МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра Математического и компьютерного моделирования

Разработка и реализация информационной системы «магазин электроники»

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 441 группы

направление 09.03.03 — Прикладная информатика

механико-математического факультета

Печенова Дениса Сергеевича

Научный руководитель
профессор, д.э.н., профессор Л.В. Кальянов

Зав. кафедрой
зав. каф., д.ф.-м.н., доцент Ю.А. Блинков

Саратов 2023

Введение. Развитие электронной коммерции привело к росту спроса на разработку информационных систем, которые предоставляют покупателям простой и удобный способ приобретения товаров. Однако, чтобы оставаться конкурентоспособными на этом быстро меняющемся рынке, владельцы площадок электронной коммерции должны иметь эффективный и действенный способ управления своими запасами, продажами и взаимодействием с клиентами. Одним из способов достижения этой цели является разработка и внедрение информационной системы.

Основной целью работы является проектирование и разработка информационной системы, которая улучшит общую производительность магазина. Эта система будет включать различные функции, такие как управление взаимоотношениями с клиентами, обработка заказов и обработка платежей. Внедрив такую систему, предприятие сможет оптимизировать свою деятельность, сократить количество ошибок, допускаемых сотрудниками магазина, и повысить уровень удовлетворенности клиентов. Для достижения этой цели необходимо выполнить следующие задачи:

- Определить ключевые требования к эффективной информационной системе для магазина.
- Провести анализ предметной области.
- Определить требования к средствам разработки.
- Выбрать средства разработки.
- Спроектировать и разработать прототип предлагаемой системы.

Актуальность работы очевидна в условиях современных рыночных тенденций, когда переход к электронной коммерции продолжает ускоряться. Эффективная информационная система для компании может обеспечить конкурентное преимущество за счет улучшения общего опыта покупок для клиентов. Кроме того, анализ средств и методов разработки может внести вклад в существующий объем знаний по электронной коммерции, предоставив представление о проектировании и разработке информационной системы для электронной коммерции.

Для реализации подобного проекта требуется проанализировать технологии разработки, обосновать решение применения выбранной технологии. После этого подробно описать процесс разработки, включая проектирование и

архитектуру приложения, реализацию с использованием выбранных средств. Кроме того, обсудить важность следования четко определенному процессу разработки программного обеспечения и соблюдения лучших практик, таких как повторное использование кода, масштабируемость и сопровождаемость.

В целом, полученные выводы могут быть полезны владельцам площадок электронной коммерции, стремящимся улучшить работу своего предприятия, и разработчикам, заинтересованным в создании подобных приложений. Также подчеркивается важность соблюдения систематического и итерационного процесса разработки программного обеспечения для создания высококачественных и надежных программных систем.

Основное содержание работы

Первый раздел посвящен анализу предметной области, для дальнейшего проектирования информационной системы. В начале описывается сам процесс анализа, далее подробно рассматриваются этапы составляющие этот процесс.

- Первым шагом в этом процессе является определение требований клиента. Это предполагает сбор информации о пользователях системы, их потребностях и бизнес-процессах, которые будет поддерживать система.
- Далее создаётся концептуальная модель поведения системы, построением диаграммы «сущность-связь», вариантов использования. Основная цель этого шага понять высокоуровневое представление функциональности системы.
- Следующим шагом после создания концептуальной модели требуется написать подробную функциональную спецификацию, описывающую поведение системы.
- Последним этапом анализа предметной области является проверка на соответствие системы требованиям. Обычно для этого требуется создать тестовые примеры, которые позволяют проверить это соответствие.

В первом подразделе определяются требования клиента, что является первым этапом в построении информационной системы. Вся работа информационной системы во многом зависит от того, насколько хорошо она удовлетворяет потребности пользователей и требованиям заказчика. Собирая и ана-

лизируя требования клиента, становится возможно разработать информационную систему, соответствующую потребностям и предпочтениям конечных пользователей, что приведет к повышению эффективности работы предприятия клиента. Установим требования клиента, обычно они заключаются в следующем:

- Информационная система должна обладать удобным визуальным интерфейсом, в котором легко ориентироваться и пользователям, и владельцам.
- Интернет-приложение должно иметь хорошо организованный каталог товаров, который позволяет пользователям искать и просматривать товары по категориям, брендам, ценовому диапазону и другим критериям.
- Приложение должно иметь подробные описания и характеристики каждого товара, включая особенности, технические характеристики и изображения, это поможет покупателям выбрать подходящий товар.
- Площадка электронной коммерции должна иметь корзину, которая позволяет покупателям добавлять товары в корзину, обновлять их количество и переходить к оформлению заказа. Процесс оформления заказа должен быть простым и безопасным.
- Интернет-магазин должен иметь систему управления запасами, которая отслеживает уровень запасов каждого продукта и предупреждает вас о наличии товара на складе.
- Площадка должна быть безопасной. В ней должны быть предусмотрены меры по защите данных клиентов и предотвращению несанкционированного доступа или действий мошенников.
- Интернет-магазин должен иметь систему управления заказами, которая позволяет управлять заказами клиентов, получать информацию о состоянии их заказов.

Во втором подразделе строится концептуальная модель поведения системы, создаётся диаграмма «сущность-связь» с помощью языка UML. Язык UML - это стандартизированный язык, используемый для моделирования программных систем. Он используется для визуального моделирования, позволяет разработчикам создавать диаграммы и модели, представляющие структуру и поведение программной системы.

Он был разработан в конце прошлого века группой экспертов по разработке программного обеспечения как способ позволяющий стандартизировать множество различных подходов к моделированию. На сегодняшний день, этот язык очень частно применяется во время работы разработчиками программного обеспечения, архитекторами для создания визуальных моделей информационных систем.

Одним из основных преимуществ использования UML при разработке является то, что благодаря этому обеспечивается стандартизированный способ создания визуальных моделей, которые, в свою очередь, могут улучшить у разработчиков понимание системы, которую они создают. Эти модели также позволяют определить потенциальные проблемы или области для улучшения, а также донести информацию о системе до других. Диаграммы UML также могут использоваться для определения требований, проектирования архитектуры программного обеспечения и автоматической генерации кода.

Для создания «Entity-relationship diagram» нужно пошагово сделать следующее:

- Определить сущности, которые имеют отношение к системе. Например сущности: «пользователи», «продукты», «заказы».
- Определить атрибуты сущности. Например сущность «клиент» может иметь такие атрибуты как: номер клиента, имя, адрес электронной почты, телефон, адрес места жительства.
- Определить связи между сущностями. Например, существует связь между сущностями «пользователи» и «заказы», ведь каждый заказ связан с определённым клиентом.
- Определить вид связи. Он описывает количество экземпляров каждой сущности, которые могут быть связаны с экземплярами другой сущности. В частности, у «пользователя» может быть много заказов, но каждый «заказ» связан только с одним «пользователем».
- Создать диаграмму, используя язык UML.

В третьем подразделе создаётся диаграмма вариантов использования. Диаграмма вариантов использования - это тип диаграммы UML, которая позволяет увидеть функциональные требования системы, отображая взаимодействие между участниками и системой. Она обеспечивает чёткий и лако-

ничный способ донесения функциональности системы до заинтересованных сторон - заказчиков информационной системы и её разработчиков.

На этом виде диаграмм действующие лица схематично изображаются в виде людей, а варианты использования - в виде овалов. Стрелки между действующими лицами и вариантами использования указывают на взаимодействие между ними. Диаграмма вариантов использования может также включать границы системы, которые указывают на границы проектируемой системы и внешних систем.

Участники системы или «Акторы» представляют пользователей или внешние системы, которые взаимодействуют с проектируемой системой, а варианты использования представляют действия или задачи, которые могут быть выполнены этими участниками.

Созданная «use case» диаграмма должна отвечать на вопросы:

- Каковы основные функции системы?
- Какие действующие лица задействованы в системе?
- Каковы различные варианты использования системы?

Исходя из требований клиента, можно установить, что интернет-магазин имеет клиентов, товары, заказы и платежи.

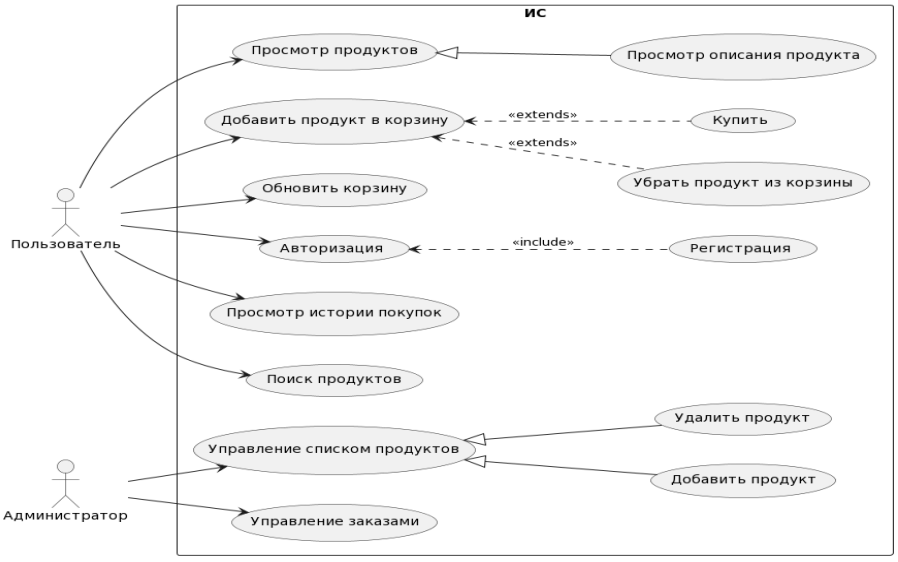

Рисунок 1 — Диаграмма вариантов использования

В соответствии с рисунком 1 можно описать действующих лиц диаграммы. Она включает двух акторов, актор «Клиент» представляет пользовате-

ля информационной системы, который хочет посмотреть и приобрести товары в интернет-магазине. Актор «Администратор» соответственно обозначает администраторы информационной системы, который управляет интернет-магазином, добавляет или исключает товары из списка товаров, обрабатывает заказы. Полученная диаграмма также отображает следющие варианты использования для Пользователя:

- Пользователь может просматривать продукты.
- Искать продукты на основе ключевых слов, категорий или других критериев.
- Пользователь также может прочитать описание продукта.
- Добавить продукт в корзину.
- Удалить продукт из корзины.
- Обновить количество товара в корзине.
- Пользователь также может оформить заказ.
- Он может зарегистрироваться в системе.
- Также пользователь может авторизироваться в системе.

Варианты использования Администратора отображают его возможность управлять списком продуктов, добавляя и удаляя продукты, а также упправлять заказами.

Следует отметить также виды связи между вариантами использования. Например, вариант использования «убрать продукт из корзины» расширяет вариант использования «добавить продукт в корзину», сообщая о том, что после добавления продукта в корзину его возможно удалить из корзины, но удалить продукт из корзины, без добавления его туда - нельзя. Вариант использования «Авторизация» включает вариант использования «Регистрация», ведь авторизироваться без регистрации нельзя.

Второй раздел посвящен выбору и сравнению разных средств разработки, языков программирования, которые потребуются для создания информационно системы. В первом подразделе определяются требования к средствам разработки. Реализация проекта подразумевает создание комплексного веб-приложения, для которого требуется язык программирования, фреймворк с несколькими ключевыми функциями и возможностями, а в некоторых случаях и модель объектно-реляционного отображения. Во втором подразделе

анализируются положительные и отрицательные стороны языка программирования Java и фреймворка Spring, положительные и отрицательные стороны языков программирования и фреймворков Python и Django, C# и ASP.NET Core. Сравниваются SQL и NoSQL базы данных и рассматриваются их положительные и отрицательные стороны.

Третий раздел посвящен созданию информационной системы для предприятия по продаже электроники. В первом подразделе анализируется платформа ASP.NET Core. Частью это платформы является встроенная реализация паттерна MVC.

MVC — паттерн разработки, означающий разделение функциональностей, в противовес нечётким архитектурам, которые были до него, например в ранних версиях ASP.NET использовались web-страницы на html и http, но после того как этот фреймворк пересобрали под платформу .NET Core он был перестроен на открытую, расширенную кросс-платфомренную основу.

В данный момент этот паттерн стал не так важен из-за развития spa (single-page applications), приложений, где выполняется только один http запрос и система получает только один html документ, который отправляет многофункциональный клиент например Angular или React, написанные на JavaScript. Несмотря на это, поддержка паттерна MVC остаётся до сих пор.

MVC и SPA имеют некоторые недостатки, например их долго разрабатывать; нечёткие архитектуры, где весь код «перемешан», этого недостатка лишены и создание простого приложения может занять пару часов. В противовес этому, даже маленькое приложение с MVC архитектурой и SPA требует много времени на доведение до рабочего состояния. Razor берёт паттерн нечётких архитектур и реализует его с использованием функции платформы, изначально разработанных для MVC framework. Код и содержимое страницы смешиваются, образуя автономные страницы, а это повышает скорость разработки. Razor и MVC можно использовать вместе. Например писать основную логику на MVC, а дополнительные функции через Razor.

Во втором подразделе анализируется структура программы, реализации паттерна MVC на платформе ASP.NET Core. MVC расшифровывается как Model-View-Controller, этот паттерн подразумевает разделение кода приложения на три отдельных компонента: модель, представление и контроллер.

В ASP.NET Core модель представляет данные и бизнес-логику приложения. Модель обычно состоит из классов, которые определяют структуру и поведение данных, которыми управляет приложение.

Представление в ASP.NET Core определяет пользовательский интерфейс приложения. Оно отвечает за отображение данных модели для пользователя и получение пользовательского ввода. Представления обычно определяются с помощью HTML, CSS и синтаксиса Razor, который представляет собой язык шаблонов, позволяющий динамически отображать содержимое.

Контроллер в ASP.NET Core действует как посредник между моделью и представлением. Он получает пользовательский ввод от представления, обрабатывает его и обновляет модель и представление по мере необходимости. Контроллеры обычно определяются как классы, которые обрабатывают HTTP-запросы и ответы.

Итак, когда поступает запрос, он направляется в класс контроллера. Этот класс по желанию получает некоторые данные через модель или получает данные из запроса в качестве модели, и эта модель может взаимодействовать с базой данных. Затем эти данные отправляются обратно в контроллер для выполнения любой логики, которую необходимо выполнить. Затем контроллер отправит эти данные в представление, а представление отобразит их и вернет в качестве ответа пользователю.

В третьем подразделе анализируются классы связанные с паттерном Repository, который упрощяет работу с базой данных и Entity Framework Core. Паттерн Repository - это шаблон проектирования, используемый при разработке программного обеспечения для отделения уровня доступа к данным от уровня бизнес-логики приложения. Целью этого паттерна является создание уровня абстракции между приложением и базой данных, что облегчает сопровождение и тестирование приложения.

В шаблоне Repository уровень доступа к данным представлен репозиторием, который предоставляет набор методов для доступа и манипулирования данными в базе данных. Репозиторий выступает в качестве посредника между приложением и базой данных, предоставляя приложению интерфейс для взаимодействия с базой данных без необходимости разбираться с базовыми деталями реализации.

Entity Framework Core — это популярная система объектно-реляционного отображения или ORM для приложений .NET. Эта система позволяет разработчикам определять схему базы данных с помощью классов C#, которые затем отображаются на таблицы базы данных во время выполнения, другими словами она позволяет легко выполнять операции CRUD над базой данных без необходимости написания SQL-кода.

В четвёртом подразделе создаётся простой API или интерфейс прикладного программирования - это набор протоколов, процедур и инструментов, позволяющих программным приложениям взаимодействовать друг с другом. Он определяет, как разные программные компоненты должны взаимодействовать друг с другом, обеспечивая стандартизированный способ обмена данными и функциональностью для приложений.

API обычно реализуется как набор методов в контроллере. Эти методы называются Action или действиями и отвечают за обработку входящих HTTP-запросов и возврат соответствующего ответа.

Чтобы вызвать действие, обычно необходимо отправить HTTP-запрос на конечную точку API с помощью такого инструмента, как веб-браузер, утилита командной строки, например cURL. Точный формат запроса зависит от HTTP-глагола, используемого действием API, и параметров, требуемых действием.

Например, если API имеет действие под названием «GetCustomerById», которое принимает «ID» клиента в качестве параметра и возвращает соответствующие данные клиента в формате JSON, то становится возможно вызвать это действие, отправив HTTP GET запрос на URL конечной точки API с параметром «ID» клиента, включенным в строку запроса.

В пятом подразделе рассматривается готовая информационная система, её пользовательский интерфейс.

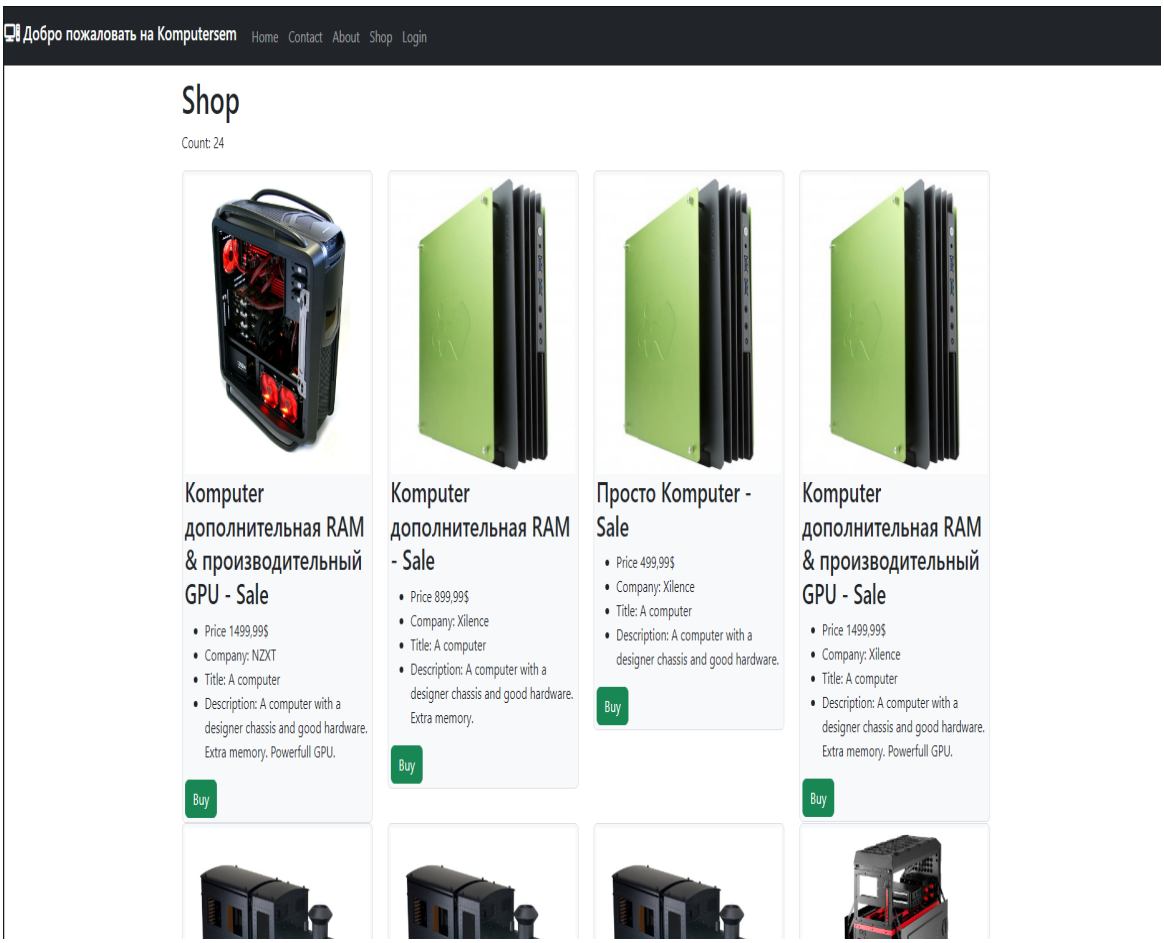

Рисунок 2 — Пользовательский интерфейс информационной системы

Внешний вид созданной информационной системы показан в соответствии с рисунком 2

Заключение. В ходе работы была проанализирована предметная область. При её анализе, проведённом на разных уровнях, было построено несколько UML диаграмм. Были глубоко проанализированы средства разработки информационных систем, создан прототип системы для интернет-магазина.

В ходе работы были достигнуты все поставленные задачи: определены ключевые требования к информационной системе, проведён анализ предметной области, определены требования к средствам разработки, выбраны средства разработки и спроектирован прототип информационной системы. Проведенная работа позволила понять принципы создания систем электронной коммерции, улучшить свои навыки решения задач в области проектирования таких систем.