**Реферат**

Пояснительная записка содержит 63 страницы, 51 рисунок, 24 таблицы,  
22 источника, 7 приложений.

ВЕБ-ПРИЛОЖЕНИЕ, EXPRESS, NODE.JS, MONGODB, REACT.JS, JETBRAINS WEBSTORM 2020.3

Объектом дипломного проекта является веб-приложение для аренды услуг программиста.

Основной целью дипломного проекта является разработка веб-приложения для аренды услуг программиста. В разработке дипломного проекта была использована платформа *NodeJS*, язык программирования *JavaScrpit*, технология *React.js*, протокол обмена данными *HTTP*.

Пояснительная записка состоит из введения, шести разделов и заключения.

В первом разделе описаны цель и задачи дипломного проекта, обзор аналогов и обзор средств разработки.

Во втором разделе представлены архитектура приложения, проектирование и структура таблиц базы данных.

Третий раздел посвящен разработке программного средства.

Четвертый раздел посвящен тестированию программного средства.

В пятом разделе приведено руководство пользователя.

В шестом разделе приводится расчет экономических параметров и себестоимость программного продукта.

В заключении представлены итоги дипломного проекта и задачи, которые были решены в ходе разработки программного средства.

Графическая часть представлена в объеме 0,75 листов формата А1.

**Abstract**

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*3*

*БГТУ 00.00.ПЗ*

Разраб*.*

*Чистякова Ю.А.*

Пров*.*

*.*

*Годун А.В.*

Консульт.

*Годун А.В.*

Н*.* контр*.*

*Рыжанкова А.С.*

Утв*.*

*Пацей Н.В.*

Реферат

*Лит.*

*Листов*

*1*

74417011, 2021

The explanatory note of the diploma project contains 63 pages of explanatory note, 51 illustrations, 24 tables, 22 sources used, 7 appendices.

WEB-APPLICATION, EXPRESS, NODE.JS, MONGODB, REACT.JS, JETBRAINS WEBSTORM 2020.3

The object of the graduation project is web-application for rent of programmer services.

The main goal of the diploma project is to develop a web-application for rent of programmer services. In developing the thesis project, the NodeJS platform, the JavaScript programming language, React technology, and the HTTP data exchange protocol were used.

The explanatory note consists of an introduction, six sections, and a conclusion.

The first section describes the purpose and objectives of the diploma project, an overview of analogs, and an overview of development tools.

The second section presents the application architecture, design, and structure of database tables.

The third section is devoted to software development.

The fourth section is devoted to testing the software tool.

The fifth section contains the user's guide.

The sixth section provides a calculation of economic parameters and the cost of the software product.

In conclusion, the results of the diploma project and the tasks that were solved   
during the development of the software tool are presented**.**

The graphic part is presented in the volume of 0,75 sheets of A1 format.

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*4*

*БГТУ 00.00.ПЗ*

Разраб*.*

*Чистякова Ю.А.*

Провер*.*

*.*

*Годун А.В.*

Консульт.

*Годун А.В.*

Н. контр.

*Рыжанкова А.С.*

Утв.

*Пацей Н.В.*

Abstract

*Лит.*

*Листов*

*1*

74417011, 2021

Содержание

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*5*

*БГТУ 00.00.ПЗ*

Разраб*.*

*Чистякова Ю.А.*

Провер*.*

*.*

*Годун А.В.*

Консульт.

*Годун А.В.*

Н*.* контр*.*

*Рыжанкова А.С.*

Утв*.*

*Пацей Н.В.*

Содержание

*Лит.*

*Листов*

*2*

74417011, 2021

[Введение 7](#_Toc72502087)

[1 Обзор аналогичных решений и средств разработки 8](#_Toc72502088)

[1.1 Обзор аналогов 8](#_Toc72502089)

[1.1.1 Сайт TrueTech.by 9](#_Toc72502090)

[1.1.2 Сайт RedSale.by 9](#_Toc72502091)

[1.1.3 Сайт FreelanceHunt.by 10](#_Toc72502092)

[1.1.4 Сайт Flagma.by 11](#_Toc72502093)

[1.2 Патентный поиск 12](#_Toc72502094)

[1.3 Обзор средств разработки 13](#_Toc72502095)

[1.3.1 Обоснование выбора языка программирования 13](#_Toc72502096)

[1.3.2 Обоснование выбора средства программирования 13](#_Toc72502097)

[1.3.3 Обоснование выбора платформы разработки 14](#_Toc72502098)

[1.4 Выбор технологий и библиотек 14](#_Toc72502099)

[1.4.1 Фреймворк Express 14](#_Toc72502100)

[1.4.2 Библиотека React для разработки пользовательских интерфейсов 15](#_Toc72502101)

[1.5 Вывод по разделу 15](#_Toc72502102)

[2 Проектирование веб-приложения 16](#_Toc72502103)

[2.1 Проектирование диаграммы развертывания 16](#_Toc72502104)

[2.2 Проектирование диаграммы вариантов использования 17](#_Toc72502105)

[2.3 Проектирование базы данных 18](#_Toc72502106)

[2.4 Описание алгоритма бронирования услуги 22](#_Toc72502107)

[2.5 Описание алгоритма создания услуги 23](#_Toc72502108)

[2.6 Вывод по разделу 23](#_Toc72502108)

[3 Разработка веб-приложения 24](#_Toc72502109)

[3.1 Физическая структура веб-приложения 24](#_Toc72502110)

[3.2 Технологии, используемые в серверной части приложения 26](#_Toc72502111)

[3.3 Реализация серверной части приложения 26](#_Toc72502112)

[3.3.1 Взаимодействие с базой данных 26](#_Toc72502113)

[3.3.2 Модели данных 27](#_Toc72502114)

[3.3.3 Функции промежуточной обработки 28](#_Toc72502115)

[3.3.4 Контроллеры 29](#_Toc72502116)

[3.3.5 Основные обработчики функций 30](#_Toc72502117)

[3.3.6 Сервисы 31](#_Toc72502118)

[3.3.7 Реализация отправки писем 32](#_Toc72502119)

[3.3.8 Реализация аутентификации 33](#_Toc72502120)

[3.4 Технологии, используемые в клиентской части приложения 35](#_Toc72502121)

[3.4.1 Реализация клиентской части приложения 35](#_Toc72502122)

[3.4.2 Маршрутизация в клиентской части приложении 36](#_Toc72502123)

[3.4.3 Конечные точки HTTP API 37](#_Toc72502124)

[3.5 Вывод по разделу 39](#_Toc72502125)

[4 Тестирование веб-приложения 40](#_Toc72502126)

[4.1 Негативное тестирование страницы входа 40](#_Toc72502127)

[4.2 Негативное тестирование страницы регистрации 42](#_Toc72502128)

[4.3 Негативное тестирование форм создания услуги 44](#_Toc72502129)

[4.4 Негативное тестирование импортирования категорий 46](#_Toc72502130)

[4.5 Вывод по разделу 46](#_Toc72502131)

[5 Руководство пользователя 47](#_Toc72502132)

[5.1 Роль «Гость» 47](#_Toc72502133)

[5.2 Роль «Администратор» 49](#_Toc72502134)

[5.3 Роль «Пользователь» 50](#_Toc72502135)

[5.4 Роль «Программист» 52](#_Toc72502136)

[5.5 Вывод по разделу 61](#_Toc72502137)

[6 Технико-экономическое обоснование проекта 62](#_Toc72502138)

[6.1 Общая характеристика разрабатываемого программного средства 62](#_Toc72502139)

[6.2 Исходные данные для проведения расчетов 62](#_Toc72502140)

[6.3 Методика обоснования цены 63](#_Toc72502141)

[6.3.1 Объем веб-приложения 64](#_Toc72502142)

[6.3.2 Основная заработная плата 65](#_Toc72502143)

[6.3.3 Дополнительная заработная плата 65](#_Toc72502144)

[6.3.4 Отчисления в Фонд социальной защиты населения и Белгосстрах 65](#_Toc72502145)

[6.3.5 Расходы на материалы 66](#_Toc72502146)

[6.3.6 Расходы на оплату машинного времени 66](#_Toc72502147)

[6.3.7 Прочие прямые затраты 67](#_Toc72502148)

[6.3.8 Накладные расходы 67](#_Toc72502149)

[6.3.9 Сумма расходов на разработку веб-приложения 67](#_Toc72502150)

[6.3.10 Расходы на сопровождение и адаптацию 67](#_Toc72502151)

[6.3.11 Полная себестоимость 68](#_Toc72502152)

[6.3.12 Определение цены, оценка эффективности 68](#_Toc72502153)

[6.4 Вывод по разделу 69](#_Toc72502154)

[Заключение 69](#_Toc72502155)

[Список использованных источников 71](#_Toc72502156)

[Приложение А Диаграмма развертывания 73](#_Toc72502157)

[Приложение Б Диаграмма вариантов использования 74](#_Toc72502158)

[Приложение В Скриншот работы веб-приложения 75](#_Toc72502159)

[Приложение Г Таблица расчетов экономических показателей 76](#_Toc72502160)

[Приложение Д Структурная схема веб-приложения 77](#_Toc72502161)

[Приложение Е Логическая схема базы данных 78](#_Toc72502162)

[Приложение Ж Блок-схема бронирования услуги программиста 79](#_Toc72502163)

Приложение З Блок-схема создания услуги программиста 80

Приложение И Листинг функции подтверждения заказа 81

# **Введение**

В Беларуси *IT*-индустрия – самая динамично развивающаяся отрасль, и с каждым годом превращается во все более мощный драйвер экономики страны. Во многих мировых рейтингах, оценивающих достижения *IT*-сектора, Беларусь занимает довольно высокие позиции [1].

Основной целью дипломного проекта является разработка веб-приложения для аренды услуг программистов.

Данный проект является исключительно актуальным для пользователей, которым необходимо получить срочную консультацию или помощь квалифицированного программиста.

Веб-приложение незаменимо и с точки зрения *IT*-компаний по множеству причин, основными из которых являются: необходимость быстро масштабировать свою команду разработки; короткий проект, на который нет возможности взять штатных специалистов; невозможность увеличить кадровую нагрузку; выход в отпуск или больничный одного из членов команды. Во всех случаях компания может без проблем арендовать услуги программиста на необходимый срок.

Кроме того, веб-приложение является актуальным и для самих программистов. Нередко возникают ситуации, когда в данный момент в компании нет нагрузки, и программисты простаивают, поэтому у них появляется прекрасная возможность сдать в аренду свои услуги.

Задачи дипломного проекта:

* подтверждение учетной записи через почту;
* редактирование личного кабинета пользователем;
* возможность послать запрос программисту на выполнение услуги клиентом;
* добавление услуги в избранное, создание отзыва клиентом;
* возможность принять, отклонить запрос на услугу программистом;
* создание, изменение, удаление услуги программистом;
* ведение чата между клиентом и программистом;
* возможность создания, изменения, удаления категорий программистов администратором.

Для достижения поставленной цели необходимо:

* исследовать преимущества и недостатки аналогичных приложений;
* разработать структуру базы данных;
* спроектировать диаграмму вариантов использования;
* спроектировать диаграмму развертывания;
* реализовать веб-приложение;
* рассчитать экономические показатели.

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*7*

*БГТУ 00.00.ПЗ*

Разраб*.*

*Чистякова Ю.А.*

Провер*.*

*.*

*Годун А.В.*

Консульт.

*Годун А.В.*

Н*.*контр*.*

*Рыжанкова А.С.*

Утв*.*

*Пацей Н.В.*

Введение

*Лит.*

*Листов*

*1*

74417011, 2021

# **1 Обзор аналогичных решений и средств разработки**

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*8*

*БГТУ 01.00.ПЗ*

Разраб*.*

*Чистякова Ю.А.*

Провер*.*

*.*

*Годун А.В.*

Консульт.

*Годун А.В.*

Н. контр.

*Рыжанкова А.С.*

Утв*.*

*Пацей Н.В.*

1 Обзор аналогичных решений и средств разработки

*Лит.*

*Листов*

*8*

74417011, 2021

В последнее время в нашей стране стал необычайно актуален вопрос услуг программиста. Для ИТ-компаний на это есть множество причин:

* необходимость быстро масштабировать команду разработки;
* короткий проект, на который нет возможности взять штатных специалистов;
* невозможность увеличить кадровую нагрузку;
* выход в отпуск или больничный одного из членов команды [2].

Разработанное приложение позволит пользователю забронировать наиболее подходящую ИТ-услугу нужного специалиста. На сайте будет достаточное количество информации, касающейся различных видов технологий, услуг, которые можно получить.

## 1.1 Обзор аналогов

Сегодня традиционные виды бизнеса всё чаще находят своё развитие в веб-пространстве. В частности, сайт для аренды услуг программиста может стать шагом на пути к развитию в крупное аутстаффинговое ИТ-агентство.

Сайт для аренды услуг программиста должен быть:

* функциональным. Это значит, что загрузка страниц должна быть быстрой, навигация – удобной, а интерфейс простым и понятным Функциональность позволяет внедрять на сайте различные интерактивные сервисы, производить его наполнение, вести модерацию, не затрачивая много времени и сил;
* информативным. На сайте должна быть понятно и развёрнута предоставлена информация об услугах. Все ответы по услугам клиент должен найти на сайте;
* интерактивным. Сайт аренды услуг программиста должен быть снабжен специальной системой, позволяющей выбрать наиболее подходящие услуги по заданным параметрам или произвести бронирование, не покидая страниц веб-ресурса [3].
* клиентоориентированным. На сайте должна быть актуальная информация о предлагаемых услугах, скорость поиска информации быстрой, обратная связь удобной;
* визуально привлекательным. Любую страницу в интернете человек сперва воспринимает глазами. И чтобы «зацепить» клиента визуально, сайт должен быть ярким;
* адаптивным. Это значит, что сайт адаптируется под любые устройства в зависимости от разрешения экрана. Cайт везде будет одинаково удобным и понятным. Следует отметить, что адаптивность учитывается при индексации нового сайта и дальнейшем *SEO*-продвижении [4];
* безопасным. Клиент должен чувствовать себя защищённым и доверять вашему сайту. На странице платежа и на самом сайте обязательно должен быть *SSL*-сертификат.

### 1.1.1 Сайт TrueTech.by

В качестве одного из прототипов приложения был выбран сайт *TrueTech.by*. Интерфейс сайта представлен на рисунке 1.1.

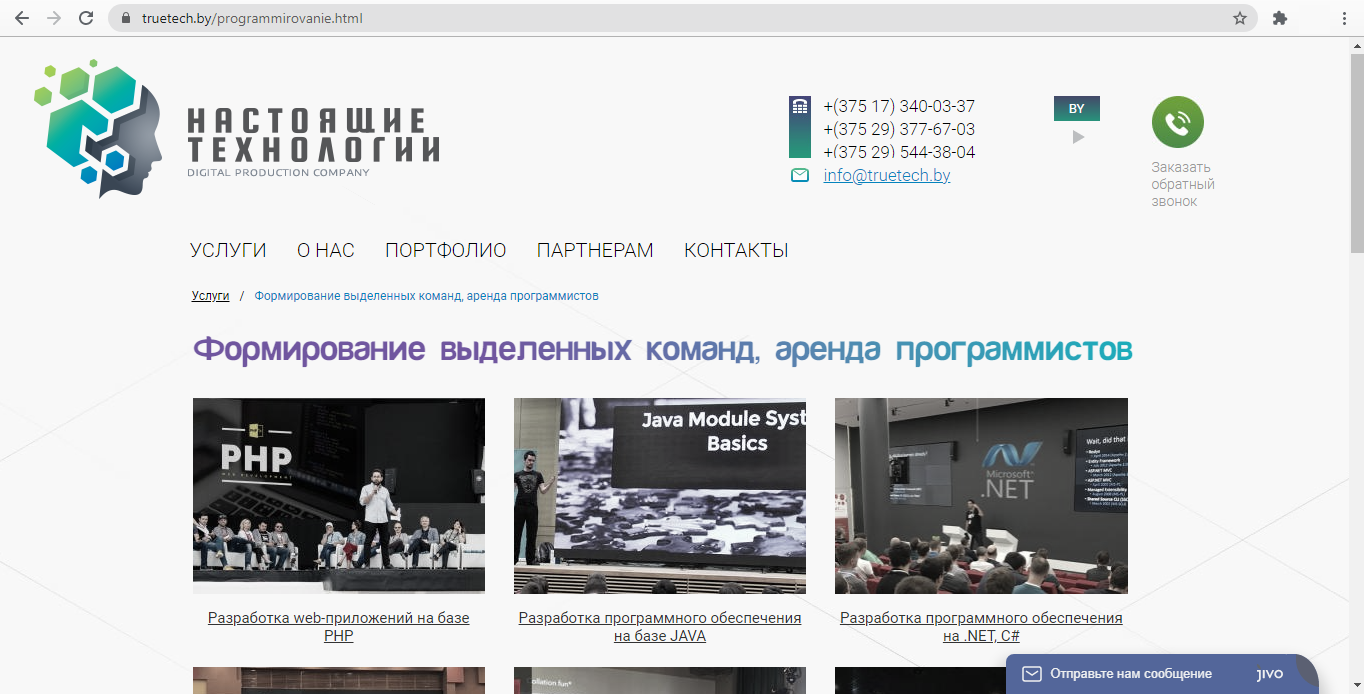


Рисунок 1.1 – Сайт *TrueTech.by*

Данный сайт предназначен для ознакомления с возможными услугами программистов для последующего бронирования.

На сайте есть фильтрация услуг по различным технологиям и языкам программирования. Для более детального ознакомления есть портфолио с примерами работ. Есть возможность заказать обратный звонок, а также получить помощь *online*-консультанта.

Из выявленных недостатков необходимо выделить неудобный пользовательский интерфейс, отсутствие непосредственного чата с исполняющим задание программистом, невозможность узнать рейтинг услуги. Возможность регистрации аккаунтов не предусмотрена.

### 1.1.2 Сайт RedSale.by

Судя по отзывам на сайте, сайт имеет хорошую репутацию среди пользователей. В сообществе социальной сети «В Контакте» указано, что они лучшие в Беларуси по количеству рекомендаций на *Otzyvy.by*.

Возможности:

* просмотр рейтинга и отзывов на программистов;
* подбор мастера по заданному описанию нужной услуги, подходящей стоимости, дате оказания услуг, района;
* наличие *online*-консультанта;
* услуга «Перезвони мне». Пользователь указывает в специальной форме свое имя и номер телефона;
* возможность отставить заявку на подбор программиста. Отправка заявки не накладывает обязательств и не является бронированием. Получив заявку, менеджер уточнит наличие подходящего программиста и свяжется с клиентом.

Интерфейс приложения представлен на рисунке 1.2.

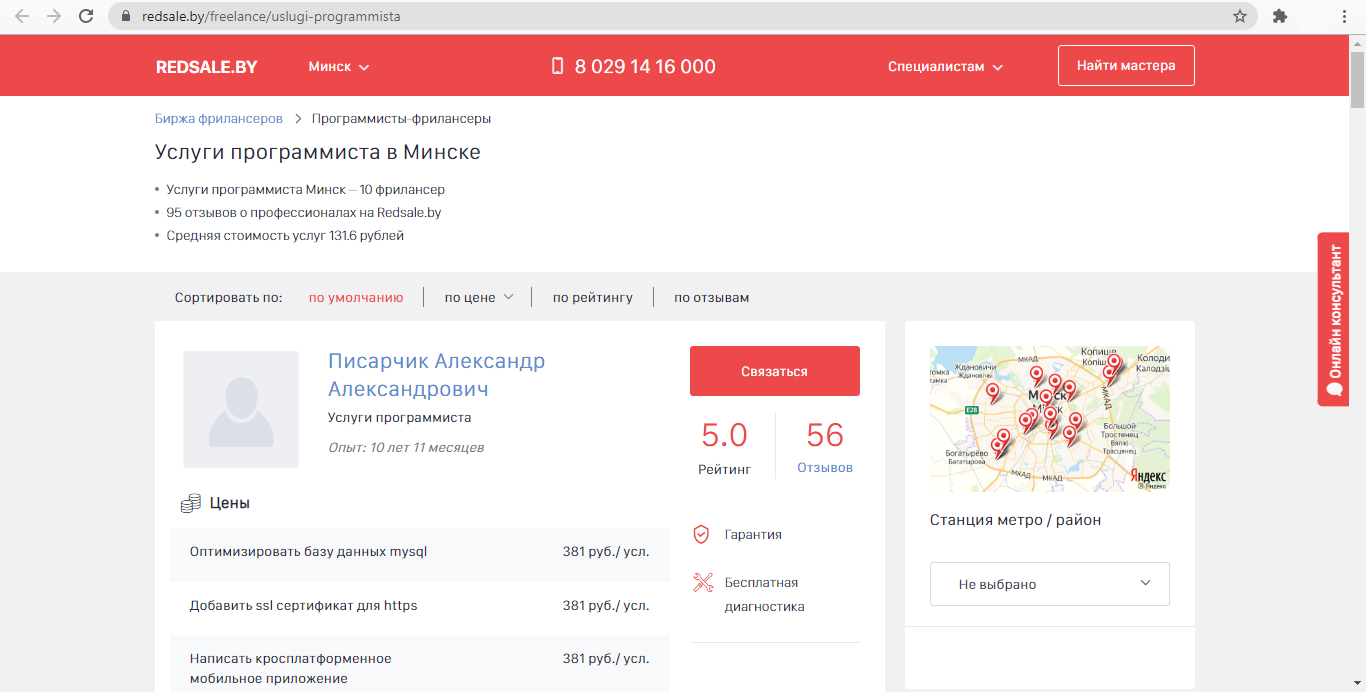


Рисунок 1.2 – Сайт *RedSale.by*

Существенным недостатком является не адаптивность данного сайта. Так как

в современном мире количество запросов с мобильных устройств преобладает над количеством запросов с персональных компьютеров, очень важно правильное отображение сайта на различных устройствах, динамически подстраивающийся под заданные размеры окна браузера.

### 1.1.3 Сайт FreelanceHunt.by

Основные возможности:

* посмотреть рейтинг, резюме и портфолио программиста;
* оставить заявку на выполнение услуги;
* есть возможность вести чат с программистом;
* найти ответы на часто задаваемые вопросы;
* зарегистрироваться на сайте;
* оставить отзыв о полученной услуге;
* добавить проект, создать вакансию, открыть конкурс;
* просмотреть топ фрилансеров;
* осуществлять быстрый поиск программиста по логину или фамилии.

Несомненными преимуществами данного сайта являются его адаптивность под мобильные устройства, динамическое подстраивание под данные окна браузера, а также удобный пользовательский интерфейс.

Главная страница сайта представлена на рисунке 1.3.

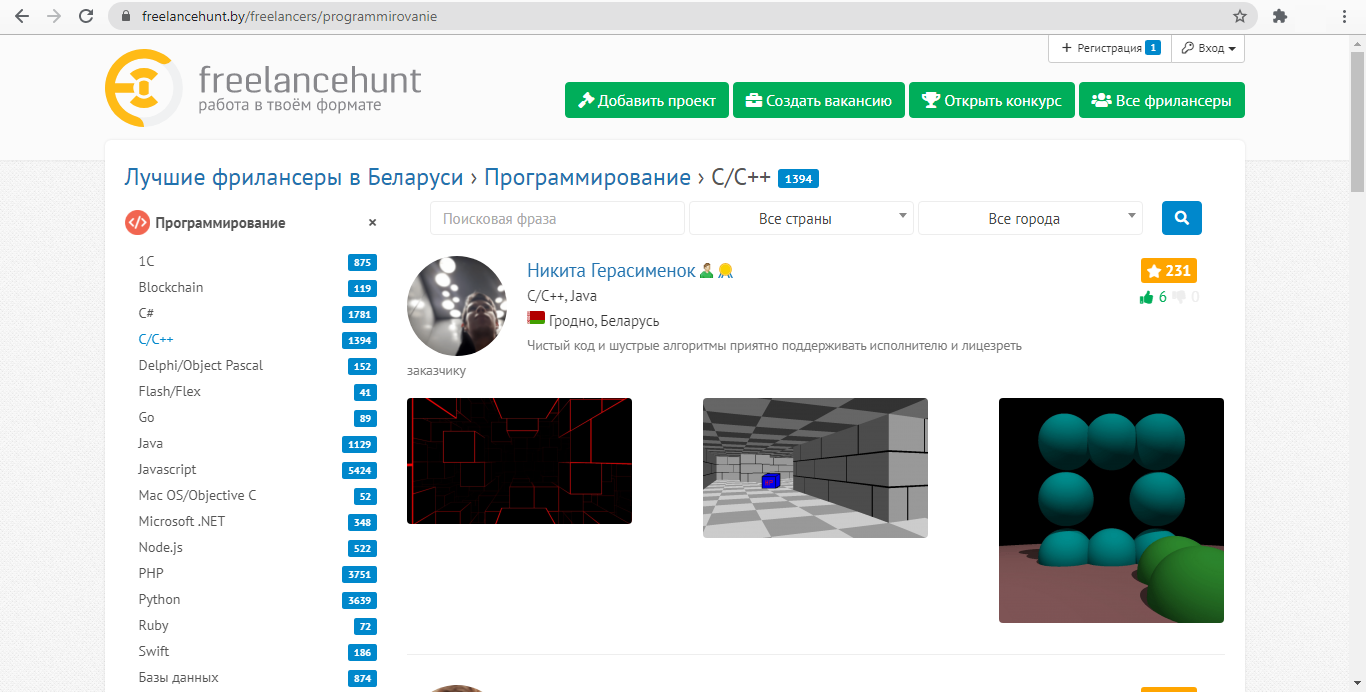


Рисунок 1.3 – Сайт *FreelanceHunt.by*

Главными преимуществами данного сайта является удобный пользовательский интерфейс, гарантия оплаты услуги клиентом, оперативная техподдержка, наличие конкурсов среди программистов, небольшая конкуренция, возможность предоставить свои услуги клиентам из Европы. Кроме того, создание своего портфолио и голосование за лучшие работы стимулируют и способствуют профессиональному росту программистов. Профили недобросовестных программистов или клиентов удаляются.

Незначительным недостатком данного сайта является комиссия 3% при выводе средств оплаты услуги программиста.

### 1.1.4 Сайт Flagma.by

Данный сайт предназначен для получения информации об услугах программистов Беларуси.

Основные возможности:

* просмотр услуг программистов;
* добавление услуги в избранное;
* регистрация личного кабинета;
* возможность пожаловаться на услугу;
* выбор категорий в зависимости от используемой программистом технологии и языка программирования;
* быстрый поиск по сайту;
* применить фильтр по цене к перечню услуг;
* возможность отправить личное сообщения программисту;
* относительно быстрая загрузка страниц с объемными фотографиями.

На рисунке 1.4 представлен снимок экрана главной страницы сайта.

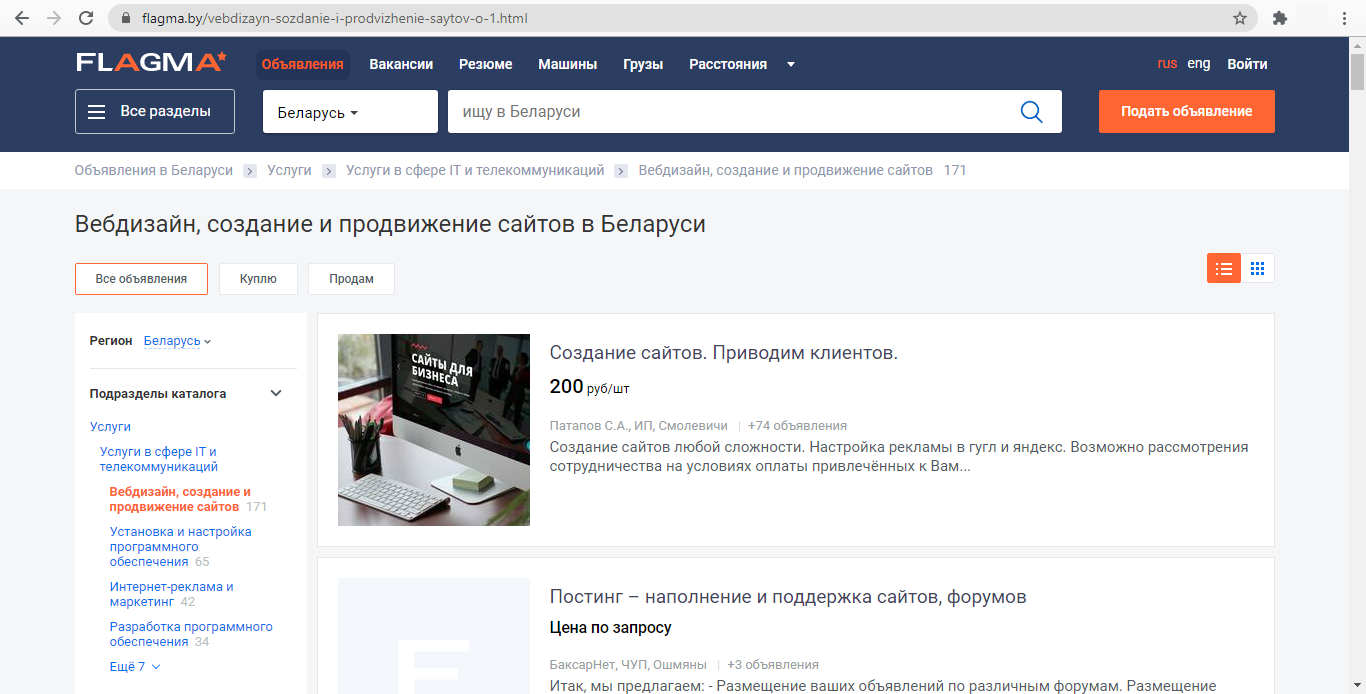


Рисунок 1.4 – Сайт *Flagma.by*

Благодаря возможности пожаловаться на клиента или оказывающего услугу программиста, практически исключена возможность попасть на недобросовестного партнера, поскольку профили с жалобами удаляются. Возможность доблавения услуги в избранное позволяет с легкостью вернуться к ней в будущем без повторного поиска, а также увидеть сколько людей уже добавило услугу в избранное и сделать вывод о ее качестве.

Недостатком сайта является недостаточно полное количество подкатегорий используемых технологий, а также неудобная навигация по сайту и отсутствие фильтрации услуг по региону или городу.

## Патентный поиск

В результате патентного поиска, был обнаружен патент, представленный в таблице 1.1.

Таблица 1.1 – Описание патента

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Номер патента | Опубликовано | Авторы |
| Способ и система приема заказов | RU2324221C2 | 10.05.2008 | Юкка Салонен |

Реферат: «Настоящее изобретение относится к области телекоммуникации. Более конкретно, настоящее изобретение относится к системе и способу оформления заказа (резервирования) в системе приема заказов и синхронизации приема заказов в нескольких системах приема заказов с участием, по меньшей мере, одного провайдера (поставщика) услуг (сервиса), службы-посредника, клиента и, по меньшей мере, одного терминального клиентского устройства, которое может являться мобильным устройством и которое поддерживает диалог. Кроме того, система по изобретению содержит линии телекоммуникации, которые используются для связывания систем приема заказов, провайдеров услуг, посредника и терминального клиентского устройства.

Возможность осуществлять прием заказов является необходимой для многих сервисов. В этой связи можно указать осуществление предварительной записи на услуги в области здравоохранения, оформление заказов (в рамках туристических услуг) на гостиницы, авиабилеты и прокат автомобилей; бронирование билетов на различные мероприятия; оформление предварительных заказов на обслуживание автомобиля; оформление предварительных заказов на обслуживание квартиры и т.д. Было бы весьма полезным, если бы перечисленные сервисы могли получать информацию друг от друга. Например, если пользователь заказывает билет на концерт, он может захотеть заказать также и столик в ресторане. В этой связи службе резервирования мест в ресторане было бы удобно получить соответствующую информацию (например, дату и имя заказчика) от системы приема заказов на театральные билеты. К сожалению, способы обмена информацией между различными системами приема заказов (резервирования) пока не разработаны.» [10].

## 1.3 Обзор средств разработки

В данном подразделе будет обоснован выбор языка программирования, среды разработки и платформы, которые будут использованы в разработке приложения.

### 1.3.1 Обоснование выбора языка программирования

Для разработки веб-приложения был выбран язык программирования *JavaScript*. *JavaScript* – это легкий, интерпретируемый язык с динамической типизацией. Является реализацией стандарта языка *ECMAScript*. Используется как язык сценариев веб-страниц, а также для разработки на сторону сервера на базе платформы *Node.js*.

Язык *JavaScript* очень популярен и еще долго будет занимать лидирующие позиции в веб-разработке. Также язык динамично развивается, в открытом доступе имеется иножетсов библиотек, фреймворков и обучающих материалов. Главными достоинствами языка *JavaScript* являются поддержка скриптов всеми популярными браузерами, быстрая скорость работы и производительность, простота и рациональность применения, а также удобство пользовательских интерфейсов [12].

### 1.3.2 Обоснование выбора средства программирования

Интегрированная среда разработки (*Integrated Development Environment – IDE*) – совокупность программных средств, с помощью которых программистами производится написание, корректировка, преобразование в машинные коды, наладка и запуск различного программного обеспечения.

*JetBrains WebStorm* – представляет собой инструмент для разработки веб-сайтов и редактирования *HTML*, *CSS* и *JavaScript* кода [13]. *WebStorm* можно использовать для создания различных типов приложений, от простых приложений для магазина и игр для мобильных клиентов до больших и сложных систем, обслуживающих предприятия и центры обработки данных.

Плагины, удобный интерфейс, хороший отладчик, удобный и быстрый редактор кода, делают *WebStorm* одним из лучших средств для разработки программного обеспечения.

### 1.3.3 Обоснование выбора платформы разработки

*Node.js* – это событийно-ориентированная платформа, применяемая для создания веб-приложений, которая обеспечивает возможность использовать *JavaScipt* на стороне сервера через движок *v8*. *Node.js* позволяет писать полноценные приложения, умеет работать с внешними библиотеками, вызывать команды из кода на *JavaScript* и выполнять роль веб-сервера.

Как асинхронное событийное *JavaScript*-окружение, *Node.js* спроектирован для построения масштабируемых сетевых приложений. Благодаря асинхронности пользователи *Node.js* могут не беспокоиться о блокировке потока процессами. Сегодня *Node.js* по праву считается одной из лидирующих платформ для веб-разработки, и большая часть веб-инструментов, серверных и клиентских, работают с этим инструментом [15].

Кроме того, к преимуществам платформы относят:

* богатая стандартная библиотека;
* большой выбор внешних библиотек и готовых модулей;
* пакетный менеджер *npm*;
* асинхронность в сочетании с событийным подходом;
* открытый исходный код;
* высокий темп развития.

## 1.4 Выбор технологий и библиотек

### 1.4.1 Фреймворк Express

*Express.js* – это один из мощнейших универсальных фреймворков для серверов. *Express.js* используется для разработки приложений достаточно давно и благодаря своей стабильности прочно занимает позицию одного из самых популярных фреймворков *Node.js*. Для этого фрейморвка существует большое количество подробных инструкций и описаний, которые составлены разработчиками, проверившими его эффективность на практике.

Основная особенность данного фреймворка заключается в том, что для *Express* характерен небольшой объем базового функционала. Все остальные нужные функции нужно использовать за счет внешних модулей. Благодаря такому минимализуму разработчик изначально получает в свое распоряжение легкий и быстрый инструмент, который он может расширять и развивать.

Главными преимуществами выбора фреймворка *Express* являются простота, гибкость, хорошая масштабируемость, подробная документация, широкий выбор подключаемых модулей.

### 1.4.2 Библиотека React для разработки пользовательских интерфейсов

Главная задача *React* – обеспечение вывода на экран того, что можно видеть на веб-страницах. *React* значительно облегчает создание интерфейсов благодаря разбиению каждой страницы на небольшие фрагменты, которые называются компонентами.

Компонент *React* – это участок кода, который представляет часть веб-страницы. Каждый компонент – это *JavaScript*-функция, которая возвращает кусок кода, представляющего фрагмент страницы.

Когда программист пытается манипулировать *html*-элементами с помощью *JavaScript*, то он может столкнуться со снижением производительности, особенно при изменении большого количества элементов. А операции над элементами могут занять некоторое время, что неизбежно скажется на пользовательском опыте. Для решения проблемы производительности появилась концепция виртуального *DOM*.

Виртуальный *DOM* – это концепция программирования, в которой идеальное или «виртуальное» представление пользовательского интерфейса хранится в памяти и синхронизируется с «реальным» *DOM* с помощью библиотеки *ReactDOM* [17]. Такой процесс называется согласованием. Данный подход обеспечивается декларативным *React* *API*: программист говорит *React*, в каком состоянии должен находиться *UI*, и он гарантирует, что *DOM* соответствует этому состоянию. Это абстрагирует манипуляции с атрибутами, обработку событий и ручное обновление *DOM*, которые в противном случае пришлось бы использовать для создания приложения.

## Вывод по разделу

В данном разделе было дано подробное описание какими качествами и свойствами должен обладать сайт аренды услуг программиста.

Был проведен патентный поиск. В качестве аналогов разрабатываемого веб-приложения были рассмотрены следующие сайты: *TrueTech.by*, *RedSale.by*, *FreelanceHunt.by*, *Flagma.by*.

Для разработки веб-приложения был обоснован выбор языка программирования, платформы разработки, интегрированной среды разработки.

В процессе разработки было решено использовать: фреймворк для создания веб-приложений *Express.js*, *ORM* технологию доступа к данным *Mongoose*, *JavaScript*-библиотеку для разработки пользовательских интерфейсов *React*.

Серверная часть приложения должна обеспечивать хранение данных, основываться на *REST*-архитектуре. В свою очередь клиентская часть должна предоставлять информацию об услугах программиста, позволять пользователю осуществлять бронирование, быть легко поддерживаемой.

# **2 Проектирование веб-приложения**

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*16*

*БГТУ 02.00.ПЗ*

Разраб*.*

*Чистякова Ю.А.*

Провер*.*

*.*

*Годун А.В.*

Консульт.

*Годун А.В.*

Н*.* контр*.*

*Рыжанкова А.С.*

Утв*.*

*Пацей Н.В.*

2 Проектирование

веб-приложения

*Лит.*

*Листов*

*8*

74417011, 2021

Проектирование архитектуры приложения является первым этапом борьбы со сложностью программного средства, на котором реализуется принцип выделения относительно независимых компонент.

## 2.1 Проектирование диаграммы развертывания

Физическое представление программной системы не может быть полным, если отсутствует информация о том, на какой платформе, а также на каких   
вычислительных средствах она реализована.

Для визуализации элементов и компонентов программы была разработана диаграмма развертывания, представленная на рисунке 2.1 и в приложении А.

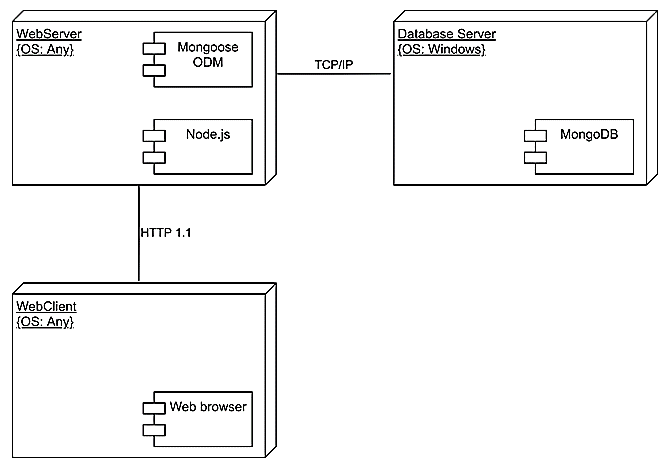


Рисунок 2.1 – Диаграмма развертывания

В главной роли, по обработке всех данных, а также в формировании и   
передаче их по удаленным каналам выступает *Web* *Server*. Клиент посылает запросы на сервер по *HTTP* протоколу версии 1.1 [17].

Все процессы, отвечающие за хранение данных возложены на *MongoDB*. Данная диаграмма более детально описывает архитектуру программного средства, указывая какие технологии и библиотеки, в каких модулях используются. Структурная схема веб-приложения представлена в приложении Д.

## 2.2 Проектирование диаграммы вариантов использования

Диаграмма вариантов использования (диаграмма прецедентов) в *UML* – диаграмма, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Основное назначение диаграммы – описание функциональности и поведения, позволяющее заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую или существующую систему.

Функциональные возможности приложения отображены на диаграмме вариантов использования, которая представлена в приложении Б и на рисунке 2.2.

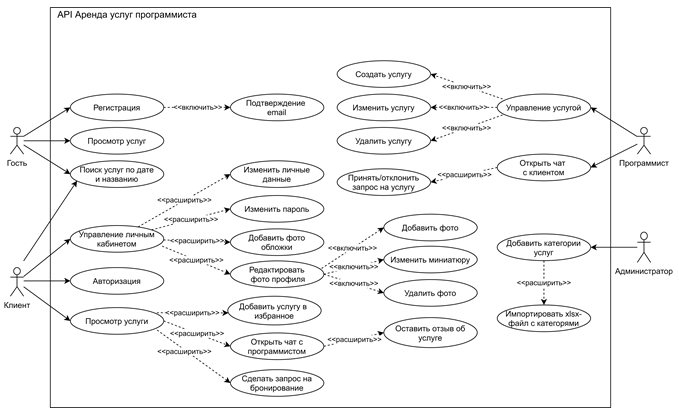


Рисунок 2.2 – Диаграмма вариантов использования

В приложении представлены четыре роли: Гость, Клиент, Программист, Администратор. В задачи программиста входят управление услугой, а также принятие или отклонение запросов на выполнение услуг от клиента после открытия чата. Возможности администратора ограничены созданием категорий. Клиент может управлять личным кабинетом, а также добавлять услугу в избранное, создавать отзывы и посылать запросы на бронирование.

В результате была разработана диаграмма вариантов использования, на которой изображен функционал разрабатываемого программного средства. С использованием данной диаграммы будет проходить дальнейшая разработка приложения.

Описание актеров разрабатываемого приложения приведено в таблице 2.1.

Таблица 2.1 – Описание актеров

|  |  |
| --- | --- |
| Актер | Описание |
| Гость | Пользователь, который впервые зашел на сайт или не прошедший этап авторизации |
| Клиент | Выполняет базовые функции приложения: создание запроса на услугу программиста, добавление услуги в избранное, создание отзыва, ведение чата |
| Программист | Выполняет функцию модерации контента: создание, удаление, изменение услуг; принятие или отклонение запроса на услугу от клиента; ведение чата |
| Администратор | Выполняет единственную функцию добавления категорий услуг программиста |

## 2.3 Проектирование базы данных

В разрабатываемом веб-приложении используется *NoSQL*-система управления базами данных – *MongoDB*, которая является документо-ориентированной системой, где центральным понятием являются коллекции и документы.

Также был использован графический интерфейс программы *Compass* для управления данными: добавления, изменения, удаления.

Для сопоставлеия объектов классов и документов коллекций из базы данных используется специальная *ODM*-библиотека – *Mongoose*.

В результате была создана база данных, которая удовлетворяет поставленным задачам. Данная реализация достаточно проста и универсальна, с ее помощью в дальнейшем можно изменять и дополнять структуру базы данных. Для реализации функций, которые были определены на стадии проектирования, были созданы следующие коллекции:

* *Bookings* (информация о заказах услуг);
* *Calendars* (информация о календаре услуги);
* *Likes* (информация о лайках);
* *Messages* (информация о сообщениях из чата);
* *Offerphotos* (информация о фотографиях услуг);
* *Offers* (информация об услугах программиста);
* *ProgrammerCategories* (информация о категориях услуг);
* *Reviews* (информация об отзывах);
* *Userphotos* (информация о фотографиях пользователей);
* *Users* (информация о пользователях).

Для реализации функционала веб-приложения было создано десять коллекций. Далее рассмотрим структуру документа в каждой коллекции базы данных.

Таблица «*Bookings*» содержит информацию о заказах услуг программиста: идентификатор, статус, идентификаторы клиента и программиста создателя, идентификатор услуги, даты начала и окончания, назначенный адрес выполнения услуги, детали адреса, время начала, дополнительная информация. Описание ее полей приведено в таблице 2.2.

Таблица 2.2 – Структура таблицы «*Bookings*»

|  |  |  |
| --- | --- | --- |
| № | Название | Описание |
| 1 | *\_id* | Идентификатор заказа |
| 2 | *Status* | Статус заказа, может принимать значения: *open* (открыт), *approve* (подтвержден) |
| 3 | *ClientId* | Идентификатор клиента |
| 4 | *OfferCreatorId* | Идентификатор исполняющего программиста |
| 5 | *OfferId* | Идентификатор услуги |
| 6 | *StartDate* | Дата начала услуги |
| 7 | *EndDate* | Дата окончания услуги |
| 8 | *Location* | Назначенный адрес выполнения услуги |
| 9 | *LocationDetails* | Детали адреса |
| 10 | *StartTime* | Время начала выполнения услуги |
| 11 | *AdditionalTerms* | Дополнительная информация |

Таблица «*Calendars*» содержит информацию о конкретном дне в календаре выполнения услуги: идентификатор, идентификаторы клиента, заказа и услуги, дата, статус, цена за день. Описание ее полей приведено в таблице 2.3.

Таблица 2.3 – Структура таблицы «*Calendars*»

|  |  |  |
| --- | --- | --- |
| № | Название | Описание |
| 1 | *\_id* | Идентификатор дня |
| 2 | *Status* | Статус возможности выполнения услуги в данный день, может принимать значения: *enabled* (возможен), *disabled* (невозможен) |
| 3 | *ClientId* | Идентификатор клиента, который забронировал услугу в данный день |
| 4 | *BookingId* | Идентификатор заказа, если на данный день услуга забронирована |
| 5 | *OfferId* | Идентификатор услуги |
| 6 | *Date* | Календарная дата дня |
| 7 | *PricePerDay* | Стоимость выполнения услуги за день |

Таблица «*Likes*» содержит информацию об услугах, добавленных в избранное: идентификатор, идентификатор пользователя, идентификатор услуги, дата создания. Описание ее полей приведено в таблице 2.4.

Таблица 2.4 – Структура таблицы «*Likes*»

|  |  |  |
| --- | --- | --- |
| № | Название | Описание |
| 1 | *\_id* | Идентификатор лайка |
| 2 | *UserId* | Идентификатор пользователя, который поставил лайк |
| 3 | *OfferId* | Идентификатор услуги, добавленной в избранное |
| 4 | *CreatedAt* | Дата добавления услуги в избранное |

Таблица «*Messages*» содержит информацию о сообщениях в чате: идентификатор, статус, идентификатор заказа, текст, идентификатор отправителя, номер. Описание ее полей приведено в таблице 2.5.

Таблица 2.5 – Структура таблицы «*Messages*»

|  |  |  |
| --- | --- | --- |
| № | Название | Описание |
| 1 | *\_id* | Идентификатор сообщения |
| 2 | *Status* | Статус сообщения, может принимать значения: *read* (прочитано), *pending* (в ожидании) |
| 3 | *BookingId* | Идентификатор заказа, к которому привязан чат |
| 4 | *Text* | Текст сообщения |
| 5 | *SenderId* | Идентификатор отправителя |

Таблица «*OfferPhotos*» содержит информацию о фотографиях, прикрепленным к услугам: идентификатор, флаг фотографии обложки, идентификатор услуги, путь к файлу с фотографией. Описание ее полей приведено в таблице 2.6.

Таблица 2.6 – Структура таблицы «*OfferPhotos*»

|  |  |  |
| --- | --- | --- |
| № | Название | Описание |
| 1 | *\_id* | Идентификатор фотографии |
| 2 | *CoverPhoto* | Является ли данная фотография обложкой услуги |
| 3 | *OfferId* | Идентификатор услуги |
| 4 | *PhotoUrl* | Путь к файлу с фотографией |

Таблица «*Offers*» содержит информацию об услугах: идентификатор, статус, номер шага, url видео, количество лайков, идентификатор пользователя, категория, подкатегория, название, описание, дата создания, максимальное и минимальное количество раз выполнения услуги, валюта оплаты, цена. Описание ее полей приведено в таблице 2.7.

Таблица 2.7 – Структура таблицы «*Offers*»

|  |  |  |
| --- | --- | --- |
| № | Название | Описание |
| 1 | *\_id* | Идентификатор услуги |
| 2 | *Status* | Статус услуги, принимает значения: open (открыта), closed (закрыта) |
| 3 | *Step* | Текущий шаг создания услуги |
| 4 | *VideoUrl* | Ссылка на видео YouTube |
| 5 | *CountLikes* | Количество лайков |

Окончание таблицы 2.7

|  |  |  |
| --- | --- | --- |
| № | Название | Описание |
| 6 | *UserId* | Идентификато программиста |
| 7 | *Category* | Категория услуги |
| 8 | *SubCategory* | Подкатегория услуги |
| 9 | *Name* | Название услуги |
| 10 | *Summary* | Описание услуги |
| 11 | *CreatedAt* | Дата и время создания услуги |
| 12 | *maxNumberOfActs* | Максимальное количество раз выполнения услуги |
| 13 | *minNumberOfActs* | Минимальное количество раз выполнения услуги |
| 14 | *Currency* | Валюта оплаты |
| 15 | *PricePerAct* | Цена за услугу |

Таблица «*ProgrammerCategories*» содержит информацию о категориях услуг программиста: идентификатор, категория, родительская категория. Описание ее полей приведено в таблице 2.8.

Таблица 2.8 – Структура таблицы «*ProgrammerCategories*»

|  |  |  |
| --- | --- | --- |
| № | Название | Описание |
| 1 | *\_id* | Идентификатор категории |
| 2 | *Category* | Название категории |
| 3 | *ParentCategory* | Название родительской категории, если такая есть |

Таблица «*Reviews*» содержит информацию об отзывах на услуги: идентификатор, текст, идентификаторы пользователя, услуги и заказа, дата создания. Описание ее полей приведено в таблице 2.9.

Таблица 2.9 – Структура таблицы «*Reviews*»

|  |  |  |
| --- | --- | --- |
| № | Название | Описание |
| 1 | *\_id* | Идентификатор отзыва |
| 2 | *Text* | Текст отзыва |
| 3 | *UserId* | Идентификатор пользователя, написавшего отзыва |
| 4 | *OfferId* | Идентификатор услуги |
| 5 | *BookingId* | Идентификатор заказа |
| 6 | *CreatedAt* | Дата и время создания отзыва |

Таблица «*UserPhotos*» содержит информацию о фотографиях галереи пользователей: идентификатор, идентификатор пользователя, путь к файлу с фотографией. Описание ее полей приведено в таблице 2.10.

Таблица 2.10 – Структура таблицы «*UserPhotos*»

|  |  |  |
| --- | --- | --- |
| № | Название | Описание |
| 1 | *\_id* | Идентификатор фотографии |
| 2 | *UserId* | Идентификатор пользователя |
| 3 | *PhotoUrl* | Путь к файлу с фотографией |

Таблица «*Users*» содержит информацию о пользователях: идентификатор, роль, флаг подтверждения почты, статус, *email*, имя, фамилия, пароль, дата рождения, токен, пол, фотография профиля и обложки. Описание ее полей приведено в таблице 2.11.

Таблица 2.11 – Структура таблицы «*Users*»

|  |  |  |
| --- | --- | --- |
| № | Название | Описание |
| 1 | *\_id* | Идентификатор пользователя |
| 2 | *Role* | Роль пользователя, может принимать одно или несколько значений: *user* (клиент), *admin* (администратор), *programmer* (программист) |
| 3 | *ConfirmMail* | Флаг подтверждения электронной почты |
| 4 | *Status* | Статус активации пользователя, может принимать значения: *isActive* (активирован), *inActive* (неактивирован) |
| 5 | *Email* | Почтовый адрес пользователя |
| 6 | *FirstName* | Имя пользователя |
| 7 | *LastName* | Фамилия пользователя |
| 8 | *Password* | Пароль пользователя для входа в учетную запись |
| 9 | *Birthday* | Дата рождения пользователя |
| 10 | *Token* | Битовая последовательность, которую сервер может проверить на валидность, имеет время жизни |
| 11 | *Gender* | Пол пользователя |
| 12 | *ProfilePhotoUrl* | Путь к фотографии профиля |
| 13 | *CoverPhoto* | Фотография обложки |

Логическая структура базы данных для данного веб-приложения представлена в приложении Е.

## 2.4 Описание алгоритма бронирования услуги

Для того, чтобы забронировать услугу пользователю необходимо авторизироваться в веб-приложении с ролью Клиент, затем выбрать одну из возможных услуг. Может быть так, что на текущий момент у программиста нет свободных дат для бронирования. В таком случае пользователю будет предложено выбрать другую услугу интересующей технологии.

Выбрав услугу, пользователь выбирает нужные даты бронирования из возможных свободных дат в календаре, и нажимает кнопку «*Book dates*». После этого пользователь подробно указывает критерии бронирования, точный адрес и время встречи с программистом, и отправляет запрос на бронирование услуги нажатием кнопки «*Send request to programmer*». Заказ при этом приобретает статус «*Respond In*». Далее происходит автоматический переход на страницу с сообщениями, где создается чат между данными клиентом и программистом. Там пользователи могут договориться о деталях бронирования, а также отследить текущий статус выполнения услуги. Блок-схема, иллюстрирующая бронирование услуги, представлена в приложении Ж.

После выполнения описанных в блок-схеме действий, в учетной записи программиста также появится чат с данным клиентом, где он сможет принять или отклонить запрос нажатием кнопки «*Approve*» или «*Decline*», соответственно. Программная реализация описанного алгоритма будет приведена в следующем разделе, который посвящен разработке веб-приложения.

## 2.5 Описание алгоритма создания услуги

Для того, чтобы создать услугу, пользователю необходимо авторизироваться в веб-приложении с ролью Программист. После успешной авторизации происходит считывание данных об услугах текущего пользователя. Если ранее услуги не были созданы, на экран будет выведено соответствующее сообщение «*You have no offers yet*», иначе будет выведен список услуг данного пользователя.

Далее необходимо выбрать категорию и подкатегорию услуг из созданных ранее администратором. Затем ввести всю необходимую подробную информацию об услуге, добавить фотографии, видео, выбрать дни выполнения услуги в календаре. После ввода валидных данных, услуга будет создана в базе данных, и система проинформирует пользователя об успешном создании услуги отправкой письма на почту. Далее произойдет автоматическая переадресация на страницу новой созданной услуги. Блок-схема, иллюстрирующая создание новой услуги, представлена в приложении З.

## 2.6 Вывод по разделу

Веб-приложение будет включать в себя следующие компоненты: *MongoDB Server*, сервер приложения на базе *Node.js*, клиентская часть приложения, разработанная с помощью библиотеки *React*.

В данной главе были представлены:

* диаграмма развертывания веб-приложения;
* диаграмма вариантов использования веб-приложения;
* описание алгоритма бронирования услуги;
* описание алгоритма создания услуги.

Для серверной части приложения спроектировано 10 таблиц базы данных. Для каждой спроектированной таблицы дано подробное описание.

В следующем разделе будет представлена сама реализация веб-приложения.

# **3 Разработка веб-приложения**

В первую очередь программа, должна решать поставленные задачи и хорошо выполнять свои функции, причем в различных условиях. Сюда можно отнести такие характеристики, как надежность, безопасность, производительность, способность справляться с увеличением нагрузки (масштабируемость).

## 3.1 Физическая структура веб-приложения

Серверная часть реализована с использованием *Node.js*, и представляет собой *API*, при помощи которого клиентская часть может получить необходимые данные. Архитектура серверной части веб-приложения представлена на рисунке 3.1.

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*24*

*БГТУ 03.00.ПЗ*

Разраб*.*

*Чистякова Ю.А.*

Провер*.*

*.*

*Годун А.В.*

Консульт.

*Годун А.В.*

Н*.* контр*.*

*Рыжанкова А.С.*

Утв*.*

*Пацей Н.В.*

3 Разработка

веб-приложения

*Лит.*

*Листов*

*17*

74417011, 2021

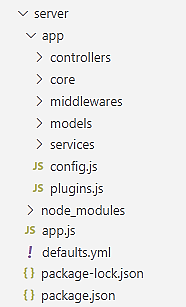


Рисунок 3.1 – Архитектура серверной части веб-приложения

– файл *package.json* использует пакетный менеджер *npm* и содержит в себе перечень модулей, которые использует приложение;

– файл *defaults.yml* содержит конфигурационные настройки приложения;

– файл *app.js* стартует при запуске приложения и содержит подключение базы данных и необходимых модулей;

– папка *node\_modules* в которой содержатся зависимости, установленные пакетным менеджером;

– папка *app/controllers* содержит обработчики маршрутов запросов разных методов, с указанием необходимых *middlewares* для каждого из них и имя обработчика из папки app/cores;

– папка *app/core* содержит определения тел основных обрабатывающих запросы функций;

– папка *app/middlewares* содержит функции промежуточной обработки;

– папка *app/models* содержит модели данных, ассоциированные с сущностями базы данных programmer-booking;

– папка *app/services* содержит файлы с описанием вспомогательных функций приложения;

– файл *app/config.js*, содержит подключение к файлу *default.yml* и обращения к *default*-переменным.

Клиентская часть разработанного приложения имеет структуру, изображенную на рисунке 3.2.

– папка *admin* содержит страницы для администратора;

– папка *booking* содержит страницы для оформления заказа услуги;

– папка *inbox* содержит страницы для управления сообщениями;

– папка *offers* содержит страницы для управления услугой;

– папка *profile* содержит страницы для управления профилем;

– файл *preview-offer.js* описывает страницу предпросмотра услуги;

– файл *registration.js* описывает страницу регистрации;

– файл *index.js* описывает домашнюю страницу со строкой поиска услуг;

– файл *login.js* описывает страницу авторизации.

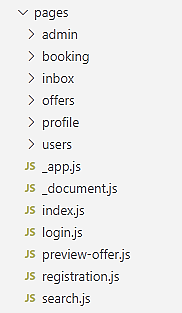


Рисунок 3.2 – Архитектура клиентской части веб-приложения

Помимо выше описанных файлов, структура клиентской части приложения включает в себя разработанные при помощи *React.js* компоненты, подключаемые файлы .*css* с набором определенных стилей, статические ресурсы файловой системы, подгружаемые на веб-страницу.

## 3.2 Технологии, используемые в серверной части приложения

В первом разделе был определен стек технологий серверной и клиентской части системы. Технологии, с помощью которых происходила разработка серверной части программного средства, представлены в таблице 3.1.

Таблица 3.1 – Технологии серверной части приложения

|  |  |
| --- | --- |
| Название | Назначение |
| *Node.js* | Серверная платформа для исполнения *JavaScript* |
| *REST API* | Подход к построению архитектуры *API* с передачей данных по протоколу *HTTP* |
| *MongoDB* | Система управления нереляционными базами данных. |
| *NodeMailer* | Библиотека для осуществления рассылки писем. |
| *Nodemon* | Технология следящая за обновлениями файлов и перезапускающая сервер |
| *Express* | Библиотека предоставляющая простой и мощный инструмент для создания веб-серверов и *API* на базе прокола *HTTP* |

Для взаимодействия с системой управления базами данных *MongoDB* использовался инструмент *MongoDBCompass* версии 1.26.1. Для написания серверной части кода и работы с *REST API* использовалась среда разработки *Microsoft* *Visual* *Studio* *Code* версии 1.45.0. *Nodemon* и дополнительные пакеты для *Node.js* добавлялись в проект при помощи менеджера пакетов *npm*. *NPM* – это менеджер пакетов, входящий в состав *Node.js*.

Он позволяет также легко устанавливать и обновлять сторонние библиотеки и инструменты в *Visual* *Studio Сode*. *NPM* не является обязательным для использования, но значительно упрощает управление зависимостями в приложении.

## 3.3 Реализация серверной части приложения

В данном разделе будут рассмотрены элементы, зависимости и принцип работы серверной части.

### 3.3.1 Взаимодействие с базой данных

Для администрирования базой данных использовался графический клиент *MongoDBCompass* поставляющийся в базовый комплект установки *MongoDB* и утилита для командной строки *mongo*.

После создания таблиц чтобы реализовать функциональную возможность подключения базы данных к приложению Express, необходимо загрузить модуль Node.js для соответствующей базы данных, в данном случае это *mongoose*.

Библиотека *mongoose* позволяет приложению *Node.js* общаться с *MongoDB*. При подключении базы данных будет использоваться локальное подключение.

Подключение базы данных из файла *app.js* представлено на рисунке 3.3.

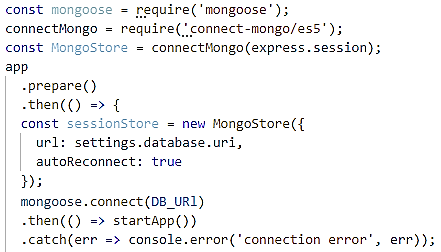


Рисунок 3.3 – Подключение базы данных

В данном файле запрашивается база данных, строка подключения к которой описана в файле *defaults.yml*, содержащем конфигурационные настройки.

### 3.3.2 Модели данных

Для работы с сущностями базы данных были разработаны модели данных, которые ассоциированы с данными сущностями и находятся в папка *app/models*. В таблице 3.2 приведен перечень моделей данных разработанного приложения.

Таблица 3.2 – Описание моделей данных приложения

|  |  |
| --- | --- |
| Название модели | Описание модели |
| *Booking* | Содержит все необходимые свойства для создания и удаления заказа услуги клиентом |
| *Calendar* | Содержит все необходимые свойства для создания и удаления рабочего дня в календаре услуги |
| *Like* | Содержит свойства для добавления услуги в избранное |
| *Message* | Содержит все необходимые свойства для отправки сообщения в чат программистом или клиентом |
| *OfferPhoto* | Содержит все необходимые свойства для создания и удаления фотографий услуги программиста |
| *Offer* | Содержит все необходимые свойства для создания и удаления услуги программиста |
| *ProgrammerCategory* | Содержит все необходимые свойства для создания и удаления категории услуг администратором |
| *Review* | Содержит все необходимые свойства для создания отзыва клиента на услугу программиста |
| *UserPhoto* | Содержит все необходимые свойства для создания и удаления фотографии пользователя |
| *User* | Содержит свойства однозначно иденцифицирующие зарегистрированного пользователя |

Каждая модель данных ассоциирована с соответствующей сущностью базы данных, а их свойства – с полями данной сущности. Для примера в листинге 3.1 приведен исходный код модели *Like*. Данная модель расположена к папке *app/models* и описывает сущность *Likes* базы данных.

|  |
| --- |
| const LikeSchema = new mongoose.Schema({    userId: {      type: ObjectId,      ref: 'User',      required: true,      trim: true    },    offerId: {      type: ObjectId,      ref: 'Offer',      required: true,      trim: true    },    createdAt: {      type: Date,      default: Date.now    }  }); |

Листинг 3.1 – Модель *Like*

После определения схемы *Mongoose* дает возможность создать модель, основанную на определенной схеме. Затем модель синхронизируется с документом *MongoDB* с помощью определения схемы модели. В данном примере схема имеет три свойства.

С помощью ссылки *ref* в схеме указывается, какая модель должна быть связана с этим полем. В данном случае поле *userId* будет ссылаться на идентификатор пользователя модели *User*, которорый добавил услугу в избранное, а поле *offerId* будет ссылаться на идентификатор понравившейся услуги.

С помощью атрибута *trim* мы указываем схеме, что следует удалять ведущие и хвостовые пробелы. Атрибут *required* указывает на обязательность заполнения данного поля при создании объекта модели. С помощью атрибута *default* можно указать значение по умолчанию, которое будет присвоено данному полю при создании модели, если оно не было указано.

### 3.3.3 Функции промежуточной обработки

Для промежуточной обработки запросов были разработаны промежуточные обработчики (*middleware*), которые расположены в папке *app/middlewares*. Функции промежуточной обработки – это функции, имеющие доступ к объекту запроса, объекту ответа и к следующей функции обработки в цикле «запрос-ответ» приложения.

Описание промежуточных обработчиков представлено в таблице 3.3.

Таблица 3.3 – Описание промежуточных обработчиков приложения

|  |  |
| --- | --- |
| Название обработчика | Описание обработчика |
| *CleanupFiles* | Содержит обработку функции предварительного удаления файлов в директории |
| *ReqAdmin* | Содержит проверку на наличие у текущего пользователя роли «Администратор» |
| *ReqClient* | Содержит проверку на наличие у текущего пользователя роли «Клиент» |
| *ReqDeserializeUser* | Содержит проверку на прохождение авторизации |
| *ReqProgrammer* | Содержит проверку на наличие у текущего пользователя роли «Программист» |

Для примера в листинге 3.2 приведен исходный код промежуточного обработчика *ReqAdmin*. Данный обработчик обеспечивает проверку пользователя на наличие роли администратора.

|  |
| --- |
| function getMiddleware(fail) {    return function (req, res, next) {      if (req.user && ~req.user.role.indexOf('admin')) {        next();        return;      }      fail(req, res);    };  }  module.exports = getMiddleware(function (req, res) {    res.sendStatus(401);  }); |

Листинг 3.2 – Промежуточный обработчик *ReqAdmin*

Если текущая функция промежуточной обработки не завершает цикл «запрос-ответ», она должна вызвать *next()* для передачи управления следующей функции промежуточной обработки.

Вызов промежуточных обработчиков осущеставляется из контроллеров, которые играют роль маршрутизатора, перехватывая запрос с определенным uri, вызывая промежуточные обработчики и передающие запрос далее по цепочке на основные функции обработки. Далее перейдем к рассмотрению контроллеров приложения.

### 3.3.4 Контроллеры

Контроллеры являются ключевым звеном в веб-приложении и предназначены для принятия запроса и формирования ответа. В данном приложении контроллер перехватывает запрос с определенным *url*, вызывает промежуточные обработчики функций (*middleware*), если это необходимо, и передает запрос далее по цепочке основному обработчику функции в папке *app/core*. Таким образом, контроллер реализует функции маршрутизатора запросов. Описание контроллеров приложения представлено в таблице 3.4.

Таблица 3.4 – Описание контроллеров приложения

|  |  |
| --- | --- |
| Название контроллера | Описание контроллера |
| *Admin* | Предназначен для работы с функциями администратора |
| *Chat* | Предназначен для работы с чатом |
| *Client* | Предназначен для работы с функциями клиента |
| *Offer* | Предназначен для работы с заказами |
| *User* | Предназначен для работы с данными пользователей |

Для примера в листинге 3.3 приведен исходный код одной из функций контроллера *Offers*.

|  |
| --- |
| app.route('/api/v1/offer/:id/price')      .all(expressJwt({        secret: settings.server.secret, fail: function (req, res, next) {          if (!req.headers.authorization) res.send(400, errorMessages.MISSING\_AUTHORIZATION\_HEADER);          res.send(401);        }      }))      .put(middlewares.reqDeserializeUser, middlewares.reqOrganiserProgrammer, (req, res) => {        core.offer.addPrice(req.user, req.body, req.params.id)          .then(offer => res.status(200).json(offer))          .catch(err => res.status(err.status || 500).json(err))      }); |

Листинг 3.3 – Контроллер *Offers*

Данная функция перехватывает запросы с заданным в параметре *uri*, проверяет действующий токен. Далее если запрос имеет метод *put*, функция контроллера применяет промежуточный обработчик *reqDeserializeUser* для проверки текущей авторизации пользователя, после чего передает запрос далее по цепочке обработчику функций *core.offer*, в котором дано описание тела функции *addPrice()* для установки цены услуги. Далее рассмотрим реализацию данной функции подробнее.

### 3.3.5 Основные обработчики функций

Конечным звеном в цепочке обработки запроса являются функции в папке *app/core*, которая содержит определения тел основных обрабатывающих запросы функций. Реализация функции подтверждения заказа приведена в приложении Ж.

В таблице 3.5 представлен перечень обработчиков функций разрабатываемого приложения.

Таблица 3.5 – Описание обработчиков функций приложения

|  |  |
| --- | --- |
| Название обработчика | Описание обработчика |
| *Admin* | Содержит обработчики функций администратора |
| *Chat* | Содержит обработчики функций чата |
| *Client* | Содержит обработчики функций клиента |
| *Offer* | Содержит обработчики функций заказа услуг |
| *User* | Содержит обработчики функций пользователей |

Для примера в листинге 3.4 приведен исходный код функции *addPrice()* обработчика функций *Offer*, которая реализует установку цены услуги программиста.

|  |
| --- |
| Offer.prototype.addPrice = function (creator, params, offerId) {    const Offer = mongoose.model('Offer');    return Offer.findOne({      \_id: offerId,      userId: creator.id    })      .then(existOffer => {        if (!existOffer) throw new AppError({status: 400, message: errorMessages.OFFER\_NOT\_FOUND});        existOffer.set('pricePerAct', params.pricePerAct);        existOffer.set('currency', params.currency);        return existOffer.save();      })      .catch(err => {        if (err instanceof AppError) throw err;        throw new AppError({status: 500, message: errorMessages.SERVER\_ERROR, err: err});      })  }; |

Листинг 3.4 – Обработчик функций *Offer*

Данная функция *addPrice()* вызывается из контроллера *offer*, описанного ранее. С помощью *ODM* *mongoose* функция находит нужную услугу модели *offer* с заданным идентификатором, после чего устанавливает новое значение полей *pricePerAct, currency* и сохраняет изменения в базе данных.

### 3.3.6 Сервисы

Под сервисами понимается такой слой приложения, назначением которого является выполнение какой-либо логики. Для того, чтобы не перегружать исходный код основных обрабатывающих функций в папке *app/core*, некоторая логика или повторно вызываемые участки кода были вынесены в папку *app/services*.

В таблице 3.6 представлен перечень сервисов разрабатываемого приложения.

Таблица 3.6 – Описание сервисов приложения

|  |  |
| --- | --- |
| Название сервиса | Описание сервиса |
| *DbHelper* | Содержит функции для наполнения базы данных данными и категорями услуг при создании |
| *Error* | Содержит функции для обработки возникающих ошибок |
| *ErrorMessages* | Содержит перечень сообщений ошибок |
| *FilesHelper* | Содержит функции для работы с файловой системой |
| *MailHelper* | Содержит функции для отправки писем на *email* пользователей |
| *MailTemplate* | Содержит шаблон сообщения подтверждения учетной записи |

В разработанном веб-приложении в данную папку также были вынесены шаблоны сообщений подтверждения учетной записи, которые пересылаются на *email* пользователей при регистрации. Далее будет подробнее рассмотрен один из сервисов *MailHelper*, предназначенный для отправки писем на *email* пользователей.

### 3.3.7 Реализация отправки писем

Для реализации отправки писем была использована библиотека *NodeMailer*. Эта популярная библиотека для приложений *Node*.*js* для простой отправки электронных писем.

*NodeMailer* это модуль с нулевыми зависимостями, простым и понятным кодом, с поддержкой юникода, обширным выбором методов транспортировки, поддержкой *SMTP*, наличие автоматически сгенерированных почтовых тестовых аккаунтов от *Ethereal*.*email*.*com.* Несомненными преимуществами *NodeMailer* являются непрерывная поддержка последних обновлений платформы, встроенный *sendmail* транспорт, поддержка *OAuth2* аутентификации, а также уведомления о статусе доставки.

Перед началом использования необходимо скачать модуль *npm* *nodemailer*. Сначала создается *transporter*, который хранит всю конфигурацию *SMTP*. Для его создания используется метод *createTransport*(), который принимает объект со следующими полями:

* *host*. Адрес почтового сервера, который будет отправлять посылаемые письма;
* *port*. Порт почтового сервера, по умолчанию может быть 25, 465 или 587;
* *secure*. Булевое значение, задается *true*, если используется *SSL*, в таком случае значением *port* должно быть 465;
* *auth*. Объект со свойствами *user* и *pass*, в которых указывается логин и пароль используемого почтового аккаунта.

В разработанном проекте в качестве транспорта для отправки сообщений использовался сервис *Yandex*. В поле *auth* были указаны почтовый адрес и пароль созданной учетной записи *yandex.com*.

Пример реализации метод *createTransport()* приведен на рисунке 3.4.



Рисунок 3.4 – Пример реализации метод *createTransport()*

Отправка письма осуществляется с помощью метода *sendMail()*. Он принимает объект message со следующей структурой:

* *to* – адрес получателя;
* *subject* – тема сообщения;
* *text* – текст сообщения;
* *html* – текст сообщения в формате *HTML*;
* *attachments* – прикрепленные к сообщению файлы.

Приведенная структура не обязательна и может изменяться в зависимости от желания вида конечного письма у пользователей.

### 3.3.8 Реализация аутентификации

Для реализации аутентификации и авторизации использован алгоритм *JSON Web Token*. Он применяется для создания токенов доступа, основан на формате JSON.

*JSON Web Tokens* – это стандарт аутентификации, который работает путем назначения и передачи зашифрованного токена в запросах, который помогает идентифицировать пользователя, вошедшего в систему, вместо сохранения пользователя в сеансе на сервере. Информация о пользователе и пароле передаются в *cookies*. Процедура аутентификации выглядит следующим образом:

* пользователь регистрируется и в базе данных создается запись о нем;
* пользователь входит в систему используя пароль, и при успешном вводе логина и пароля получает в ответ сгенерированный на основе секрета *jwt* отсервера;
* клиент сохраняет токен локально и отправляет его обратно при каждом последующем запросе, требующем аутентификацию;
* все запросы, требующие аутентификации, проходят через промежуточное ПО, которое проверяет предоставленный токен и разрешает запрос только в том случае, если токен проверен. Пользователь заходит на ресурс, отсылая свой *jwt*, по которому и авторизуется уже без ввода пароля.

*JWT*– это *JSON* в котором может храниться, например адрес почты пользователя. Но этот *JSON* подписывается секретным ключом, что не позволяет адрес почты изменить, но позволяет его прочитать.

Рассмотрим подробнее алгоритм хеширования паролей при регистрации нового пользоваеля и сравнивание хеша введенного пароля с хешем пароля пользователя из базы данных в процессе авторизации. Код алгоритма авторизации пользователя, расположенный в файле *models/user.js* представлен на рисунке 3.5.

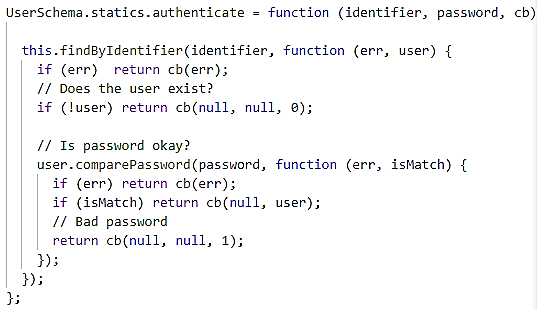


Рисунок 3.5 – Функция авторизации

Сначала система находит пользователя с нужным уникальным идентификатором. Если он найден, система сравнивает пароли с помощью разработанной функции *comparePassword*, исходный код которой представлен на рисунке 3.6.

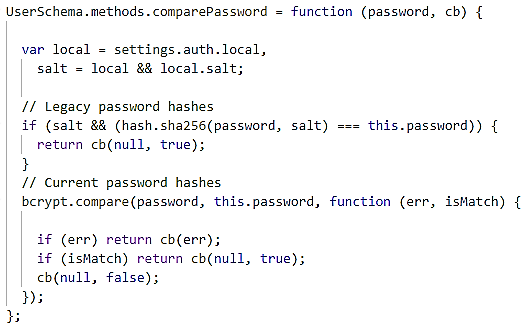


Рисунок 3.6 – Функция сравнения паролей

В приложении пароли хешируются при помощи алгоритма хеширования *sha256* и соли. Затем введенный пароль сравнивается с паролем пользователя из базы данных с помощью функции *compare() js*-библиотеки *bcrypt*.

## 3.4 Технологии, используемые в клиентской части приложения

Технологии, с помощью которых происходила разработка клиентской части программного средства представлены в таблице 3.7.

Таблица 3.7 – Технологии клиентской части приложения

|  |  |
| --- | --- |
| Название | Назначение |
| *React.js* | Серверная платформа для исполнения *JavaScript* |
| *Moment* | *JavaScript* библиотека для разбора и форматирования дат |
| *React circular progressbar* | Широко настраиваемый компонент *React.js* круговой прогрессбар, построенный с использованием *SVG* |
| *Semantic ui react* | Фреймворк для создания переносимых интерфейсов, который поможет повторно использовать *UI* в своих проектах |
| *Semantic ui css* | Компонент входящий в *Semantic ui react* с дефолтной темой |
| *Axios API* | Предоставляет интерфейс *JavaScript* для работы с запросами и ответами *Http* |

Для написания клиентской части кода и работы с *Axios API* использовалась среда разработки *Microsoft* *Visual* *Studio* *Code* версии 1.45.0. *Semantic-ui-react* и дополнительные пакеты для *React.js* добавлялись в проект при помощи менеджера пакетов *npm*.

### 3.4.1 Реализация клиентской части приложения

Библиотека *React* создана для пользовательских интерфейсов и построена вокруг концепции компонентов. Создание многократно используемых компонентов пользовательского интерфейса помогает поддерживать код при работе над крупномасштабными проектами, а также улучшает читаемость.

Компоненты – это небольшие части пользовательского интерфейса, которые

можно использовать повторно и работать отдельно. Во многом компоненты ведут себя как обычные функции *JavaScript*. Они реализуют метод *render(),* который принимают произвольные входные данные (*props*) и возвращает ответ для вывода. Помимо *props*, компонент поддерживает внутренние данные состояния *state*. Когда данные состояния компонента поменяются, *React* еще раз вызовет и обновит отрендеренную разметку. Для рендеринга *React*-элемента в корневой узел *DOM*, вызывается [*ReactDOM.render()*](https://ru.reactjs.org/docs/react-dom.html#render)*.*

На рисунке 3.7 показаны все реализованные компоненты приложения.

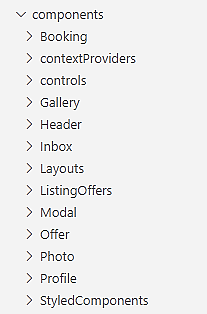


Рисунок 3.7 – Перечень компонентов приложения

Компоненты сгруппированы для удобства по ролям или по и нахождению. В дальнейшем готовые компоненты, к примеру, для регистрации пользователя можно переиспользовать в других проектах. Это является большим плюсом в использовании *React.js.*

### 3.4.2 Маршрутизация в клиентской части приложении

В разрабатываемом приложении для правильности получения данных используется *Axios* *API*, который предоставляет интерфейс *JavaScript* для работы с запросами и ответами *HTTP*. *Axios* *API* позволяет легко и понятно получать ресурсы по сети асинхронно. Он является современным и мощным инструментом, который поддерживают все современные браузеры. Применение вызова *axios* представлено на рисунке 3.8.

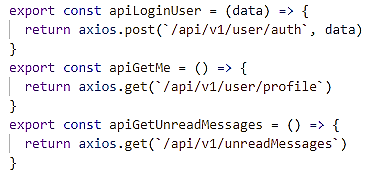


Рисунок 3.8 – Пример использования *axios*

В базовый синтаксис входит *URL* от конечной точки *API*, чтобы получить промис, возвращающий объект ответа, и дополнительные параметры.

Одна из особенностей, которая делает *axios* лучше *fetch()* – автоматическое преобразование *JSON*-данных. При использовании *fetch()* для передачи данных в *JSON*, необходимо сначала сделать фактический запрос, а затем вызвать метод *json()* для полученных данных с сервера. С помощью *axios* можно обойтись без передачи *HTTP*-запроса в метод *json()*. *Axios* возвращает именно тот объект с данными, который мы ожидаем.

### 3.4.3 Конечные точки HTTP API

Для взаимодействия клиентского приложения с серверным было разработано *REST* *API*. Данное *API* используется в клиентском приложении, которое разработано с помощью библиотеки *React*.

*REST* (*Representational State Transfer*) – это передача состояния представления. *REST API* представляется конечному пользователю как набор *uri*, к которым он может делать четыре типа запросов: *get*, *post*, *put*, *delete*. Главными преимуществами *REST*-приложений являются:

* надежность, за счет отсутствия необходимости сохранять информацию о состоянии клиента, которая может быть утеряна;
* производительность, за счет использования кэша;
* масштабируемость;
* легкость внесения изменений;
* простота интерфейсов;
* прозрачность системы взаимодействия.

В таблице 3.8 описаны методы разработанного *API*.

Таблица 3.8 – Описание методов REST API

|  |  |  |
| --- | --- | --- |
| Адрес | Метод | Описание |
| */api/v1/user* | *POST* | Метод предназначен для создания пользователя |
| */api/v1/user/auth* | *POST* | Метод предназначен для авторизации пользователя |
| */api/v1/user/profile* | *GET* | Метод предназначен для получения текущего пользователя |
| */api/v1/unreadMessages* | *GET* | Метод предназначен для получения непрочитанных сообщений |
| */api/v1/user/${id}* | *GET* | Метод предназначен для получения пользователя по идентификатору |
| */api/v1/user/profile* | *PUT* | Метод предназначен для изменения профиля пользователя |
| */api/v1/user/${id}/profilePhoto* | *POST* | Метод предназначен для изменения фотографии пользователя |
| */api/v1/user/${id}/profilePhoto* | *DELETE* | Метод предназначен для удаления фотографии профиля пользователя |

Продолжение таблицы 3.8

|  |  |  |
| --- | --- | --- |
| Адрес | Метод | Описание |
| */api/v1/user/${id}/offers* | *GET* | Метод предназначен для получения перечня услуг заданного пользователя |
| */api/v1/user/myLikeOffers* | *GET* | Метод предназначен для получения добавленных в избранное услуг |
| */api/v1/user/updatePassword* | *PUT* | Метод предназначен для изменения пароля |
| */api/v1/user/photo* | *POST* | Метод предназначен для добавления фотографий в галерею |
| */api/v1/user/photo* | *GET* | Метод предназначен для получения фотографий из галереи |
| */api/v1/user/photo/${id}* | *DELETE* | Метод предназначен для удаления фотографии из галереи |
| */api/v1/user/photo/${id}/coverPhoto* | *PUT* | Метод предназначен для изменения фотографии обложки |
| */api/v1/user/programmer* | *POST* | Метод предназначен для создания программиста |
| */api/v1/user/programmerCategories* | *GET* | Метод предназначен для получения категорий услуг |
| */api/v1/offer* | *POST* | Метод предназначен для создания услуги |
| */api/v1/offer/${id}* | *PUT* | Метод предназначен для редактирования услуги |
| */api/v1/offer/${id}* | *DELETE* | Метод предназначен для удаления услуги |
| */api/v1/offer/${id}* | *GET* | Метод предназначен для получения услуги |
| */api/v1/offer/details/photo?offerId=${id}* | *GET* | Метод предназначен для получения фотографии услуги |
| */api/v1/offer/details/photo/${id}* | *DELETE* | Метод предназначен для удаления фотографии услуги |
| */api/v1/offer/details/photo/${data.id}* | *PUT* | Метод предназначен для изменения фотографии услуги |
| */api/v1/offer/${offerId}/setAvatar* | *PUT* | Метод предназначен для установки фотографии обложки услуги |
| */api/v1/offer/${data.offerId}/calendar* | *POST* | Метод предназначен для создания календаря услуги |
| */api/v1/offer/${data.offerId}/calendar/disabledDays* | *POST* | Метод предназначен для создания неактивных дней в календаре услуги |
| */api/v1/offer/${data.offerId}/price* | *PUT* | Метод предназначен для установки цены услуги |

Окончание таблицы 3.8

|  |  |  |
| --- | --- | --- |
| Адрес | Метод | Описание |
| */api/v1/offer/copyOffer* | *POST* | Метод предназначен для клонирования услуги |
| */api/v1/preChat* | *POST* | Метод предназначен для создания чата об услуге |
| */api/v1/user/${offerId}/like* | *POST* | Метод предназначен для добавления услуги в избранное |
| */api/v1/user/offer/${offerId}/reviews* | *GET* | Метод предназначен для получения отзывов об услуге |
| */api/v1/chats* | *GET* | Метод предназначен для получения чатов |
| */api/v1/chat/${id}* | *GET* | Метод предназначен получения чата |
| */api/v1/chat/${id}/messages* | *GET* | Метод предназначен для получения сообщений из чата |
| */api/v1/chat/${chatId}/messages* | *POST* | Метод предназначен для отправки сообщения в чат |
| */api/v1/chats/updateStatus* | *PUT* | Метод предназначен для изменения статуса сообщений |
| */api/v1/offer/${offerId}/approve* | *PUT* | Метод предназначен для подтверждения заказа услуги |
| */api/v1/offer/${offerId}/decline* | *PUT* | Метод предназначен для отклонения заказа услуги |
| */api/v1/client/${bookingId}/review* | *POST* | Метод предназначен для создания отзыва об услуге |
| */api/v1/admin/newUploadProgrammerCategories* | *POST* | Метод предназначен для добавления категорий услуг |

## 3.5 Вывод по разделу

В данной главе была подробно описана разработка серверной и клиентской части приложения.

Для серверной части были описаны технологии, примененные для ее разработки. Рассмотрены этапы разработки, а именно: подключение и взаимодействие с базой данных, разработка моделей, контроллеров, промежуточных и основных обработчиков, сервисов.

Для клиентской части также приведен перечень использованных технологий. Описаны элементы, из которых состоит клиентская часть приложения, компоненты и конечные точки *HTTP API*.

Благодаря рассмотренным компонентам реализованного приложения можно представить общий процесс работы.

# **4 Тестирование веб-приложения**

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*40*

*БГТУ 04.00.ПЗ*

Разраб*.*

*Чистякова Ю.А.*

Провер*.*

*.*

*Годун А.В.*

*Консульт*.

*Годун А.В.*

Н*.* контр*.*

*Рыжанкова А.С.*

Утв*.*

*Пацей Н.В.*

*Лит.*

*Листов*

*7*

74417011, 2021

## 4.1 Негативное тестирование страницы входа

Негативным называют тестирование, которое применяет сценарии, соответствующие внештатному поведению тестируемой системы. Примерами такого тестирования могут быть исключительные ситуации или неверные данные. Некоторый функционал приложения доступен авторизованному пользователю, по этой причине протестируем страницу входа для пользователя.

На экране входа в приложение находится поля *email* и пароля, кнопка входа в приложение. Изначально попытаемся оставить пустой форму *email* и пароля, и попробуем войти в приложение. Результат выполнения данного теста можно увидеть на рисунке 4.1.

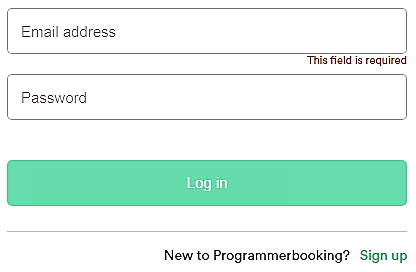


Рисунок 4.1 – Пример работы валидации формы входа

Веб-приложение, как и было запланировано, отобразило сообщения правил валидации о необходимости заполнения обязательных полей. При этом вся проверка валидации осуществлялась «на лету» по мере введения данных в форму.

Поле *email* в данном веб-приложении имеет три ограничения валидации: длина должна быть не менее 3 символов, электронный адрес должен подходить по стандарту и поле должно быть заполненным. Протестируем правило на ввод невалидного *email*.

4 Тестирование

веб-приложения

Результат выполнения данного теста можно увидеть на рисунке 4.2.

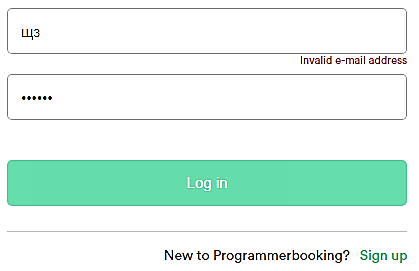


Рисунок 4.2 – Пример тестирования на не валидный *email*

Также форма имеет валидационное правило на существующий *email*, протестируем его. Попытаемся ввести несуществующий *email* пользователя, которого не существует в базе данных. Результат выполнения данного теста можно увидеть на рисунке 4.3.

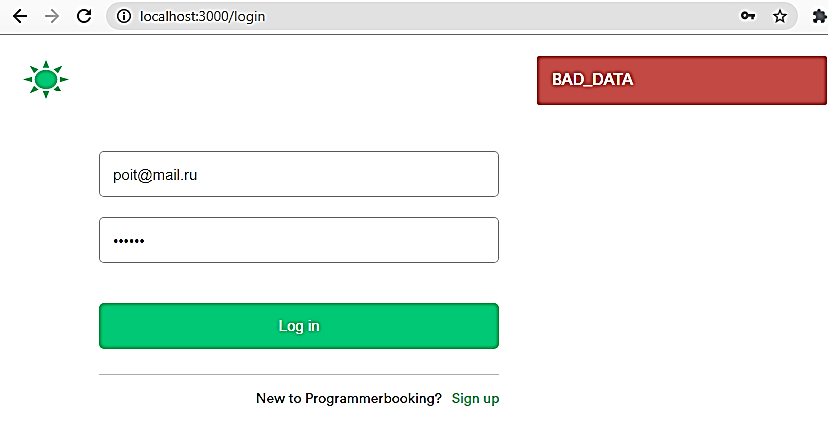


Рисунок 4.3 – Пример тестирования на несуществующий *email*

Веб-приложение, как и было запланировано, оповестило пользователя о вводе неверных данных с помощью уведомление «*BAD\_DATA*». Аналогичный результат будет получен при вводе существующего *email* и неверного пароля.

Реализация данной валидации осуществляется вручную на серверной стороне и на клиентской, посредством прямого вывода ошибок на форму входа.

## 4.2 Негативное тестирование страницы регистрации

Следующим объектом тестирования рассмотрим форму регистрации нового пользователя. На экране регистрации находятся поля ввода *email*, имени, фамилии, пароля и даты рождения.

Все поля являются обязательными для заполнения. Проверка факта заполнения полей формы регистрации также осуществляется «на лету». Попытаемся оставить пустым одно из полей. Результат выполнения данного теста можно увидеть на рисунке 4.4.

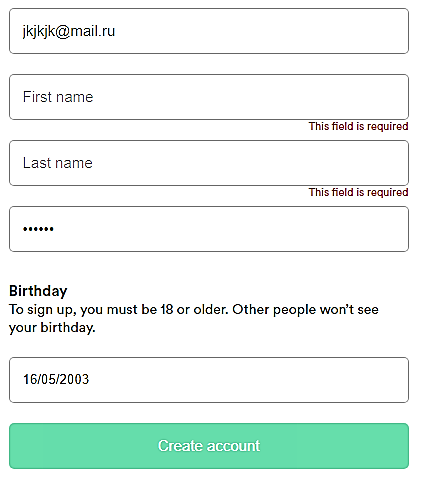


Рисунок 4.4 – Пример работы валидации формы регистрации

Веб-приложение предназначено для лиц старше восемнадцати лет. Данная проверка осуществляется уже на этапе регистрации при выборе даты рождения пользователя в календаре. Выбор невалидной даты в календаре невозможен, поскольку будущие даты, даты менее чем восемнадцатилетней давности и даты рождения, лишенные смысла, являются неактивными для нажатия.

Аналогично с формой авторизации, для формы регистрации предусмотрено валидационное правило ввода корректного *email*, который соответствует стандартам. Протестируем данное правило. Для этого в поле *email* введем набор символов, которые нарушают одно или несколько ограничений валидации электронных адресов: имеют длину менее трех символов или не соответствуют стандарту *email*-адресов.

Результат данного теста продемонстрирован на рисунке 4.5.

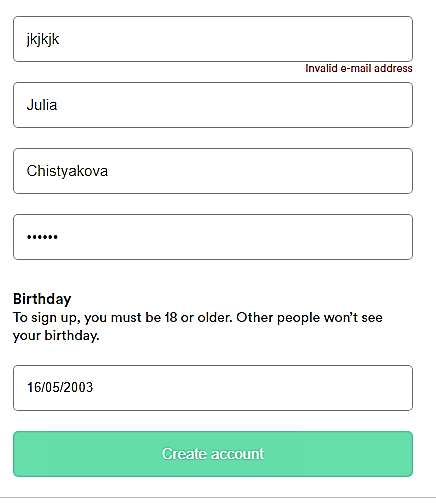


Рисунок 4.5 – Пример тестирования на не валидный *email*

Также предусмотрено валидационное правило для ввода пароля, длина которого должна быть не менее шести символов. Результаты данного теста представлены на рисунке 4.6.

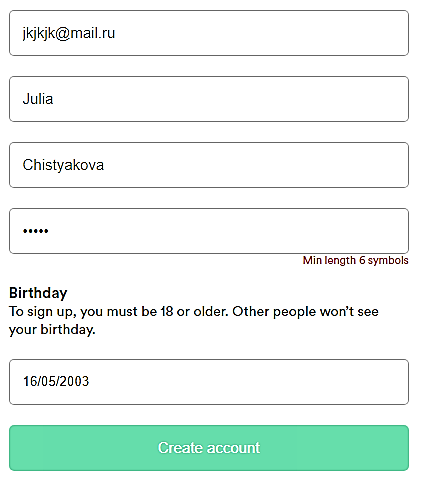


Рисунок 4.6 – Пример тестирования на короткий пароль

В случае, если пользователь при регистрации укажет email, который уже зарегистрирован в системе, и нажмет кнопку «*Create account*», будет выведена ошибка, представленная на рисунке 4.7.

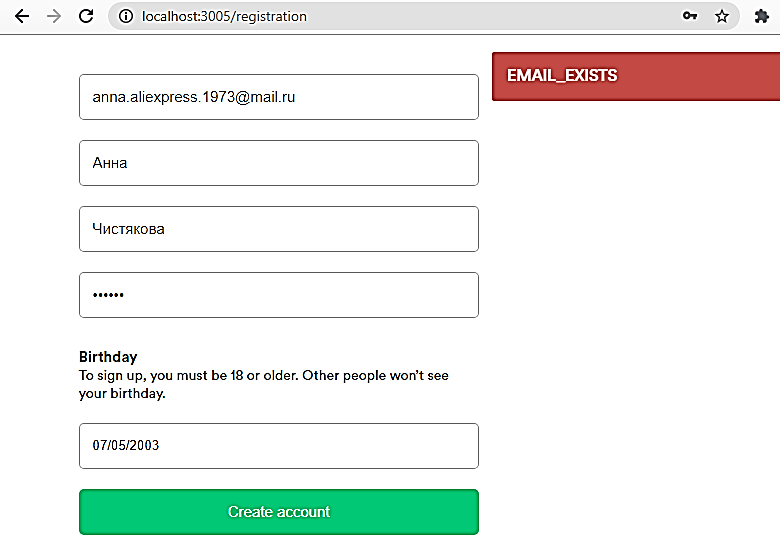


Рисунок 4.7 – Пример тестирования на существующий *email*

## 4.3 Негативное тестирование форм создания услуги

Следующим объектом тестирования рассмотрим формы создания новой услуги. При первом шаге создания услуги – заполнении категории, подкатегории, названия и описания услуги происходит мгновенная проверка введенных в форму данных. Выбор категории и заполнение полей названия и описания являются обязательными. В случае невыполнения одного из этих условий, кнопка «*Continue*» для перехода на следующий этап создания услуги программиста будет недоступной для нажатия. Результат данного теста продемонстрирован на рисунке 4.8.

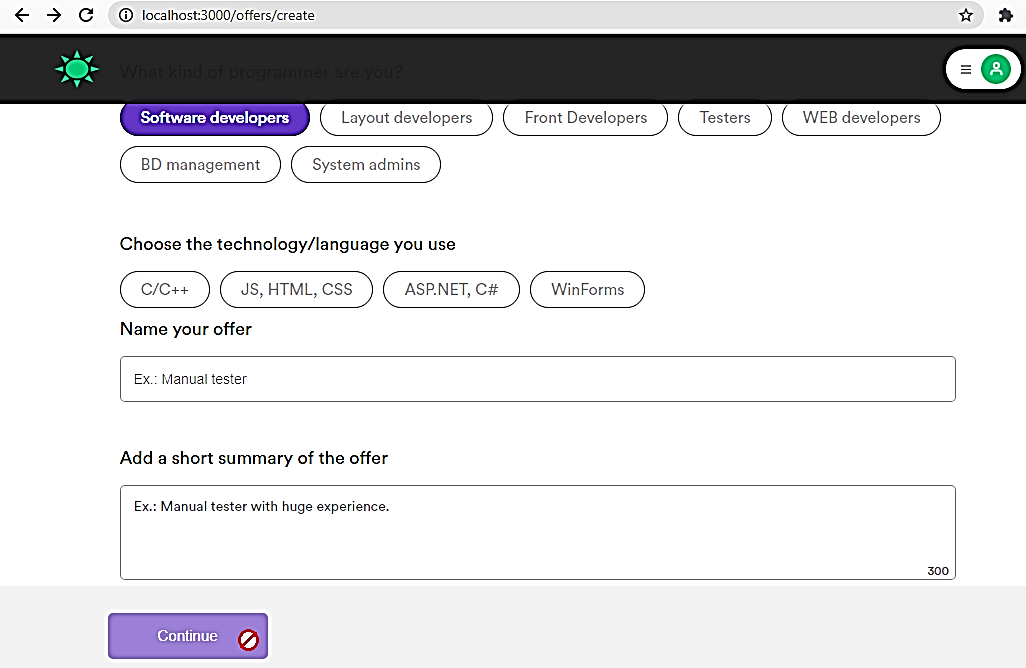


Рисунок 4.8 – Пример тестирования на пустые поля услуги

Одним из шагов создания услуги является выбор возможных дат оказания услуги в календаре. Возможность выбора невалидной даты также исключена за счет неактивности курсора в области дат в прошлом времени. Результат данного теста представлен на рисунке 4.9.

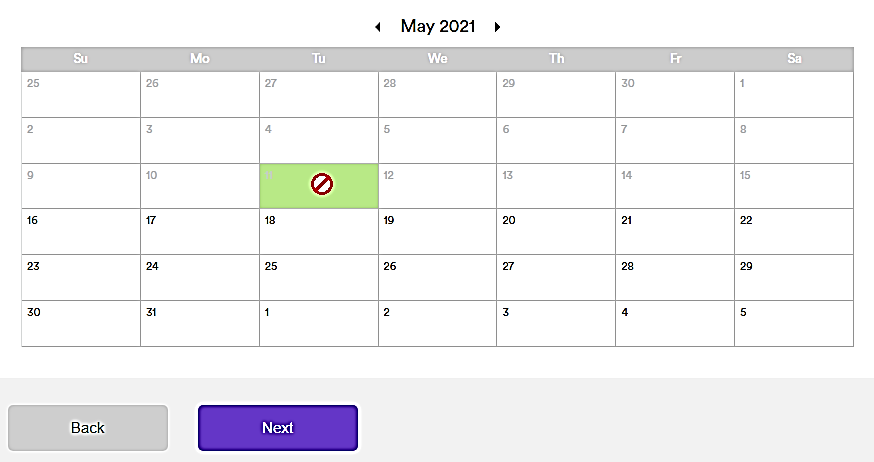


Рисунок 4.9 – Пример тестирования невалидной даты услуги

Также возможна ситуация, когда на шаге ввода цены за услугу программист введет некорректное отрицательное число в поле ввода. Данный случай также предусмотрен – система выведет соответствующее уведомление об ошибке. Результат данного теста представлен на рисунке 4.10.

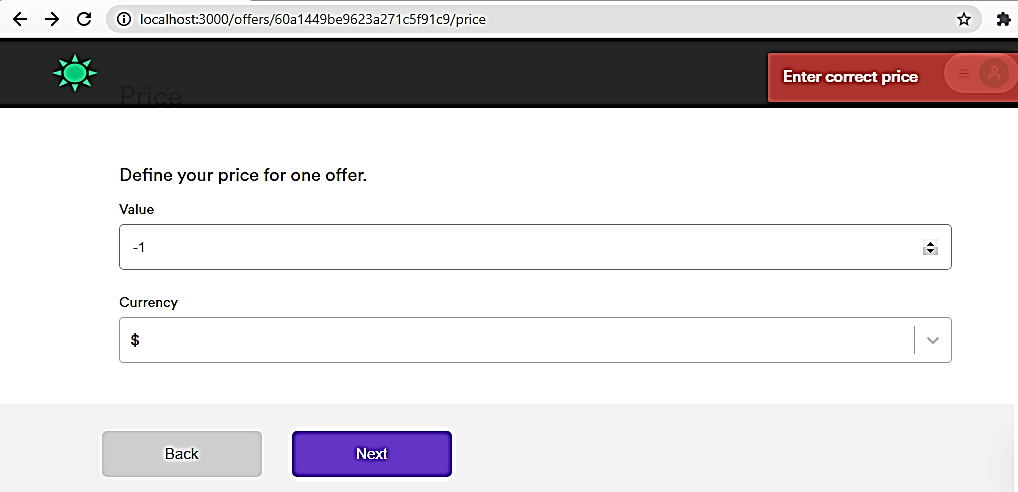


Рисунок 4.10 – Пример тестирования невалидной цены услуги

Аналогичные валидационные правила действуют для формы редактирования уже созданной услуги.

## 4.4 Негативное тестирование импортирования категорий

Импортировать категории имеют право лишь пользователи с ролью «Администратор». Для импорта категорий услуг необходимо нажать кнопку «*programmer.xlsx*» на странице администратора в пункте «*Admin page*» главного меню. Далее необходимо выбрать файл с расширением .*xlsx* для импорта. Выберем файл с другим расширением и попробуем нажать кнопку «*Upload categories*». Результат данного теста представлен на рисунке 4.11.

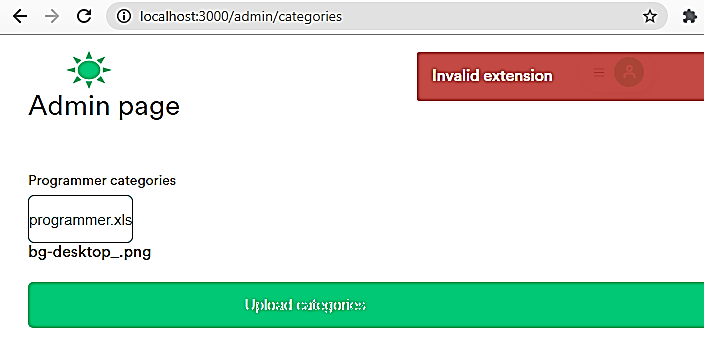


Рисунок 4.11 – Пример тестирования невалидного импорта

Веб-приложение, как и планировалось, предотвратило попытку импортировать невалидный файл и вывело предупреждение «*Invalid extension*».

4 Тестирование

веб-приложения

## 4.5 Вывод по разделу

В данном разделе были рассмотрены примеры негативного тестирования страниц входа и регистрации.

Также были приведены примеры негативного тестирования форм «Создание услуги программиста» и «Импортирование категорий услуг». Данные формы имеет право заполнять только программист и администратор, соответственно.

В ходе работы тестирования были протестированы все формы, которые подразумевают заполнение и ввод данных, и были продемонстрированы все возможные ситуации неверной эксплуатации программного средства пользователем, вследствие чего была минимизирована вероятность появления исключительных ситуация в веб-приложении.

# **5 Руководство пользователя**

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*47*

*БГТУ 05.00.ПЗ*

Разраб*.*

*Чистякова Ю.А.*

Провер*.*

*.*

*Годун А.В.*

Консульт.

*Годун А.В.*

Н. контр.

*Рыжанкова А.С.*

Утв*.*

*Пацей Н.В.*

5 Руководство пользователя

*Лит.*

*Листов*

*15*

74417011, 2021

В данном разделе представлено подробное руководство по использованию веб-приложения для ролей «Гость», «Пользователь», «Программист» и «Администратор».

## 5.1 Роль «Гость»

При запуске веб-приложения неавторизованный пользователь попадает на главную страницу приложения. На главной странице пользователь может посмотреть актуальные предложения программистов, а также воспользоваться строкой быстрого поиска услуг по тексту или дате. На рисунке 5.1 и в приложении В представлен скриншот главной страницы приложения.

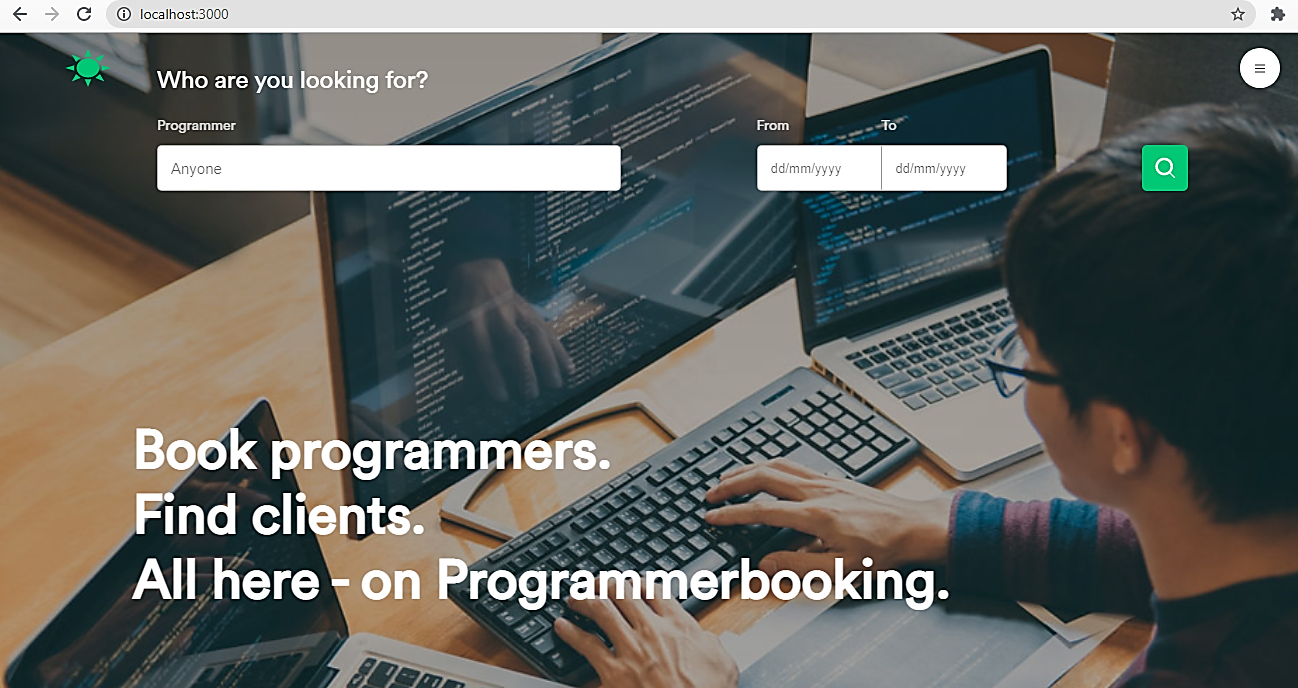


Рисунок 5.1 – Главная страница

Для осуществления бронирования услуг гостю следует авторизоваться. Если пользователь пользуется приложением впервые, ему следует зарегистрироваться.

Для того чтобы зарегистрироваться пользователю следует перейти по пункту меню «*Sign up*». Для успешной регистрации пользователь должен ввести валидный *email* и пароль, а также ввести свои имя, фамилию и дату рождения и нажать на кнопку «*Sign up*».

На рисунке 5.2 представлен скриншот страницы регистрации пользователя.

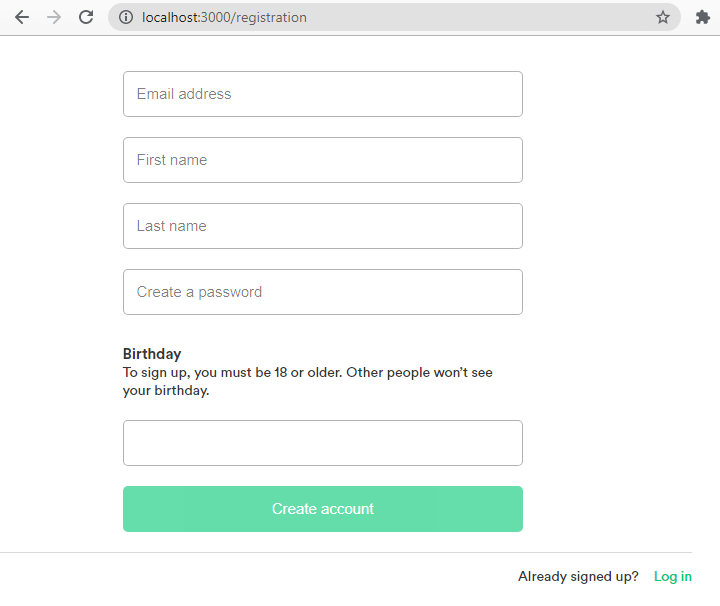


Рисунок 5.2 – Страница регистрации

После заполнения всех необходимых полей валидным данными и нажатия кнопки «*Create account*», пользователю будет показано уведомление «*Check your mail to activate account*», после чего произойдет переадресация на страницу авторизации. Содержимое письма для активации аккаунта представлено на рисунке 5.3.

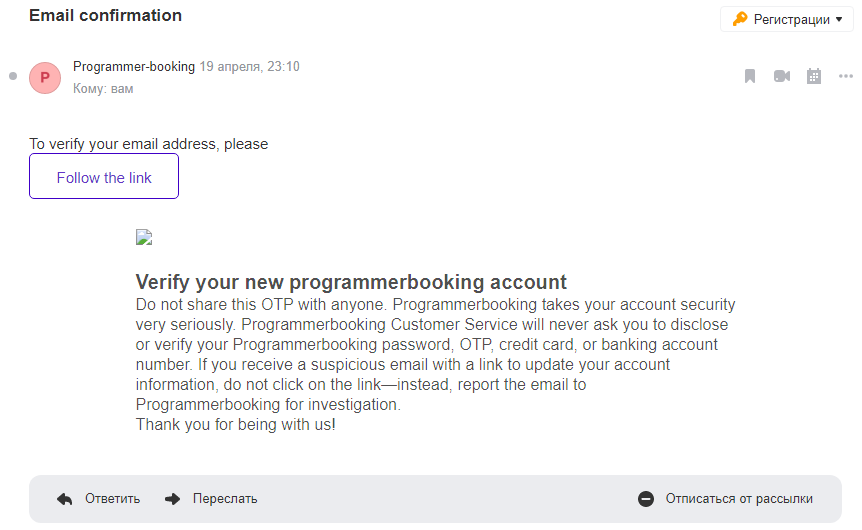


Рисунок 5.3 – Письмо подтверждения аккаунта

В случае, если у пользователя уже есть учетная запись, ему следует нажать на ссылку «*Log in*» внизу страницы регистрации. Пользователю будет предоставлена форма для авторизации, которая представлена на рисунке 5.4.

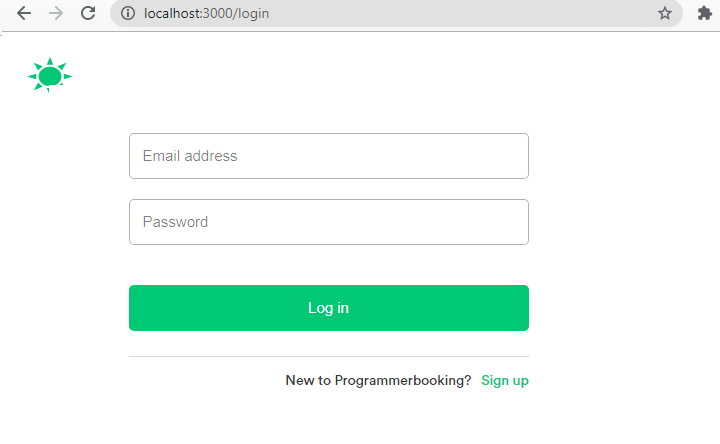


Рисунок 5.4 – Страница авторизации

Далее рассмотрим функциональные возможности для роли «Администратор».

## 5.2 Роль «Администратор»

Роль администратора есть в базе данных. Выдача пользователю прав администратора не предусмотрена.

После успешного прохождения этапа авторизации под учетной записью администратора можно заметить, что пользователю стали доступны дополнительные вкладки в навигационном меню, а именно: *Profile*, *My offers*, *Messages*, *Wishlist*, *Admin page*. Перейдем в пункт меню «*Admin page*». Результат представлен на рисунке 5.5.

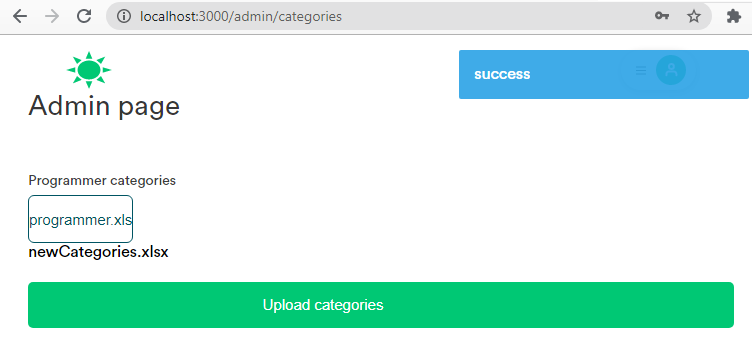


Рисунок 5.5 – Страница администратора

На данной странице администратор может добавить новые категории и подкатегории услуг программиста, импортировав соответствующий файл с расширением .*xlsx*. Пример содержимого файла с новыми категориями представлен на рисунке 5.6.

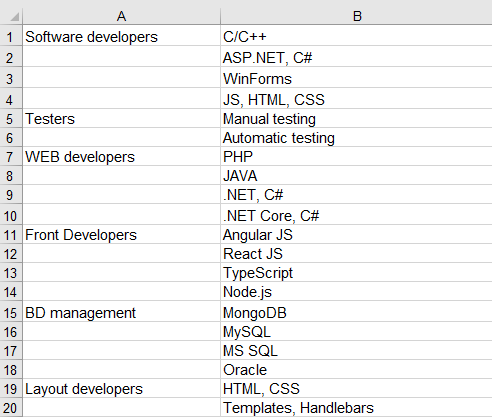


Рисунок 5.6 – Страница администратора

В данном файле в столбце *А* указаны названия категорий, а в столбце *В* – соответствующие им подкатегории. При импорте новых категорий администратором удаление старых не предусмотрено.

## 5.3 Роль «Пользователь»

Далее рассмотрим каждый пункт меню пользователя более подробно.

На рисунке 5.7 представлена страница редактирования профиля пользователя. На данной странице пользователь может изменить имя, фамилию, пол или дату рождения и подтвердить внесенные изменения нажатием кнопки «*Save*». После этого система оповестит об успешном сохранении изменений уведомлением в верхнем правом углу экрана.

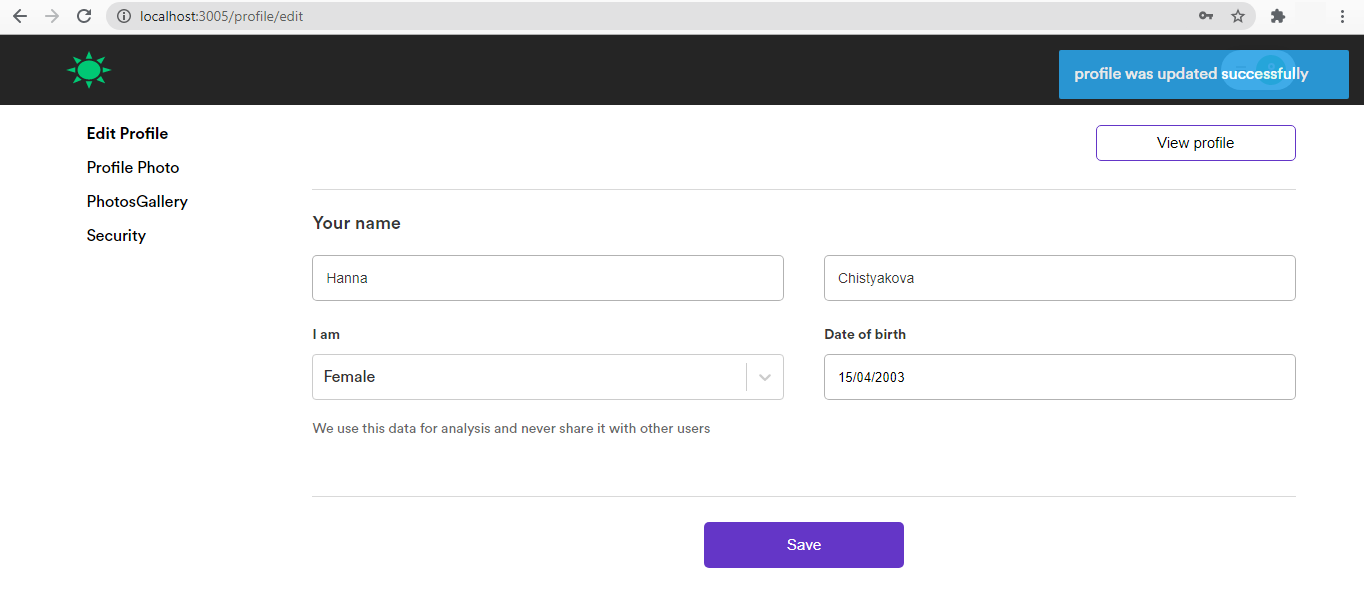


Рисунок 5.7 – Страница редактирования профиля

На рисунке 5.8 изображен фрагмент страницы после перехода по пункту меню «*Profile Photo*». На данной странице пользователь может загрузить новую фотографию профиля или удалить существующую, а также отредактировать миниатюру.

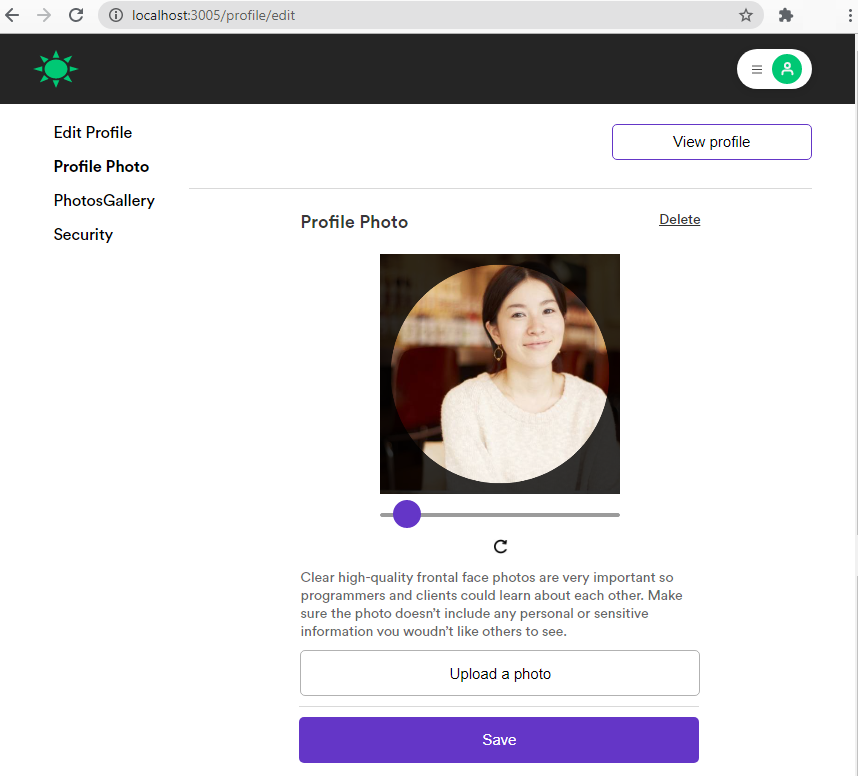


Рисунок 5.8 – Страница редактирования фото профиля

При переходе по пункту меню «*Photos Gallery*» результат будет схожий. Единственная разница только в том, что на вкладке «*Profile Photo*» пользователь загружает фотографию профиля, а в «*Photos Gallery*» – фотографию обложки.

Нажав на пункт меню «*Security*», пользователь попадает на страницу изменения пароля, представленную на рисунке 5.9. Здесь пользователь может ввести новый пароль и нажать на кнопку «*Update Password*», после чего система оповестит пользователя об успешном изменении соответствующим уведомлением.

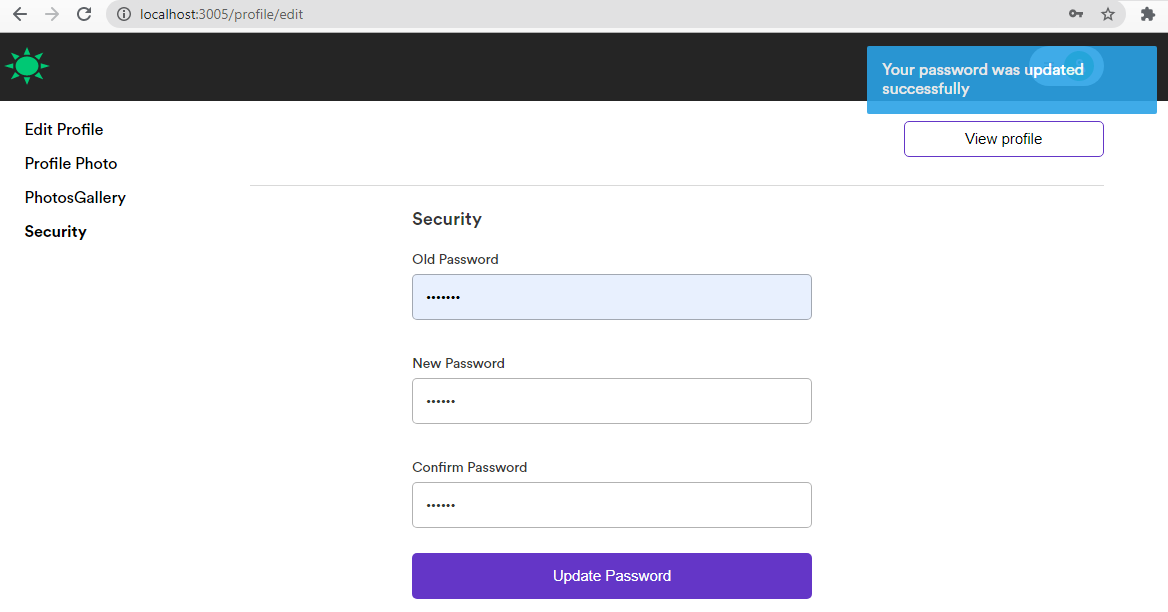


Рисунок 5.9 – Страница изменения пароля

Также на данной странице доступна кнопка «*View profile*», позволяющая посмотерть профиль текущего пользователя. Результат представлен на рисунке 5.10.

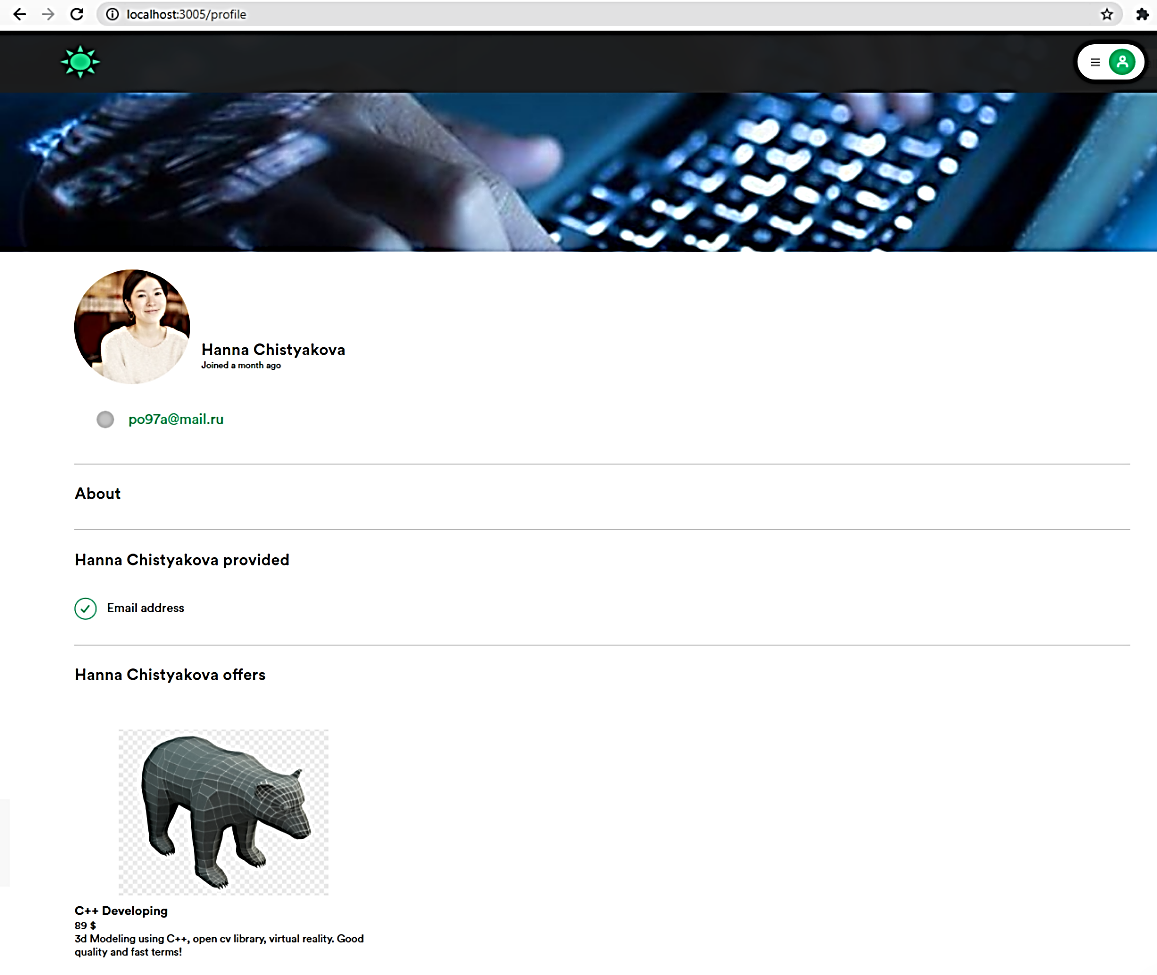


Рисунок 5.10 – Страница просмотра профиля

На данной странице можно увидеть описание профиля, фотографии, подтверждение учетной записи, а также все предложения профиля, если ему доступна роль «Программист». Далее рассмотрим возможности роли «Программист».

## 5.4 Роль «Программист»

Роль программиста присваивается каждому новому пользователю по умолчанию. Помимо всех возможностей роли «Пользователь», роли «Программист» также присваивается право создавать предложения услуг. Главная страница программиста выглядит аналогично, как и главная страница пользователя, представленная на рисунке 5.1.

Если у текущего программиста отсутствуют предложения услуг, будет выведен текст «*You have no offers yet*», а также кнопка «*Create offer*» для создания новой услуги. Однако если программист создавал услуги ранее, они будут выведены на страницу.

На рисунке 5.11 представлен результат перехода на страницу «*My offers*» после нажатия на соответствующий пункт меню.

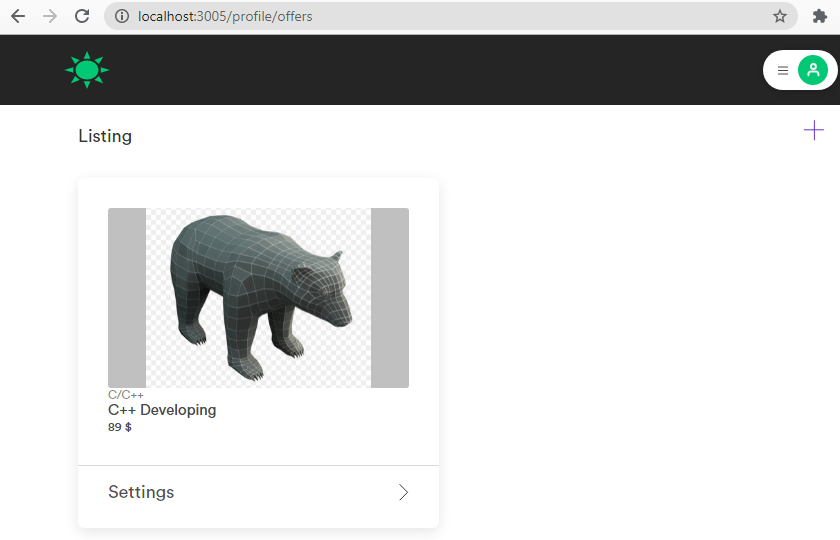


Рисунок 5.11 – Страница предложений программиста

Далее рассмотрим подробнее процесс создания предложения. Для этого на странице предложений нажмем на кнопку «+». Результат представлен на рисунке 5.12.

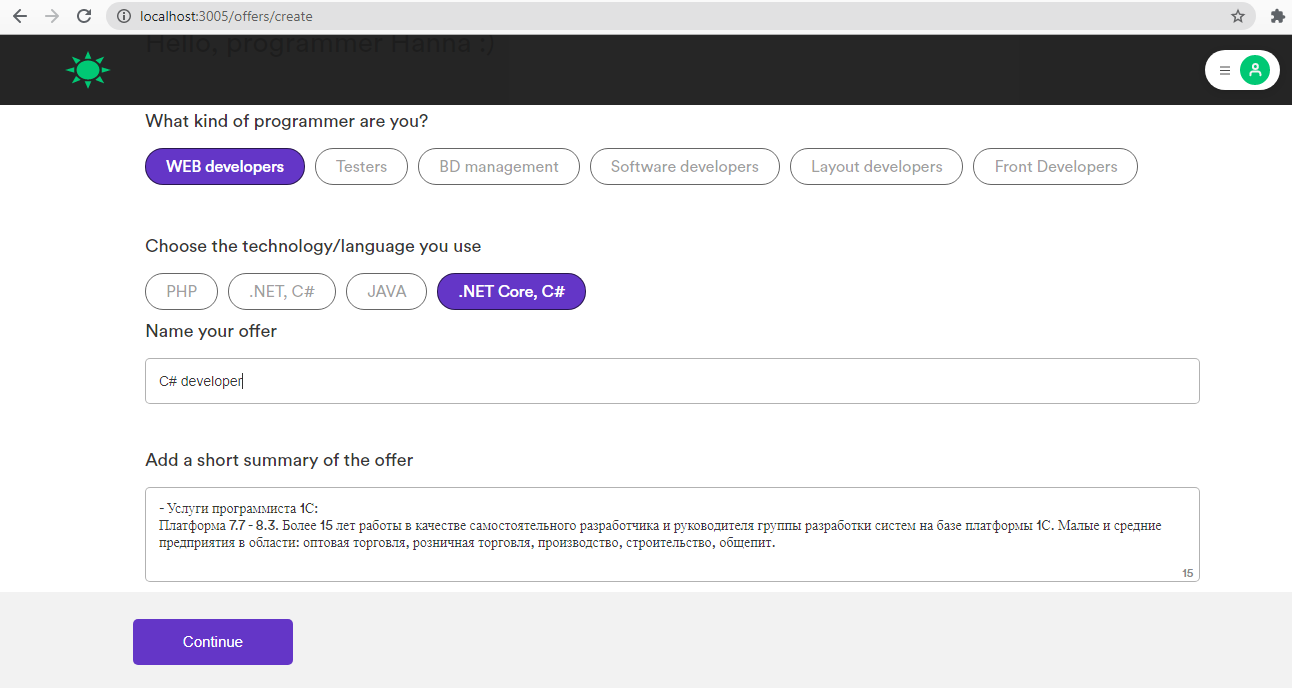


Рисунок 5.12 – Страница создания предложения

На данной странице программист выбирает категорию и подкатегорию своей услуги из добавленных ранее администратором. После заполнения названия и описания услуги становится активной кнопка «*Continue*». После нажатия на данную кнопку происходит переход на страницу текущего прогресса, фрагмент которой представлен на рисунке 5.13.

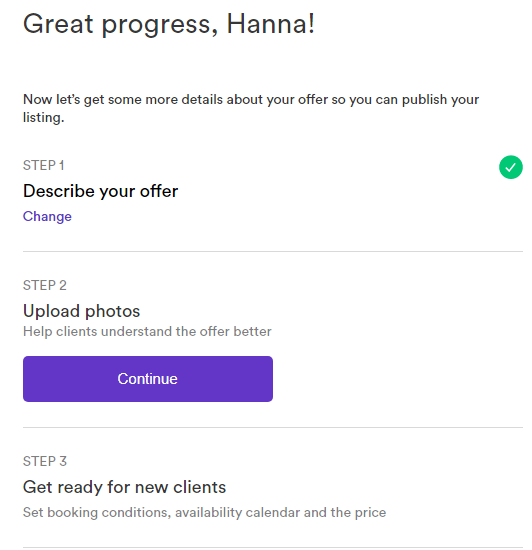


Рисунок 5.13 – Страница прогресса создания

На данной странице представлен текущий прогресс создания услуги. При нажатии на кнопку «*Continue*» пользователь переходит ко второму шагу – добавлению фотографий услуги. Фрагмент страницы представлен на рисунке 5.14.

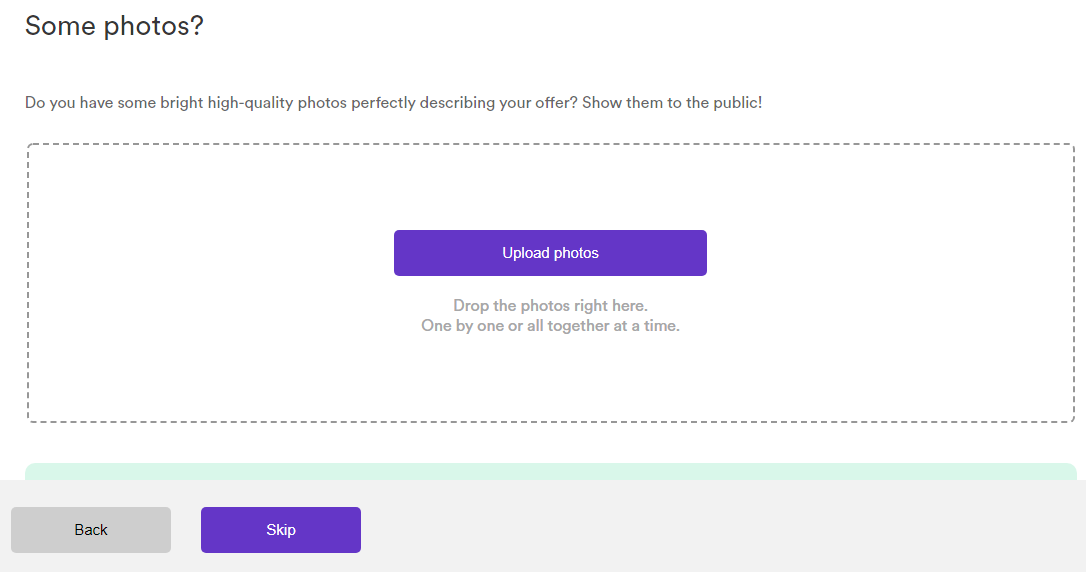


Рисунок 5.14 – Страница добавления фотографий услуги

При нажатии кнопки «*Skip*» пользователь автоматически перейдет к третьему шагу создания услуги, и текущая услуга останется без фотографий. При выборе фотографий и нажатии кнопки «*Next*» пользователь будет переадресован на страницу добавления видео услуги, фрагмент которой представлен на рисунке 5.15.

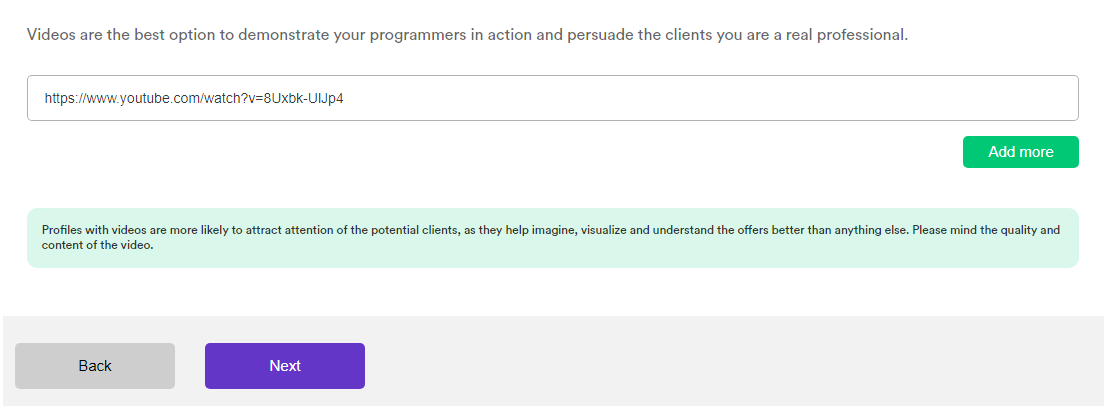


Рисунок 5.15 – Страница добавления видео услуги

На данной странице программист может добавить ссылки на видео услуги. При нажатии на кнопку «*Next*», пользователь вновь попадет на странице текущего прогресса, которая представлена на рисунке 5.13, после чего перейдет к последнему шагу создания услуги – указанию дополнительной информации. Результат перехода на данную страницу представлен на рисунке 5.16.

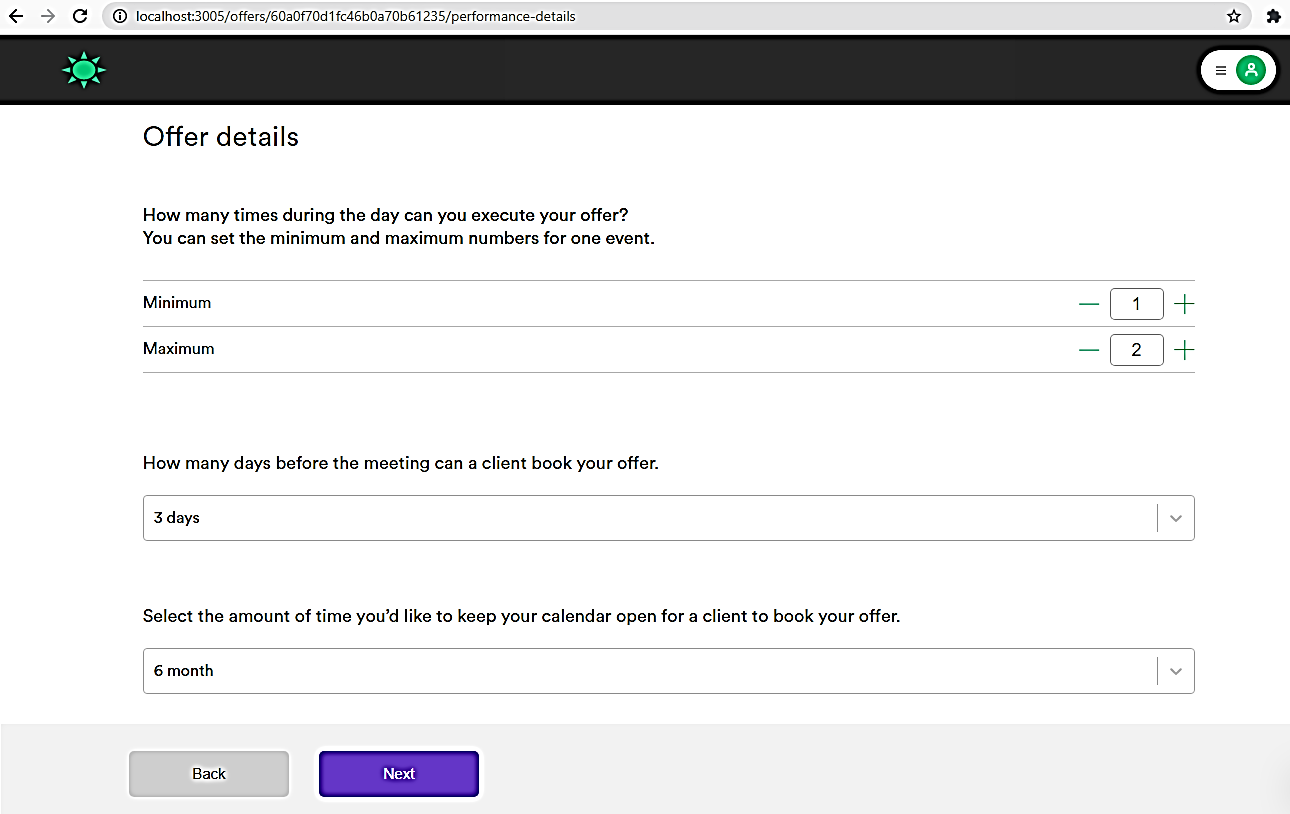


Рисунок 5.16 – Страница заполнения подробностей услуги

На данной странице программист заполняет максимальное количество раз выполнения услуги в день, количество дней между бронированием и выполнением услуги, а также срок, в течении которого текущая услуга остается активной.

После заполнения всех полей и нажатия кнопки «*Next*» происходит переход на страницу выбора дней услуги, фрагмент которой представлен на рисунке 5.17.

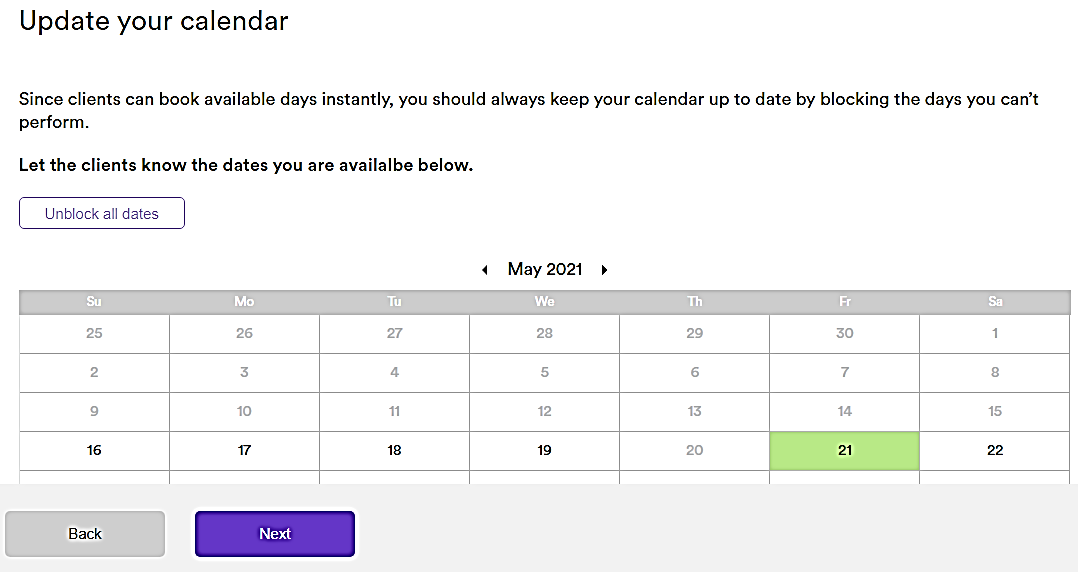


Рисунок 5.17 – Выбор дней услуги в календаре

На данной странице пользователь выбирает возможные дни услуги. После нажатия кнопки «*Next*» происходит переход на страницу ввода цена и валюты за услугу. После ввода цены и нажатия кнопки «*Publish offer now*», предложение об услуге публикуется на сайте, о чем пользователя информирует уведомление «*Your offer was published*». Теперь пользователь может изменить или удалить услугу на странице «*My offers*» нажатием соответствующих кнопок.

Далее происходит переадресация на страницу созданного предложения услуги, которая представлена на рисунке 5.18.

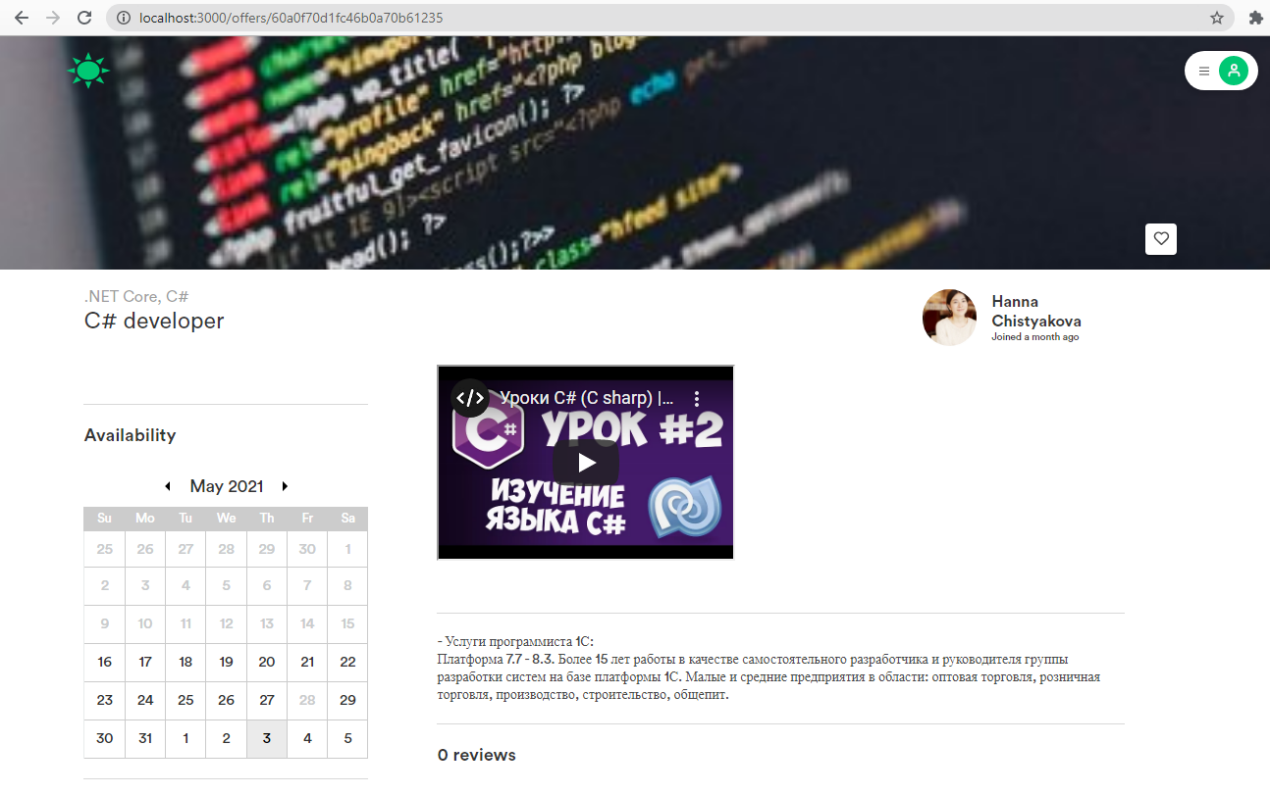


Рисунок 5.18 – Страница предложения услуги

На данной странице пользователь может ознакомиться с информацией об услуге, возможных днях бронирования, профилем программиста, а также отзывами других пользователей о данной услуге. Также при нажатии на кнопку «сердце», данная услуга добавляется в избранное.

Для того, чтобы посмотреть добавленные в избранное услуги, необходимо нажать на соответствующий пункт «*Wishlist*» в главном меню. Страница с избранными объявлениями представлена на рисунке 5.19.

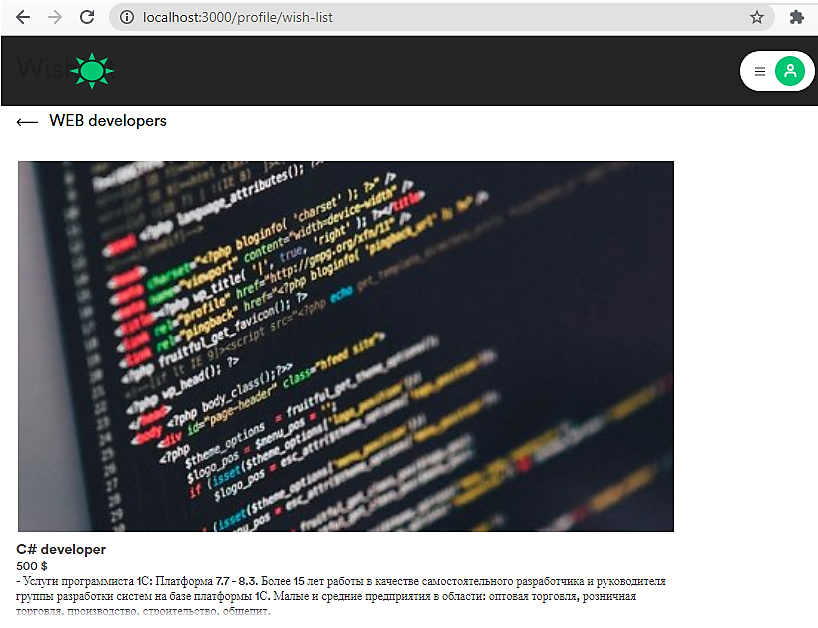


Рисунок 5.19 – Страница избранных услуг

Все избранные услуги отсортированы по соответствующим категориям технологий, выбранным при создании услуги. Таким образом, при переходе в пункт «*Wishlist*» главного меню пользователю представлен список категорий, услуги которых были добавлены в избранное. При нажатии на категорию, раскрывается вложенный список конкретных объявлений услуг данной категории, которые пользователь добавил в избранное. Для того, чтобы удалить услугу программиста из избранного, необходимо вновь перейти на страницу услуги и повторно нажать на кнопку «Сердце».

Далее рассмотрим подробнее процесс бронирования услуги. Для этого вновь авторизируемся в учетной записи пользователя и перейдем на страницу услуги по ссылке на главной странице, смотри рисунок 5.1. На данной странице пользователь может выбрать нужные даты начала и конца бронирования данной услуги.

Результат перехода на страницу услуги представлен на рисунке 5.20.

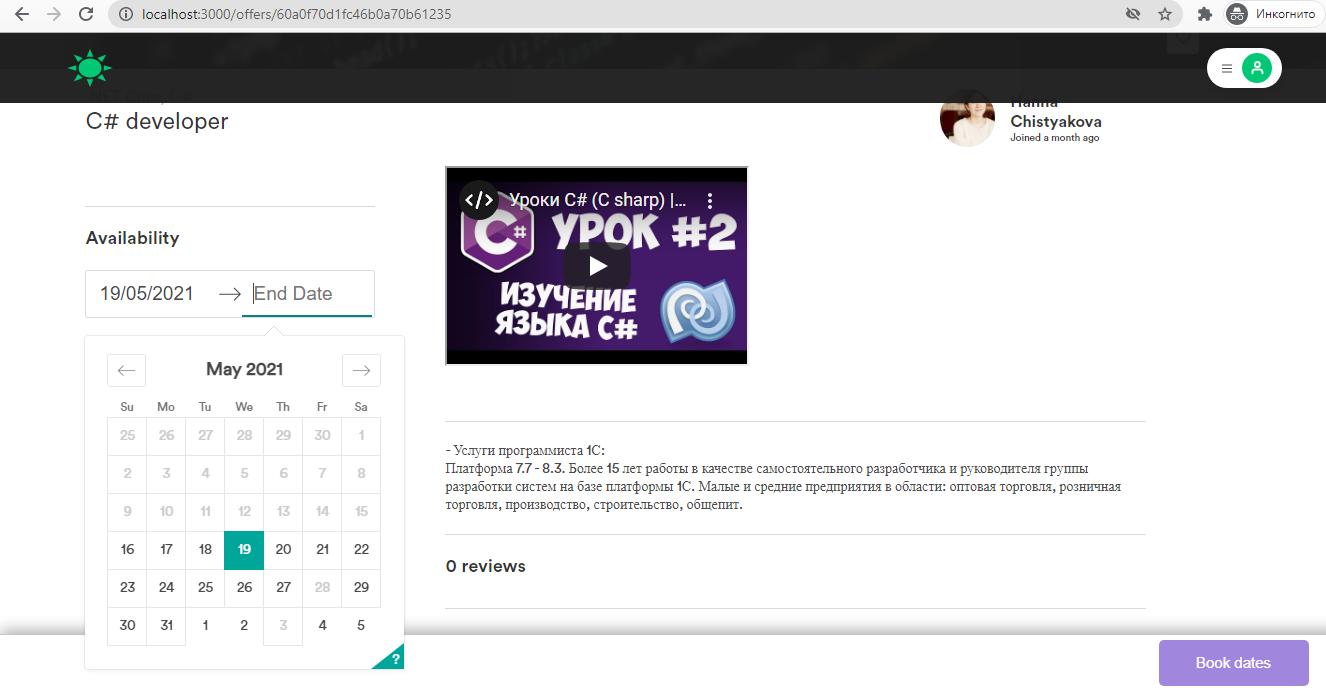


Рисунок 5.20 – Страница услуги

Следует обратить внимание, что при выборе дат подсвечены лишь те возможные даты бронирования, которые программист указал при создании услуги. После нажатии на кнопку «*Book dates*» пользователь будет перенаправлен на страницу заполнения подробностей бронирования, которая представлена на рисунке 5.21.

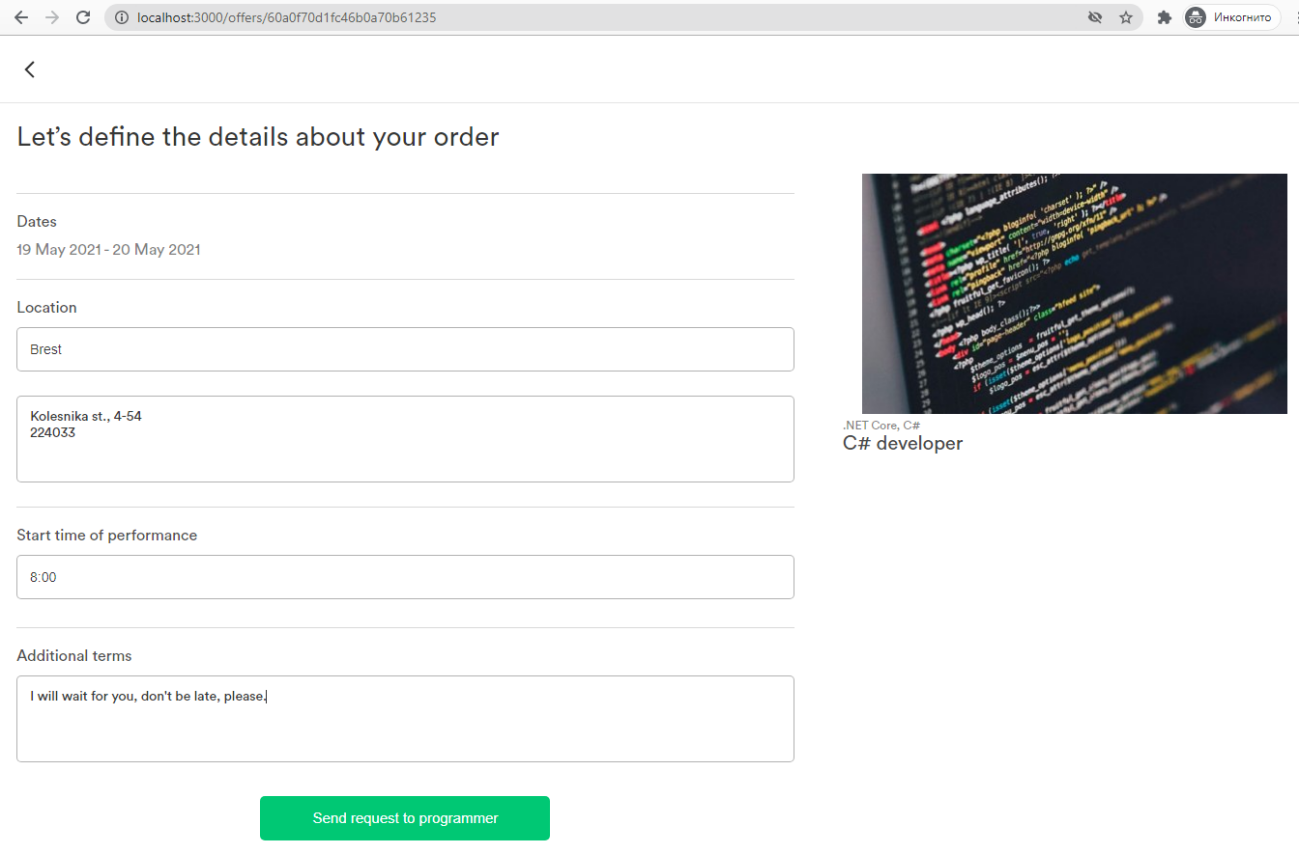


Рисунок 5.21 – Страница бронирования услуги

На данной странице пользователь может указать точный адрес для выполнения услуги, а также время начала и дополнительную информацию. При нажатии на кнопку «*Send request to programmer*», пользователь будет перенаправлен на страницу чата с программистом. Страница чата продемонстрирована на рисунке 5.22.

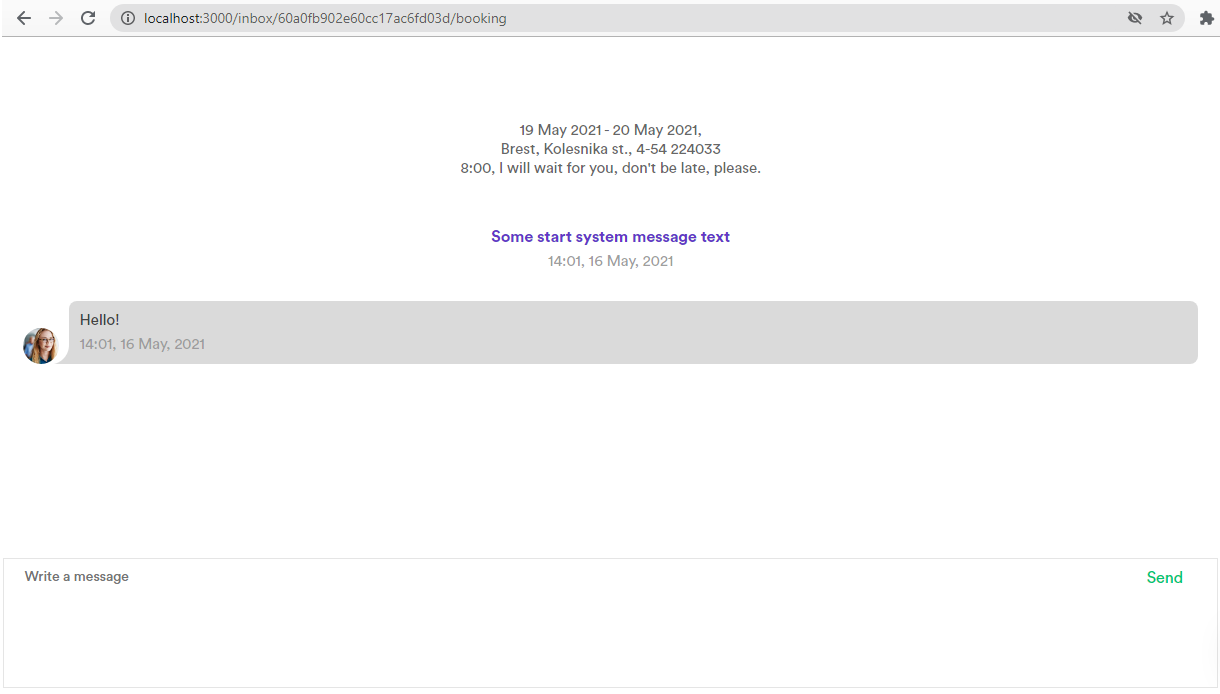


Рисунок 5.22 – Страница чата

На данной странице пользователь может написать сообщение в чат, а также посмотреть текущий статус заказа услуги.

Вернемся к учетной записи программиста. В главном меню будет высвечено напоминание о новом сообщении, которое представлено на рисунке 5.23.

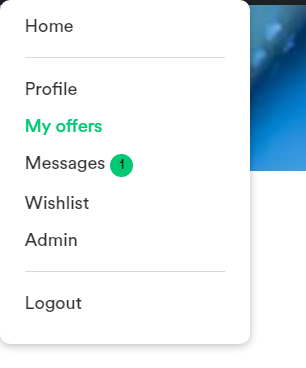


Рисунок 5.23 – Новое сообщение в чате

На странице чатов в пункте меню «*Messages*», программист может посмотреть чаты с клиентами, а также текущий статус бронирования услуги для каждого чата. Поскольку, программист еще не подтвердил и не отклонил бронирование услуги, текущий заказ будет иметь статус «*Respond In*». Далее, когда программист подтвредит услугу она будет иметь статус «*Approved*», иначе «*Declined*». Страница с чатами пользователя представлена на рисунке 5.24.

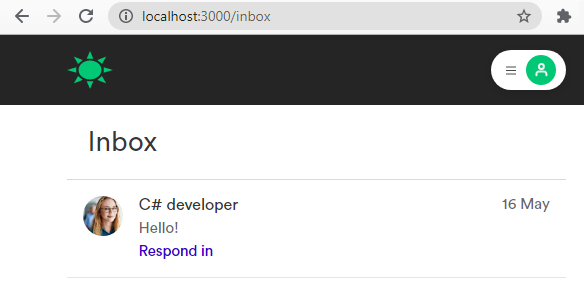


Рисунок 5.24 – Страница чатов

Перейдем на страницу рассматриваемого чата. Результат представлен на рисунке 5.25. На данной странице программист может ответить на задаваемые вопросы пользователя, а также принять или отклонить бронирование услуги нажатием кнопки «*Approve or Decline*». После нажатия программист будет направлен на страницу, где, в зависимости от ответа, он может нажать «*Approve*» или «*Decline*». Заказ при этом примет статус *approved* или *declined*, соответственно.

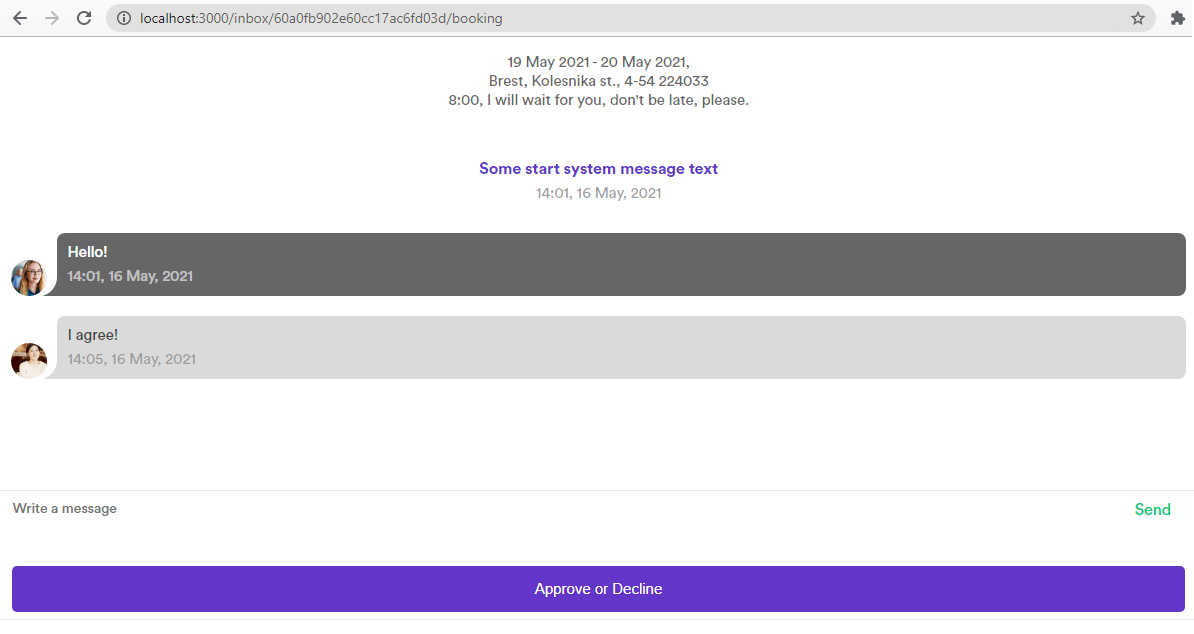


Рисунок 5.25 – Страница чата

В случае подтверждения программистом бронирования на выполнение услуги, в учетной записи клиента на странице чата появится кнопка «*Leave review*», которая перенаправит пользователя на страницу для создания отзыва.

Страница чата в учетной записи клиента после подтверждения бронирования услуги представлена на рисунке 5.26.

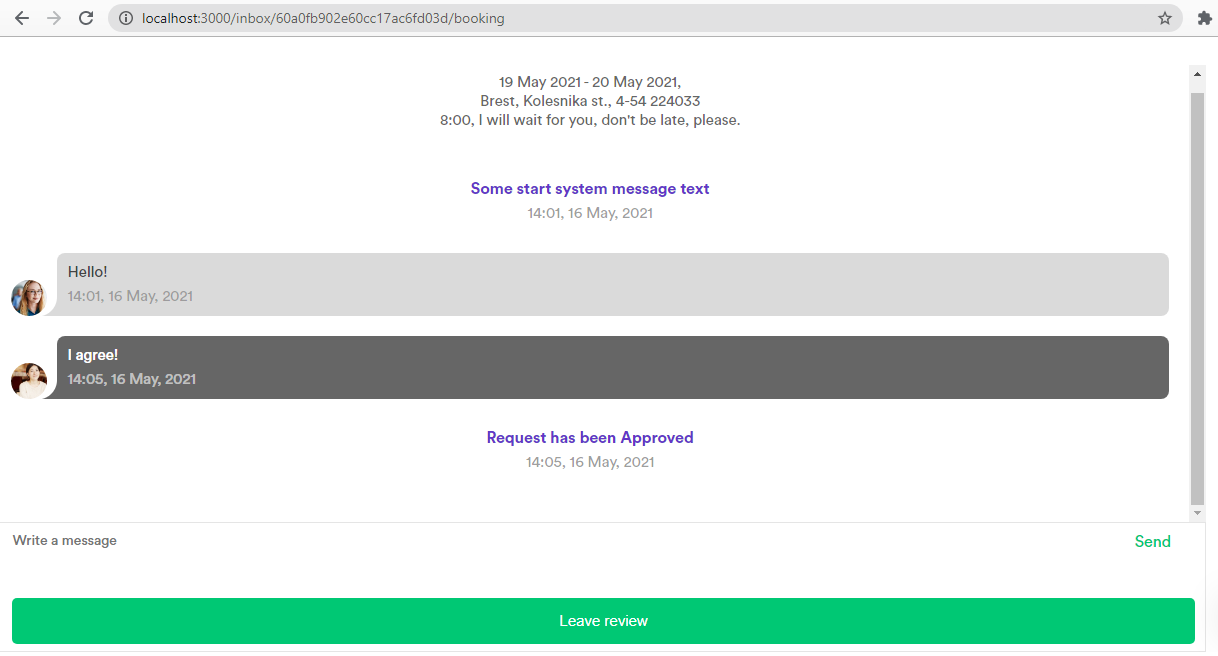


Рисунок 5.26 – Страница чата после подтверждения

После того, как пользователь заполнит поле на странице создания отзыва и нажмет кнопку «*Send review*», система уведомит об успешном отправлении отзыва уведомлением «*Review was sent successfully*». После этого отзыв будет добавлен на страницу услуги.

Таким образом, мы рассмотрели все функциональные возможности данного программного средства.

## 5.5 Вывод по разделу

В данном разделе представлена основная информация о работе с веб-приложением. Так как интерфейс приложения достаточно простой, у пользователя не должно возникнуть проблем с эксплуатацией приложения.

Данное веб-приложение позволяет пользователям редактировать личный кабинет, бронировать нужные им услуги программиста, вести чат с программистом и оставлять отзывы на оказанные им услуги.

Для программистов есть возможность создания новых услуг, подтверждения или отклонения бронирования услуг, а также ведение чата с клиентами.

Для администраторов есть возможность добавления новых категорий услуг.

В следующем разделе будет рассмотрена экономическая целесообразность разработки данного веб-приложения.

# **6 Технико-экономическое обоснование проекта**

ФИО.

Подпись

Дата

Лист

1

БГТУ 06.00.ПЗ

Разраб.

*Чистякова Ю.А.*

Провер.

*Годун А.В.*

Консульт.

*Евлаш А.И.*

Н. котр.

*Рыжанкова А.С.*

Утв.

Пацей Н.В.

6 Технико-экономическое обоснование проекта

Лит.

Листов

8

74417011, 2021

## 6.1 Общая характеристика разрабатываемого программного средства

Основной целью экономического раздела является экономическое обоснование целесообразности разработки веб-приложения, представленного в дипломном проекте. В данном разделе проводится расчет затрат на всех стадиях разработки, а также анализ экономического эффекта в связи с использованием данного веб-приложения.

Разработанный программный продукт позволяет пользователям найти подходящую для них услугу программиста и осуществить ее предварительное бронирование.

Во время разработки дипломного проекта использовалась технология *Node.js* для написания серверной части приложения и библиотекой *React* для написания клиентской части приложения. Данное веб-приложение разработано для последующего использования в коммерческих целях.

## 6.2 Исходные данные для проведения расчетов

Источниками исходных данных для данных расчетов выступают действующие законы и нормативно-правовые акты. Исходные данные для расчета стоимости программного продукта представлены в таблице 6.1.

Таблица 6.1 — Исходные данные для расчета

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование показателя | Единица измерения | Условные обозначения | Норматив |
| Численность разработчиков | чел. | Чр | 1 |
| Норматив дополнительной заработной платы | % | Ндз | 12 |
| Ставка отчислений в Фонд социальной защиты населения | % | Нфсзн | 34 |
| Ставка отчислений в БРУСП «Белгосстрах» | % | Нбгс | 0,4 |
| Цена одного машино-часа | руб. | Смч | 0,06 |
| Норматив прочих затрат | % | Нпз | 10,5 |
| Норматив накладных расходов | % | Нобп*,* обх | 122 |
| Норматив расходов на сопровождение и адаптацию | % | Нрса | 10 |
| Ставка НДС | % | Нндс | 20 |

В ходе проведения маркетингового анализа была выявлена стоимость разработки программного продукта по аренде услуг программиста. Цены разработки аналогичных программных средств представлены в таблице 6.2.

Таблица 6.2 — Маркетинговый анализ аналогов

|  |  |
| --- | --- |
| Название программного средства | Стоимость разработки, руб. |
| *TrueTech.by* | 28 000 |
| *RedSale.by* | 30 000 |
| *FreelanceHunt.by* | 29 500 |

Таким образом, для получения рентабельного продукта стоимость разработки должна быть менее, чем стоимости, приведенные в таблице выше и должна составлять максимум 27 500 руб.

## 6.3 Методика обоснования цены

В современных рыночных экономических условиях веб-приложение выступает преимущественно в виде продукции организаций, представляющей собой функционально завершенные и имеющие товарный вид веб-приложения, реализуемые покупателям по рыночным отпускным ценам. Все завершенные разработки веб-приложения являются научно-технической продукцией.

Широкое применение вычислительных технологий требует постоянного обновления и совершенствования веб-приложения. Выбор эффективных проектов веб-приложения связан с их экономической оценкой и расчетом экономического эффекта, который может определяться как у разработчика, так и у пользователя.

У разработчика экономический эффект выступает в виде чистой прибыли от реализации приложения, остающейся в распоряжении организации, а у пользо-вателя – в виде экономии трудовых, финансовых ресурсов, получаемой за счет:

* снижения трудоемкости расчетов и алгоритмизации программирования и отладки программ;
* снижения расходов на материалы;
* ускорение ввода в эксплуатацию новых систем;
* улучшения показателей основной деятельности в результате использования веб-приложения.

Стоимостная оценка веб-приложения у разработчиков предполагает определение затрат, что включает следующие статьи:

* заработная плата исполнителей – основная и дополнительная;
* отчисления в фонд социальной защиты населения;
* отчисления по обязательному страхованию от несчастных случаев на производстве и профессиональных заболеваний;
* расходы на оплату машинного времени;
* накладные расходы;
* прочие прямые затраты.

На основании затрат рассчитывается себестоимость и отпускная цена конечного веб-приложения.

### 6.3.1 Объем веб-приложения

Для оценки объема веб-приложения, все его функции классифицируются с использованием специального каталога функций, который определяет их объем. Общий объем веб-приложения *Vo*, вычисляется как сумма объемов *Vi* каждой из n его функций. Ниже приведена формула 6.1 для вычисления.

|  |  |  |
| --- | --- | --- |
|  |  | (6.1) |

где *Vi* – объем *i*-ой функции ПС, условных машинных команд;

*n* – общее число функций.

В таблице 6.3 представлены функции, присутствующие в рассматриваемом программном средстве и соответствующий им объем в условных машино-командах.

Таблица 6.3 – Содержание и объем функций в программном средстве

|  |  |  |
| --- | --- | --- |
| Номер функции | Содержание функции | Объем, условных машино-команд |
| 101 | Организация ввода информации | 230 |
| 102 | Контроль, предварительная обработка | 450 |
| 111 | Управление вводом/выводом | 2000 |
| 203 | Формирование базы данных | 2100 |
| 506 | Обработка ошибочных и сбойных ситуаций | 410 |
| 507 | Обеспечение интерфейса между компонентами | 970 |
| 707 | Графический вывод результатов | 550 |

Опираясь на данные таблицы 6.3, можно определить объем условных машино-команд веб-приложения, разработанного в ходе дипломного проектирования:

*V*o = 230 + 450 + 2000 + 2100 + 410 + 970 + 550 = 6710 (маш. команд).

Уточненный объем веб-приложения *V*o/ равен произведению объема веб-приложения *V*o на коэффициент изменения скорости обработки информации Кск согласно формуле 6.2

|  |  |  |
| --- | --- | --- |
|  |  | (6.2) |

Исходя из вычисленного объема веб-приложения, можно определить уточненный объем веб-приложения:

*V*o/ = 6710 ⋅ 0,6 = 4026.

Установили, что уточненный объем веб-приложения равен 4026 (условных машино-команд).

### 6.3.2 Основная заработная плата

Для определения величины основной заработной платы, было проведено исследование величин заработных плат для специалистов в сфере веб-программирования на *JavaScript*. Источником данных служили открытые веб-порталы, различные форумы, официальная отчетность и общий средний уровень заработке в сфере информационных технологий. В итоге было установлено, что средняя месячная заработная плата на позиции *junior*/*middle* составляет 1600 рублей.

Проект разрабатывался одним человеком на протяжении трех месяцев. Таким образом, основная заработная плата будет рассчитываться по формуле 6.3

*С*оз = Траз · Краз · Сзп, (6.3)

где Соз – основная заработная плата, руб.;

Траз – время раработки, месяцев;

Краз – количество разработчиков, человек;

Сзп – средняя месячная заработная плата, руб.

Соз = 3 · 1 · 1600 = 4800 руб.

### 6.3.3 Дополнительная заработная плата

Дополнительная заработная плата на конкретное программное средство включает выплаты, предусмотренные законодательством о труде, и определяется по нормативу в процентах к основной заработной плате по формуле 6.4

|  |  |  |
| --- | --- | --- |
|  |  | (6.4) |

где Соз– основная заработная плата, руб.;

Ндз – норматив дополнительной заработной платы, %.

Cдз = 4800 ⋅ 12 / 100 = 576 руб.

### 6.3.4 Отчисления в Фонд социальной защиты населения и Белгосстрах

Отчисления в Фонд социальной защиты населения и Белгосстрах (ФСЗН) определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей и вычисляются по формуле 6.5:

|  |  |  |
| --- | --- | --- |
|  |  | (6.5) |

где Соз – основная заработная плата, руб.;

Сдз – дополнительная заработная плата за веб-приложение, млн. руб.;

Нфсзн– норматив отчислений в Фонд социальной защиты населения, %.

Отчисления в БРУСП «Белгосстрах» вычисляются по формуле 6.6

|  |  |
| --- | --- |
|  | (6.6) |

руб.

руб.

Таким образом, общие отчисления в «Белгосстрах» составили 21,5 руб., а в фонд социальной защиты населения – 1827,84 руб.

### 6.3.5 Расходы на материалы

Сумма расходов на материалы СМ определяется как произведение нормы расхода материалов в расчете на сто строк исходного кода НМ на уточненный объем веб-приложения *V*o/.

|  |  |  |
| --- | --- | --- |
|  |  | (6.7) |

где См – сумма расходов на материалы, руб.;

Нм – норма расхода материалов в расчете на 100 строк исходного кода веб-приложения, руб.;

*V*o/ – уточненный объем веб-приложения, условных машино-команд.

Учитывая, что норма расхода материалов в расчете на сто строк исходного кода равна 0,46 руб (по данным, приведенным в приложении 2 таблице П 2.10 «Оценка значений среднего расхода материалов на разработку и отладку 100 строк кода применения веб-приложения» методического пособия [22]), можно определить сумму расходов на материалы:

СМ = 0,46 ⋅ 4026 / 100 = 18,52 руб.

### 6.3.6 Расходы на оплату машинного времени

Сумма расходов на оплату машинного времени Смв определяется как произведение стоимости одного машино-часа Смч на уточненный объем веб-приложения *Vo/* и на норматив расхода машинного времени на отладку ста строк исходного кода НМВ согласно формуле 6.8

|  |  |  |
| --- | --- | --- |
|  | . | (6.8) |

Учитывая, что норматив машинного времени на отладку ста строк исходного кода равен 15 (по данным, приведенным в приложении 2 таблице П 2.11 «Оценка значений среднего машинного времени на отладку 100 строк исходного кода без применения ПС» методического пособия [22]), можно определить сумму расходов на оплату машинного времени.

Смв = 0,07 ⋅ 6018 ⋅ 15 / 100 = 63,19 руб.

### 6.3.7 Прочие прямые затраты

Сумма прочих затрат Спз определяется как произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив прочих затрат в целом по организации Нпз согласно формуле 6.9

|  |  |  |
| --- | --- | --- |
|  |  | (6.9) |

Спз = 4800 ⋅ 10,5 / 100 = 504 руб.

### 6.3.8 Накладные расходы

Сумма накладных расходов Cобп, обх – произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив накладных расходов в целом по организации Нобп, обх согласно формуле 6.10

|  |  |  |
| --- | --- | --- |
|  |  | (6.10) |

Все данные необходимые для вычисления есть, поэтому можно определить сумму накладных расходов:

Собп, обх = 4800 ⋅ 122 / 100 = 5856 руб.

### 6.3.9 Сумма расходов на разработку веб-приложения

Сумма расходов на разработку веб-приложения Ср определяется как сумма основной и дополнительной заработных плат исполнителей на конкретное программное средство, отчислений на социальные нужды, расходов на материалы, расходов на оплату машинного времени, суммы прочих затрат и суммы накладных расходов согласно формуле 6.11

|  |  |  |
| --- | --- | --- |
|  | Ср = Соз + Сдз + Сфсзн + Сбгс + См + Смв + Спз + Собп, обх | (6.11) |

Ср = 4800 + 576 +1827,84 + 21,5 + 18,52 + 36,23 + 504 + 5856 = 13 640,09 руб.

Сумма расходов на разработку веб-приложения была вычислена на основе данных, рассчитанных ранее в данном разделе. Таким образом, сумма расходов на разработку веб-приложения составила 13 640,09 рублей.

### 6.3.10 Расходы на сопровождение и адаптацию

Сумма расходов на сопровождение и адаптацию веб-приложения Срса определяется как произведение суммы расходов на разработки на норматив расходов на сопровождение и адаптацию Нрса согласно формуле 6.12

|  |  |  |
| --- | --- | --- |
|  |  | (6.12) |

где Срса – сумма расходов на сопровождение и адаптацию веб-приложения, руб.;

Ср – сумма расходов на разработку веб-приложения, руб.;

Нрса – норматив расходов на сопровождение и адаптацию, %.

Срса = 13 640,09 ⋅ 10 / 100 = 1364,01 руб.

Таким образом, сумма расходов на сопровождение и адаптацию веб-приложения составляют 1364,01 рублей.

### 6.3.11 Полная себестоимость

Полная себестоимость Сп определяется как сумма двух элементов: суммы расходов на разработку Ср и суммы расходов на сопровождение и адаптацию веб-приложения Срса согласно формуле 6.13

|  |  |  |
| --- | --- | --- |
|  |  | (6.13) |

где Сп – полная себестоимость веб-приложения, руб.;

Ср – сумма расходов на разработку веб-приложения, руб.;

Срса – сумма расходов на сопровождение и адаптацию веб-приложения, руб.

Сп = 13 640,09 + 1364,01 = 15 004,1 руб.

Полная себестоимость составила = 15 004,1 рублей.

### 6.3.12 Определение цены, оценка эффективности

При расчете прибыли будем опираться на среднюю рыночную цену, по которой можно разработать данное программное обеспечение. Средняя рыночная цена составляет 27 500 руб. Прибыль от реализации веб-приложения рассчитывается по формуле 6.14

|  |  |  |
| --- | --- | --- |
|  | , | (6.14) |

где Ппс – прибыль от реализации программного средства, руб.;

Цр – средняя рыночная цена продукта, руб.;

Сп – полная себестоимость программного средства, руб.;

Ппс = (27 500 / 1,2) – 15 004,1 = 7912,57 (руб.).

Уровень рентабельности разработанного программного средства определяется по формуле 6.15

|  |  |  |
| --- | --- | --- |
|  | , | (6.15) |

где Урент – уровень рентабельности программного средства, %

Сп – полная себестоимость программного средства, руб.;

Ппс – прибыль от реализации программного средства, руб.

Урент = (7912,57 / 15 004,1) ⋅ 100 = 52,74 %

Рентабельность разработки программного средства была вычислена на основе данных, рассчитанных ранее в данном разделе.

## 6.4 Вывод по разделу

В таблице 6.4 и приложении Г представлены результаты расчетов для основных показателей данной главы в краткой форме.

Таблица 6.4 – Результаты расчетов

|  |  |
| --- | --- |
| Наименование показателя | Значение |
| Время разработки, мес. | 3 |
| Количество программистов, чел. | 1 |
| Зарплата с отчислениями, руб. | 7225,34 |
| Расходы на материалы, оплату машинного времени, прочие, руб | 558,75 |
| Накладные расходы, руб | 5856 |
| Себестоимость разработки веб-приложения, руб | 13 640,09 |
| Расходы на сопровождение и адаптацию, руб. | 1364,01 |
| Полная себестоимость, руб. | 15 004,1 |
| Цена аналога, руб. | 27 500 |
| Прибыль от реализации, руб. | 7912,57 |
| Рентабельность разработки, % | 52,74 |

Разработанное веб-приложение позволяет подобрать варианты услуг программиста в соответствии с требованиями и финансовыми возможностями клиента.

В результате проведенных вычислений можно сделать вывод о том, что разрабатываемое веб-приложение является экономически целесообразным, так как его полная себестоимость 15 004,1 руб. куда ниже среднерыночной цены в 27 500 руб. Реализация данного приложения принесет прибыль организации в сумме 7912,57 рублей, уровень рентабельности составит 52,74%.

Заключение

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*70*

*БГТУ 00.00.ПЗ*

Разраб*.*

*Чистякова Ю.А.*

Провер*.*

*.*

*Годун А.В.*

Консульт.

*Годун А.В.*

Н. контр.

*Рыжанкова А.С.*

Утв*.*

*Пацей Н.В.*

Заключение

*Лит.*

*Листов*

*1*

74417011, 2021

В рамках работы над дипломным проектом был проведен обзор аналогичных решений, проведен патентный поиск, выбраны язык и платформа для разработки сервера, технологии для разработки клиентской части веб-приложения.

В итоге было разработано веб-приложение, которое с одной стороны позволяет программисту создать услугу, с другой предоставляет клиентам простой и удобный способ для бронирования услуг программиста онлайн.

Для выполнения поставленных задач были спроектированы:

* логическая схема базы данных;
* диаграмма вариантов использования;
* диаграмма развертывания;
* структурная схема веб-приложения;
* блок-схема алгоритма бронирования услуги программиста;
* блок-схема алгоритма создания услуги программиста.

Было дано описание негативных тестов, которые были проведены на этапе тестирования веб-приложения. Были протестированы все случаи неправильной эксплуатации веб-приложения.

В пояснительной записке приведено руководство пользователя по использованию веб-приложения, которое описывает основные сценарии работы, что позволяет пользователям легко ориентироваться в приложении и работать с ним.

Рассчитаны затраты на разработку представленного в дипломном проекте программного средства. На основании полученных данных можно сделать вывод, что разработанный проект является экономически выгодным.

Конечный продукт предоставляет информацию об услугах программиста, позволяет осуществлять бронирование услуг.

Веб-приложение является законченным программным продуктом, который реализует все перечисленные возможности в полной форме. Веб-приложение поддерживает архитектуру, которая позволяет в дальнейшем быстро и безопасно расширить функционал.

Результаты дипломного проекта доложены на 72-ой научно-технической конференции учащихся, студентов и магистрантов БГТУ с публикацией материалов конференции в сборнике работ.

# **Список использованных источников**

1. И­Т в Беларуси [Электронный ресурс] // belarus.by. – Режим доступа: https://www.belarus.by/ru/business/doing-business/it-belarus/. – Дата доступа 10.04.2021.
2. Зачем компании берут в аренду программистов [Электронный ресурс] // vc.ru – Режим доступа: https://vc.ru/tribuna/102469-marketpleys-zhivyh-lyudey/. – Дата доступа 10.04.2021.
3. Каким должен быть сайт фирмы [Электронный ресурс] // softformula.com. – Режим доступа: https://www.softformula.com/entsiklopedia/kakim-dolzhen-byt-sayt-turfirmy/. – Дата доступа: 11.04.2021.
4. Каким должен быть сайт. Информация, внешний вид, документы [Электронный ресурс] // travelsoft.by. – Режим доступа: https://travelsoft.by/blog/5-sovetov-kakim-dolzhen-byt-sayt-turfirmy-v-2019-godu/. – Дата доступа: 11.04.2021.
5. TrueTech [Электронный ресурс] // truetech.by. – Режим доступа: http://truetech.by/programmirovanie.html/. – Дата доступа: 11.04.2021.
6. Redsale [Электронный ресурс] // redsale.by. – Режим доступа: https://redsale.by/freelance/uslugi-programmista/. – Дата доступа: 11.04.2021.
7. FreelanceHunt [Электронный ресурс] // freelancehunt.by. – Режим доступа: https://freelancehunt.by/freelancers/programmirovanie/. – Дата доступа: 12.04.2021.
8. Flagma [Электронный ресурс] // flagma.by. – Режим доступа: https://flagma.by/vebdizayn-sozdanie-i-prodvizhenie-saytov-o-1.html/. – Дата доступа: 12.04.2021.
9. Рекомендации по выполнению патентного поиска [Электронный ресурс] // bsuir.by. – Режи­м доступа: https://bsuir.by/m/12\_100229\_1\_98996.htm – Дата доступа: 12.04.2021.
10. Патент RU2324221C2 [Электронный ресурс] // patents.google.com. – Режим доступа: https://patents.google.com/patent/RU2324221C2/ru/. – Дата доступа: 12.04.2021.
11. Flexberry [Электронный ресурс] flexberry.github.io. – Режим доступа: https://flexberry.github.io/ru/gbt\_javascript.html/. – Дата доступа: 13.04.2021.
12. Язык программирования JavaScript [Электронный ресурс] // vc.ru. – Режим доступа: https://vc.ru/hr/145461-yazyk-programmirovaniya-javascript-osoben–nosti-i-preimushchestva/. – Дата доступа: 13.04.2021.
13. Описание продукта JB WebStorm [Электронный ресурс] // itpro.ua. – Режим доступа: https://itpro.ua/product/jetbrains-webstorm/. – Дата доступа 13.04.2021.
14. О Node.js [Электронный ресурс] // nodejs.org. – Режим доступа: https://nodejs.org/ru/about/. – Дата доступа: 13.04.2021.

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*71*

*БГТУ 00.00.ПЗ*

Разраб*.*

*Чистякова Ю.А.*

Провер*.*

*.*

*Годун А.В.*

Консульт*.*

*Годун А.В.*

Н. контр.

*Рыжанкова А.С.*

Утв.

*Пацей Н.В.*

Список использованных

источников

*Лит.*

*Листов*

*2*

74417011, 2021

1. Чем хорош Node.js [Электронный ресурс] // techrocks.ru. – Режим доступа: https://techrocks.ru/2019/01/20/why-do-you-need-node-js/. – Дата доступа: 13.04.2021
2. Выбор лучшего Node.js фреймворка [Электронный ресурс] // umbrellait.com. – Режим доступа: https://umbrellait.com/ru/blog/choosing-the-best-nodejs-framework/. – Дата доступа: 13.04.2021
3. Виртуальный DOM и его реализация в React [Электронный ресурс] // ru.reactjs.org. – Режим доступа: https://ru.reactjs.org/docs/faq-internals.html/. – Дата доступа 14.04.2021.
4. RFC 2616 Hypertext Transfer Protocol – HTTP 1/1 [Электронный ресурс] // tools.ietf.org. – Режим доступа: https://tools.ietf.org/html/rfc2616/. – Дата доступа: 16.04.2021.
5. Пустовалов А.К. Использование сценариев JavaScript в HTML / Пустовалов А.К. – Минск: УО «Белорусский государственный технологический университет», 2018. – 257 с.
6. JavaScript [Электронный ресурс] / developer.mozilla.org – Режим доступа: https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide/JavaScript\_Overview/ – Да-та доступа: 19.04.2021.
7. About Node.js [Электронный ресурс] / Node.js – 2018. Режим доступа: https://nodejs.org/en/about. – Дата доступа: 20.04.2021.
8. Каштелян, Т.В. Экономическое обоснование дипломных проектов / Каштелян Т.В. – Минск: УО «Белорусский государственный технологический университет», 2013. – 88 с.

# ПРИЛОЖЕНИЕ А

Диаграмма развертывания

# ПРИЛОЖЕНИЕ Б

Диаграмма вариантов использования

# ПРИЛОЖЕНИЕ В

Скриншот работы веб-приложения

# ПРИЛОЖЕНИЕ Г

Таблица расчетов экономических показателей

# ПРИЛОЖЕНИЕ Д

Структурная схема веб-приложения

# ПРИЛОЖЕНИЕ Е

Логическая схема базы данных

# ПРИЛОЖЕНИЕ Ж

Блок-схема бронирования услуги программиста

# ПРИЛОЖЕНИЕ З

Блок-схема создания услуги программиста

# ПРИЛОЖЕНИЕ И

Листинг функции подтверждения заказа

|  |
| --- |
| const mongoose = require('mongoose');  function Offer(options) {    this.core = options.core;  }  Offer.prototype.approve = function (creator, params, offerId) {    const Booking = mongoose.model('Booking');    const Calendar = mongoose.model('Calendar');    let booking = {};    return Booking.findOne({      \_id: params.bookingId,      offerCreatorId: creator.id,      offerId: offerId,      status: 'pending'    }).populate('clientId')      .populate({path: 'offerId', populate: {path: 'category'}})      .populate({path: 'offerId', populate: {path: 'subCategory'}})      .then(existBooking => {        if (!existBooking) throw new AppError({status: 400, message: errorMessages.BOOKING\_NOT\_FOUND});        booking = existBooking;        existBooking.set('status', 'approve');        return existBooking.save();      })      .then(booking => Calendar.updateMany({bookingId: booking.id}, {$set: {status: 'disabled',}}))      .then(() => this.core.chat.sendSystemMessage('Request has been Approved', params.bookingId))      .then(() => {        return new Promise((resolve, reject) => {          const subject = 'Request has been Approved';          const data = {            offerPhotoURL: `${appURL}/uploads/${booking.offerId.avatarUrl}`,            category: booking.offerId.category.category,            offerName: booking.offerId.name,            date: new Date(Date.now()),            title: 'Offer has been Approved',            bookingLink: `${settings.appURL}/inbox/${booking.id}/booking`,          };        });      })      .catch(err => {        if (err instanceof AppError) throw err;        throw new AppError({status: 500, message: errorMessages.SERVER\_ERROR, err: err});      })  }; |