# 1.3. Постановки на реализацию системы и ее компонентов

## Описание архитектуры решения.

1. Клиентский уровень:

* Реализован на основе HTML, CSS и JavaScript с использованием фреймворка, такого как React или Angular, для создания динамического и адаптивного пользовательского интерфейса.

1. Серверный уровень:

* Реализован на языке программирования, таком как Python или Java, с использованием фреймворка, такого как Django или Spring, для обработки бизнес-логики и взаимодействия с базой данных.
* Серверный уровень включает в себя API для взаимодействия с клиентским уровнем и внешними системами.

1. Уровень базы данных:

* Используется реляционная база данных, такая как MySQL, для хранения данных о товарах, заказах, пользователях и прочих сущностях.
* Реализована репликация и балансировка нагрузки для обеспечения высокой доступности и производительности.

1. Безопасность:

* Используется HTTPS для защиты передаваемых данных.

1. Мониторинг и аналитика:

* Используются инструменты мониторинга (например, Grafana) для контроля над производительностью системы.
* Интегрирована система аналитики (например, Google Analytics) для отслеживания поведения пользователей и оптимизации конверсии.

1. Масштабируемость:

* Использование облачных сервисов вычислений (например, Amazon EC2, Google Cloud Compute Engine или Microsoft Azure Virtual Machines) для легкого масштабирования ресурсов в зависимости от потребностей интернет-магазина.

1. Производительность:

* Использование сервисов кэширования для уменьшения нагрузки на базу данных и ускорения отклика системы.
* Использование сервисов объектного хранения (например, Amazon S3) для хранения и быстрого доступа к большим объемам данных, таких как изображения товаров.

1. Гибкость:

* Разделение функциональности интернет-магазина на независимые микросервисы (например, каталог товаров, корзина, оплата, доставка), что позволяет изменять и расширять функциональность без значительных затрат времени и ресурсов.
* Использование API для взаимодействия между микро сервисами и упрощения интеграции с другими сервисами.

1. Интеграция с другими сервисами:

* Использование API для интеграции интернет-магазина с различными сторонними сервисами, такими как платежные системы, службы доставки, системы управления контентом и др.

## Описание бизнес-архитектуры.

**Бизнес-стратегия:**

Цель: завоевание лидирующих позиций на рынке товаров одежды, обуви, товаров для дома и т.д., увеличение объема продаж и расширение клиентской базы.

Целевая аудитория: семьи и индивидуальные покупатели, интересующиеся качественными и современными товарами для дома.

Конкурентные преимущества: широкий ассортимент, доставка в короткие сроки, персональные скидки и бонусы для постоянных клиентов.

**Продуктовая линейка:**

Каталог товаров:мужская,женская и детская одежда и обувь, аксессуары, игрушки, косметика, мебель, посуда, текстиль, электроника и др.

Организация процессов закупки, хранения и доставки товаров от производителей и дистрибьюторов.

**Продажи и маркетинг:**

Стратегии продвижения: контекстная реклама, SEO, SMM, email-маркетинг, партнерские программы.

Процессы оформления заказов, оплаты (кредитные карты, электронные кошельки, PayPal).

**Финансы и учет:**

Управление финансовыми потоками, учет выручки, расходов, налогов и взаиморасчетов с поставщиками и клиентами.

Интеграция с платежными системами и банками для обработки платежей от клиентов.

**Технологическая платформа:**

Выбор и реализация трехуровневой клиент-серверной архитектуры с использованием веб-сервера, сервера приложений и базы данных.

Использование современных технологий и инструментов для разработки, развертывания и поддержки интернет-магазина (например, React, Node.js, MySQL).

**Управление и персонал:**

Организация процессов управления и мотивации персонала (администраторы, маркетологи, разработчики, менеджеры по продажам).

Система контроля качества и оценки эффективности работы интернет-магазина, включая анализ продаж, отзывов клиентов и статистики посетителей.

**Законодательство и стандарты:**

Соответствие законодательным требованиям и стандартам в области электронной коммерции и защиты персональных данных.

## Описание программной архитектуры

## C4 Diagram - контекстный уровень и уровень контейнеров

1. **Контекстный уровень (Context Level):**

На контекстном уровне представлена общая схема взаимодействия интернет-магазина с внешними пользователями и системами.

Основные элементы контекстного уровня:

Интернет-магазин вещей: трехуровневая клиент-серверная архитектура, включающая клиентскую часть, промежуточный слой и серверную часть.

Пользователи: покупатели, администраторы, менеджеры по продажам и другие роли, взаимодействующие с интернет-магазином.

Внешние системы: платежные системы, службы доставки, системы управления контентом, аналитические сервисы и другие сторонние сервисы, с которыми интегрируется интернет-магазин.

Взаимодействие между элементами контекстного уровня:

Пользователи взаимодействуют с интернет-магазином через веб-приложение, выполняя покупки, просматривая каталог товаров, оставляя отзывы и т.д.

Интернет-магазин взаимодействует с внешними системами для обработки платежей, оформления доставки, управления контентом и аналитики**(см.Приложение №7)**.

1. **Уровень контейнеров (Container Level):**

На уровне контейнеров представлена разбивка интернет-магазина на отдельные контейнеры (веб-приложение, мобильное приложение, сервер приложений, база данных и др.), каждый из которых выполняет определенные функции и взаимодействует с другими контейнерами через API.

Основные контейнеры:

Веб-приложение: клиентская часть интернет-магазина, реализованная на основе HTML, CSS, JavaScript и фреймворков (например, React, Angular или Vue.js).

Мобильное приложение: клиентская часть интернет-магазина для мобильных устройств, реализованная на основе соответствующих платформ (iOS, Android или универсальные приложения).

Сервер приложений: промежуточный слой, обрабатывающий бизнес-логику, управляющий сессиями пользователей, аутентификацией и авторизацией. Может быть реализован на Node.js, Java, Python и др.

База данных: серверная часть, хранящая информацию о товарах, заказах, пользователях и других сущностях интернет-магазина. Может быть реализована на MySQL.

Взаимодействие между контейнерами:

Веб-приложение и мобильное приложение взаимодействуют с сервером приложений через API (REST или GraphQL), отправляя запросы на получение и отправку данных.

Сервер приложений взаимодействует с базой данных для выполнения запросов и операций с данными**(см.Приложение №8)**.

## Описание архитектуры данных

## Концептуальное - Class Diagram, логическое и физическое - ER-diagram

Концептуальная архитектура данных, представленная в виде диаграммы классов (Class Diagram), является абстрактным представлением сущностей и их взаимосвязей в системе. Этот вид диаграммы используется на ранних этапах проектирования для определения основных объектов, их атрибутов и отношений между ними, независимо от конкретной реализации или СУБД**(см.Приложение №9).**

Логическая архитектура данных — более детальное представление, которое определяет структуру данных, включая типы данных атрибутов и точные определения связей между сущностями**(см.Приложение №10).**

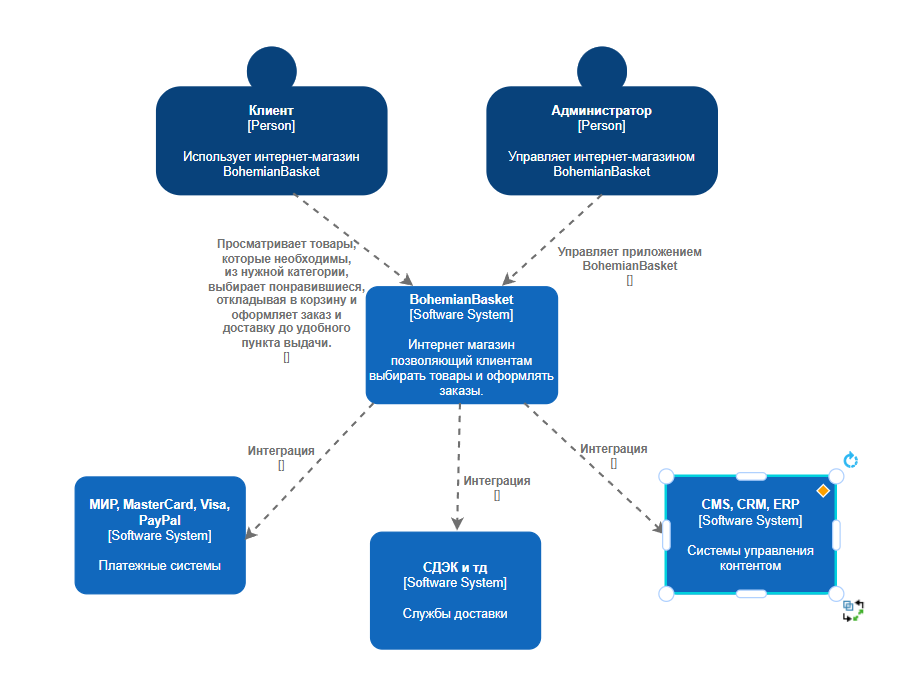
Физическая архитектура данных — это представление данных на уровне хранения, включая конкретные механизмы и структуры, используемые для хранения данных на носителях информации**(см.Приложение №11).**

**Вывод:**в результате выполнения практики были приобретены практические навыки в проектировании и разработке сложных информационных систем, основанных на трехуровневой клиент-серверной архитектуре. Также была освоена методика написания документации, включая системные требования, архитектуру и функциональность системы. Этот опыт является неотъемлемой частью профессионального роста и подготовки специалистов в области информационных технологий.

**Приложение**

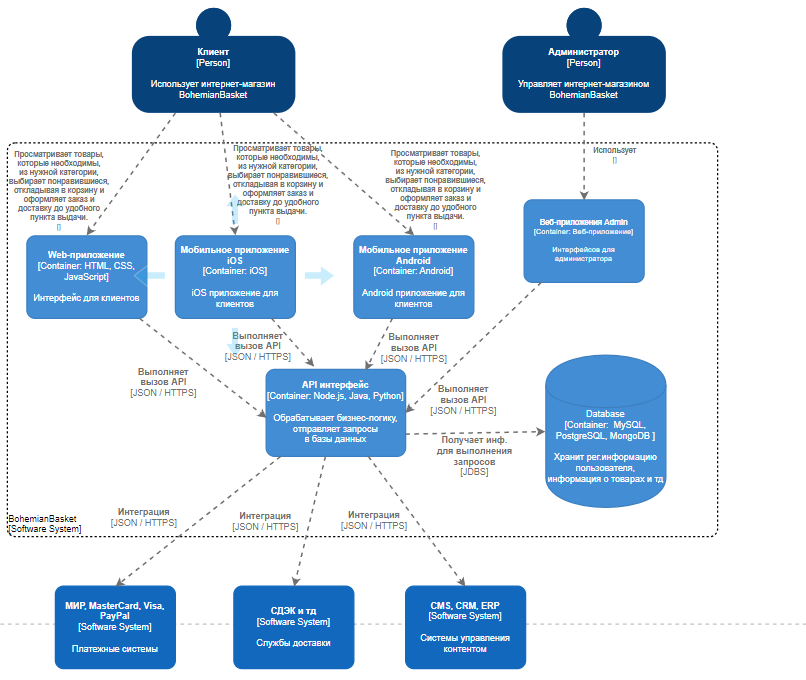
**Приложение №1.**

Контекстная диаграмма системы модели C4.



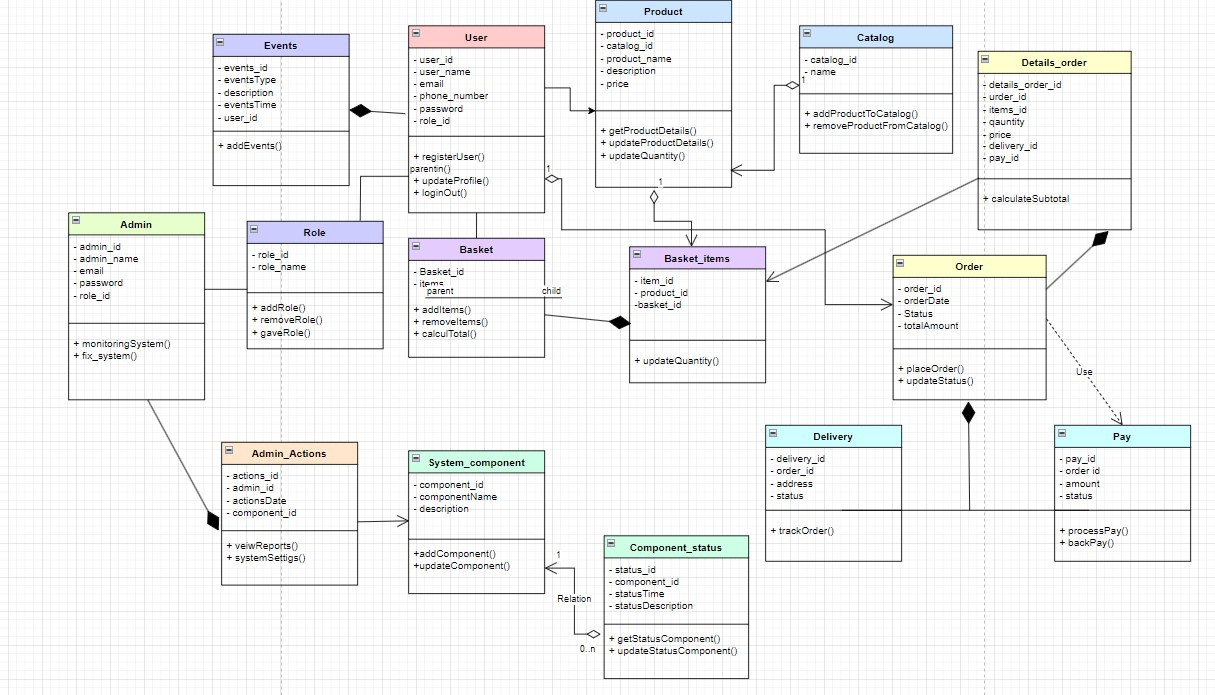
**Приложение №2.**

Схема контейнеров модели C4.



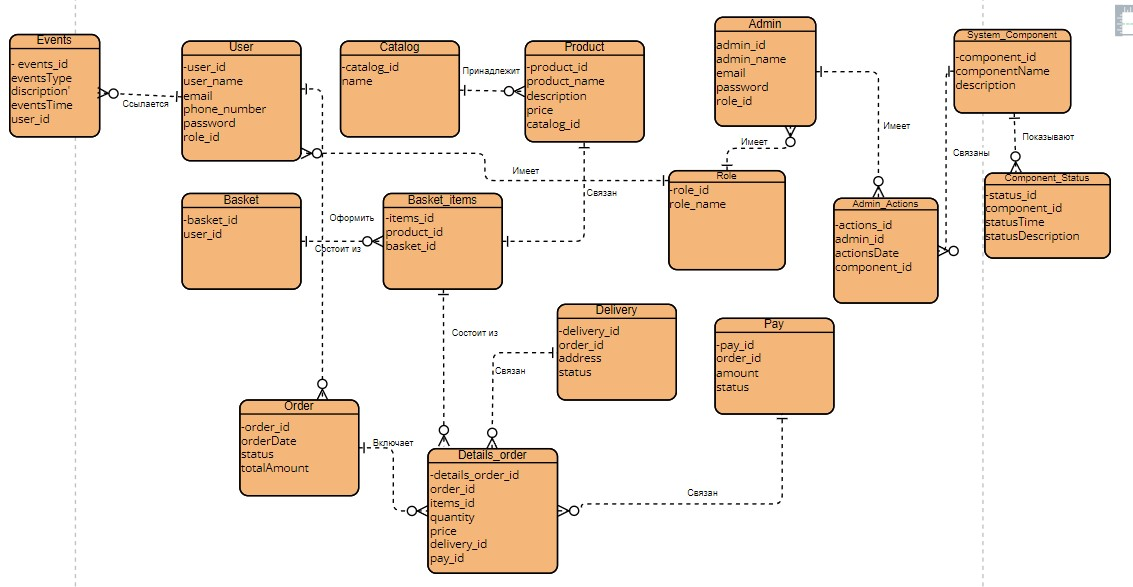
**Приложение №3.**

Class Diagram.



**Приложение №10.**

Логическая ER-diagram.



**Приложение №11.**

Физическая ER-diagram.

