МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**

Кафедра теории функций и стохастического анализа

# РАЗРАБОТКА АДАПТИВНОГО WEB-ПРИЛОЖЕНИЯ ИНТЕРНЕТ-МАГАЗИНА

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 451 группы
направления 38.03.05 — Бизнес-информатика

механико-математического факультета
Гнатюка Андрея Евгеньевича

Научный руководитель
ст. преподаватель _____ Н. В. Сергеева

Заведующий кафедрой
д. ф.-м. н., доцент _____ С. П. Сидоров

Саратов 2024

# ВВЕДЕНИЕ

**Актуальность работы:** актуальность выбранной темы обусловлена тем, что онлайн-коммерция становится все более распространенной с каждым годом, вдохновляя многие компании на переход к онлайн-торговле, создавая свои интернет-магазины. Это явление не случайно, поскольку развитие технологий IT способствует развитию различных сфер жизни, включая торговлю. Интернет-магазин представляет собой удобную альтернативу традиционному посещению магазина, предлагая доступ к более широкому ассортименту товаров, удобство и возможность совершать покупки в любое удобное время, включая ночное время. Это позволяет сократить время, затрачиваемое на ожидание в очередях, и обеспечивает удобство для покупателей, делая процесс покупки более эффективным и приятным

**Целью бакалаврской работы** является разработка web-приложения интернет-магазина электроники с использованием современных средств и подходов к разработке програмного обеспечения.

**Объектом исследования** являются процессы и подходы к разработке веб-приложений, включая разработку пользовательского интерфейса, выбор и реализацию архитектуры приложения, создание серверной части и проектирование схемы базы данных.

**Предметом исследования** являются ключевые аспекты разработки веб-приложения, включая выбор технологического стека, определение механизмов взаимодействия между клиентом и сервером, выбор стратегии хранения данных, обеспечение их согласованности и защиты, а также выбор методов аутентификации и авторизации пользователей.

Для достижения необходимой цели в работе, необходимо решить следующие **задачи**:

— Определение технологического стека приложения: необходимо выбрать технологический стек, который будет включать в себя как клиентскую (UI), так и серверную части, а также выбрать подходящую базу данных;

— Обеспечение высокого usability приложения: интерфейс должен быть интуитивно понятным и удобным для пользователей, обеспечивая легкий доступ к функциям поиска, добавления товаров в корзину, оформ-

ления заказа. Это включает в себя разработку интуитивно понятных форм регистрации и входа, а также удобных интерфейсов для просмотра и выбора товаров;

— Организация масштабируемой архитектуры: разработка архитектуры, которая позволит легко внедрять новый функционал в приложение по мере его расширения. Это обеспечит гибкость и возможность масштабирования приложения в соответствии с изменяющимися требованиями и потребностями пользователей;

— Обеспечение обмена данными в формате JSON по протоколу HTTP: необходимо обеспечить надежный и безопасный обмен данными между клиентом и сервером, используя протокол HTTP и формат данных JSON. Это обеспечит эффективное и безопасное взаимодействие между различными компонентами приложения;

— Проектирование и разработка схемы базы данных: разработка схемы базы данных, которая будет обеспечивать эффективное хранение и обработку данных приложения. Это включает в себя выбор подходящей СУБД и проектирование схемы, которая будет соответствовать требованиям приложения и обеспечивать высокую производительность и надежность.

**Практическая значимость** данной работы заключается в том, что сфера веб-разработки продолжает активно развиваться, предоставляя новые возможности для бизнеса и улучшения взаимодействия с клиентами. В современном мире веб-приложения становятся основой для многих сервисов, включая интернет-магазины, банковские услуги, доставку и многие другие, поскольку они обеспечивают удобство и доступность для пользователей, имеющих доступ к интернету через любое устройство. Кроме этого, многие традиционные мобильные и компьютерные приложения уже функционируют как веб-приложения, что демонстрирует тенденцию к унификации и упрощению доступа к услугам. Это подчеркивает необходимость разработки веб-приложений, способных адаптироваться к изменяющимся потребностям пользователей и бизнеса, обеспечивая высокую степень удобства и эффективности в использовании.

**Структура и содераж**ание бакалаврской работы. Работа состоит из введения, трех разделов, заключения и списка использованных источников, содержащего 20 наименований и 6 приложений. Общий объем работы составляет 56 страниц.

## ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Во **введении** обосновывается актуальность темы работы, а также формулируются цели работы.

В **первом** разделе приводится описание используемых технологий и инструментов, которые используются для реализации web-приложения.

Во **втором** разделе описывается архитектура приложения. В приложении используется чистая архитектура, которая включает в себя:

— уровень домена;
— уровень бизнес-логики;
— уровень инфраструктуры;
— уровень представления.

*Уровень домена* хранит все бизнес-сущности приложения. Важно понимать, что данный уровень не может ссылаться на вышележащие уровни. Задача верхних слоев, это инкапсуляция бизнес сущностей от объектов реального мира. Также на данном уровне определены специальные классы DTO (data transfer object), которые содержат только ту информацию, которая необходима для того или иного слоя приложения.

*Уровень бизнес-логики* содержит в себе классы-сервисы, которые отвечают за обработку данных, полученных от уровня представления, и манипулирование ими до того, как они будут представлены пользователю или сохранены в базе данных.

*Инфраструктурный уровень* содержит все необходимое для общения приложения с внешним миром (пользователями, сторонними сервисами, железом и т.д). Данный уровень может очень быстро разростаться ввиду настройки большого количества интеграций.

Инфраструктурный код позволяет соединить ядро приложения с:

— Файловой системой;
— Сетью;
— ORM;
— Брокерами сообщений.

На данном уровне не может быть никакой бизнес-логики.

*Уровень представления* представляет собой Web API и включает в себя следующие элементы:

— Контроллеры - представляют собой endpoint-ы, принимающие запросы от клиента и передающие данные для обработки нижним слоям приложения, а именно в слой Application (приложения);
— Middleware - представляют собой компонент конвеера обработки запроса, позволяя обработать запрос до того, как он попадет в метод контроллера или после того, как запрос был обработан контроллера;
— Хранение глобальных настроек приложения;

Кроме этого, проект Web API подключает к приложению все используемые в приложении зависимости, находящиеся на нижих уровнях, а также является точкой запуска приложения.

В **третьем** разделе описывается реализация web-приложения. Данный раздел включает в себя 5 подразделов:

— Реализация регистрации и авторизации пользователей.
— Реализация отображения продуктов на странице.
— Реализация отображения подробной информации о продукте.
— Реализация добавление товара в корзину.
— Реализация оформления заказа.

**Регистрации и авторизации пользователей.** Для представления пользователя в приложении реализован специальный класс.

```
public sealed class User : IAuditable
{
    public Guid Id { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string Login { get; set; }
    public string Password { get; set; }
    public Cart Cart { get; set; }
    public UserToken UserToken { get; set; }
    public IEnumerable<Order> Orders { get; set; }
    public DateTime CreatedAt { get; set; }
    public DateTime? UpdatedAt { get; set; }
}
```

Процесс авторизации и аутентификации пользователя в приложении реализован на основе Bearer авторизации с использованием access и refresh токенов. Использование данного механизма авторизации позволяет пользователю длятельное время иметь доступ к защищенным ресурсам, а также иметь возможность выполнять действия в приложения, требующие авторизации без необходимости повторого входа в аккаунт.

*Access - токен* - это короткоживущий токен, который используется для получения доступа к ресурсам, требующие авторизацию. В зависимости от приложения, время жизни данного токены может составлять от нескольких минут до нескольких часов, в зависимости от нужного уровня защиты данных пользователя.

*Refresh - токен* - это долгоживущий токен, который используется для обновления access токена пользователя. Срок жизни refresh токена может составлять от нескольких дней до нескольких месяцев. После истечения времени жизни refresh токена, приложение запрашивает повторную авторизацию и выдает новую пару access и refresh токенов.

Данные токены хранятся в cookie на стороне пользователя и имеют флаг HttpOnly, что позволяет избежать межсайтового скриптинга (XSS - внедрение вредоносного клиентского кода), делая недопустимым получение данных кук через JavaScript.

В приложении на стороне клиента определены специальные формы для регистрации/авторизации. Для отправки данных форм на клиенте опредены специальные методы, которые отправляют запрос для обработки на сервер.

На стороне сервера определены специальные конечные точки, которые обрабатывают данные запросы (на регистрацию/авторизацию). Данные конечные точки вызывают соответствующие методы сервиса, которые отвечают за логику ругистрации или авторизации пользователя.

Важно отметить, что конфиденциальная информация, такая как пароль пользователя не хранится в базе данных в чистом виде. Для защиты пароля в приложении реализован механизм его хэширования.

При обращения клиента к конечным точкам, которые требует авторизации, то токены приложению необходимо получать из отправляемых на сервер cookie. Для этого реализован специальный middleware, который встроен в конвеер обработки запросов, поэтому, при каждом обращении к приложению данный middleware будет извлекать и валидировать содержащийся в cookie токен и, в случае, если токен не валиден, отправлять соотвествующий статус код клиенту.

В случае, если access - токен истек, а refresh нет, то на стороне клиента встроен интерсептор, который автоматически отправляет запрос на сервер для обновления access токена и повторяет предыдущий запрос. А в случае, если истек и access токен, и refresh токен, то пользователю необходимо будет заново авторизоваться в приложении.

**Реализация отображения продуктов на странице**. Для реализации отображения товаров на странице в приложении определен специальный класс, который отвечает за сущность продукта.

```
public sealed class Product : IAuditable
{
    public long Id { get; set; }
    public int BrandId { get; set; }
    public Brand Brand { get; set; }
    public int CategoryId { get; set; }
    public Category Category { get; set; }
    public string Model { get; set; }
    public decimal  Price { get; set; }
    public string MainImage { get; set; }
    public IEnumerable<ProductImage> ProductImages { get; set; }
```

```
        public IEnumerable<Specification> Specifications { get; set; }
        public int CountPurchase { get; set; }
        public int AvailableQuantity { get; set; }
        public DateTime CreatedAt { get; set; }
    }
```

Затем в приложении реализованы специальные методы, которы получают всю необходимую информацию о товаре из базы данных с помощью классов - репозиториев, передают эту информацию в классы DTO.

После получения DTO конечная точка отправляет их на сторону клиента. В резульате чего на клиенте будет изображена необходимая для страницы информация.

**Реализация отображения подробной информации о продукте**. Для реализации возможности отображении подробной информации о конкретном товаре в классе - сервисе определен специальный метод, который, используя классы - репозитория, получает определенный товар из базы данных по его идентификатору в системе. Затем данный метод получает всю связанную с данным товаром информацию, а именно его характеристики и картинки.

После чего, данный метод передает эту информацию в виде DTO на конечную точку, а конечная отправляет эти DTO на сторону клиента.

**Реализация добавление товара в корзину**. Для реализации возможности добавить товар в корзину на странице с детальной информацией о товаре определена специальная кнопка. Стоит отметить, что данная кнопка доступна только для авторизованных пользователей, а также при наличии данного товара.

На стороне сервера в приложении определен класс корзины, а также специальный класс, который представляет конкретный товар в корзине. Код его реализации изображен ниже.

```
public sealed class CartProduct
{
    public long Id { get; set; }
    public long CartId { get; set; }
    public Cart Cart { get; set; }
    public long ProductId { get; set; }
```

```
    public Product Product { get; set; }
    public int Count { get; set; }
}
```

Для добавления товара в корзину в приложении определена специальная конечная точка, на которую клиент отправляет запрос на добавление товара в корзину. Далее данная конечная точка вызывает специальный метод класса - сервиса, куда передает всю необходимую информацию о добавляемом товаре. После чего метод сервиса вызывает метод класса - репозитория для сохранения этого товара в корзине конкретного пользователя. После чего, если пользователь перейдет на страницу корзины, то у него отобразятся все товары, которые он добавил в корзину, а также их общая стоимость. Помимо этого, пользователь может с данной страницы увеличить/уменьшить количество товара для заказа, а также удалить товар из корзины.

**Реализация оформления заказа** Для оформмления заказа на странице корзины пользователя изображена специальная кнопка, по нажатию на которую, пользователя перенаправялет на страницу для оформления заказа. На данной странице изображена информация о заказа, такая как адрес доставки (заполняется пользователем), количество продуктов в заказе, а также общую стоимость заказа. После оформления заказа пользователя перенаправляет на страницу его заказов.

На стороне сервера для реализации оформления заказа в приложении определен специальный класс заказа. Его реализация изображена ниже.

```
public sealed class Order : IAuditable
{
    public long Id { get; set; }
    public Guid UserId { get; set; }
    public int OfficeId { get; set; }
    public Office Office { get; set; }
    public string Status { get; set; }
    public decimal TotalPrice { get; set; }
    public DateTime CreatedAt { get; set; }
    public DateTime? UpdatedAt { get; set; }
    public ICollection<OrderProduct> OrderProducts { get; set; }
}
```

Для оформления заказа и передачи о нем данных в базу данных в приложении определена специальная конечная точка, которая вызывает метод-сервиса для создания заказа в системе, куда передает всю необходиму информацию о товаре. Данный метод, используя методы класса - репозитория, сохраняет товары из заказа в специальную таблицу, а также создает запись о заказе в БД.

В **заключении** приведены результаты бакалаврской работы.

### Основные резльтаты

1. Спроектирована масштабируемая архитектура приложения, обеспечивающая его гибкость и возможность расширения при необходимости.
2. Создан интуитивно понятный и удобный пользовательский интерфейс (UI) для приложения.
3. Спроектирована и реализована схема базы данных, обеспечивающая эффективное хранение и обработку данных.
4. Реализован полный набор функций, необходимых для работы интернет-магазина.
5. Обеспечен надежный обмен данными между клиентом и сервером в формате JSON через протокол HTTP.