Минобрнауки России

Юго-Западный государственный университет

Кафедра программной инженерии

ОТЧЕТ

о преддипломной (производственной) практике

наименование вида и типа практики

на (в) ООО «МЦОБ ОНЛАЙН-СЕРВИСЫ»

наименование предприятия, организации, учреждения

Студента 5 курса, группы ПО-12з

курса, группы

Пирцхалава Андрея Романовича

фамилия, имя, отчество

Руководитель практики от предприятия, организации, учреждения

Оценка _____

_____

должность, звание, степень

_____ _____

фамилия и. о. подпись, дата

Руководитель практики от университета

Оценка _____

к.т.н. доцент

должность, звание, степень

Чаплыгин А. А. _____

фамилия и. о. подпись, дата

Члены комиссии

_____ _____

подпись, дата фамилия и. о.

_____ _____

подпись, дата фамилия и. о.

_____ _____

подпись, дата фамилия и. о.

Курск 2026 г.

СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

БД – база данных.

ИС – информационная система.

ИТ – информационные технологии.

ПО – программное обеспечение.

РП – рабочий проект.

СУБД – система управления базами данных.

ТЗ – техническое задание.

ТП – технический проект.

UML (Unified Modelling Language) – язык графического описания для объектного моделирования в области разработки программного обеспечения.

API (Application Programming Interface) – программный интерфейс приложения, набор методов для взаимодействия между программными компонентами.

МойСклад – облачная система управления складским учётом и торговлей.

Кэш (Cache) – промежуточный буфер с быстрым доступом, содержащий информацию для ускорения работы приложения.

JSON (JavaScript Object Notation) – текстовый формат обмена данными, основанный на синтаксисе JavaScript.

1 Анализ предметной области

1.1 Характеристика предприятия и его деятельности. Тест длинного заголовка, не должен содержать переносы

Предприятие, для которого осуществляется разработка, является розничной торговой точкой, специализирующейся на продаже светодиодной продукции. Ассортимент магазина включает широкий спектр товаров: светодиодные ленты, модули, контроллеры, блоки питания, профили и комплектующие для монтажа. Деятельность предприятия характеризуется высокой оборачиваемостью товаров, разнообразием номенклатуры (сотни наименований) и необходимостью оперативного предоставления актуальной информации о наличии и ценах как сотрудникам, так и конечным клиентам.

Ключевой особенностью бизнес-процесса является работа с клиентами, требующая быстрого поиска товаров, консультирования по техническим характеристикам и предоставления точной информации о стоимости и остатках. В условиях розничной торговли время обслуживания одного клиента является критическим параметром, влияющим на общую пропускную способность магазина и уровень удовлетворенности покупателей.

1.2 Анализ проблемной ситуации и постановка задачи

В рамках оптимизации визуального оформления торгового зала руководством предприятия было принято решение об отказе от традиционных бумажных ценников. Данная мера, однако, привела к возникновению ряда операционных трудностей:

1. **Увеличение времени обслуживания:** Сотрудникам магазина стало необходимо использовать альтернативные, менее удобные каналы получения информации о товаре (личный компьютер, мобильное приложение «МойСклад»), что увеличило длительность процесса консультации.

2. **Снижение доступности информации для клиентов:** Покупатели лишились возможности самостоятельного ознакомления с базовой информа-

цией (название, цена), что повысило нагрузку на персонал и снизило комфорт совершения покупок.

3. **Риск ошибок:** Использование устаревших или несинхронизированных источников данных (распечатки, файлы Excel) могло приводить к предоставлению клиентам некорректной информации о цене или наличии товара.

Параллельно на предприятии появился технический ресурс для решения данной проблемы – телевизор с сенсорной панелью, установленный в торговом зале. Однако его потенциальная эффективность была нивелирована отсутствием специализированного программного обеспечения, способного в автоматическом режиме отображать актуальные данные из основной системы учета

Таким образом, сформировалась основная задача: разработать клиентское приложение для сенсорной панели, которое бы в режиме реального времени отображало актуальную информацию из системы складского учета, предоставляя интуитивно понятный интерфейс для быстрого поиска и просмотра данных о товарах. Ключевым требованием являлась полная автоматизация процесса обновления данных, исключающая необходимость ручного вмешательства сотрудников.

1.3 Обзор существующих систем и обоснование выбора платформы интеграции

Для интеграции с системой учета предприятия был выбран облачный сервис «МойСклад» (moysklad.ru), который уже активно использовался для управления номенклатурой, складскими остатками, ценами и заказами. Выбор был обусловлен следующими факторами:

– наличие полнофункционального REST API: «МойСклад» предоставляет детализированный API для программного доступа ко всем основным сущностям (товары, склады, цены, остатки), что соответствует требованиям по автоматизации;

– доступность и надежность: Как облачный сервис, «МойСклад» обеспечивает высокую доступность и отказоустойчивость, необходимые для работы приложения в торговом зале;

– актуальность данных: Интеграция на уровне API гарантирует, что отображаемые в приложении данные (цены, остатки) будут идентичны данным в основной учетной системе в момент запроса;

– отсутствие необходимости в промежуточном ПО: Прямое взаимодействие по HTTP(S) протоколу исключает потребность в установке дополнительного серверного программного обеспечения на стороне предприятия.

Анализ альтернативных решений (статические информационные киоски, таблицы на основе CMS, мобильные приложения «МойСклад») показал их недостаточную функциональность для решения поставленной задачи: статические решения требовали ручного обновления, мобильные приложения не были оптимизированы для большого сенсорного экрана и сценария коллективного использования.

2 Техническое задание

2.1 Основание для разработки

Полное наименование системы: «Разработка клиентского приложения для взаимодействия с системой складского учета на основе REST API».

Основанием для разработки программы является приказ ректора ЮЗГУ от «____» _____2025 г. №_____ «Об утверждении тем выпускных квалификационных работ».

2.2 Цель и назначение разработки

Основной задачей выпускной квалификационной работы является разработка и внедрение клиентского приложения для интерактивной сенсорной панели, обеспечивающего доступ к актуальной информации о товарах в торговом зале светодиодного магазина.

Посредством внедрения приложения планируется устранить существующие проблемы, связанные с отсутствием физических ценников: снизить время обслуживания клиентов, повысить доступность информации и исключить ошибки, вызванные использованием устаревших данных.

Исходя из этого, основную цель предлагается рассмотреть в разрезе двух групп подцелей:

– **Для сотрудников магазина:** создание удобного инструмента для быстрого поиска информации о товарах в процессе консультации клиентов.

– **Для клиентов:** предоставление возможности самостоятельного ознакомления с ассортиментом, ценами и наличием товаров через интуитивно понятный интерфейс.

Задачами данной разработки являются:

– создание модуля аутентификации и авторизации через API «МойСклад»;

– реализация модуля загрузки и синхронизации данных о товарах, ценах и остатках в реальном времени;

– разработка системы кэширования для оптимизации производительности и работы в условиях нестабильного интернет-соединения;

– создание адаптивного пользовательского интерфейса с двумя режимами отображения (сетка и список), оптимизированного для сенсорного ввода;

– реализация функционала поиска и фильтрации товаров по названию, артикулу, складу и категории.

2.3 Актуальность темы разработки

Современная розничная торговля требует не только наличия товара, но и обеспечения его максимальной информационной доступности. В условиях высокой конкуренции скорость и удобство получения информации становятся ключевыми факторами при принятии клиентом решения о покупке. Отказ от традиционных бумажных носителей (ценников) в пользу цифровых решений является общемировым трендом, направленным на снижение издержек, повышение эстетики пространства и оперативности обновления данных.

Разрабатываемое приложение актуально, так как решает конкретную производственную проблему, возникшую в результате модернизации торгового процесса. Интеграция с уже используемой системой учета («МойСклад») через открытое API позволяет создать легковесное, автономное клиентское решение, не требующее сложного внедрения или изменения существующих бизнес-процессов. Преимущество перед статическими дисплеями или использованием личных устройств сотрудников заключается в полной автоматизации обновления данных, высокой скорости работы и специализированном интерфейсе, оптимизированном для сенсорного взаимодействия в условиях торгового зала.

2.4 Требования пользователя к интерфейсу приложения

Приложение должно включать в себя:

– экран ввода API-токена для авторизации в системе;

– главный экран с областью поиска и фильтрации товаров;

– два режима отображения товаров: адаптивная сетка и компактный список;

– информативные карточки товаров с изображением, названием, ценой и остатком;

– диалоговые окна для детального просмотра информации и перехода на страницы товаров;

– элементы управления, адаптированные для сенсорного ввода (крупные кнопки, удобные для касания).

Композиция основного экрана приложения представлена на рисунке 2.1.

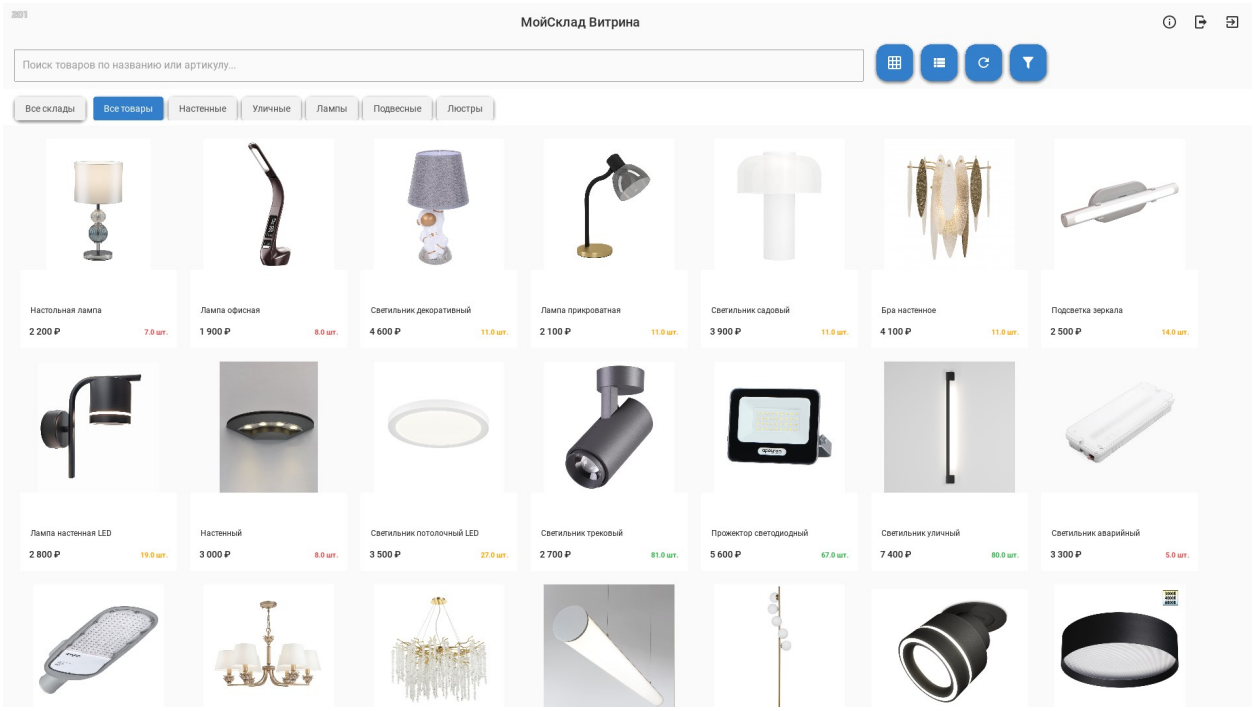

Рисунок 2.1 – Основной экран приложения

2.5 Моделирование вариантов использования

Для разрабатываемого приложения была реализована модель, которая обеспечивает наглядное представление вариантов использования. Она помогает в физической разработке и детальном анализе взаимосвязей объектов. При построении диаграммы вариантов использования применяется унифицированный язык визуального моделирования UML.

Диаграмма вариантов описывает функциональное назначение разрабатываемой системы. То есть это то, что система будет непосредственно делать в процессе своего функционирования. Она является исходным концептуальным представлением системы в процессе ее проектирования и разработки. Проектируемая система представляется в виде ряда прецедентов, предоставляемых системой актерам или сущностям, которые взаимодействуют с системой. Актером или действующим лицом является сущность, взаимодействующая с системой извне (например, человек, техническое устройство). Прецедент служит для описания набора действий, которые система предоставляет актеру.

На основании анализа предметной области в программе должны быть реализованы следующие прецеденты:

- ввод и проверка API-токена для доступа к данным «МойСклад»;
- просмотр товаров в режиме адаптивной сетки;
- просмотр товаров в режиме компактного списка;
- поиск товаров по названию или категории;
- фильтрация товаров по складу и группе;
- переход на внешнюю страницу товара через встроенную ссылку;
- обновление данных из системы «МойСклад»;

На рисунке 2.2 изображены прецеденты для покупателя.

Рисунок 2.2 – Прецеденты для покупателя

2.6 Функциональные требования (сценарии прецедентов)

В разрабатываемом клиентском приложении выделен один основной актёр — **Пользователь** (объединяющий сотрудника магазина и клиента). Ниже представлены сценарии ключевых прецедентов.

2.6.1 Сценарий прецедента «Авторизация в системе»

Предусловие: Приложение запущено впервые или токен доступа не сохранен.

Основной успешный сценарий:

1. Пользователь вводит действительный API-токен в поле ввода.
2. Нажимает кнопку «Войти».
3. Система проверяет токен через API «МойСклад».

4. При успешной проверке осуществляется переход на главный экран.

5. Токен сохраняется в локальной конфигурации для последующих запусков.

2.6.2 Сценарий прецедента «Просмотр каталога товаров»

Предусловие: Пользователь авторизован.

Основной успешный сценарий:

1. Система автоматически загружает и отображает товары с учетом текущих фильтров.

2. Пользователь выбирает режим отображения «Сетка» или «Список».

3. Приложение отображает товары в выбранном формате с базовой информацией (изображение, название, цена, остаток).

2.6.3 Сценарий прецедента «Поиск товара»

Предусловие: Открыт главный экран с каталогом.

Основной успешный сценарий:

1. Пользователь вводит текст в поле поиска.

2. Система в реальном времени фильтрует отображаемые товары.

3. В результатах отображаются товары, в названии, артикуле или названии группы которых содержится введенный текст.

2.6.4 Сценарий прецедента «Фильтрация по складу»

Предусловие: Открыт главный экран с каталогом.

Основной успешный сценарий:

1. Пользователь нажимает кнопку выбора склада.

2. Система отображает диалоговое окно со списком доступных складов.

3. Пользователь выбирает конкретный склад или опцию «Все склады».

4. Приложение обновляет отображение товаров с учетом остатков на выбранном складе.

2.6.5 Сценарий прецедента «Фильтрация по группе товаров»

Предусловие: Открыт главный экран с каталогом.

Основной успешный сценарий:

1. Пользователь выбирает категорию товаров из горизонтального списка групп.
2. Система немедленно фильтрует каталог, оставляя только товары выбранной категории.
3. При выборе опции «Все товары» фильтр сбрасывается.

2.6.6 Сценарий прецедента «Открыть ссылку товара»

Предусловие: Открыто диалоговое окно с деталями товара, для которого указана ссылка.

Основной успешный сценарий:

1. Пользователь нажимает кнопку «Перейти» в диалоговом окне.
2. Система запрашивает подтверждение действия.
3. После подтверждения открывается браузер по умолчанию с переходом по указанной ссылке.

2.6.7 Сценарий прецедента «Обновление данных»

Предусловие: Приложение подключено к сети Интернет.

Основной успешный сценарий:

1. Пользователь нажимает кнопку «Обновить».
2. Система очищает локальный кэш данных.
3. Выполняется новый запрос к API «МойСклад» для получения актуальной информации.
4. Обновленные данные отображаются в интерфейсе и сохраняются в кэш.

2.7 Требования пользователя к интерфейсу приложения

Приложение должно содержать следующие экраны и элементы управления:

– **Экран авторизации:** поле ввода токена, кнопка входа, справочная информация.

– **Главный экран:**

– Панель поиска с текстовым полем.

– Панель фильтров (кнопка выбора склада, горизонтальный скроллируемый список групп товаров).

– Панель управления (кнопки переключения режимов отображения, обновления данных).

– Основная область отображения товаров (адаптивная сетка или вертикальный список).

– **Диалоговые окна:**

– Окно выбора склада с поиском.

– Окно детальной информации о товаре.

– Окно подтверждения перехода по внешней ссылке.

– Окна уведомлений и сообщений об ошибках.

2.8 Требования к оформлению документации

Разработка программной документации и программного изделия должна производиться согласно ГОСТ 19.102-77 и ГОСТ 34.601-90. Единая система программной документации.

3 Технический проект

3.1 Общая характеристика организации решения задачи

Необходимо спроектировать и разработать клиентское приложение для интерактивного сенсорного киоска, которое обеспечит доступ к актуальной информации о товарах из системы складского учёта «МойСклад». Приложение должно работать в автономном режиме на устройстве с сенсорным экраном, установленном в торговом зале, и предоставлять пользователю (сотруднику или клиенту) интуитивно понятный интерфейс для поиска, фильтрации и просмотра данных о товарах.

Клиентское приложение представляет собой кроссплатформенное настольное приложение, написанное на языке Python с использованием фреймворка KivyMD для графического интерфейса. Приложение взаимодействует с облачным REST API системы «МойСклад» для получения данных и реализует локальное кэширование для обеспечения работы при временном отсутствии сетевого соединения.

Решение разрабатывается с учетом специфических требований розничной торговли, включая необходимость оперативного доступа к информации, минимальное время отклика интерфейса и устойчивость к временным потерям сетевого соединения. Архитектура приложения построена по принципу разделения ответственности, где каждый модуль выполняет строго определенный набор функций, что упрощает сопровождение и дальнейшее развитие системы.

3.2 Обоснование выбора технологии проектирования

Для реализации поставленной задачи был выбран стек технологий, обеспечивающий кроссплатформенность, быструю разработку графического интерфейса и надёжное взаимодействие с внешним API. Выбор конкретных технологических решений осуществлялся на основе анализа требований к производительности, удобству использования и простоте развертывания приложения в условиях реальной эксплуатации в торговом зале.

При проектировании системы учитывались такие факторы, как необходимость работы на оборудовании с ограниченными ресурсами (в случае использования неспециализированного потребительского телевизора с сенсорной панелью), требования к отказоустойчивости при временной потере соединения с интернетом, а также потребность в минимальном техническом обслуживании после внедрения. Выбранный технологический стек позволяет удовлетворить все перечисленные требования при сохранении приемлемой сложности реализации.

3.2.1 Описание используемых технологий и языков программирования

3.2.1.1 Язык программирования Python

Python — интерпретируемый язык программирования высокого уровня с динамической типизацией и автоматическим управлением памятью. Был выбран в качестве основного языка разработки благодаря совокупности преимуществ, которые критически важны для успешной реализации проекта:

– богатой экосистеме библиотек для различных задач, что позволяет быстро интегрировать готовые решения без необходимости разработки с нуля;

– высокой скорости разработки прототипов, что особенно важно при итеративном подходе к созданию пользовательского интерфейса;

– кроссплатформенности (поддержка Windows, Linux, macOS), обеспечивающей возможность запуска приложения на различных устройствах без существенной модификации кода;

– простому синтаксису, облегчающему сопровождение кода и снижающему порог вхождения для новых разработчиков;

– широкому распространению и активному сообществу, что гарантирует доступность документации и поддержку при возникновении проблем.

Использование Python также позволяет применять современные методологии разработки, такие как объектно-ориентированное программирова-

ние и модульное тестирование, что повышает надежность и сопровождаемость конечного продукта.

3.2.1.2 Фреймворк KivyMD

KivyMD — фреймворк для создания кроссплатформенных приложений с интерфейсом в стиле Material Design. Основан на фреймворке Kivy, который специализируется на разработке приложений с мультитач-интерфейсами. Выбор KivyMD обусловлен следующими соображениями:

– поддержкой Material Design 3, обеспечивающего современный внешний вид интерфейса, соответствующий актуальным тенденциям в дизайне пользовательских интерфейсов;

– оптимизацией для сенсорных устройств, включая корректную обработку жестов, адаптацию размеров элементов под различные плотности пикселей и учет особенностей тактильного взаимодействия;

– наличием готовых UI-компонентов (кнопки, диалоги, списки, карточки), что значительно ускоряет процесс разработки интерфейса и обеспечивает единообразие визуального представления;

– возможностью создания адаптивных интерфейсов, способных корректно отображаться на экранах с различными соотношениями сторон и разрешениями;

– открытой лицензией и активным сообществом разработчиков, что гарантирует долгосрочную поддержку и развитие фреймворка.

Важным дополнительным преимуществом KivyMD является его способность эффективно работать на устройствах с ограниченными вычислительными ресурсами, что актуально для неспециализированного оборудования, которое может использоваться в качестве сенсорного киоска.

3.2.1.3 Библиотека Requests

Requests — библиотека для выполнения HTTP-запросов, которая стала де-факто стандартом для работы с веб-API в экосистеме Python. Используется

для взаимодействия с REST API «МойСклад». Преимущества выбора данной библиотеки включают:

– простой и интуитивно понятный API, позволяющий выполнять сложные HTTP-запросы с минимальным количеством кода;

– поддержку сессий, что обеспечивает сохранение состояния между запросами (например, авторизационных токенов) и повышает производительность за счет переиспользования соединений;

– встроенные механизмы авторизации, обработки ошибок и повторных попыток соединения, что повышает надежность взаимодействия с внешним API;

– широкое распространение и активное сообщество, что гарантирует высокое качество кода, регулярные обновления и наличие решений для большинства типовых сценариев использования;

– полную совместимость со стандартами HTTP/1.1 и HTTP/2, что обеспечивает оптимальную производительность при работе с современными вебсервисами.

Библиотека Requests также обеспечивает прозрачную работу с прокси-серверами и поддержку SSL-сертификатов, что важно для корпоративных сред, в которых может развертываться приложение.

3.2.1.4 Система кэширования JSON

Для хранения временных данных реализована собственная система кэширования на основе формата JSON. Выбор JSON в качестве формата данных обусловлен следующими причинами:

– человекочитаемым форматом, что упрощает отладку и анализ содержимого кэша в процессе разработки и эксплуатации;

– простотой сериализации/десериализации в Python благодаря встроенной поддержке этого формата в стандартной библиотеке языка;

– возможностью хранения как в оперативной памяти, так и в файловой системе с минимальными накладными расходами на преобразование данных;

– широкой поддержкой в различных инструментах и системах, что обеспечивает гибкость при дальнейшей интеграции или миграции данных;

– эффективным балансом между размером данных и скоростью их обработки, что важно для обеспечения быстрого отклика пользовательского интерфейса.

Разработанная система кэширования реализует механизм времени жизни (TTL) для записей, что предотвращает использование устаревших данных, и включает алгоритмы ограничения размера кэша для предотвращения чрезмерного потребления дискового пространства.

3.3 Диаграмма компонентов и схема взаимодействия

Диаграмма компонентов описывает архитектурные компоненты разрабатываемой системы и их взаимодействие. Данная диаграмма является важным инструментом визуализации архитектурных решений и позволяет четко определить границы ответственности каждого модуля системы. Основными компонентами являются:

– **Модуль UI (Пользовательский интерфейс)** — отвечает за отображение данных и обработку пользовательского ввода, обеспечивая интуитивное взаимодействие с системой через сенсорный экран.

– **Модуль бизнес-логики** — реализует основные сценарии работы приложения, включая обработку данных, применение фильтров и реализацию алгоритмов поиска.

– **Модуль работы с API** — обеспечивает взаимодействие с REST API «МойСклад», выполняя запросы данных и обработку ответов с учетом специфики внешнего интерфейса.

– **Модуль кэширования** — управляет хранением временных данных, обеспечивая быстрый доступ к часто используемой информации и работу приложения в автономном режиме.

– **Модуль конфигурации** — хранит настройки приложения, включая параметры подключения к внешним сервисам и пользовательские предпочтения отображения информации.

Каждый компонент разработан с соблюдением принципа единственной ответственности, что упрощает тестирование, отладку и дальнейшее развитие системы. Взаимодействие между компонентами организовано через четко определенные интерфейсы, что снижает степень связанности и повышает модульность архитектуры.

На рисунке 3.1 изображена диаграмма компонентов системы, демонстрирующая взаимосвязи между основными модулями приложения и внешними системами.

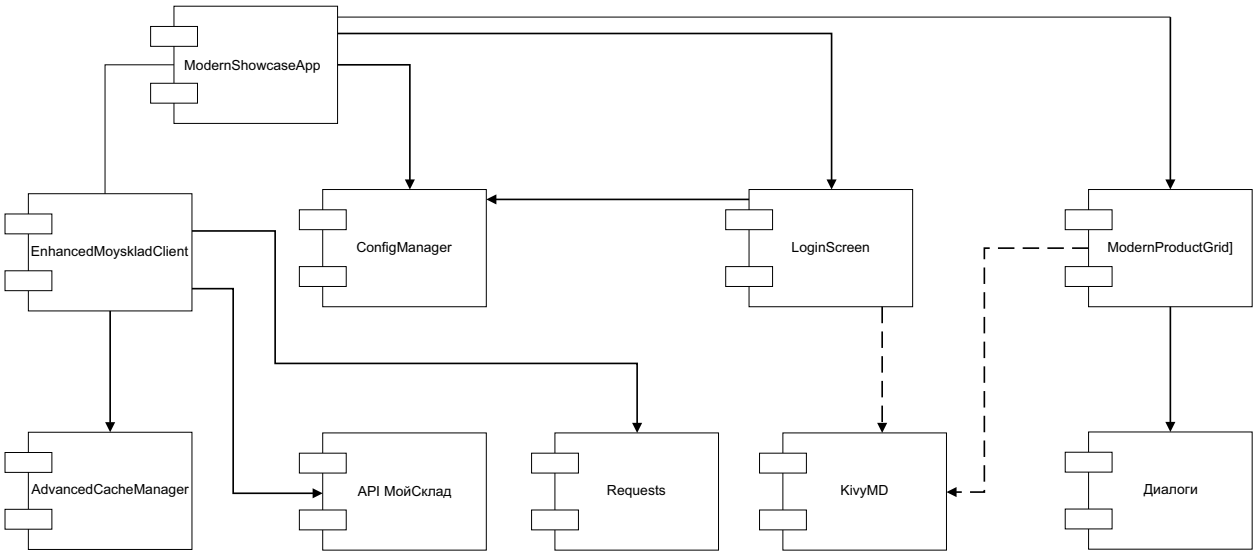

Рисунок 3.1 – Диаграмма компонентов клиентского приложения

Диаграмма наглядно показывает, что архитектура системы построена по многоуровневому принципу, где каждый уровень предоставляет определенный набор сервисов для вышележащего уровня. Такой подход обеспечивает четкое разделение ответственности и упрощает модификацию отдельных компонентов без необходимости переработки всей системы.

3.4 Диаграмма размещения

Диаграмма размещения отражает физическую архитектуру системы и показывает, как программные компоненты распределены по аппаратным узлам. Понимание физической архитектуры важно для оценки требований к инфраструктуре, планирования развертывания и анализа потенциальных узких мест в производительности.

Система состоит из следующих физических компонентов:

– **Клиентское устройство** — компьютер или планшет с сенсорным экраном, на котором установлено разработанное приложение. Данное устройство размещается непосредственно в торговом зале и является точкой взаимодействия конечных пользователей с системой.

– **Облачный сервис «МойСклад»** — внешняя система, предоставляющая данные через REST API. Данный сервис функционирует в инфраструктуре провайдера облачных услуг и обеспечивает хранение и обработку основной бизнес-информации предприятия.

– **Локальное хранилище** — файловая система клиентского устройства, используемая для хранения кэша данных, конфигурации приложения и временных файлов (например, загруженных изображений товаров).

Взаимодействие между компонентами осуществляется через стандартные сетевые протоколы: HTTPS для обмена данными с облачным сервисом и локальные файловые операции для работы с кэшем. Архитектура размещения предполагает минимальные требования к инфраструктуре на стороне предприятия — достаточно устройства с сенсорным экраном и стабильного интернет-соединения.

На рисунке 3.2 представлена диаграмма размещения системы, иллюстрирующая физическое распределение компонентов и направления обмена данными между ними.

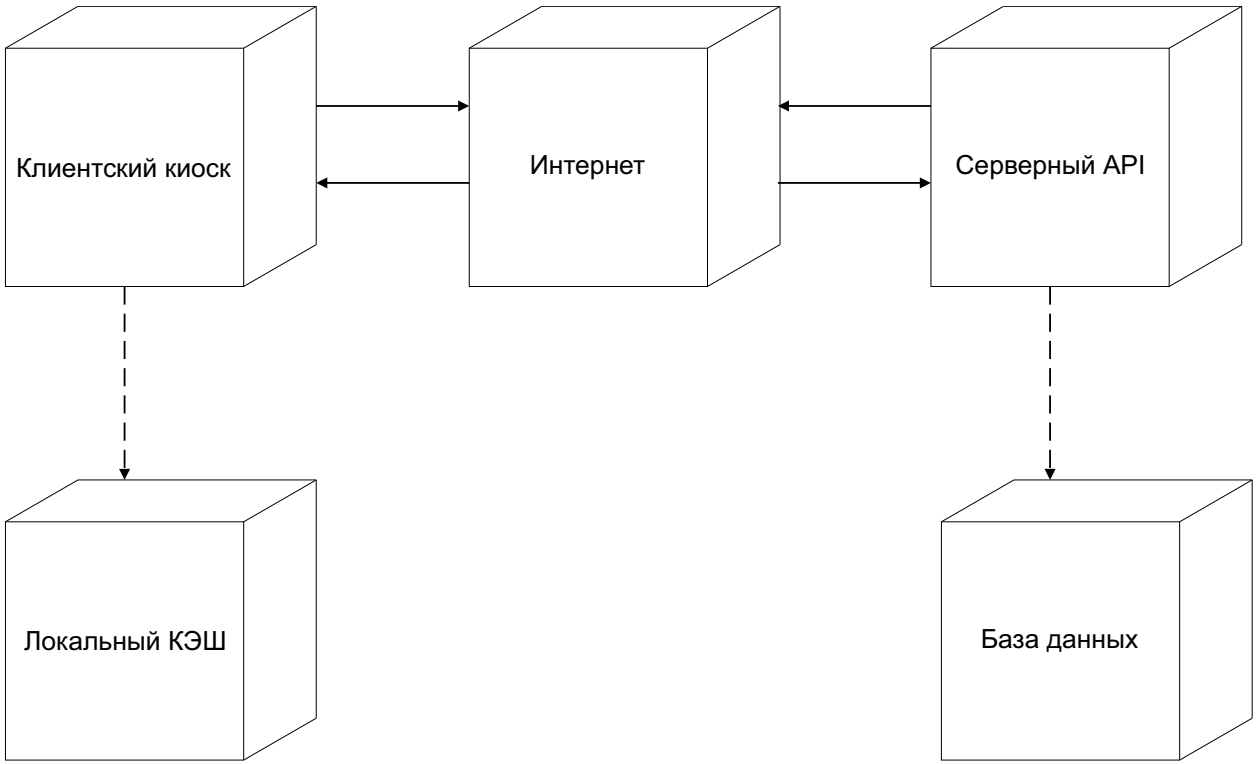

Рисунок 3.2 – Диаграмма размещения системы

Особенностью предложенной архитектуры размещения является ее ориентация на работу в условиях возможных перебоев с интернет-соединением. Приложение способно продолжать функционирование, используя кэшированные данные, и автоматически синхронизировать информацию при восстановлении связи. Это свойство критически важно для обеспечения непрерывности работы в розничной торговле.

3.5 Содержание данных. Основные сущности

Проанализировав требования к системе и особенности предметной области розничной торговли светодиодной продукцией, можно выделить следующие основные сущности системы, которые представляют ключевые бизнес-объекты и информационные сущности:

– **Товар (Product)** — основная сущность, представляющая товарную единицу. Данная сущность содержит информацию о конкретном изделии, предлагаемом к продаже, и является центральным объектом в информационной модели приложения.

– **Склад (Store)** — сущность, представляющая физическое или виртуальное место хранения товаров. В контексте системы складского учета «МойСклад» данная сущность может соответствовать как реальным складским помещениям, так и виртуальным складам, используемым для учета товаров в различных состояниях.

– **Группа товаров (ProductGroup)** — категория для классификации товаров по различным признакам: функциональному назначению, производителю, техническим характеристикам или другим критериям, значимым для бизнес-процессов предприятия.

– **Пользовательские настройки (UserSettings)** — конфигурация приложения, хранящая параметры, специфичные для конкретной установки или пользовательские предпочтения отображения информации.

Каждая сущность обладает определенным набором атрибутов, которые описывают ее свойства и характеристики. Атрибуты выбраны с учетом как требований к функциональности приложения, так и структуры данных, предоставляемых через API системы «МойСклад».

В таблице 3.1 представлены атрибуты сущности «Товар», которые были выделены в результате анализа структуры данных API «МойСклад» и определения информационных потребностей пользователей приложения.

Таблица 3.1 – Атрибуты сущности «Товар»

Поле	**Тип**	**Обязательное**	**Описание**
id	String	Да	Уникальный идентификатор товара в системе «МойСклад», используемый для однозначной идентификации товарной позиции во всех операциях
name	String	Да	Наименование товара, представляющее собой краткое, но информативное описание товарной позиции для ее идентификации пользователями
price	Float	Да	Цена товара в рублях, отображаемая с учетом применяемых наценок и скидок, актуальная на момент запроса информации
stock	Integer	Да	Общее количество товара на всех складах, рассчитанное как сумма остатков по всем местам хранения, учитываемым в системе
article	String	Нет	Артикул товара — уникальный идентификатор, присваиваемый производителем или поставщиком для упрощения учета и поиска товаров
description	String	Нет	Описание товара, содержащее дополнительную техническую информацию, характеристики или особенности применения продукции
link	String	Нет	Ссылка на страницу товара во внешнем источнике (например, на сайте производителя), предоставляющая доступ к расширенной информации
groupId	String	Нет	Идентификатор группы товара, используемый для связи товара с соответствующей категорией в иерархической структуре классификации

Атрибуты сущности «Товар» выбраны таким образом, чтобы обеспечить баланс между полнотой информации и удобством ее восприятия на экране сенсорного киоска. Обязательные поля гарантируют отображение минимально необходимой информации для принятия решения о покупке, в то время как необязательные поля предоставляют дополнительный контекст при его наличии.

В таблице 3.2 представлены атрибуты сущности «Склад», которые определяют свойства мест хранения товаров в контексте их отображения в приложении и использования для фильтрации товарного ассортимента.

Таблица 3.2 – Атрибуты сущности «Склад»

Поле	Тип	Обязательное	Описание
id	String	Да	Уникальный идентификатор склада в системе «МойСклад», используемый для однозначной идентификации места хранения при выполнении запросов
name	String	Да	Наименование склада, представляющее собой удобочитаемое название места хранения, понятное пользователям приложения
address	String	Нет	Адрес склада — физическое местоположение места хранения товаров, предоставляемое для информационных целей
archived	Boolean	Да	Флаг архивности склада, указывающий на то, что данное место хранения более не используется для текущих операций, но сохраняется в системе для исторических данных

Сущности реализуются в виде классов Python, а их атрибуты — в виде свойств этих классов. Такой подход обеспечивает типобезопасность, инкапсуляцию логики работы с данными и возможность расширения функциональности через наследование и полиморфизм. Взаимодействие между сущностями осуществляется через методы соответствующих модулей, что обеспечивает четкое разделение ответственности и упрощает тестирование отдельных компонентов системы.

Выделенные сущности и их атрибуты образуют информационную модель приложения, которая служит основой для разработки модулей работы с данными, пользовательского интерфейса и бизнес-логики системы. Данная модель спроектирована с учетом как текущих функциональных требований, так и возможных направлений развития системы в будущем.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Лутц, М. Изучаем Python / М. Лутц. – 5-е изд. – Санкт-Петербург : Символ-Плюс, 2022. – 1648 с. – ISBN 978-5-93286-256-6. – Текст: непосредственный.

2. Саммерфилд, М. Программирование на Python 3. Подробное руководство / М. Саммерфилд. – Санкт-Петербург : Символ-Плюс, 2019. – 608 с. – ISBN 978-5-93286-211-5. – Текст: непосредственный.

3. Россум, Г. Язык программирования Python / Г. ван Россум, Ф. Л. Дрейк. – Москва : ДМК Пресс, 2020. – 454 с. – ISBN 978-5-93700-107-1. – Текст: непосредственный.

4. Ричардсон, Л. RESTful Web API. Проектирование для всего цикла жизни / Л. Ричардсон, М. Амюнден, С. Рубь. – Москва : Вильямс, 2016. – 448 с. – ISBN 978-5-8459-2045-4. – Текст: непосредственный.

5. Масала, М. Проектирование RESTful API / М. Масала. – Санкт-Петербург : Питер, 2018. – 240 с. – ISBN 978-5-4461-0530-8. – Текст: непосредственный.

6. Гурьянов, А. А. Протокол HTTP: теория и практика / А. А. Гурьянов. – Москва : Наука и техника, 2019. – 320 с. – ISBN 978-5-94387-974-6. – Текст: непосредственный.

7. Крокфорд, Д. JSON: Основы / Д. Крокфорд. – Москва : Символ-Плюс, 2017. – 128 с. – ISBN 978-5-93286-213-9. – Текст: непосредственный.

8. Купер, А. Интерфейс: основы проектирования взаимодействия / А. Купер, Р. Рейман, Д. Кронин. – 4-е изд. – Санкт-Петербург : Питер, 2021. – 720 с. – ISBN 978-5-4461-1231-3. – Текст: непосредственный.

9. Тидвелл, Д. Разработка пользовательских интерфейсов / Д. Тидвелл. – 2-е изд. – Санкт-Петербург : Питер, 2020. – 528 с. – ISBN 978-5-4461-1152-1. – Текст: непосредственный.

10. Мельников, С. В. Технологии кэширования в распределенных системах / С. В. Мельников. – Москва : ДМК Пресс, 2018. – 256 с. – ISBN 978-5-97060-623-7. – Текст: непосредственный.

11. Таненбаум, Э. С. Распределенные системы. Принципы и парадигмы / Э. С. Таненбаум, М. ван Стеен. – 2-е изд. – Санкт-Петербург : Питер, 2020. – 1120 с. – ISBN 978-5-4461-1456-0. – Текст: непосредственный.

12. Соммервилл, И. Инженерия программного обеспечения / И. Соммервилл. – 10-е изд. – Москва : Вильямс, 2019. – 736 с. – ISBN 978-5-8459-2077-5. – Текст: непосредственный.

13. Куликов, С. Ю. Тестирование программного обеспечения / С. Ю. Куликов. – Москва : Бином. Лаборатория знаний, 2018. – 280 с. – ISBN 978-5-9963-3604-2. – Текст: непосредственный.

14. МойСклад. Документация REST API [Электронный ресурс]. – Режим доступа: https://dev.moysklad.ru/doc/api/remap/1.2/ (дата обращения: 15.12.2024). – Текст: электронный.

15. Kivy Framework. Официальная документация [Электронный ресурс]. – Режим доступа: https://kivy.org/doc/stable/ (дата обращения: 15.12.2024). – Текст: электронный.

16. Библиотека Requests. Документация [Электронный ресурс]. – Режим доступа: https://requests.readthedocs.io/ru/latest/ (дата обращения: 15.12.2024). – Текст: электронный.

17. Material Design 3. Руководство по дизайну [Электронный ресурс]. – Режим доступа: https://m3.material.io/ (дата обращения: 15.12.2024). – Текст: электронный.

18. Стандарт JSON [Электронный ресурс]. – Режим доступа: https://www.json.org/json-ru.html (дата обращения: 15.12.2024). – Текст: электронный.

19. RFC 7230-7237. Протокол HTTP/1.1 [Электронный ресурс]. – Режим доступа: https://www.rfc-editor.org/rfc/rfc7230 (дата обращения: 15.12.2024). – Текст: электронный.