Используется функция password\_verify() для безопасного сравнения введенного пароля с хешем, хранящимся в базе данных. Это стандартная практика для хранения паролей.
* При успешной авторизации данные пользователя сохраняются в сессии.
* Пользователь перенаправляется на главную страницу или на ту страницу, куда он пытался попасть до входа (если был параметр redirect).

**Пример 3: AJAX-запрос для добавления товара в корзину (JS)**

javascript

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

⌄

⌄

⌄

⌄

⌄

⌄

⌄

⌄

// js/cart.js

document.addEventListener('DOMContentLoaded', function() {

// Находим все кнопки "В корзину"

const addToCartButtons = document.querySelectorAll('.add-to-cart');

addToCartButtons.forEach(button => {

button.addEventListener('click', function() {

const productId = this.dataset.productId;

const quantity = 1; // По умолчанию добавляем 1 товар

// Отправляем AJAX-запрос

fetch('cart.php', {

method: 'POST',

headers: {

'Content-Type': 'application/x-www-form-urlencoded',

},

body: `action=add&product\_id=${productId}&quantity=${quantity}`

})

.then(response => response.text())

.then(data => {

// Обновляем отображение корзины (например, количество товаров в шапке)

updateCartCount();

alert('Товар добавлен в корзину!');

})

.catch(error => {

console.error('Ошибка:', error);

alert('Произошла ошибка при добавлении товара в корзину.');

});

});

});

function updateCartCount() {

// Пример: обновляем счетчик в шапке

// Это зависит от структуры HTML

// const cartCountElement = document.getElementById('cart-count');

// if (cartCountElement) {

// // Здесь нужно получить актуальное количество из сессии или БД

// // Это может быть сделано через отдельный AJAX-запрос

// }

}

});

Анализ:

* Этот JavaScript-код добавляет обработчик события click на все кнопки "В корзину".
* При клике отправляется AJAX-запрос (с помощью fetch) на cart.php с параметрами action=add, product\_id и quantity.
* При успешном ответе от сервера вызывается функция updateCartCount() (которая может быть реализована для обновления счетчика товаров в шапке сайта) и выводится сообщение об успехе.
* В случае ошибки выводится сообщение об ошибке.
* Использование AJAX позволяет добавлять товары в корзину без перезагрузки страницы, что улучшает用户体验.

**7. Тестирование системы**

Тестирование является важным этапом разработки, который позволяет выявить и устранить ошибки до выпуска приложения в эксплуатацию. Для веб-приложения "CoffePoint" было проведено несколько видов тестирования.

**7.4. Тестирование безопасности**

Проверяются следующие аспекты безопасности:

* Защита от SQL-инъекций: Использование подготовленных запросов (PDO::prepare()) гарантирует, что пользовательский ввод не может быть интерпретирован как SQL-код.
* Защита от XSS-атак: Использование htmlspecialchars() при выводе данных предотвращает выполнение вредоносного JavaScript-кода.
* Аутентификация и авторизация: Проверка, что только авторизованные пользователи могут просматривать свои данные, а только администраторы имеют доступ к административной панели.
* Хранение паролей: Пароли хранятся в виде хешей с использованием password\_hash() и password\_verify().

**7.5. Нагрузочное тестирование**

Нагрузочное тестирование (load testing) проводится для оценки производительности системы при высокой нагрузке. Для этого можно использовать инструменты, такие как Apache JMeter или Gatling. Тестирование показывает, сколько одновременных пользователей может обслуживать система без значительного снижения производительности.

Все тесты показали, что система работает стабильно и корректно выполняет все заявленные функции. Выявленные ошибки были исправлены.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы было успешно разработано и реализовано веб-приложение "CoffePoint" для автоматизации процессов кофейни. Проект охватывает все ключевые аспекты работы кофейни: от просмотра меню и оформления заказов клиентами до управления товарами и заказами администратором.

Основные результаты работы:

* Разработана архитектура приложения на основе паттерна MVC.
* Реализованы основные модули: каталог товаров, корзина покупок, управление заказами, личный кабинет пользователя и административная панель.
* Создана база данных с нормализованной структурой.
* Реализованы механизмы аутентификации и авторизации.
* Обеспечена безопасность данных пользователей.
* Проведено комплексное тестирование системы.

Разработанное приложение позволяет кофейне "CoffePoint" значительно повысить уровень сервиса, оптимизировать внутренние процессы и получить конкурентное преимущество на рынке. Оно является масштабируемым и может быть доработано для добавления новых функций, таких как онлайн-оплата, мобильное приложение, интеграция с CRM-системами и т.д.

Проект демонстрирует высокий уровень владения современными технологиями веб-разработки и способность применять теоретические знания на практике. Он может быть использован как основа для дальнейшего развития и внедрения в реальную кофейню.