

**МИНОБРНАУКИ РОССИИ**  
**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**  
**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**  
**ВЫСШЕГО ОБРАЗОВАНИЯ**  
**“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

Факультет компьютерных наук

Кафедра технологий обработки и защиты информации

YouFree

Разработка веб-приложения для фриланса

Курсовая работа

09.03.02 Информационные системы и технологии

Обучающийся \_\_\_\_\_ Воробьёв А.Н, б.о, 4курс

Руководитель \_\_\_\_\_ В.С. Тарасов, старший преподаватель

Воронеж 2022

## Содержание

Введение	3
1. Постановка задачи	4
2. Анализ предметной области	5
2.1. Глоссарий	5
2.2. Анализ аналогов	7
2.3. Анализ потребности	9
3. Диаграммы	10
3.1. Диаграмма прецедентов	10
3.2. Диаграмма последовательностей	11
3.3. Диаграмма взаимодействий	13
3.4. Диаграмма состояний	14
3.5. Диаграмма развертывания	16
3.6. Схема базы данных	17
4. Реализация	18
4.1. Средства реализации	18
4.2. Frontend-составляющая приложения	20
4.3. Backend-составляющая приложения	20
4.4. Сценарии воронок конверсии	24
4.5. Описание функциональной части приложения и графического пользовательского интерфейса	25
5. Тестирование	31
Список используемой литературы	35

## **Введение**

В настоящее время интернет становится все более развитой средой для осуществления коммуникаций. В связи с глобальным развитием сети Интернет, в программировании все более резко начала выделяться отдельная его отрасль web-программирование.

Сейчас, чтобы привлечь внимание клиентов, покупателей или партнёров, просто необходимо заявить о себе в интернете, путём создания web-сайта. Для этих целей как раз и служит web- сайт, содержащий основную информацию об организации, частном лице, компании, товарах или услугах, прайс-листы, контактные данные. Сайты позволяют хранить, передавать, продавать различные типы информации, не отходя от экрана компьютера.

Wide web - глобальная компьютерная сеть, на сегодняшний день содержит миллионы сайтов, на которых размещена всевозможная информация. Люди получают доступ к этой информации посредством использования технологий Internet. Для поиска по интернету используют специальные программы - Web-браузеры, которые существенно облегчают путешествие по бескрайним просторам интернета.

## 1. Постановка задачи

Цель данного курсового проекта – разработать приложение для осуществления услуг просмотра и выставления заказов.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Собрать необходимую информацию о исследуемой предметной области;
2. Провести анализ полученных данных;
3. Создать диаграммы, отражающие основные аспекты приложения;
4. Организовать хранение и модификацию данных о:
  - Пользователе (логин, пароль, емейл, тип пользователя)
  - Заказах (тема, заголовок, описание, стоимость, длительность выполнения)
5. Обеспечить возможность просмотра информации о заказах;
6. Обеспечить защищенность сессии пользователя;
7. Создать удобный и понятный интерфейс;
8. Провести тестирование веб-приложения.

Таким образом, требуется разработать приложение, обеспечивающее:

1. Интуитивно понятный дизайн;
2. Организацию создания, хранения и модификации информации, включающей в себя:
  - Данные о пользователях (данные авторизации, тип пользователя);
  - Данные о заказах (название, описание, стоимость).
3. Предоставление информации о заказах;
4. Авторизацию, основанную на JWT токенах.

## **2. Анализ предметной области**

### **2.1. Глоссарий**

В курсовом проекте будут использоваться следующие термины:

- Проект - разрабатываемое веб приложение.
- Фрилансер - самозанятый человек.
- Веб-приложение (сайт) - Клиент-серверное приложение, в котором клиент взаимодействует с веб-сервером при помощи браузера;
- Посетитель - пользователь, который не прошел регистрацию.
- Создатель - пользователь, который прошел регистрацию и выбрал роль создателя, позволяющая создавать заказы
- Исполнитель - пользователь, который прошел регистрацию и выбрал роль исполнителя, позволяющая брать и выполнять заказы клиентов.
- Front-end - клиентская часть приложения. Отвечает за получение информации с программно-аппаратной части и отображение ее на устройстве пользователя
- СУБД - Система управления базами данных. Комплекс программно-языковых средств, позволяющих создать базы данных и управлять данными;
- Сервер, серверная часть - компьютер, обслуживающий другие компьютеры (клиентов) и предоставляющий им свои ресурсы для выполнения определенных задач
- Клиентская сторона - компьютер, использующий ресурсы сервера и предоставляющий пользователю возможность взаимодействия с системой

- Bootstrap - Свободный набор инструментов для создания сайтов и веб-приложений;
- Back-end – программно-аппаратная часть приложения. Отвечает за функционирование внутренней части приложения
- GitHub - Сервис для совместной разработки и хостинга проектов.
- Пользователь – авторизованный в системе человек, пользующийся функционалом веб-приложения.
- C# (произносится как "си шарп") — современный объектно-ориентированный и типобезопасный язык программирования. C# позволяет разработчикам создавать разные типы безопасных и надежных приложений, выполняющихся в .NET.
- TypeScript (TS, TScript или «тайпскрипт») — это язык программирования для веб-разработки, основанный на JavaScript. Делает код понятнее и надежнее, добавляет статическую типизацию (переменные привязаны к конкретным типам данных), а также может быть скомпилирован в JavaScript. TypeScript используют фронтенд- и бэкенд-разработчики
- Microsoft SQL Server - система управления реляционными базами данных, разработанная корпорацией Microsoft. Основным используемым языком запросов – Transact-SQL.
- React - это фреймворк для создания реактивных веб-приложений.
- Windows 10 – Операционная система для персональных компьютеров и рабочих станций, разработанная корпорацией Microsoft в рамках семейства Windows NT.

## 2.2. Анализ аналогов

Для формулирования более конкретных требований к веб-приложению необходимо было провести анализ аналогов. К ним можно отнести такие веб-приложения как, например: Kwork, FL и Workzilla

Kwork - не совсем традиционное приложение для поиска работы в формате маркетплейса. Здесь размещают заказы разного уровня сложности, в том числе и простые задачи, не требующие глубоких знаний в теме.

Сильные стороны сервиса:

- Бесплатное обучение в работе с платформой;
- Широкий спектр задач;
- Отсутствие платных подписок
- Продуманный интерфейс.

Слабые:

- Высокая комиссия заказов;
- Большая конкуренция среди исполнителей
- При Решении споров и разногласий арбитраж зачастую поддерживает покупателя.

FL – это крупное многопрофильное приложение с сотнями удаленных проектов, конкурсов и вакансий.

Сильные стороны сервиса:

- Возможность обмена контактами с работодателем;
- Огромный выбор заданий;
- Удобные фильтры в веб-приложении.

Слабые:

- Для доступа ко всем заданиями нужно покупать PRO-аккаунт;
- Вывести средства можно только на банковскую карту;
- Небольшой потенциал для роста на бесплатном аккаунте.

Workzilla – это приложение, с объемной базой типовых заданий для начинающих и опытных фрилансеров.

Сильные стороны сервиса:

- Гарантия оплаты исполнителю при выполнении задания;
- Полезные советы для исполнителей;
- Возможность добавлять доп. контакты.

Слабые:

- Обязательное тестирование и платная подписка;
- Высокие комиссии;
- Нельзя посмотреть подробное ТЗ проектов до оплаты подписки;
- Часто заниженная оплата.



### **2.3. Анализ потребности**

После изучения аналогов разрабатываемого веб-приложения можно сделать вывод, что при наличии достаточно высокой конкуренции на рынке похожих проектов, ни один из них не является идеальным примером и решением. Но существование большого количества похожих сервисов показывает, что пользователь нуждается в этом в современном мире. Сейчас многие хотят работать на себя в меру своих возможностей и желания. Свободный график и выбор задач, которые нужно решить, реализуется в соответствии с желанием пользователей данных сервисов.

Именно поэтому разработка проекта YouFree является актуальной. Команде разработчиков необходимо вдохновиться другими проектами и создать что-то собственное, учитывая ошибки других проектов.

### 3. Диаграммы

#### 3.1 Диаграмма прецедентов

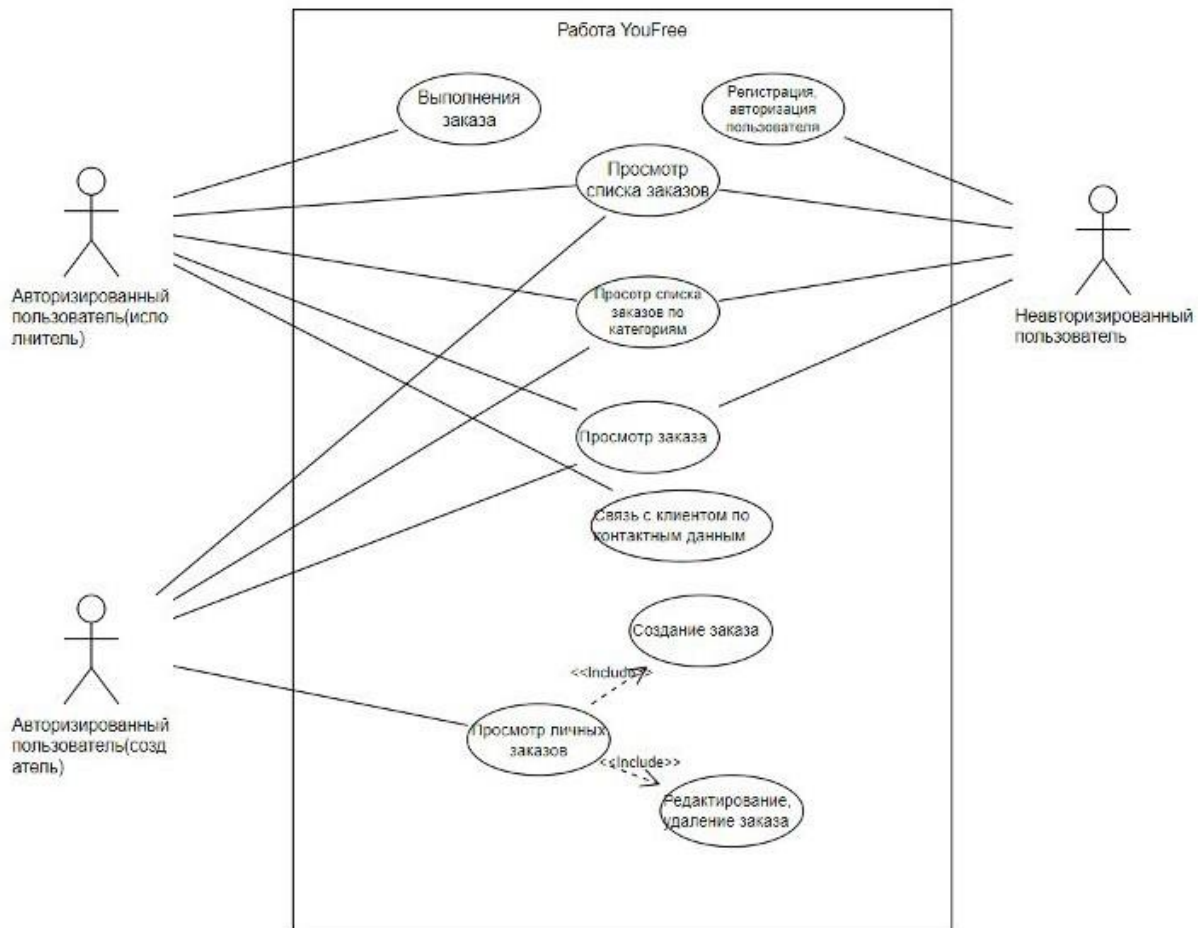


Рисунок 1 - Диаграмма прецедентов (use-case)

На рисунке 1 представлены основные функции пользователей в веб приложении YouFree.

Неавторизованный пользователь: регистрация/авторизация в системе, просмотр списка заказов, использования фильтра для просмотра заказов нужной категории;

Авторизованный пользователь (роль исполнитель): просмотр списка заказов, использования фильтра для просмотра заказов нужной категории, связываться с клиентом через контактные данные, выполнять подходящие заказы.

Авторизованный пользователь (роль создатель): просмотр списка заказов, просмотр собственного списка заказов с возможностью редактирования этих заказов. Еще возможность создавать и удалять личные заказы пользователя.

### 3.2 Диаграмма последовательностей

Диаграмма последовательностей служит для описания взаимодействия объектов системы по мере их проявления в сценарии.

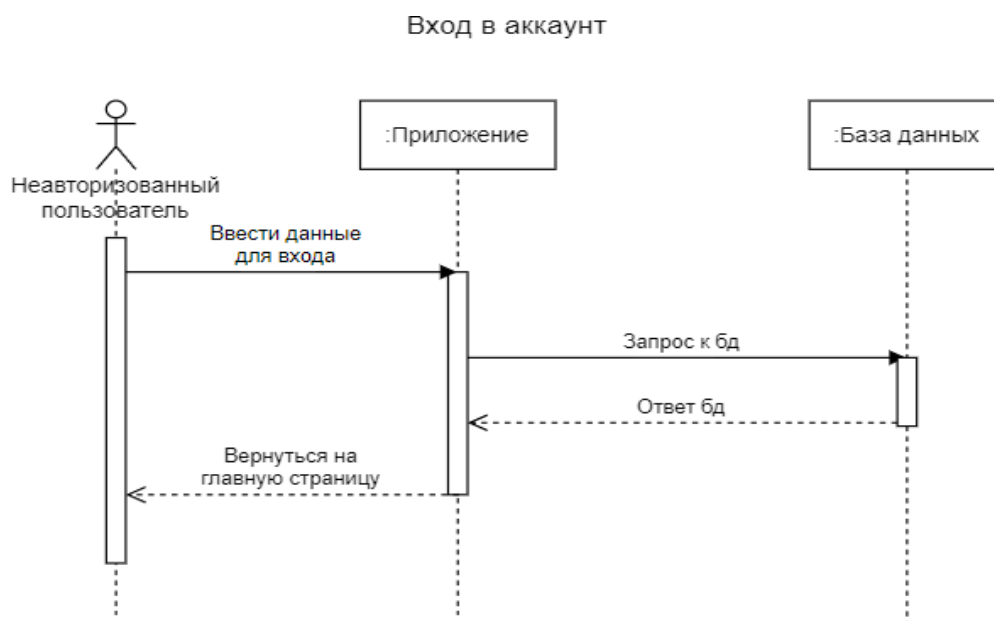


Рисунок 2 - Диаграмма последовательностей для авторизации

Неавторизованный пользователь вводит данные для входа (логин и пароль) в форму приложения, которое отправляет запрос в БД и в случае положительного ответа позволяет пользователю войти в приложение (с возвратом на главную страницу), в противном случае будет предложено ввести данные заново.

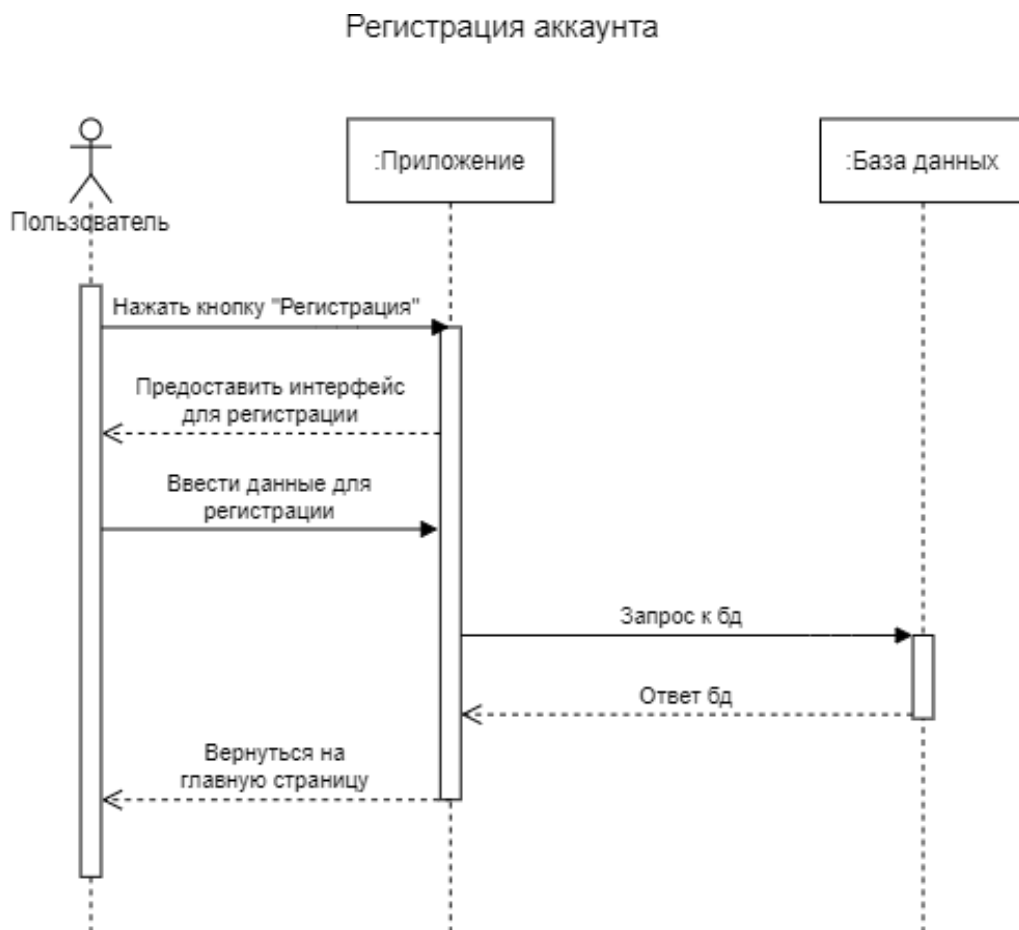


Рисунок 3 - Диаграмма последовательности для регистрации

Незарегистрированный пользователь после нажатия на кнопку регистрации получает от приложения форму для ввода данных регистрации. После отправки этих данных приложение делает запрос к БД и в случае успешного ответа (т.е. данные введены корректно и не совпадают с существующим) пользователь получает зарегистрированный аккаунт.

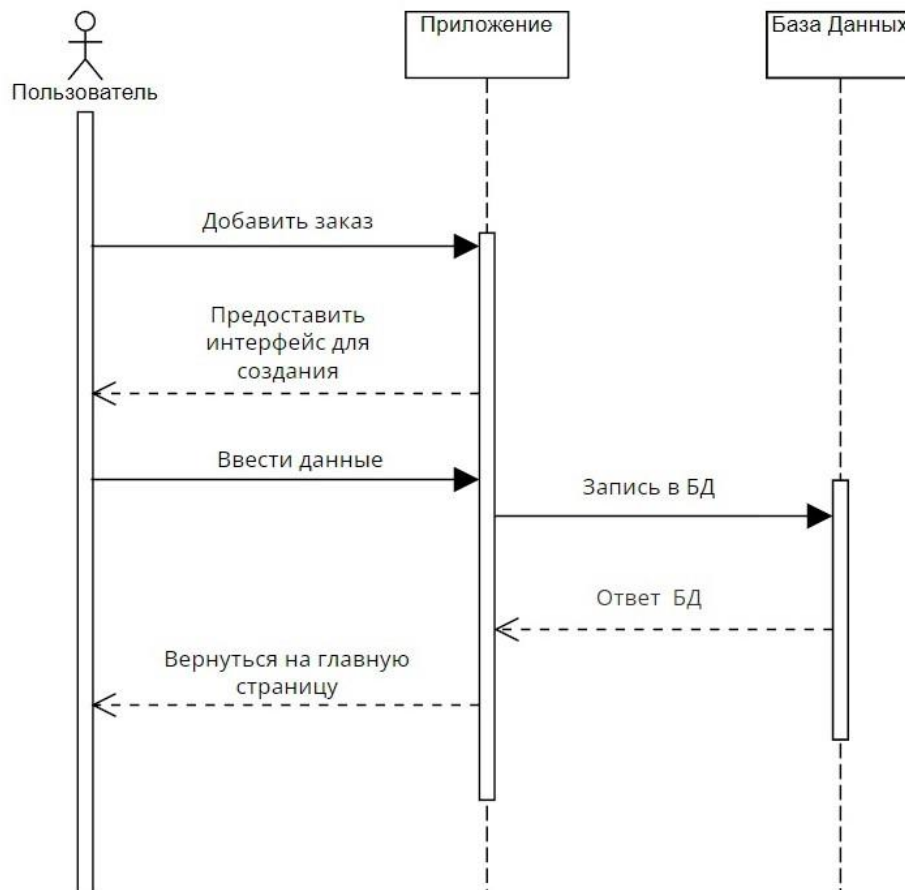


Рисунок 4 - Диаграмма последовательностей для заполнения заказа

Зарегистрированный пользователь под ролью создателя, выбирает функцию создания нового заказа, после чего приложение предлагает пользователю заполнить его. Пользователь вводит данные, после чего приложение посылает запрос в базу данных В случае положительного ответа в БД записывается информация о данных заказа.

### 3.3 Диаграмма взаимодействий

Данная диаграмма определяет возможные способы взаимодействия пользователей с системой. Позволяет видеть все взаимодействия запросов в системе, служит дополнением к диаграмме последовательностей

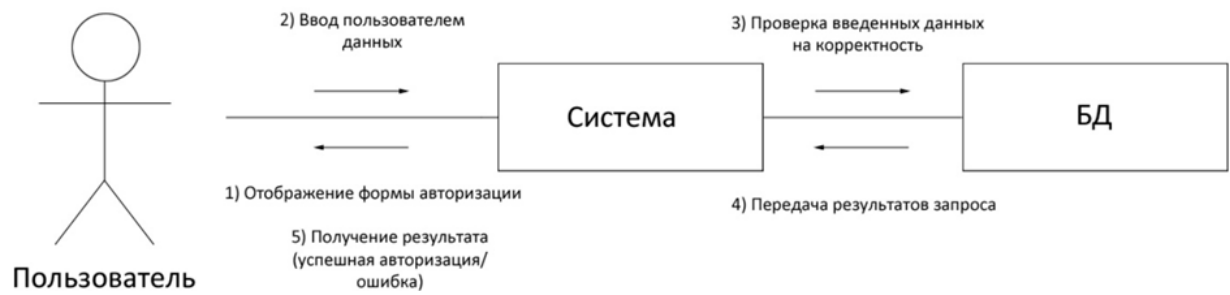


Рисунок 5 - Диаграмма взаимодействия для авторизации

### 3.4 Диаграмма состояний

Данные диаграммы описывают переход объектов из одних состояний в другие.

Диаграмма состояния для пользователя

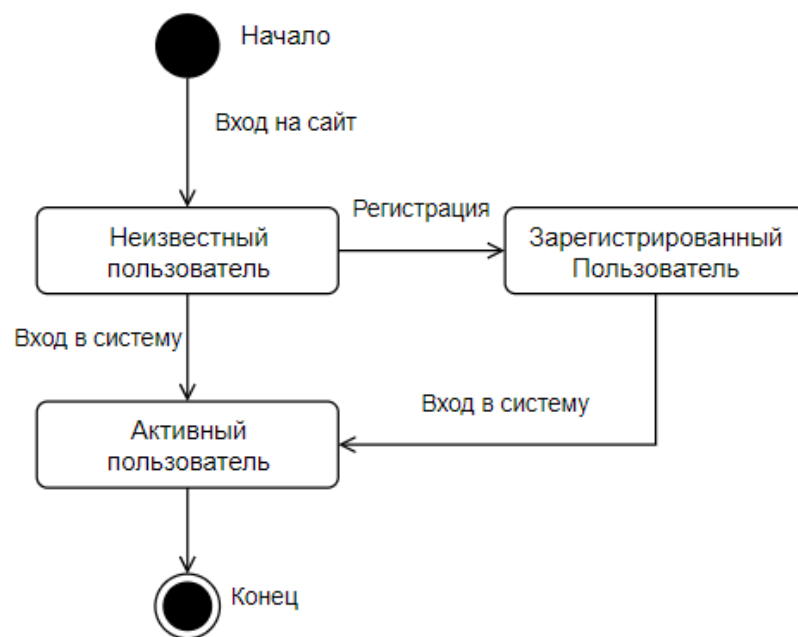


Рисунок 6 - Диаграмма состояний для авторизации

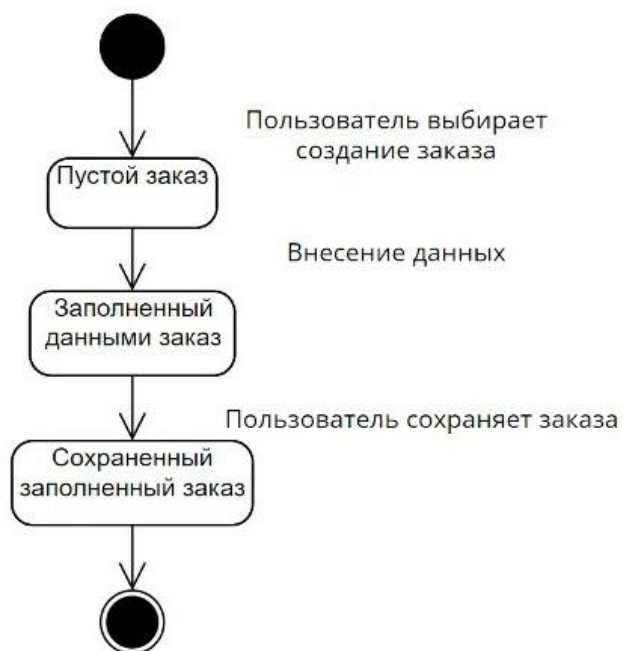


Рисунок 7 - Диаграмма состояний для заполнения заказа

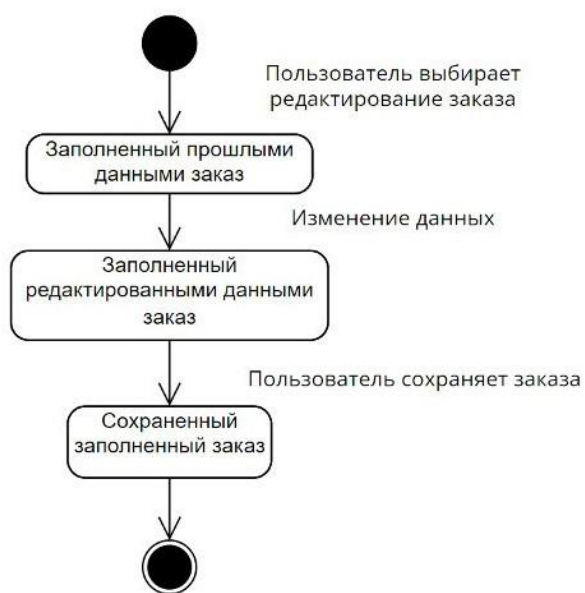


Рисунок 8 - Диаграмма состояний для редактирования заказа

### 3.5 Диаграмма развертывания

На рисунке 9 изображена диаграмма развертывания для того, чтобы продемонстрировать, какие узлы существуют в рамках данной системы и какие программные компоненты работают на этих узлах.

На данной диаграмме можно увидеть, что Backend и База Данных развернуты на сервере. Frontend же разворачивается в браузере за счет браузера и осуществляются запросы к серверной части приложения.

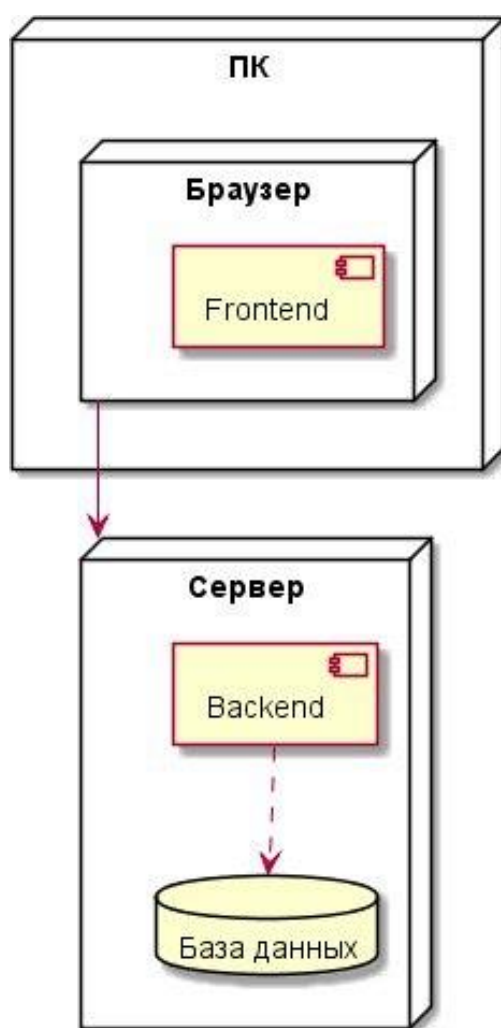


Рисунок 9 – Диаграмма развертывания



### 3.6 Схема базы данных

Для хранения информации о заказах и пользователях была создана база данных.

Структура БД (Рисунок 10) состоит из 3х таблиц:

Users – сущность (Пользователь).

Order – сущность (Заказ).

Categories – сущность (Категории).

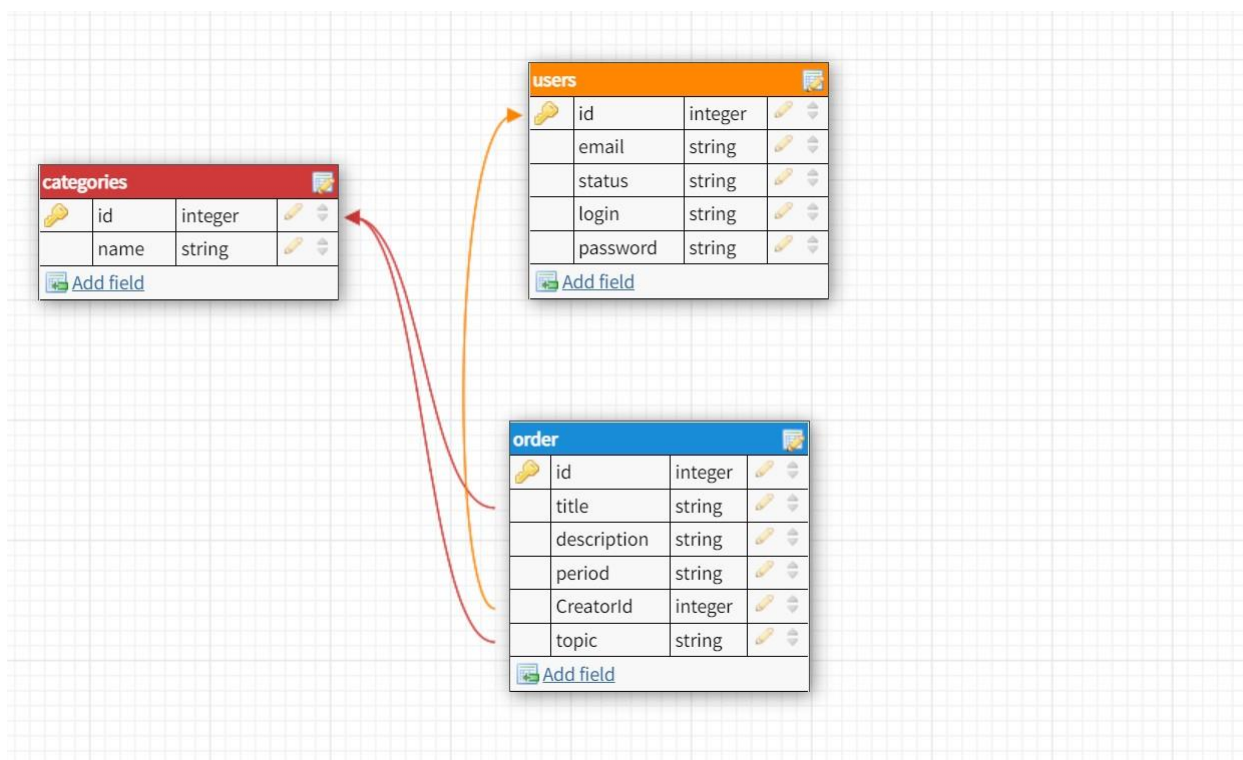


Рисунок 10 – Схема базы данных

## 4. Реализация

### 4.1 Средства реализации

В качестве средств реализации были использованы:

1. Microsoft SQL Server - система управления реляционными базами данных, разработанная корпорацией Microsoft. Основным используемым языком запросов—Transact-SQL. Сильными сторонами Microsoft SQL Server традиционно считаются:
  - Интеграция структурированных и неструктурированных данных;
  - Высокая производительность;
  - Безопасность и соответствие требованиям;
  - SQL Server упрощает развертывание, передачу и интеграцию больших данных;
  - Поддержка постоянной памяти.
2. React - это фреймворк для создания реактивных веб-приложений. Был выбран так как имеет расширение синтаксиса Java Script – JSX, которое позволяет использовать HTML, однонаправленную передачу данных и виртуальный DOM. А методы жизненного цикла позволяют на разных стадиях жизни компонента запускать необходимый код.
3. Для создания серверной части был выбран язык C#.C# изначально был придуман компанией Microsoft для собственных целей и служб. Он предусматривает следующие преимущества:
  - Строгую типизацию;
  - Функциональность;
  - Достаточно мощный инструментарий.

4. Для реализации клиентской стороны был выбран язык TypeScript.

Преимущества и особенности выбранного языка:

- Явная типизация и проверка согласования типов на этапе компиляции;
- Обратная совместимость с JS;
- Явная типизация позволяет IDE анализировать код, облегчая его поддержку и уменьшая вероятность сделать ошибку;
- Поддержка таких JS-конструкций, как `spread` и `rest` операторов, деструктуризации;
- Поддержка ко конструкций статически типизированных языков: интерфейсов, перечисляемых типов, обобщенного программирования и т.д.;
- Гибкая настройка компилятора.

5. Так же использовалась платформа .NET Core. Платформа .NET Core. Платформа представляет технологию от компании Microsoft, предназначенную для оздания различного рода веб-приложений: от небольших веб-сайтов до крупных веб-порталов и веб-сервисов. .NET Core может работать по верх кросс-платформенной среды .NET Core, которая может быть развернута на основных популярных операционных системах: Windows, Mac OS, LINUX. И таким образом, с помощью .NET Core мы можем создавать кросс-платформенные приложения .

## **4.2 Frontend-составляющая приложения**

Веб-клиент написан на React с использованием Redux ToolKit. Отладка такого проекта с использованием расширений React Developer Tools и Redux DevTools становится значительно проще – можно просматривать как и состояние отдельного компонента, так и глобального хранилища в любой момент.

Все страницы разделены на переиспользуемые компоненты. На клиентской стороне для управления сборкой проекта используется Webpack, для управления зависимостями и их версиями – пакетный менеджер Npm

## **4.3 Backend-составляющая приложения**

На серверной стороне сборка проекта обеспечивается фреймворком ASP.NET Core и платформой .Net. Сервер вместе с базой данных развернут в Docker.

Docker – это платформа контейнеризации с открытым исходным кодом, с помощью которой можно автоматизировать создание приложений, их доставку и управление. Платформа позволяет быстрее тестировать и выкладывать приложения, запускать на одной машине требуемое количество контейнеров. Благодаря контейнеризации и использованию Docker разработчики больше не задумываются о том, в какой среде будет функционировать их приложение и будут ли в этой среде необходимые для тестирования опции и зависимости. Достаточно упаковать приложение со всеми зависимостями и процессами в контейнер, чтобы запускать в любых системах: Linux, Windows и MacOS

Платформа Docker позволила отделить приложения от инфраструктуры. Контейнеры не зависят от базовой инфраструктуры, их можно легко перемещать между облачной и локальной инфраструктурами. Приложение построено по многослойной архитектуре (Рисунок 11).



Рисунок 11 – Многоуровневая архитектура

Нижний уровень многослойной архитектуры отвечает за взаимодействие с базой данных с помощью SQL-запросов и отображение результатов в C#-объекты. Такое отображение возможно благодаря Entity Framework Core. Entity Framework Core (EF Core) представляет собой объектно-ориентированную, легковесную и расширяемую технологию от компании Microsoft для доступа к данным. EF Core является ORM-инструментом (object-relational mapping - отображения данных на реальные объекты). То есть EF Core позволяет работать базами данных, но представляет собой более высокий уровень абстракции: EF Core позволяет абстрагироваться от самой базы данных и ее таблиц и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работаем с объектами

Для удобства работы с объектами используется AutoMapper. AutoMapper - это объектно-ориентированный редактор. Объектно-объектное сопоставление работает путем преобразования входного объекта одного типа в выходной объект другого типа. Что делает AutoMapper удобным, так это то, что он предоставляет несколько соглашений, позволяющих избавиться от рутинной работы по выяснению того, как сопоставить тип А с типом В. Пока тип В соответствует установленному соглашению AutoMapper для сопоставления двух типов требуется почти нулевая конфигурация.

DTO - то объект, который используется для инкапсуляции данных и отправки их из одной подсистемы приложения в другую(приложение 3). DTO чаще всего используются уровнем служб в N-уровневом приложении для передачи данных между собой и уровнем пользовательского интерфейса. Основное преимущество здесь заключается в том, что это уменьшает объем данных, которые необходимо передавать по уровням в приложениях. Они также создают отличные модели в шаблоне MVC. Совокупность всех моделей и DTO-объектов составляет уровень хранения данных (Data Access Layer)

Уровнем выше располагается слой бизнес-логики. Здесь выполняются все манипуляции с данными и реализуется значительная часть функциональности приложения. Слой бизнес-логики знает все о нижестоящем слое хранения данных, а значит может использовать его для работы. Классы, реализующие операции на уровне бизнес-логики называются сервисными классами, и, как минимум, реализуют базовые операции с данными: добавление, получение, обновление, удаление. Также, при необходимости, сервис нагружается дополнительной функциональностью, который можно отнести к бизнес-логике

Уровнем выше располагается слой представления данных. Он получает данные от клиента, при необходимости преобразует их в DTO-объект и передает их на дальнейшую обработку сервису слоя бизнес-логики, а также отправляет данные в понятном клиенту формате. В приложении слой представления реализован с помощью контроллеров – классов, которые ответственны за управление входящими запросами и отправкой ответов на них. Каждый метод контроллера, помеченный декоратором соответствующего HTTP-метода ([HttpGet],[HttpPost] и т.д.), становится ответственным за обработку конкретного маршрута с конкретным HTTP-методом. Также контроллер знает о нижестоящем слое, что позволяет ему

пользоваться методами сервисов

Для защиты сессии пользователя на сервере используется выдача JWT токенов. JWT состоит из трех основных частей: заголовка (header), нагрузки (payload) и подписи (signature). Заголовок и нагрузка формируются отдельно в формате JSON, кодируются в base64, а затем на их основе вычисляется подпись. Закодированные части соединяются друг с другом, и на их основе вычисляется подпись, которая также становится частью токена (Рисунок 12).

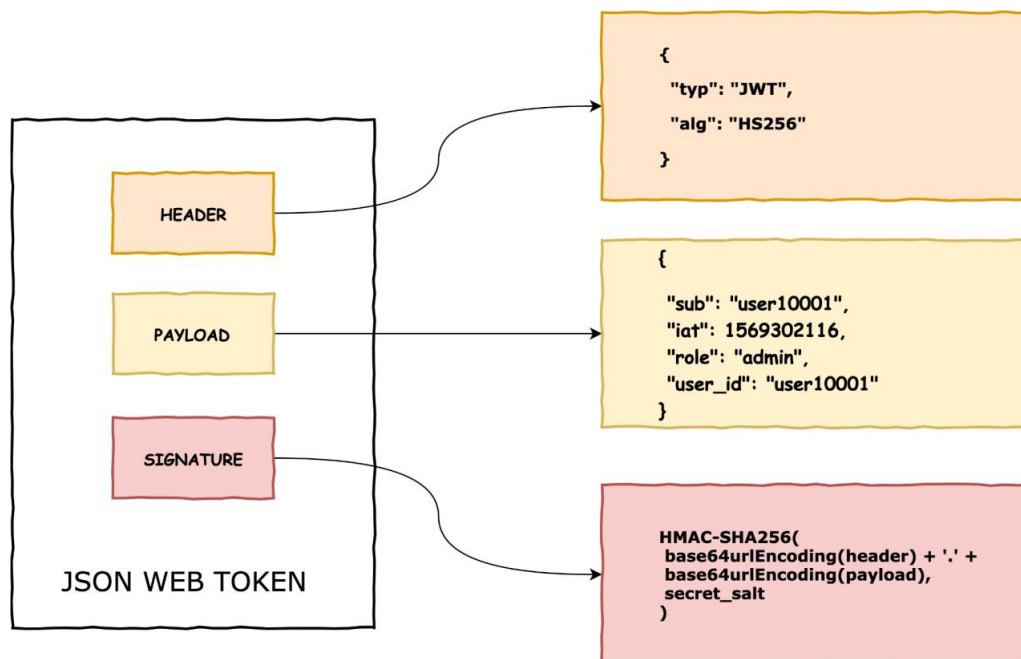


Рисунок 12 – Составляющие JWT токена.

Схема аутентификации с использованием JWT предельно проста. Пользователь вводит свои учетные данные в приложении или доверенном сервисе аутентификации. При успешной аутентификации сервис предоставляет пользователю токен, содержащий сведения об этом пользователе.

При последующем обращении токен передается приложению в запросах от пользователя: заголовках запроса, POST или GET параметрах и т. д. Получив токен, приложение сперва проверяет его подпись. Убедившись, что подпись действительна приложение извлекает из части полезной

нагрузки сведения о пользователе и на их основе авторизует его(Рисунок 13)

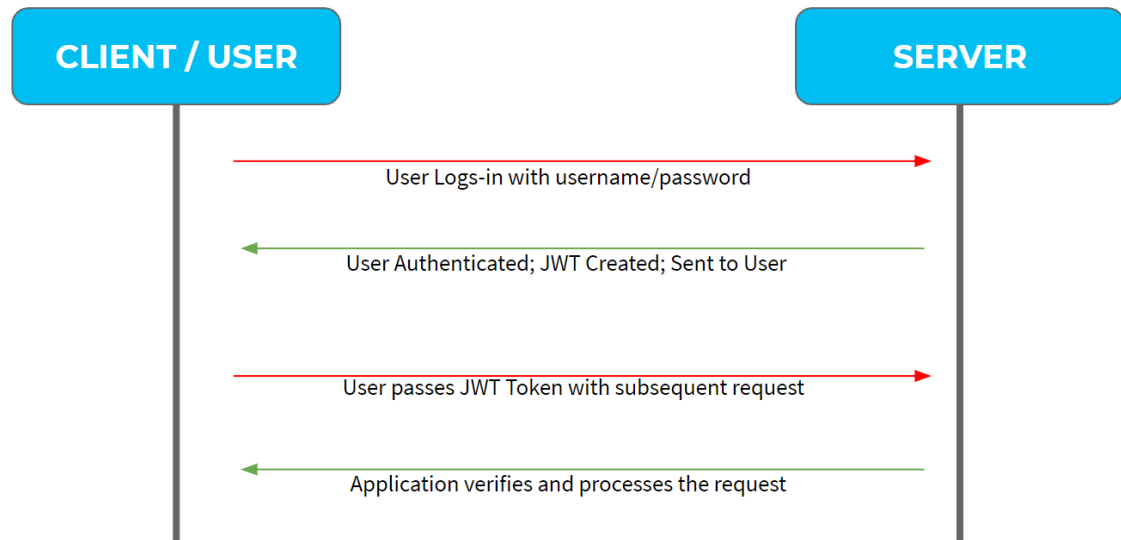


Рисунок 13 – Жизненный цикл JWT токен;

#### 4.4 Сценарии воронок конверсии

- a. Посетил главную страницу – Посетил страницу регистрации – Зарегистрировался.
- b. Посетил главную страницу – Авторизовался – Перешёл в список заказов – Перешёл на страницу заказа.
- c. Посетил главную страницу – Авторизовался – Перешел на страницу с личными заказами.



## 4.5 Описание функциональной части приложения и графического пользовательского интерфейса

Для рассмотрения интерфейса можно пройти все этапы использования приложения: от просмотра заказов неавторизованным пользователем до создания заказа пользователем с правами создателя. Рассмотрим шапку сайта (Рисунок 14).



Рисунок 14 – Верхняя часть сайта

Страница отображения заказов (Рисунок 15) со списками категорий, при помощи которой можно отсортировать заказы по нужному вам критерию. На ней пользователь может выбрать нужный ему заказ, выбрав его из списка.

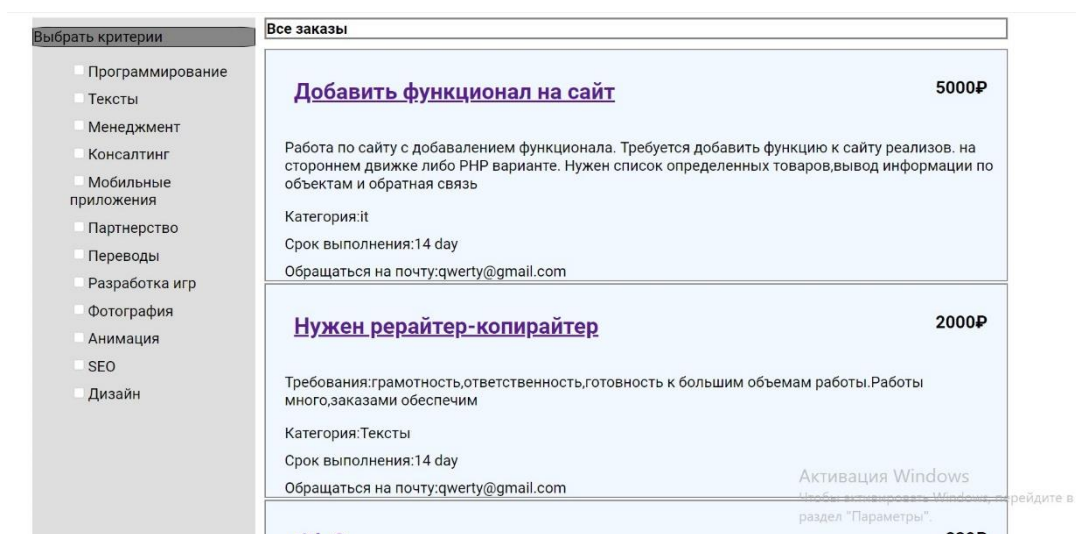


Рисунок 15 – Страница списков заказа

Для решения проблем с пагинацией создан компонент, который отображает доступное количество страниц с элементами (Рисунок 16).

Категория:games

Срок выполнения:14 day

Обращаться на почту:qwerty@gmail.com

Previous 1 2 Next

## Рисунок 16 – Пагинация

При выборе нужной вам категории, через список критериев находящийся слева от доступных заказов, вы получите список подходящих вам заказов (Рисунок 17).

Выбрать критерии

- ☒ Программирование
- ☐ Тексты
- ☐ Менеджмент
- ☐ Консалтинг
- ☐ Мобильные приложения
- ☐ Партнерство
- ☐ Переводы
- ☐ Разработка игр
- ☐ Фотография
- ☐ Анимация
- ☐ SEO
- ☐ Дизайн

Все заказы

**Добавить функционал на сайт** 5000₽

Работа по сайту с добавалением функционала. Требуется добавить функцию к сайту реализов. на стороннем движке либо PHP варианте. Нужен список определенных товаров,вывод информации по объектам и обратная связь

Категория:Программирование

Срок выполнения:14 day

Обращаться на почту:qwerty@gmail.com

**Браузерная онлайн-игра на спортивную тематику** 100000₽

Требуется front-end программист, работающий с интерфейсами и обладающий навыками в технологиях: Backbone.js (обязательно), HTML5, CSS3, Javascript, , JSON, jQuery, AJAX, socket.io.Базовая алгоритмическая подготовка, умение писать понятный, самодокументированный, реюзабельный и расширяемый код

Категория:Программирование

Срок выполнения:14 day

Обращаться на почту:qwerty@gmail.com

Активация Windows  
Чтобы активировать Windows, перейдите в раздел "Параметры".

## Рисунок 17 – Список заказа на основе выбранных фильтров.

Если выбрать нужный вам заказ, вы перейдете на страницу этого заказа.

У каждого заказа есть определенные данные, такие как цена срок выполнения его категория и контактные данные для связи.

Страницу заказа мы можем рассмотреть на рисунке 18

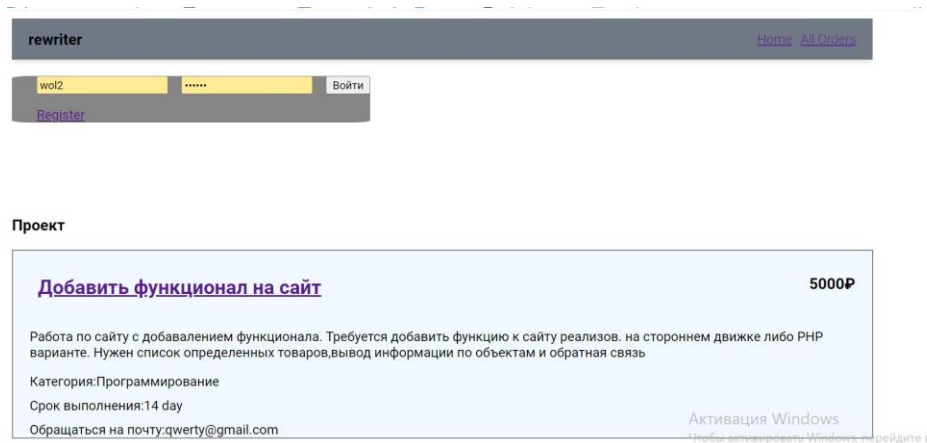


Рисунок 18 - Страница заказа

Для того, чтобы войти в аккаунт пользователю нужно ввести данные от своего аккаунта в форме для авторизации (Рисунок 19).



Рисунок 19 – Форма для авторизации.

В случае ввода правильных данных пользователь успешно авторизуется и форма автоматически измениться на личный кабинет (Рисунок 20)

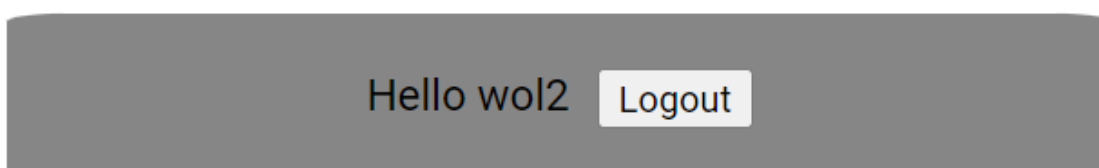


Рисунок 20 - Форма личного кабинета

Если же у пользователя нет аккаунта, то он может нажать на кнопку Register и тогда он перейдет на страницу регистрации (Рисунок – 21)

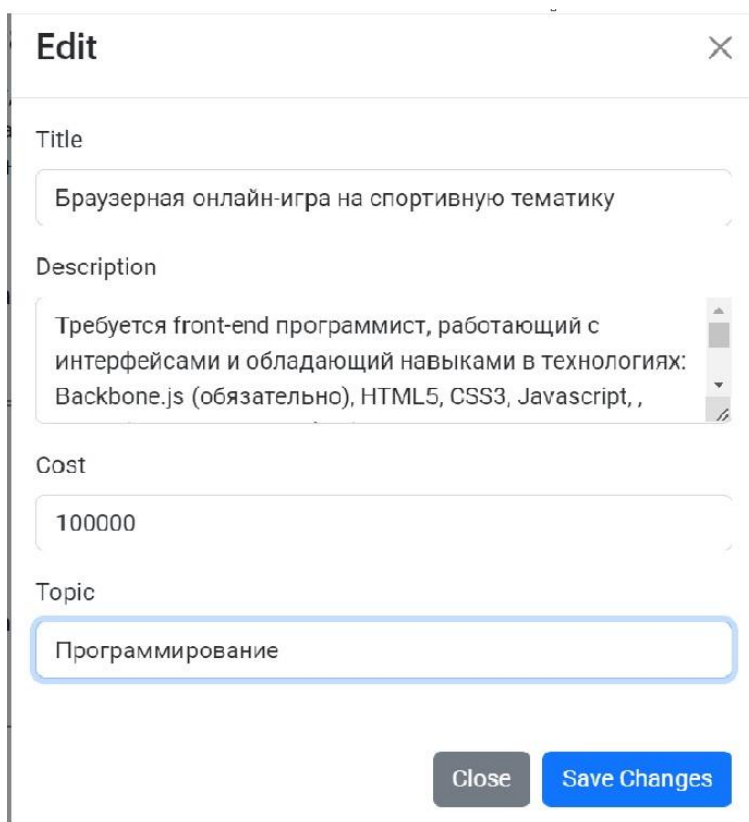
Рисунок 21 - Страница регистрации.

В форме регистрации пользователь должен ввести свои данные и выбрать тип аккаунта для регистрации: исполнитель заказов-это тот, кто выполняет заказы, или же создатель заказов тот, кто может создавать заказы. В случае успешной регистрации аккаунта пользователь опять попадет в личный кабинет. Если пользователь успешно авторизуется как создатель, то в его верхней части сайта появится дополнительная страничка личных заказов- в ней он может управлять заказами (Рисунок 22).

Страница личных заказов позволяет пользователю информацию о его заказах возможность удаления и их редактирования (Рисунок 23)

Рисунок 22 – Личные заказы пользователя

Если пользователь вошедший под ролью создатель захочет редактировать свой заказа, он может нажать кнопку Edit, тогда откроется модальная форма редактирования заказа.

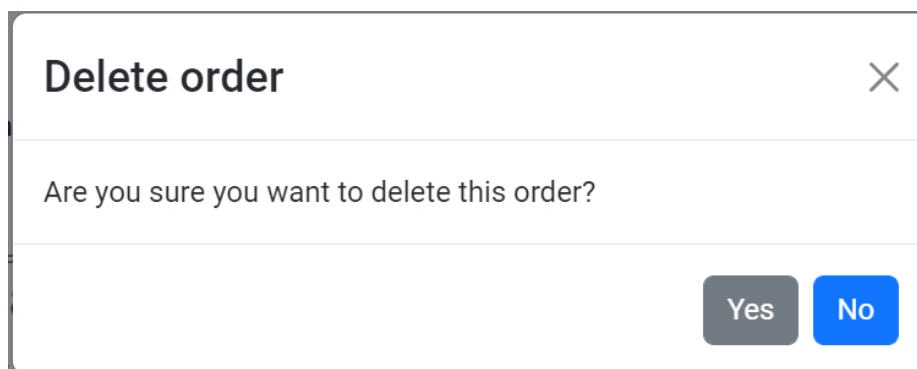


The 'Edit' modal form contains the following fields and controls:

- Title:** A text input field containing 'Браузерная онлайн-игра на спортивную тематику'.
- Description:** A text area containing 'Требуется front-end программист, работающий с интерфейсами и обладающий навыками в технологиях: Backbone.js (обязательно), HTML5, CSS3, Javascript, ,'. A vertical scrollbar is visible on the right.
- Cost:** A text input field containing '100000'.
- Topic:** A text input field containing 'Программирование', which is highlighted with a blue border.
- Buttons:** At the bottom right, there are two buttons: a grey 'Close' button and a blue 'Save Changes' button.

Рисунок 23 – Форма редактирования заказа.

При нажатии кнопки Delete, появиться окно с подтверждением удаления заказа изображенное на рисунке 24.



The 'Delete order' modal contains the following elements:

- Title:** 'Delete order' with a close button (X) in the top right corner.
- Text:** A confirmation message: 'Are you sure you want to delete this order?'.
- Buttons:** At the bottom right, there are two buttons: a grey 'Yes' button and a blue 'No' button.

Рисунок 24 – Окно удаления заказа

Если пользователь хочет создать новый заказ, то ему нужно нажать на кнопку Add Order и тогда появиться форма создания с критериями для заполнения нужного заказа (Рисунок 25).

The image shows a web form titled "Add Order" with a close button (X) in the top right corner. The form contains the following fields:

- Title**: A single-line text input field.
- Description**: A multi-line text input field with a rich text editor icon (two diagonal lines) in the bottom right corner.
- Cost**: A single-line text input field.
- Topic**: A single-line text input field.
- Period**: A single-line text input field.

At the bottom of the form, there are two buttons: a grey "Close" button and a blue "Save Changes" button.

Рисунок 25 – Форма создания заказа

## 5. Тестирование

После работ по созданию приложения было произведено тестирование его backend сервисов. Результаты тестирования представлены в таблице 1.

Тип запроса	Функция	Результат тестирования
POST	Регистрация пользователя	Успешно
POST	Авторизация пользователя	Успешно
GET	Выбор просмотра заказа из списка личных заказов	Успешно
GET	Выбор просмотра заказа на последующих страницах	Успешно
POST	Заполнение данных при создании нового заказа	Успешно
GET	Выбор просмотра определенной категории заказов по критериям	Успешно
GET	Выбор просмотра определенного заказа	Успешно

POST	Обработка поступивших данных по созданию нового заказа	Успешно
POST	Изменение данных при редактировании существующего заказа	Успешно
GET	Выбор просмотра заказа из списка заказов	Успешно



## Заключение

В ходе выполнения данной курсовой работы были выполнены следующие задачи:

- Собрана необходимая информация об исследуемой предметной области;
- Проведен анализ полученных данных;
- Созданы диаграммы, отражающие основные аспекты приложения;
- Организовано хранение и модификация данных о пользователе и заказах;
- Обеспечена возможность просмотра информации о заказах;
- Обеспечена защищенность сессии пользователя;
- Создан удобный и понятный интерфейс;
- Проведено тестирование веб-приложения.

Цель курсовой работы считается достигнутой, разработанное веб-приложение удовлетворяет поставленным перед ним требованиям:

- Интуитивно понятный дизайн;
- Обеспечена организация создания, хранения и модификации информации, включающей в себя данные о пользователе и данные о заказах;
- Обеспечено представление информации о заказах;
- Обеспечена авторизация, основанная на JWT токенах.

Итого, в результате проделанной работы было создано приложение для просмотра и создания фриланс заказов. Была выбрана клиент-серверная архитектура, клиент реализован в виде веб-клиента. Для хранения необходимой информации была разработана база данных, реализован

механизм ввода, сохранения, просмотра, редактирования и удаления введенной информации: данные о категориях, пользователях и заказах. Внедрена JWT аутентификация и авторизация на основе ролей

В дальнейшем планируется расширить функциональность приложения добавлением возможности становиться ответственным за выполнение заказа пользователем-исполнителем. Планируется добавление просмотра пользователей их рейтинга на сайте, а также добавление возможности оплаты.

### Список используемой литературы

1. Троелсен, Э. Язык программирования C# 5.0 и платформа .NET 4.5 / Э. Троелсен; Пер. с англ. Ю.Н. Артеменко. – М.: Вильямс, 2016. – 1312 с.
2. Рихтер, Джеффри, (1964). CLR via C# / Джеффри Рихтер ; 4-е изд.- Москва [и др.] : Питер, 2019. – 895 с.
3. Розенталс, Натан. Изучаем TypeScript 3: создавайте промышленные веб-приложения корпоративного класса с использованием TypeScript 3 и современных фреймворков / Натан Розенталс. – Москва : ДМК Пресс, 2019. – 623 с.
4. Чиннатамби, Кирупа. Изучаем React : практическое руководство по созданию веб-приложений при помощи React и Redux / Кируп Чиннатамби ; пер. с англ. М. А. Райтмана. - 2-е изд. - Москва : Эксмо, 2019. - 365 с.
5. Решаем проблемы REST с помощью Redux Toolkit Query. – Режим доступа: свободный. <https://habr.com/ru/company/netcracker/blog/646163/>. - (Дата обращения: 15.10.2022).
6. Building an ASP.NET Web API with ASP.NET Core. – Режим доступа: свободный. <https://www.toptal.com/asp-dot-net/asp-net-web-api-tutorial> – (Дата обращения 20.09.2022)
7. JWT validation and authorization in ASP.NET Core . – Режим доступа: свободный. <https://devblogs.microsoft.com/dotnet/jwt-validation-and-authorization-in-asp-net-core/>. – (Дата обращения 29.08.2022).
8. Redux Toolkit как средство эффективной Redux-разработки. – Режим доступа: свободный. <https://habr.com/ru/company/inobitec/blog/481288/>. – (Дата обращения: 27.09.2022)