Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой ВТ
_____ О.В. Непомнящий
подпись инициалы, фамилия
«_____» _____2019 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование направления

Интернет-магазин. Серверная часть
тема

Руководитель	_____ подпись, дата	доц. канд.физ.-мат.наук должность, ученая степень	К.В. Коршун инициалы, фамилия
Выпускник	_____ подпись, дата		С.А. Муравьёв инициалы, фамилия
Нормоконтролер	_____ подпись, дата	доцент, канд. техн. наук должность, ученая степень	В.И. Иванов инициалы, фамилия

Красноярск 2019

РЕФЕРАТ

Выпускная квалификационная работа по теме «Интернет-магазин. Серверная часть» содержит 37 страниц текстового документа, 4 иллюстрации, 7 листингов, 26 использованных источников.

ПРОГРАММНЫЙ КОМПЛЕКС, ПРОГРАММНЫЙ ИНТЕРФЕЙС, ИНФОРМАЦИОННАЯ СИСТЕМА ИНТЕРНЕТ-МАГАЗИНА, ПРОЕКТИРОВАНИЕ АЛГОРИТМА, ВЕБ ПРОГРАММИРОВАНИЕ, PHP, GRAPHQL, LARAVEL.

Целью выпускной квалификационной работы является разработка программного комплекса, позволяющего клиентским приложениям (например, приложению интернет-магазина для Android) получать данные и выполнять операции в интернет-магазине.

Задачи, решенные в ходе выполнения бакалаврской работы:

– разработано задание на разработку программного комплекса, позволяющего клиентским приложениям взаимодействовать с интернет-магазином;

– выполнено проектирование и разработка программного комплекса в клиент-серверной архитектуре;

– выполнена разработка инструментов разработчика клиентских приложений программного комплекса.

Общие результаты и выводы: в ходе выполнения данной бакалаврской работы был выполнен проект и разработан программный комплекс, позволяющий клиентским приложениям получать данные и выполнять операции в интернет-магазине.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью выпускной квалификационной работы является разработка программного комплекса, позволяющего клиентским приложениям (например, приложению интернет-магазина для Android) получать данные и выполнять операции в интернет-магазине. Программный комплекс обладает следующими особенностями:

– возможность доступа по протоколам HTTP(S);

– возможность авторизованного и анонимного доступа с различными наборами доступных операций;

– возможность запуска как в среде разработки, так и в рабочем окружении интернет-магазина;

– возможность динамически контролировать объём и структуру ответа от программного интерфейса.

Для достижения цели в работе решаются следующие задачи:

– провести обзор существующих технических решений интернет-магазина

– составить требования к программному комплексу;

– выполнить проектирование архитектуры и реализовать программный комплекс, придерживаясь разработанных ранее архитектурных решений;

– составить инструкции по подготовке к работе и использованию.

1 Анализ технического задания

Ознакомимся с существующей бизнес-моделью и техническими решениями, разработанными для интернет-магазина.

1.1 Обзор бизнес-модели интернет-магазина

В интернет-магазине существуют несколько моделей объектов, которые описывают сценарии взаимодействия. Рассмотрим детально каждую из основных моделей.

1.1.1 Покупатель

Основные сценарии взаимодействия (прецеденты) пользователя с интер-

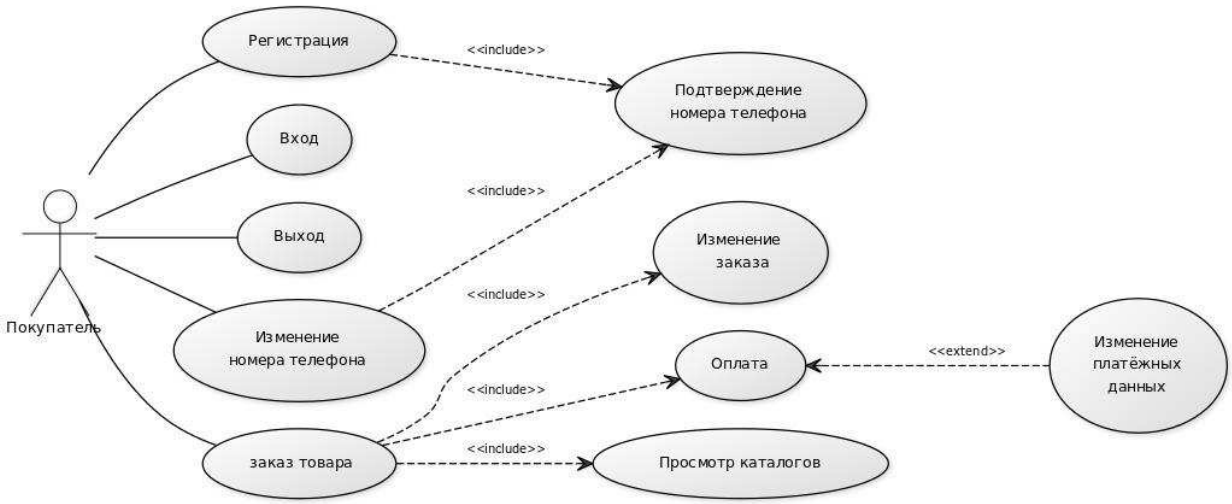

Рисунок 1 – упрощенная диаграмма прецедентов покупателя

нет-магазином представлены на рисунке 1

Текстовое описание прецедентов:

Название прецедента: Подтверждение номера телефона

Предусловия: клиентское приложение получает номер телефона пользователя.

Основной сценарий:

a) Пользователь отправляет запрос на подтверждение номера телефона, передавая в нём номер.

b) При правильном формате телефона приложению высылается значение «ИСТИНА».

Постусловия: пользователь получает СМС с кодом подтверждения на номер мобильного телефона

Условие 1. Клиентское приложение передало номер в неверном формате или значение, которое невозможно распознать как номер мобильного телефона.

Основной сценарий:

a) Приложению высылается ошибка валидации поля номера телефона.

Постусловия: клиентское приложение обрабатывает ошибку и отображает её пользователю, после чего прецедент повторяется.

Название прецедента: Регистрация

Предусловия: пользователь нажимает кнопку «Регистрация».

Основной сценарий:

a) Пользователь отправляет поля формы регистрации: псевдоним, пароль, адрес электронной почты, код из смс с подтверждением номера телефона, опционально свои имя и фамилию.

b) Сервер проверяет введённые поля, в том числе код из смс, полученный в прецеденте «подтверждение номера телефона».

c) При правильном формате введённых данных сервер создаёт нового пользователя в информационной системе.

d) Сервер в ответ на запрос предоставляет клиенту запрошенные им поля о новом пользователе.

Постусловия: В информационной системе имеется новый пользователь, а клиентское приложение имеет к нему доступ.

Условие 1. Клиентское приложение передало любое из полей в неверном формате, или код подтверждения по смс не соответствует высланному.

Основной сценарий:

a) Приложению высылается ошибка валидации, в которой содержится название неверно полученного поля и объяснение ошибки.

Постусловия: клиентское приложение обрабатывает ошибку и отображает её пользователю, после чего прецедент повторяется.

Название прецедента: Вход

Предусловия: пользователь нажимает кнопку «Вход».

Основной сценарий:

a) Пользователь отправляет поля формы авторизации: псевдоним и пароль.

b) Сервер проверяет введённые поля.

c) Сервер в ответ на запрос предоставляет клиенту запрошенные им поля об авторизированном пользователе.

Постусловия: В информационной системе пользователь помечается авторизированным, а клиентское приложение имеет к нему доступ.

Условие 1. Клиентское приложение передало любое из полей в неверном формате или пользователь не найден по введённым данным.

Основной сценарий:

a) Приложению высылается ошибка валидации, в которой содержится объяснение ошибки.

Постусловия: клиентское приложение обрабатывает ошибку и отображает её пользователю, после чего прецедент повторяется.

Название прецедента: Выход

Предусловия: пользователь нажимает кнопку «Выход».

Основной сценарий:

a) Клиентское приложение отправляет запрос на завершение сессии.

b) При авторизованном запросе клиентскому приложению высылается значение «ИСТИНА».

Постусловия: В информационной системе сессия пользователя помечается оконченной

Условие 1. Клиентское приложение не передало ключ авторизации.

Основной сценарий:

a) Приложению высылается ошибка выполнения, в которой содержится объяснение ошибки.

Постусловия: клиентское приложение обрабатывает ошибку и отображает её пользователю, после чего прецедент повторяется.

Название прецедента: Изменение номера телефона

Предусловия: пользователь нажимает кнопку «Изменить номер телефона».

Основной сценарий:

a) Клиентское приложение исполняет прецедент «Подтверждение номера телефона».

b) Клиентское приложение отправляет запрос на изменение номера телефона.

c) Сервер проверяет введённый номер телефона и код подтверждения, в случае успешной проверки сохраняет изменения в информационной системе.

Постусловия: В информационной системе изменяется номер телефона пользователя

Условие 1. Клиентское приложение не передало ключ авторизации.

Основной сценарий:

a) Приложению высылается ошибка выполнения, в которой содержится объяснение ошибки.

Постусловия: клиентское приложение обрабатывает ошибку и отображает её пользователю, после чего прецедент повторяется.

Условие 2. Клиентское приложение передало неверный код подтверждения или неприемлемый номер телефона.

Основной сценарий:

a) Приложению высылается ошибка выполнения, в которой содержится объяснение ошибки.

Постусловия: клиентское приложение обрабатывает ошибку и отображает её пользователю, после чего прецедент повторяется.

Название прецедента: Просмотр каталогов

Предусловия: пользователь просматривает список каталогов.

Основной сценарий:

a) Клиентское приложение запрашивает лоты из каталога, который просматривает пользователь.

b) Сервер возвращает список лотов в запрошенном каталоге.

Постусловия: Пользователь может просматривать товары в каталоге

Название прецедента: Заказ товара

Предусловия: Пользователь нажимает кнопку «добавить к заказу».

Основной сценарий:

a) Клиентское приложение получает от пользователя количество заказываемых единиц, свойства заказываемых единиц и предпочитаемое место получения товара.

b) Клиентское приложение делает запрос на сервер с полученными от пользователя данными.

Постусловия: Товар добавляется к заказу пользователя.

Условие 1. Клиентское приложение не передало ключ авторизации.

Основной сценарий:

a) Приложению высылается ошибка выполнения, в которой содержится объяснение ошибки.

Постусловия: клиентское приложение обрабатывает ошибку и отображает её пользователю, после чего прецедент повторяется.

Условие 2. Клиентское приложение передало несуществующий лот или любые иные некорректные данные.

Основной сценарий:

a) Приложению высылается ошибка выполнения, в которой содержится объяснение ошибки.

Постусловия: клиентское приложение обрабатывает ошибку и отображает её пользователю, после чего прецедент повторяется.

Название прецедента: Изменение заказа

Предусловия: Пользователь нажимает кнопку «изменить заказ».

Основной сценарий:

a) Клиентское приложение получает от пользователя новые данные заказа.

b) Клиентское приложение делает запрос на сервер с полученными от пользователя данными.

Постусловия: Заказ изменяется в соответствии с запросом пользователя

Условие 1. Клиентское приложение не передало ключ авторизации.

Основной сценарий:

a) Приложению высылается ошибка выполнения, в которой содержится объяснение ошибки.

Постусловия: клиентское приложение обрабатывает ошибку и отображает её пользователю, после чего прецедент повторяется.

Условие 2. Клиентское приложение передало несуществующий лот или любые иные некорректные данные

Основной сценарий:

a) Приложению высылается ошибка выполнения, в которой содержится объяснение ошибки.

Постусловия: клиентское приложение обрабатывает ошибку и отображает её пользователю, после чего прецедент повторяется.

Название прецедента: Оплата

Предусловия: Пользователь нажимает кнопку «добавить отписку об оплате»

Основной сценарий:

a) Клиентское приложение получает от пользователя данные о проведённой оплате.

b) Клиентское приложение делает запрос на сервер с полученными от пользователя данными.

Постусловия: Отписка об оплате пользователя добавляется в информационную систему

Условие 1. Клиентское приложение не передало ключ авторизации.

Основной сценарий:

a) Приложению высылается ошибка выполнения, в которой содержится объяснение ошибки.

Постусловия: клиентское приложение обрабатывает ошибку и отображает её пользователю, после чего прецедент повторяется.

Условие 2. Клиентское приложение передало несуществующий заказ или любые иные некорректные данные

Основной сценарий:

a) Приложению высылается ошибка выполнения, в которой содержится объяснение ошибки.

Постусловия: клиентское приложение обрабатывает ошибку и отображает её пользователю, после чего прецедент повторяется.

Название прецедента: Изменение платёжных данных

Предусловия: Пользователь нажимает кнопку «изменение платёжных данных».

Основной сценарий:

a) Клиентское приложение получает от пользователя новые платёжные данные.

b) Клиентское приложение делает запрос на сервер с полученными от пользователя данными.

Постусловия: Новые платёжные данные сохраняются в информационной системе.

Условие 1. Клиентское приложение не передало ключ авторизации.

Основной сценарий:

a) Приложению высылается ошибка выполнения, в которой содержится объяснение ошибки.

Постусловия: клиентское приложение обрабатывает ошибку и отображает её пользователю, после чего прецедент повторяется.

Условие 2. Клиентское приложение передало данные, которые не проходят проверки на валидность

Основной сценарий:

a) Приложению высылается ошибка выполнения, в которой содержится объяснение ошибки.

Постусловия: клиентское приложение обрабатывает ошибку и отображает её пользователю, после чего прецедент повторяется.

1.1.2 Организатор

Организатор – оператор обработки заказов покупателя. Организаторы занимаются поиском поставщиков, установлением с ними бизнес взаимодействия, получением от них каталогов товаров.

После вышеописанных процедур организаторы заполняют информационную систему товарами поставщика и создают закупку у этого поставщика. Закупка – элемент итерационной модели работы с поставщиком, выражаемый в конечном списке покупателей и их заказов, передаваемый поставщику. Такое устройство взаимодействия с поставщиком нужно в силу того, что поставщик предоставляет пониженные цены для оптовых партий товара, поэтому заказы покупателей собираются в партии по достижению минимального объема партии, определяемого каждым поставщиком индивидуально.

После получения списка заказов поставщик сверяет со списком наличие желаемых позиций и выставляет счёт организатору. Организатор уведомляет покупателей о подтверждении или отказе в покупке данного товара и начинает сбор средств для оплаты счёта.

После периода сбора средств организатор проверяет все оплаты от покупателей – поскольку оплата принимается на счёт организатора без использования эквайринга[1], клиенты самостоятельно отчитываются об оплатах, а организатор сверяет их отчёты с выпиской по банковскому счёту. В случае подтверждения всех оплат денежные средства направляются по счёту к поставщику.

После получения денежных средств поставщик высылает (отгружает) товар организатору. При получении товара организатор уведомляет об этом факте покупателя с помощью смены статуса закупки. Организатор рассортировывает общую посылку от поставщика на заказы для каждого покупателя, упаковывает

[1] Эквайринг – приём к оплате платёжных карт в качестве средства оплаты товара. Осуществляется уполномоченным банком-эквайером путём установки платёжных терминалов или предоставления программного интерфейса [1].

их и передаёт в службу логистики – центры раздач. Дальнейшие действия работников центров раздач будут описаны в следующей главе.

На рисунке 2 представлены вышеописанные операции в виде диаграммы прецедентов. Детальное описание данных прецедентов опущено в силу наличия детального описания процедуры взаимодействий организаторов в бизнес-модели.

Кроме того, на рисунке описаны прецеденты установки статуса закупок. Данные статусы используются для учёта процесса обработки заказа. Кроме того, установка этих статусов информирует покупателей об этапе обработки их заказов. Прецеденты добавления каталогов и лотов в каталог являются классикой операцией создания объекта в информационной системе.

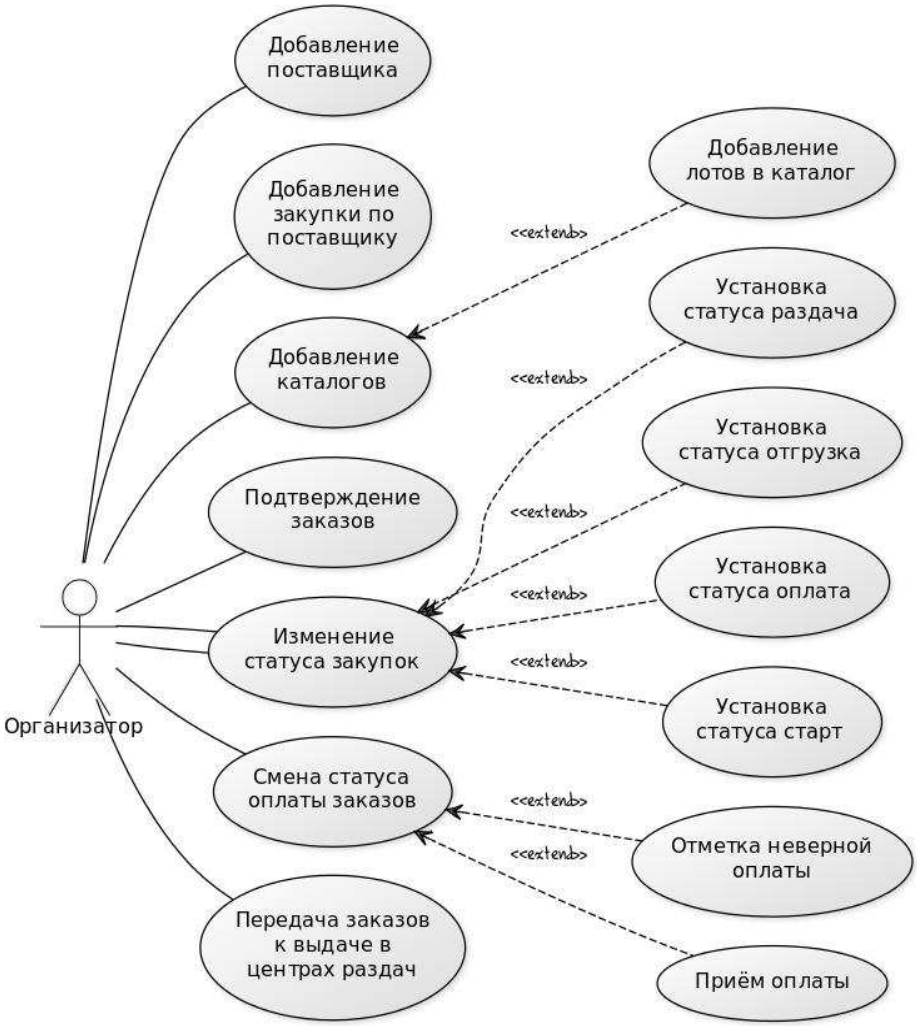

Рисунок 2 – диаграмма прецедентов организатора

Большинство из прецедентов организаторов не будет реализовано в программном интерфейсе, поскольку по заданию заказчика требуется обеспечить интерфейс только для клиентских операций и операций работников центров раздач (ЦР), речь о которых пойдёт в следующей главе.

1.1.3 Работник центра раздач

Основные сценарии взаимодействия работника центра раздач с

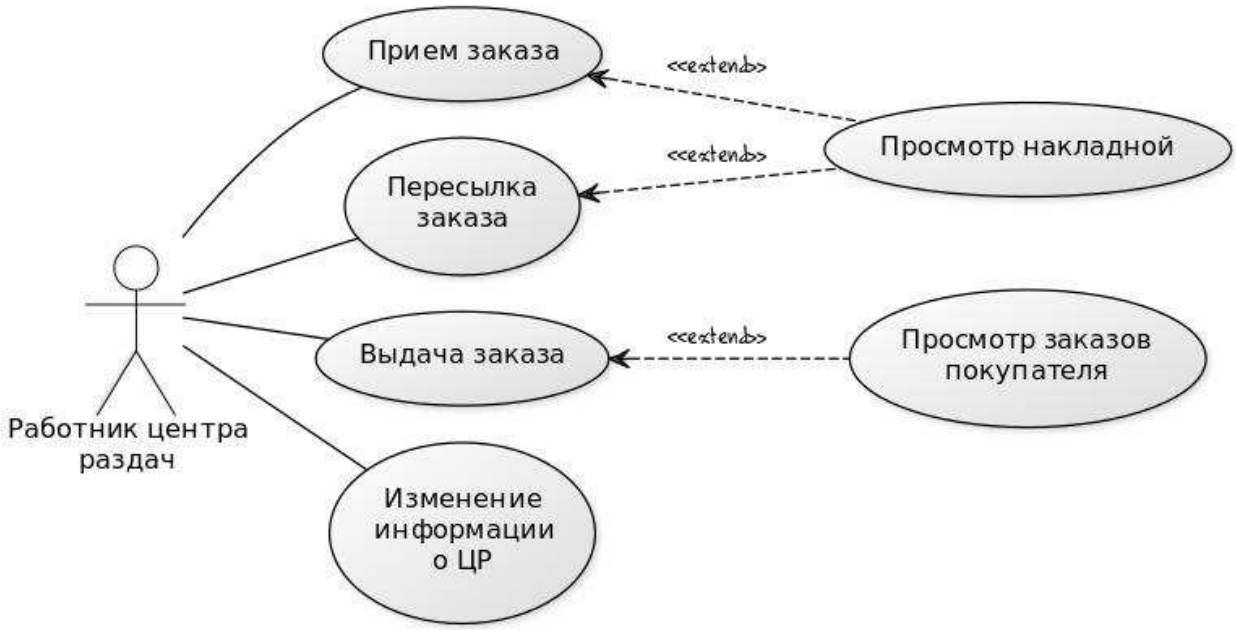

Рисунок 3 – прецеденты работника центра раздач

информационной системой представлены на рисунке 3

Текстовое описание прецедентов:

Название прецедента: Просмотр накладной

Предусловия: работник центра раздач запрашивает информацию о накладной для определённой закупки

Основной сценарий:

a) Клиентское приложение отправляет запрос на получении информации о накладной – списке заказов и информации по закупке.

b) При авторизованном запросе клиентскому приложению высылаются запрошенные значения.

Постусловия: работник центра раздач получает информацию о заказах и закупке – накладной.

Условие 1. Клиентское приложение не передало ключ авторизации или передало несуществующую, или не принадлежащую этому ЦР номеру накладной.

Основной сценарий:

a) Приложению высылается ошибка выполнения, в которой содержится объяснение ошибки.

Постусловия: клиентское приложение обрабатывает ошибку и отображает её работнику центра раздач, после чего прецедент повторяется с правильными данными.

Название прецедента: Просмотр заказов покупателя

Предусловия: Покупатель при посещении центра раздач просит работника выдать ему заказы.

Основной сценарий:

a) Клиентское приложение отправляет запрос на получении информации о заказах пользователя.

b) При авторизованном запросе клиентскому приложению высылаются запрошенные заказы пользователя

Постусловия: работник центра раздач получает информацию о заказах и закупке – накладной.

Условие 1. Клиентское приложение не передало ключ авторизации или передало несуществующего пользователя

Основной сценарий:

a) Приложению высылается ошибка выполнения, в которой содержится объяснение ошибки.

Постусловия: клиентское приложение обрабатывает ошибку и отображает её работнику центра раздач, после чего прецедент повторяется с правильными данными.

Название прецедентов: Приём заказа/Пересылка заказа/Выдача заказа

Предусловия: работник центра раздач отмечает заказ как принятый/пересланный/выданный

Основной сценарий:

a) Клиентское приложение отправляет запрос на приём/пересылку/выдачу заказа.

b) При авторизованном запросе клиентскому приложению высылается запрошенное поле заказа или значение «ИСТИНА».

Постусловия: Заказ отмечен принятым/пересланным/выданным в информационной системе, и покупатель получает уведомление о том, что он может забрать заказ в случае, если запрос был на приём заказа.

Условие 1. Клиентское приложение не передало ключ авторизации или передало несуществующий, или не принадлежащий этому ЦР заказ или заказ уже принят/переслан/выдан

Основной сценарий:

a) Приложению высылается ошибка выполнения, в которой содержится объяснение ошибки.

Постусловия: клиентское приложение обрабатывает ошибку и отображает её работнику центра раздач, после чего прецедент повторяется с правильными данными.

1.1.4 Администратор

Администраторы имеют право на совершение всех вышеописанных действий от имени любого организатора, покупателя или работника центра раздач. Кроме того, администраторы имеют право изменять настройки бизнес-процессов интернет-магазина, такие как: срок бесплатного хранения товара в центре выдачи, сумма штрафа за просрочку хранения, система сбора отчислений в пользу поддержки работоспособности системы.

1.2 Описание существующих технических решений

1.2.1 Основной язык программирования

Поскольку интернет-магазин был основан на форуме, где велась продажа товара через темы и сообщения в них, основной функционал описан с использованием фреймворка форума. Фреймворк[2] форума – PHPBB [3]. Этот фреймворк написан на языке PHP и имеет слой абстракции над хранилищем данных. Максимальная поддерживаемая версия PHP для версии фреймворка, на которой работает интернет-магазин – PHP 5.6.

1.2.2 Организация хранилища данных

Основное хранилище данных работает на СУБД MySQL [4]. Временные данные хранятся в файлах на сервере. Кроме того, в интернет-магазине реализован поиск на движке Sphinx [5], который также имеет свою систему управления индексом.

1.2.3 Среда выполнения кода

Код выполняется на серверах под управлением операционной системы Debian [6]. В качестве веб-сервера используется Apache [7], так же настроен реверсивный прокси-сервер NGINX [8].

---

[2] Фреймворк — заготовка, шаблон для программной платформы, определяющий архитектуру программной системы; программное обеспечение, облегчающее разработку и объединение разных модулей программного проекта [2].

1.3 Разработка требований к системе

1.3.1 Требования к среде выполнения и языку программирования

Код готового решения должен быть переносимым в рамках unix-подобных систем. Это означает, что решение должно одинаково обрабатывать запросы и отвечать одинаковыми ответами вне зависимости от дистрибутива, на котором оно будет запущено. Конфигурация системы будет передана через переменные окружения.

Язык программирования не должен отличаться от основного языка программирования, однако допускается использовать более современные версии языка, такие как PHP 7.3, имея в виду, что на всех серверах предустановлен и настроен как основной PHP 5.6.

1.3.2 Требования к хранилищу данных

Требуется использовать основную СУБД как единственный источник истины. Все чтения и записи объектов бизнес-логики необходимо производить с ней. Допускается создавать отдельные базы хранилища данных для проекта и хранить временные данные в файловой системе.

1.3.3 Требования к структуре запросов и ответов

Различные клиенты должны иметь возможность получать разное количество объектов и полей в этих объектах в зависимости от запроса. Данные, передаваемые в запросах и ответах, должны быть строго типизированы во избежание ошибок. Все ответы интерфейса требуется возвращать в формате JSON [9].

Аутентификация должна происходить по всем правилам бизнес-логики, определённой в исходном коде фреймворка PHPBB. Авторизация должна происходить в соответствии с вышеописанными ролями.

1.4 Выбор инструментов

Основываясь на предоставленном техническом задании, рассмотрим доступные средства и системы разработки для реализации.

1.4.1 Выбор фреймворка разработки

Поскольку в требованиях было предъявлено использование PHP как основного языка разработки, будем основываться на фреймворках этого языка.

Рассмотрим в сравнительном анализе самые популярные фреймворки, написанные для разработки под PHP:

Zend: этот фреймворк от разработчиков, которые вкладывают достаточно много сил в развитие самого PHP и на хорошем счету у мирового сообщества программистов на этом языке [10]. Однако этот фреймворк менее популярен в сообществе, чем Laravel и развивается гораздо медленнее.

Yii2: этот фреймворк разработан сообществом, однако в настоящее время основные разработчики ушли из проекта и его дальнейшая судьба не ясна [11].

Symfony: этот фреймворк развивается достаточно быстро, разработан сообществом и является самым популярным решением в бизнес-среде [12]. Однако этот фреймворк сложен в освоении в сравнении с Laravel.

Laravel: этот фреймворк развивается быстрее, чем Symfony, имеет более широкую аудиторию разработчиком и является достаточно простым для освоения [13].

По результатам анализа был выбран фреймворк Laravel последней версии.

1.4.2 Выбор стиля взаимодействия компонентов системы

На данный момент существует несколько популярных архитектурных стилей взаимодействия компонентов системы. Рассмотрим плюсы и минусы каждого из них.

RESTful: Популярный стиль, обширная поддержка сообществом, большое количество готовых библиотек и подходов к разработке на этом стиле. Этот стиль никогда не был стандартизирован и воспринимается каждым разработчиком на свой лад. Кроме того, REST не позволяет менять структуру ответа без дополнительных модификаций стиля, таких как передача списка возвращаемых полей как один из аргументов метода. Поскольку RESTful в общем базируется на методах, описанных в протоколе HTTP, появляется дополнительные сложности восприятия и возможные проблемы с отсутствием поддержки этих методов передачи у клиентов [14].

SOAP: Протокол взаимодействия с программным интерфейсом. В отличии от RESTful, является полностью описанным стилем. Редко используется в сообществе разработчиков и имеет разные типы взаимодействия для разных прикладных протоколов: SMTP, FTP, HTTP. Этот стиль использует XML в качестве опорного формата представления данных, что является более затратным, в сравнении с JSON [15].

GraphQL: Современная молодая спецификация стиля взаимодействия программных интерфейсов, разработанная в Facebook. В отличии от RESTful обладает строгой типизацией аргументов и возвращаемых полей, имеет возможность определить поля, необходимые в ответе и довольно широкую поддержку в сообществе разработчиков [16].

По результатам анализа SOAP был отвергнут поскольку в требованиях заявлен JSON, а RESTful отвергнут по причине усложнённой реализации строгой типизации и гибкого формата ответа от сервера.

1.4.2 Выбор среды выполнения

Поскольку ранее был сделан выбор в пользу последней версии фреймворка Laravel, который требует минимальную версию PHP 7.1.3, было принято решение виртуализировать среду разработки из-за сложностей в настройке одновременного функционирования нескольких версий PHP.

Рассмотрим популярные методы виртуализации в Unix системах.

KVM: При данном типе виртуализации требуется аппаратная поддержка виртуализации, глубоко виртуализирует всю операционную систему и имеет больший объём образов, в сравнении с Docker.

Docker: Данный тип виртуализации не требует аппаратной поддержки виртуализации, имеет декларативный формат конфигурации и легко переносится не только между unix-подобными системами, но и Mac и Windows.

По результатам анализа было решено виртуализировать PHP последней версии в контейнере Docker.

1.5 Итоги анализа

Из обзора бизнес-модели, заявленных существующих программных реализаций и требований, формируется общее понимание вектора проведения работ по реализации проекта. Выбраны основные инструменты и подходы к разработке и определены средства работы с средой выполнения.

2 Реализация и документация

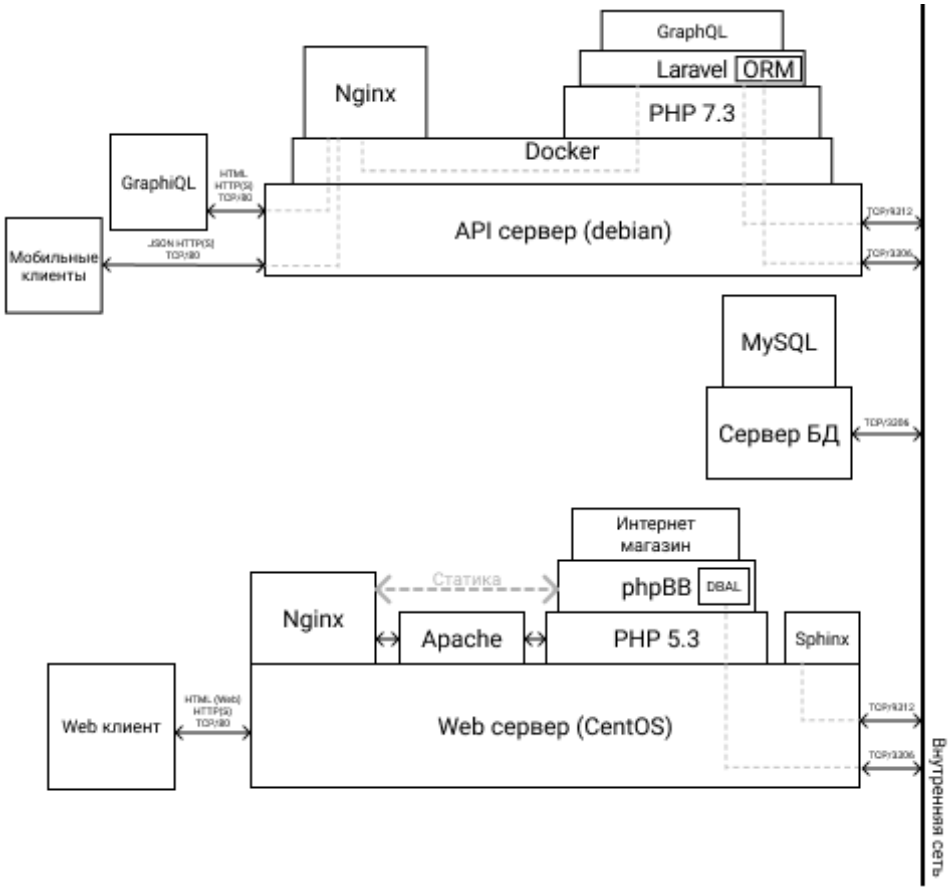

Рисунок 4 – Диаграмма архитектуры системы

На рисунке 4 предоставлена диаграмма текущей и планируемой системы. Существует три сервера, на одном из них исполняется интернет-магазин, на втором находится сервер базы данных, на третьем планируется разработка требуемого приложения.

2.1 Реализация среды выполнения

Для реализации среды выполнения (Docker, PHP 7.3) была выбрана мультиконтейнерная сборка на основе Docker compose [17]. Объявляются два сервиса: nginx, который будет обрабатывать запросы и app, внутри которого будет настроен PHP-FPM и сам PHP.

Листинг 1 – Конфигурация контейнера NGINX в Docker compose

```
nginx:
    build:
      context: ./docker/nginx
    restart: always
    working_dir: /usr/share/nginx/html
    environment:
      FPM_HOST: *php-fpm-service
      FPM_PORT: *php-fpm-port
      ROOT_DIR: '/app/public'
    volumes:
    - ./src:/app:ro
    depends_on:
    - *php-fpm-service
      network:
        ipv4_address: 192.168.10.5
```

На листинге 1 представлены настройки запуска контейнера nginx: контекст сборки, автоматический перезапуск контейнера, рабочая директория и переменные среды окружения для контейнера. Кроме того, там заданы IP-адреса и сеть, к которой подключен контейнер, и подключена директория из главной операционной системы с файлами проекта. Это нужно для того, чтобы в случае получения запроса на статический ресурс, который не требует исполнения PHP, сервер смог вернуть его из этой директории. Аналогичная настройка произведена и для образа «app», кроме того, в него добавлены настройки для прав записи в рабочей директории проекта – для создания кеш-файлов и автоматической генерации кода из командной строки контейнера.

Образ «app» будет основан на образе официального репозитория разработчиков языка – php:fpm-alpine [18]. В образ добавлена корректная настройка часового пояса, обновление и установка всех зависимостей проекта, запуск па-

кетного менеджера с задачей на установку пакетов, создание псевдонимов команд для удобной работы с консолью и требуемые расширения для PHP, такие как mysql, mbstring, opcache, iconv, bz2, xdebug и другие. Все отладочные выводы php-fpm и php перенастроены в /proc/self/fd/2 для корректной обработки службой Docker. Фоновый режим работы php-fpm отключен, для избегания возможных проблем с обработкой падений контейнера. Пул php-fpm настроен на прослушивание localhost порта, переданного из конфигурации docker-compose. Пользователь-исполнитель скриптов, и его группа так же объявлена в конфигурации менеджера процессов FastCGI из конфигурации в docker-compose для корректной работы с правами доступа к директории рабочего проекта.

Образ «nginx» будет основан на образе официального репозитория разработчиков реверсивного прокси – nginx:1.15.3-alpine [19]. В образ добавлена настройка конфигурации сервера, обработки ошибок и настройки для PHP-FPM режима. Кроме того, поскольку предполагается получать ответы по JSON, стандартные страницы ответов в случае ошибок 4xx и 5xx переопределены в формате JSON. Все отладочные выводы по аналогии с вышеупомянутым образом «php-fpm» выведены в /proc/self/fd/2. Так же отключен фоновый режим работы по вышеописанным причинам. Для лучшей отладки в формате логов сделана модификация, которая добавляет время исполнения кода в лог. Настроены CORS заголовки для правил доступа к API с других доменов сайта. SSL и gzip выключены, поскольку перед NGINX в Docker настроен реверсивный прокси-сервер, который сжимает ответы и добавляет проверку безопасности, и во внутренней сети сервера сжатие и SSL лишь усложняют схему взаимодействия.

Поскольку в обоих образах настроена общая сеть «network», и порт, на котором работает php-fpm, передаётся как yaml переменная конфигурации «docker-compose.yml» в оба контейнера, исполняемые программы в контейнерах имеют возможность взаимодействовать между собой. В случае необходимости порт взаимодействия и сеть можно изменить в конфигурации Docker compose.

2.2 Реализация механизмов работы с данными

Поскольку был использован фреймворк, нет необходимости продумывать архитектуру взаимодействия объектов в информационной системе. Однако для работы с GraphQL требуется создать структуру, описанную в спецификации. Для удобной работы с языком запросов воспользуемся библиотекой folklore/graphql, в которой уже описаны парсеры[3] и сборщики ответов, спроектированные под выбранный фреймворк. Сущности в этой библиотеке разделены по своим предназначениям – запросы (Query), мутации (Mutation), Типы (Type) и перечисления (ENUM).

Для каждой сущности необходимо создать модель работы с Eloquent ORM – встроенной в Laravel технологии связывания базы данных и ООП. На Листинге 1 представлен краткий пример примитивной модели для работы с таблицей smiles, которая имеет первичный ключ smiley_id и столбцы code, emotion, smiley_url и другие. Название таблицы присваивается переменной $table, первичный ключ присваивается переменной $primaryKey, а список столбцов, доступных для редактирования, объявляется как массив строк в переменной $fillable

Листинг 2 – Пример описания модели работы с Eloquent ORM

```
class Smile extends AbstractBaseModel{
   protected $table = 'smilies';
   protected $primaryKey = 'smiley_id';
   protected $fillable = ['code','emotion','smiley_url'];
}
```

[3] Парсер — часть программы, преобразующей входные данные (как правило, текст) в структурированный формат [20].

После создания моделей определяются типы GraphQL, которые описывают поля, название и документацию к этому типу.

В полях описываются данные, которые могут быть возвращены в стандартных типах используемой библиотеки, реализованных из спецификации языка:

– «Int» для целочисленных значений
– «Float» для числовых значений с плавающей точкой
– «String» для строковых значений
– «Boolean» для логических значений
– «ListOf» для обозначения массива типа
– «NotNull» описывает обязательность ненулевого значения этого поля. Для методов это значит, что метод не будет исполнен без этого аргумента, а для типов, что в случае отсутствия значения будет выброшена ошибка исполнения.

На листинге 3 представлена лаконичная версия описания типа GraphQL. Создаётся класс, унаследованный от базового класса типа, который предоставлен библиотекой. В переменную $attributes передаются пары ключ – значение, которые описывают уникальные признаки этого типа. Например, ключ «name» опишет имя этого типа, а «description» содержит в себе документацию этого типа. В методе «fields» возвращается массив пар ключ – значение, которые в ключе хранят название поля, а в значении передаётся массив уникальных признаков этого поля. Этот массив может содержать ключ «type», значение которого является одним из вышеупомянутых стандартных типов, либо иметь нестандартный тип используя конструкцию GraphQL::Type('Название типа'). Кроме того, этот массив также может содержать ключ «description» для документации поля. Если возникла ситуация, при которой необходимо, чтобы этот тип был аргументом метода, можно дополнительно описать значение ключа «defaultValue», которое позволяет передать в функцию-обработчик значение по умолчанию, если запрос был выполнен без этого поля.

Листинг 3 – Пример описания типа GraphQL

```
class PriceType extends BaseType{
  protected $attributes = ['name' => 'PriceType'];
  public function fields()  {
    return [
      'amount' => ['type' => Type::float()],
      'text' => ['type' => Type::string()],
    ];
  }}
```

Кроме стандартных типов есть возможность использовать модели Eloquent в качестве возвращаемых типов. Существует возможность определять перечисления, в описании которых содержится карта соответствия перечисления.

Это удобно в случаях, когда клиент должен иметь выбор аргумента только из ограниченного количества заранее определённых значений, например, «ASC/DESC» в случае с сортировкой данных. Пример реализации такого случая представлен на листинге 4.

Листинг 4 – реализация перечисления в GraphQL

```
class SortDirEnum extends GraphQLType
{
  protected $enumObject = true;
  protected $attributes = [
    'name' => 'SortDirEnum',
    'values' => ['ASC', 'DESC'],
  ];}
```

В Запросах описывается название запроса, его описание, возвращаемый тип (только один), аргументы и функция, которая занимается обработкой запроса.

Кроме того, можно задавать отдельные обработчики для полей. В стандартной ситуации в обработчике будет выполнена проверка авторизации, сделан запрос в хранилище данных, его обработка и возврат объекта в ответ.

Листинг 5 – пример обработчика запроса

```
    public function resolve($root, $args, $context,
ResolveInfo $info){
        $validator = Validator::make($args,['phone' => [new
PhoneNumber, new PhoneTaken, new PhoneTooMany]]);
        $phone = $args['phone'];
        $code = Phone::code();
        if ($validator->fails())
            throw with(new ValidationError('validation'))-
>setValidator($validator);
        $msg = ['to' => $phone,'content' =>
trans('graphql.phone_sms', ['code' => $code])];
        try {SMS::send($msg);} catch (Exception $e) {throw
new
SMSServiceUnavailable(trans('graphql.errors.phone_service_unav
ailable'));   }
        Phone::save($phone, $code);
        return trans('graphql.phone_success');
    }
```

Рассмотрим пример обработчика запроса отправки смс-сообщения проверки, представленный на листинге 5. Сперва создаётся обработчики корректности формата введённых данных, называемые валидаторами. Валидация – встроенная функциональность Laravel, позволяющая гибко определить правила проверки и использовать пакеты локализации для человеко-понятной выдачи ответа от сервера.

В валидацию передаются аргументы метода и описанные правила проверки – PhoneNumber для проверки соответствию формата поддерживаемых номеров, PhoneTaken для проверки занятости номера и PhoneTooMany для контроля частоты запросов номера. После этого осуществляется вызов статического метода code класса Phone бизнес-логики, который генерирует случайный код для проверки. Далее осуществляется проверка валидности переданных значений, создаётся структура высылаемого ответа и с помощью внешнего модуля SMS производится отправка сообщения, а клиенту отправляется сообщение о успешной отправке. Если в процессе валидации или отправки возникли ошибки, выполнение метода прекратится и клиенту будет выслан текст ошибки.

2.3 Реализация аутентификации

Для авторизации добавлен запрос входа и регистрации, который возвращает JWT [21], который клиент обязан прилагать при каждом запросе, подразумевающем аутентификацию.

Чтобы аутентификация происходила в соответствии с PHPBB, были переписаны правила авторизации Laravel, главные цели которых – определить текущего пользователя по JWT или Cookies. Кроме того, был переопределён механизм сессий Laravel. Был в точности повторён механизм создания хэш-суммы пароля оригинального фреймворка PHPBB, нестандартные функции сверки идентичности хешэй и метод эмуляции сессии пользователя в исходном хранилище PHPBB.

Благодаря этому переопределению можно отмечать пользователя онлайн при каждом новом обращении в этой сессии, а также прекращать сессию с другого устройства. Для корректной работы с JWT была использована популярная библиотека tymon/jwt-auth [22], которая позволяет осуществлять шифрование.

Листинг 6 – функция распознавания пользователя по JWT токену.

```
    public function fromToken($token){try {$userInfo = $this-
>decode($token);}
    catch ($invalidException) { throw new \('Broken api
token.');}
        if (false !== Arr::get($userInfo, 'user_id',
USER::ANONYMOUS_ID))
            return $this->resolveExistingUser($userInfo);
        return $this->anonymous();}
```

На листинге 6 представлена функция, которая получает токен JWT и возвращает пользователя системы: сперва декодируется содержимое токена для определения сессии пользователя и его идентификатора. Если эта информация не может быть извлечена, клиенту возвращается ошибка. После успешного извлечения информации происходит распознавание пользователя в системе, которое представлено на Листинге 7.

Листинг 7 – метод распознавания пользователя в системе.

```
    private function resolveExistingUser($userInfo){
        [$id, $password, $session] = [
            (int)Arr::get($userInfo, 'user.id'),
            (string)Arr::get($userInfo, 'user.password'),
            (string)Arr::get($userInfo, 'session')];
        $user = User::where('user_id', $id)
    ->where('user_password', $password)->first();
        if (!$user->tokenCheck($session))
    throw new \Exception('Invalid remember token');
        if (!$user) throw new \Exception('Invalid user
credentials.');    return $user;}
```

На представленном выше листинге выполняется поиск пользователя в базе данных используя модель User, после чего происходит проверка сессии пользователя и возврат объекта пользователя в случае успеха.

2.4 Реализация авторизации и локализации

В случае неавторизированного доступа или не валидного значения поля в запросе выбрасываются специфические исключения GraphQL, которые в зависимости от своего типа попадают в ошибки выполнения, или «ошибки, которые пользователь ожидает увидеть». Правила возвращения этих исключений описаны в главе 1.1

Для интуитивно понятных покупателю описаний ошибок был использован расширен встроенный механизм локализации, в который введён новый класс возвращаемых полей для исключений GraphQL. Встроенные механизмы локализации позволяют использовать подход шаблонизирования строк, таким образом можно определить языковую конструкцию и подставлять в неё значения, которые переданы пользователем.

2.5 Реализация поиска товаров и кеширования

При реализации поиска по товарам с использованием поискового движка Sphinx возникли дополнительные трудности, поскольку индекс движка не рассчитан на фильтрацию по параметрам, к примеру «красные носки», поэтому было принято решение обрабатывать выдачу поиска вручную – запрашивается некоторое количество товаров по возможным для индекса фильтрам, после чего идёт разбор всех параметров, группировка и сортировка по количеству встречаемости и отсечение выдачи фильтров по медианному значению количества встреч.

Из-за того, что этот процесс является весьма трудоёмким и занимает большое количество времени, было реализовано кеширование вычислений, что позволило ускорить выдачу поиска через программный интерфейс.

Кеширование реализовано стандартными средствами Laravel и осуществляется с помощью методов add и get фасада Cache. В методе add можно определить ключ хранения данных и время хранения данных. По истечению времени хранения планировщик Laravel удалит истекшие кеш-данные из хранилища. По умолчанию было использовано стандартное хранилище кеш-данных на файловой системе в рабочей папке проекта.

2.6 Реализация графического интерфейса

Поскольку GraphQL предоставляет удобные методы для документации методов и типов, было бы удобно предоставить разработчикам интерфейс для тестов и знакомства с документацией. Благодаря тому, что GraphQL – это единая спецификация, то все решения, написанные для этого языка взаимодействия, подойдут для реализованного программного интерфейса. Воспользуемся веб-клиентом Graph*i*QL [23] (Рисунок 4).

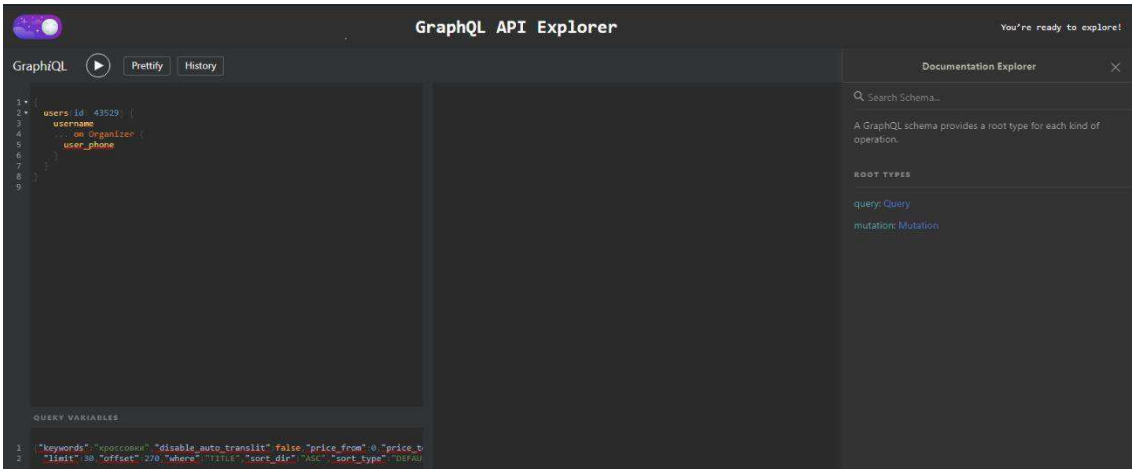

Рисунок 4 – Визуальный интерфейс Graph*i*QL [24]

Данный интерфейс позволяет:

– делать запросы;
– форматировать запросы;
– получать ответы;
– просматривать документацию программного интерфейса;
– вести историю запросов.

Все вышеописанные операции могут быть совершены прямо из браузера разработчика без необходимости установки дополнительного ПО.

2.7 Инструкция пользователя

Для взаимодействия с веб-клиентом программного интерфейса необходимо открыть путь «example.com/graphiql», заменив «example.com» на название домена, на котором функционирует программный интерфейс.

В левом поле можно писать запрос к интерфейсу, а при нажатии на кнопку Play запрос будет исполнен. Результат выполнения запроса будет отображен в правой колонке. В случае необходимости запрос можно форматировать по отступам с помощью кнопки «Prettify», историю запросов можно просмотреть, нажав кнопку «History». В меню истории можно добавлять запрос в избранное, нажав символ звёзды. При нажатии на запрос он будет вставлен в редактор как активный. Документацию к интерфейсу можно посмотреть во вкладке «Docs» слева сверху. Эта вкладка имеет поле для поиска, которое срабатывает по нажатию клавиши Enter на клавиатуре. При нажатии на типы принимаемых или возвращаемых аргументов и на типы запроса можно попасть на уровень вглубь, чтобы исследовать сущность детальнее.

2.8 Инструкция разработчика

Для запуска проекта нужно иметь Docker [25] и Docker Compose [17]. После установки перейдите в рабочую папку проекта, скопируйте файл «.env.example» в файл «.env» и измените требуемые поля в окружении. После запустите команду «docker-compose up --build -d», которая соберёт образы Docker и запустит проект.

По умолчанию проект станет доступен по HTTP с портом 9999, эти настройки можно изменить в файле docker-compose.yml

ЗАКЛЮЧЕНИЕ

В результате проделанной работы был спроектирован и реализован программный интерфейс интернет-магазина.

Разработана структура интерфейса, объекты и их взаимосвязи. Настроено рабочее окружение таким образом, чтобы его можно было удобно использовать на большинстве современных операционных систем.

В разработанном интерфейсе присутствуют недостатки, исправить их можно благодаря будущей доработке, путем добавления следующих возможностей:

– Автоматические тесты корректности выполнения задач

– Кеширование без использования файловой системы

Последняя версия приложения размещена в репозитории на GitHub [26].

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Эквайринг [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Эквайринг (дата обращения: 27.04.2019)
2. Фреймворк [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Фреймворк (дата обращения: 27.04.2019)
3. PHPBB Free and Open Forum software [Электронный ресурс]. – Режим доступа: https://www.phpbb.com/ (дата обращения: 27.04.2019)
4. MySQL [Электронный ресурс]. – Режим доступа: https://www.mysql.com/ (дата обращения: 27.04.2019)
5. Sphinx | Open Source Search Engine [Электронный ресурс]. – Режим доступа: http://sphinxsearch.com/ (дата обращения: 27.04.2019)
6. Debian -- Универсальная Операционная Система [Электронный ресурс]. – Режим доступа: https://www.debian.org/index.ru.html (дата обращения: 27.04.2019)
7. The Apache HTTP Server Project [Электронный ресурс]. – Режим доступа: https://httpd.apache.org/(дата обращения: 27.04.2019)
8. Nginx [Электронный ресурс]. – Режим доступа: https://nginx.org/ru/ (дата обращения: 27.04.2019)
9. JavaScript Object Notation [Электронный ресурс]. – Режим доступа: https://www.json.org/ (дата обращения: 27.04.2019)
10. Zend Framework [Электронный ресурс]. – Режим доступа: https://framework.zend.com/ (дата обращения: 27.04.2019)
11. Official Facebook group for Yii PHP Framework [Электронный ресурс]. – Режим доступа: https:/fb.com/groups/yiitalk/permalink/10156754090812150/ (дата обращения: 27.04.2019)
12. Symfony [Электронный ресурс]. – Режим доступа: https://symfony.com/ (дата обращения: 27.04.2019)
13. Laravel - The PHP Framework For Web Artisans [Электронный ресурс]. – Режим доступа: https://laravel.com (дата обращения: 27.04.2019)

14. REST [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/REST (дата обращения: 27.04.2019)

15. SOAP [Электронный ресурс]. – Режим доступа: https://www.w3.org/TR/soap/ (дата обращения: 27.04.2019)

16. GraphQL | A query language for your API [Электронный ресурс]. – Режим доступа: https://graphql.org/ (дата обращения: 27.04.2019)

17. Docker Compose [Электронный ресурс]. – Режим доступа: https://docs.docker.com/compose/ (дата обращения: 27.04.2019)

18. PHP – Docker hub [Электронный ресурс]. – Режим доступа: https://hub.docker.com/_/php (дата обращения: 27.04.2019)

19. NGINX – Docker hub [Электронный ресурс]. – Режим доступа: https://hub.docker.com/_/nginx (дата обращения: 27.04.2019)

20. Парсер [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Парсер (дата обращения: 27.04.2019)

21. JSON Web Tokens - jwt.io [Электронный ресурс]. – Режим доступа: https://jwt.io (дата обращения: 27.04.2019)

22. JSON Web Token Authentication for Laravel & Lumen [Электронный ресурс]. – Режим доступа: https://github.com/tymondesigns/jwt-auth (дата обращения: 27.04.2019)

23. An in-browser IDE for exploring GraphQL [Электронный ресурс]. – Режим доступа: https://github.com/graphql/graphiql (дата обращения: 27.04.2019)

24. GraphQL API Explorer | GitHub Developer Guide [Электронный ресурс]. – Режим доступа: https://developer.github.com/v4/explorer/ (дата обращения: 27.04.2019)

25. Docker: Enterprise Application Container Platform [Электронный ресурс]. – Режим доступа: https://www.docker.com/ (дата обращения: 27.04.2019)

26. GraphQL API [Электронный ресурс]. – Режим доступа: https://www.github.com/zulcom/api (дата обращения: 27.04.2019)

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой ВТ
_____ О.В. Непомнящий
подпись инициалы, фамилия
«27» 06 2019 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование направления

Интернет-магазин. Серверная часть.
тема

Руководитель	Коршун 25.06.19. подпись, дата	доц. канд.физ.-мат.наук должность, ученая степень	К.В. Коршун инициалы, фамилия
Выпускник	25.06.19 подпись, дата		С.А. Муравьёв инициалы, фамилия
Нормоконтролер	подпись, дата 26.06.19	доцент, канд. техн. наук должность, ученая степень	В.И. Иванов инициалы, фамилия

Красноярск 2019