Частное учреждение образования

«Колледж бизнеса и права»

ДОПУЩЕН К ЗАЩИТЕ

Заместитель директора

По учебной работе

\_\_\_\_\_\_\_\_\_\_\_ И.В.Малафей

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024

РЕАЛИЗАЦИЯ ВЕБ-ПРИЛОЖЕНИЯ ПО НАБОРУ ИСПОЛНИТЕЛЕЙ

ДЛЯ РЕАЛИЗАЦИИ ПРОЕКТОВ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

ДП Т.093151.401

Председатель цикловой комиссии (Т.Г.Багласова )

Руководитель проекта (А.С.Миско )

Консультант по экономической части (Е.А.Андреева )

Консультант по охране труда (В.С.Кудласевич)

Обучающийся (В.А.Черняков )

Рецензент ( )

2024

Тех задание

**СОДЕРЖАНИЕ**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

3

ДП Т.093151.401 ПЗ

Разраб.

Черняков В.А.

Провер.

Миско. А.С.

Т. Контр.

Н. Контр.

Кудрявцева М.О.

Утверд.

Багласова Т.Г.

ова Т.Г.

*Реализация веб-приложения по набору исполнителей для реализации преоктов*

Лит.

Листов

КБиП

77

У

[Введение 4](#_Toc168607328)

[1 Описание задачи 6](#_Toc168607329)

[1.1 Анализ предметной области 6](#_Toc168607330)

[1.2 Постановка задачи 6](#_Toc168607331)

[2 Проектирование веб-приложение 8](#_Toc168607332)

[2.1 Проектирование модели 8](#_Toc168607333)

[2.2 Требования к веб-приложению 10](#_Toc168607334)

[2.3 Структура веб-приложения 11](#_Toc168607335)

[2.4 Проектирование макета веб-приложения 11](#_Toc168607336)

[2.5 Программно-технические средства, необходимые для разработки приложения 13](#_Toc168607337)

[2.6 Защита и сохранность данных 13](#_Toc168607338)

[2.7 Организация и ведение информационной базы (модели) 14](#_Toc168607339)

[3 Реализация веб-приложения 18](#_Toc168607340)

[3.1 Описание разделов веб-приложения 18](#_Toc168607341)

[3.2 Разработка административной части приложения 18](#_Toc168607342)

[3.3 Разработка клиентской части приложения 18](#_Toc168607343)

[3.4 Описание используемых функций и процедур 20](#_Toc168607344)

[3.5 Функциональное тестирование 21](#_Toc168607345)

[4 Применение 23](#_Toc168607346)

[4.1 Назначение веб-приложения 23](#_Toc168607347)

[4.2 Программно-аппаратное обеспечение сервера и клиента 23](#_Toc168607348)

[5 Охрана труда и окружающей среды 25](#_Toc168607349)

[5.1 Правовые, нормативные, социально-экономические и организационные вопросы охраны труда. 25](#_Toc168607350)

[5.2 Охрана творческого труда 28](#_Toc168607351)

[5.3 Пожарная безопасность 29](#_Toc168607352)

[5.4 Охрана окружающей среды 31](#_Toc168607353)

[6 Экономический раздел 33](#_Toc168607354)

[6.1 Расчет затрат на разработку программного продукта 33](#_Toc168607355)

[6.2 Расчет показателей экономической эффективности разработки 35](#_Toc168607356)

[6.3 Расчет экономической эффективности у пользователя программного продукта 36](#_Toc168607357)

[Заключение 37](#_Toc168607358)

[Список используемых источников 38](#_Toc168607359)

[Приложение А Структура веб-приложения 39](#_Toc168607360)

[Приложение Б Текст модулей веб-приложения 40](#_Toc168607361)

[Приложение В Результаты работы приложения 77](#_Toc168607362)

Багласова Т.Г.

ова Т.Г.

# Введение

На фоне быстрого развития информационных технологий и индустрии проектного управления, эффективное формирование команды для разработки проектов становится ключевым аспектом успешной реализации идей. Научные исследования и практика показывают, что поиск подходящих участников, сопоставление их навыков с требованиями проекта, а также управление командой, это неотъемлемые компоненты успешной реализации проектных задач.

Целью дипломного проекта являться разработка веб-приложения для набора исполнителей проекта со следующими возможностями:

– поиск и набор исполнителей для проекта;

– создание профиля и заказов для выполнения работ;

– поиск с применением различных фильтров и критериев;

– просмотр личного кабинета;

– организации системы обмены сообщениями для обсуждения проекта.

Решение поставленных задач содержится в пояснительной записке из шести разделов для веб-приложения.

В первом разделе «Описание задачи» содержится описание поставленной задачи и её анализа, список основных задач, которых предстоит реализовать в рамках предметной области. Раздел так же содержит анализ предметной области.

Во втором разделе «Проектирование веб-приложения» содержится подробное описание наработок будущих прототипов и моделей, а так же другие программные решения и средства для последующей реализации веб-приложения.

В третьем разделе «Реализация веб-приложения» содержится описание разработанного веб-приложения, применяемые инструменты разработки и технологии, порядок авторизации пользователей, организация данных в рамках выбранной системе управления базами данных, логическая и физическая организация функций, входные и выходные данные. В разделе так же приводится тестирование на основании подготовленных модульных тестов, описывается справочная система.

В четвертом разделе «Применение» содержится описание назначения программного средства и условий его при применения.

В пятом разделе «Охрана труда и окружающей среды» содержится информация о правовых, нормативных, социально-экономических и организационных вопросах охраны труда, о разработке мер по нормализации и защите работающих от статического электричества при эксплуатации персональной электронно-вычислительной машины (ПЭВМ), о требованиях пожарной безопасности и охране окружающей среды.

В шестом экономическом разделе приводится обоснования необходимости выведения программного средства на рынок, составляется план на разработку программного средства и рассчитываются затраты на его разработку, а также рассчитывается экономический эффект у разработчика и пользователя программного средства.

В заключении содержится общий итог о проделанной работе, анализируется степень реализации функций, извлекаются соответствующие выводы о качестве разработанного программного средства.

В приложении А представлена структура веб-приложения.

В приложении Б представлен текст модулей веб-приложения.

В приложении В представлены результаты работы приложения.

Графическая часть представлена диаграммой вариантов использования, диаграммой последовательности, диаграммой развёртывания и структурой сайта.

# Описание задачи

## Анализ предметной области

Предметной областью решаемой задачи является сфера деятельности по поиску и координации исполнителей для проектов различной направленности. В данной предметной области осуществляется контроль и управление процессом поиска, назначения и выполнения проектных задач.

Веб-приложение направлено на обеспечение эффективного поиска исполнителей для различных проектов. Процесс работы с приложением выглядит следующим образом:

– регистрация и авторизация пользователей, предоставляющих проекты и желающих найти исполнителей;

– подача заявки на поиск исполнителя, где пользователи описывают требования к проекту и необходимые критерии исполнителя;

– выполнение задачи, где выбранные исполнители приступают к реализации проекта в соответствии с требованиями заказчика;

– завершение и отчётность, где после завершения проекта происходит закрытие заявки, а в системе делается запись о выполненной работе.

На сегодняшний день для ведения учёта проектов и поиска исполнителей может использоваться ряд различных инструментов, но важно обеспечить оперативность реагирования на запросы и эффективное ведение учёта проектов для обеспечения успешной реализации проектов заказчиками.

## Постановка задачи

Исходя из анализа предметной области и требований веб-приложения выделяются следующие задачи, подлежащие автоматизации:

– запись и систематизация информации о проектах и требованиях к исполнителям;

– отслеживание статуса проектов и доступность исполнителей.

Программа "Cukierka" предоставляет следующие преимущества:

– удобство использования, приложение обладает интуитивно понятным интерфейсом, который позволяет пользователям легко регистрировать и отслеживать проекты и соответствующих им исполнителей;

– модульность, приложение может быть настроено под конкретные потребности пользователей, включая различные типы проектов и требований к исполнителям.

На сегодняшний день решение указанных задач осуществляется разрозненно и вручную. Пользователи вынуждены использовать различные инструменты для регистрации и отслеживания проектов, что часто приводит к потере данных и неэффективности в работе.

Хотя на рынке существуют аналогичные решения, такие как "fl.ru" или "freelance.ru", они часто требуют значительных финансовых затрат и сложны в внедрении. Программа "Cukierka" разрабатывается с учетом потребностей пользователей и направлена на предоставление более гибкого и простого в использовании решения для поиска исполнителей и управления проектами.

# Проектирование веб-приложение

## Проектирование модели

Модель предметной области (инфологическая модель) – это описание предметной области, выполненное без ориентации на используемые в дальнейшем программные и технические средства. Цель инфологического проектирования заключается в представлении семантики предметной области. Инфологическая модель используется на этапе проектирования базы данных. Она помогает разработчикам базы данных понять структуру предметной области и определить, какие данные необходимо хранить в базе данных.

Диаграмма вариантов использования – это графическое представление функциональности системы с точки зрения пользователей. Диаграмма вариантов использования является видом диаграмм UML (Unified Modeling Language, Единый язык моделирования), которая показывает, как пользователи взаимодействуют с системой для достижения своих целей. Они помогают понять функциональность системы и общаться с заинтересованными сторонами. Суть диаграммы вариантов использования заключается в том, что проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью, так называемых вариантов использования.

Диаграмма вариантов использования представлена на листе 1 графической части.

Концептуальная модель веб-приложения представляет собой абстрактное представление структуры, основных компонентов и взаимосвязей веб-приложения. Она помогает разработчикам и дизайнерам лучше понять функциональные и нефункциональные требования к приложению, определить его основные возможности и поведение, а также логику взаимодействия между пользователем и системой.

В концептуальной модели веб-приложения мы определяем абстрактное представление структуры и компонентов приложения. На основе анализа были выделены сущности, "Пользователи", "Вакансии" и "Отклики", "Проекты", "Сообщения", "Комментарии" с их соответствующими атрибутами.

Сущность «Пользователи» имеет следующие атрибуты:

– почта;

– имя;

– пароль;

– дата создания;

– информация.

Сущность «Вакансии» имеет следующие атрибуты:

– заголовок;

– автор;

– информация;

– заработная плата;

– статус;

– дата создания.

Сущность «Отклики» имеет следующие атрибуты:

– пользователь;

– вакансия;

– дата создания.

Сущность «Проекты» имеет следующие атрибуты:

– пользователь;

– вакансия;

– дата создания.

Сущность «Сообщения» имеет следующие атрибуты:

– автор;

– проект;

– содержимое;

– дата создания.

Сущность «Комментарии» имеет следующие атрибуты:

– автор;

– проект;

– содержимое;

– рейтинг;

– дата создания.

Концептуальная модель реляционной базы данных представлена на рисунке 2.1

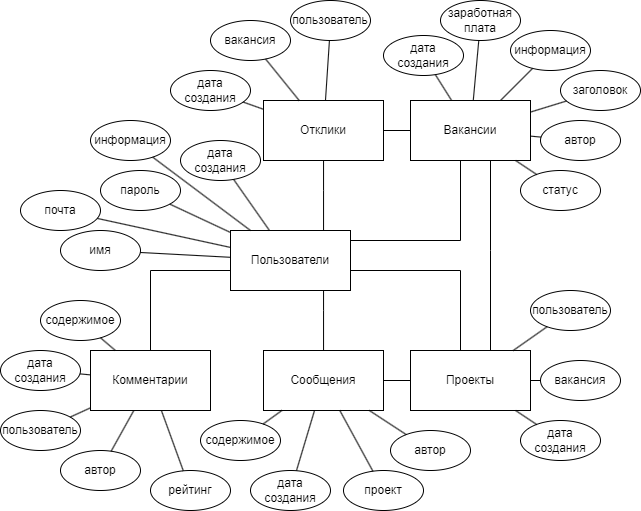


Рисунок 2.1 – Концептуальная модель базы данных

Диаграмма последовательности — вид диаграмм UML, используемых для описания взаимодействия объектов в системе в разрезе времени. Она демонстрирует, как объекты взаимодействуют друг с другом, какие сообщения передаются между ними, и в каком порядке это происходит. Диаграммы последовательности часто используются для моделирования сценариев использования, объяснения и детализации поведения системы.

Диаграмма последовательности представлена на листе 2 графической части.

## Требования к веб-приложению

Дизайн веб-приложения играет важную роль в обеспечении удобства и удовлетворительного пользовательского опыта. Основные требования к графическому интерфейсу веб-приложения включают:

– простая цветовая схема, использование 2-3 основных цветов, обеспечивающих читаемость и приятный визуальный опыт;

– деловой стиль, соответствие деловому стилю, исключая элементы, которые могут создать впечатление несерьезности или некомпетентности;

– нейтральная цветовая гамма, использование нейтральных цветов, таких как белый, зеленый и чёрный, для создания профессионального впечатления;

– минималистичный дизайн, предотвращение перегруженности интерфейса, обеспечивая четкость и ясность информации;

– логотип, включение логотипа, который отражает идентичность и название веб-приложения;

– ограниченное количество шрифтов, использование не более 2-3 шрифтов на сайте для единообразия и читаемости текста;

– кросс-браузерная и адаптивная совместимость, корректное отображение интерфейса в различных браузерах (Chrome, Firefox, Safari, Edge) и на различных устройствах (ПК, планшеты, смартфоны), обеспечивая доступность для всех пользователей.

## Структура веб-приложения

Структура веб-приложения включает в себя организацию всех компонентов и элементов, которые составляют приложение, обеспечивая его функциональность и взаимодействие с пользователями. Она определяет, как данные обрабатываются и передаются между клиентской и серверной частями, а также как элементы интерфейса организованы для оптимального взаимодействия пользователя с системой. Ниже описаны ключевые элементы структуры веб-приложения:

– веб-сервер Nginx;

– веб-приложение Cukierka;

– система управления базами данных PostgreSQL.

Диаграмма развертывания — вид диаграмм UML, который используется для отображения физического развертывания артефактов программного обеспечения на узлах системы. Такие диаграммы показывают, как различные программные компоненты, приложения и базы данных разворачиваются на различных узлах, таких как серверы, рабочие станции и другие устройства.

Диаграмма развёртывания для веб-приложения представлена на 3 листе графической части.

Главным компонентом всего приложения является веб-сервер, который принимает внешние запросы пользователей и обрабатывает их. Веб-сервер распределяет запросы между веб-приложением и ресурсами сайтов.

Структура сайта для веб-приложения представлена на 4 листе графической части.

Веб-приложение отвечает за логику всего приложения. Оно обрабатывает запросы, проверяет корректность данных, запрашивает и сравнивает изменения с базой данных и возвращает результат.

## Проектирование макета веб-приложения

Дизайн веб-приложения сосредотачивается на обеспечении простоты, удобства использования и минимализма для создания профессионального визуального опыта.

Основные аспекты дизайна:

– разделение информации, информация на странице разделена на столбцы для легкого восприятия данных;

– понятные заголовки, каждый столбец снабжен понятным заголовком, который четко выделяет суть представленной информации;

– сортировка данных, пользователи имеют возможность сортировать данные по различным критериям, обеспечивая легкий доступ к необходимой информации;

– максимальное использование экранного пространства, весь экран используется для эффективного отображения информации без перегрузки интерфейса;

– баланс информативности и удобства чтения, информация представлена в удобочитаемом формате, обеспечивая максимальную информативность при минимальном напряжении для восприятия;

– поиск, реализован эффективный механизм поиска, который обеспечивает управление большим объемом данных;

– простые цвета и формы, используются простые цвета и формы, соответствующие общему стилю приложения и обеспечивающие единый и профессиональный вид интерфейса.

Выбранный тип дизайна сосредотачивается на минимализме, удобстве использования и профессионализме, что делает его идеальным для веб-приложений, ориентированных на широкий круг пользователей. Минималистический подход позволяет избежать перегрузки интерфейса, фокусируя внимание на важной информации и функциональности. Простые формы и цвета способствуют созданию чистого и аккуратного внешнего вида, который улучшает общее восприятие приложения. Удобство использования достигается за счет интуитивно понятных элементов интерфейса и возможности гибкой настройки отображаемых данных. Все эти аспекты делают дизайн максимально эффективным для выполнения задач пользователей.

Концептуальный прототип веб-приложения предоставлен на рисунке А.1 и А.2 приложения А.

## Программно-технические средства, необходимые для разработки приложения

Комплекс программно-технических средств был тщательно подобран с учетом требований к веб-приложению и ориентированности на эффективность разработки и поддержки. Ниже приведены основные инструменты и технологии, которые были выбраны для разработки приложения:

– операционная система Windows 10;

– система контроля версий Git.

– язык программирования Rust;

– язык запросов SQL;

– система управления базами данных PostgreSQL;

– среда разработки Visual Studio Code (VSCode);

Операционная система Windows 10 – Проприетарная операционная система для персональных компьютеров, обеспечивающая удобную среду разработки и обширную поддержку инструментов и приложений [3].

Система контроля версий Git – Распределенная система управления версиями файлов, которая обеспечивает коллаборацию и управление исходным кодом проект [4].

Язык программирования Rust – Высокоуровневый системный язык программирования, который обеспечивает безопасность памяти без использования сборщика мусора [5].

Язык запросов SQL – Для взаимодействия с базой данных PostgreSQL. Так же позволяет делать запросы и к другим СУБД [6].

Система управления базами данных PostgreSQL – Мощная и расширяемая реляционная база данных, обеспечивающая надежное хранение данных приложения [6].

Среда разработки Visual Studio Code (VSCode) – Редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как легкий редактор кода для кроссплатформенной разработки веб и облачных приложений [7].

## Защита и сохранность данных

В современном мире безопасность данных является одной из главных проблем при разработке веб-приложений. Далее подробнее представленкаждый из этих аспектов.

Ограничение доступа к данным в веб-приложении осуществляется через механизмы аутентификации и авторизации. Сайт предоставляет разделение доступа для пользователей на основе от того, вошли они в аккаунт или нет. Администраторы имеют полный доступ ко всем данным и функциональности приложения, а незарегистрированные пользователи могут иметь ограниченный доступ ко всему функционалу. Все остальные пользователи получают стандартный доступ.

В программном средстве механизмы управления доступом к данным реализуются с помощью учетных записей и ролей базы данных.

## Организация и ведение информационной базы (модели)

В любой информационной системе данные хранятся в определенной структуре. Эта структура называется моделью данных. Модель данных определяет, как данные будут организованы и как к ним будет осуществляться доступ. Разрабатываемое программное средство содержит 7 таблиц для работы веб-приложения. Краткое описание таблиц проходит в таблицах 2.1-2.7.

Таблица «Users» хранит информацию о пользователях. Структура таблицы представлена в таблице 2.1.

Таблица 2.1 – Таблица «Users»

|  |  |  |  |
| --- | --- | --- | --- |
| Столбец | Тип данных | Размер | Ограничения |
| Id | UUID | 16 bytes | PRIMARY KEY |
| Email | VARCHAR | varies | NOT NULL, UNIQUE |
| Name | VARCHAR | varies | NOT NULL |
| information | VARCHAR | varies | – |
| password | VARCHAR | varies | NOT NULL |
| created | TIMESTAMPTZ | 8 bytes | NOT NULL |

Таблица «Tokens» хранит информацию о ключах авторизации. Структура таблицы представлена в таблице 2.2.

Таблица 2.2 – Таблица «Tokens»

|  |  |  |  |
| --- | --- | --- | --- |
| Столбец | Тип данных | Размер | Ограничения |
| Id | UUID | 16 bytes | PRIMARY KEY |
| user\_id | UUID | 16 bytes | FOREIGN KEY (Users.id) ON DELETE CASCADE |
| created | TIMESTAMPTZ | 8 bytes | NOT NULL |

Таблица «Vacancies» хранит информацию о созданных вакансиях. Структура таблицы представлена в таблице 2.3.

Таблица 2.3 – Таблица «Vacancies»

|  |  |  |  |
| --- | --- | --- | --- |
| Столбец | Тип данных | Размер | Ограничения |
| Id | UUID | 16 bytes | PRIMARY KEY |
| author\_id | UUID | 16 bytes | FOREIGN KEY (Users.id) ON DELETE CASCADE |
| Title | VARCHAR | varies | NOT NULL |
| information | VARCHAR | varies | NOT NULL |
| payment | INTEGER | 4 bytes | NOT NULL |
| Status | VARCHAR | varies | NOT NULL |
| created | TIMESTAMPTZ | 8 bytes | NOT NULL |

Таблица «Tikets» хранит информацию о готовности исполнителя взяться за определенную вакансию. Структура таблицы представлена в таблице 2.4.

Таблица 2.4 – Таблица «Tikets»

|  |  |  |  |
| --- | --- | --- | --- |
| Столбец | Тип данных | Размер | Ограничения |
| Id | UUID | 16 bytes | PRIMARY KEY |
| user\_id | UUID | 16 bytes | FOREIGN KEY (Users.id) ON DELETE CASCADE |
| vacancy\_id | VARCHAR | varies | FOREIGN KEY (Vacancies.id) ON DELETE CASCADE |
| created | TIMESTAMPTZ | 8 bytes | NOT NULL |

Таблица «Projects» хранит информацию о проектах. Структура таблицы представлена в таблице 2.5.

Таблица 2.5 – Таблица «Projects»

|  |  |  |  |
| --- | --- | --- | --- |
| Столбец | Тип данных | Размер | Ограничения |
| Id | UUID | 16 bytes | PRIMARY KEY |
| user\_id | UUID | 16 bytes | FOREIGN KEY (Users.id) ON DELETE CASCADE |
| vacancy\_id | VARCHAR | varies | FOREIGN KEY (Vacancies.id) ON DELETE CASCADE |
| created | TIMESTAMPTZ | 8 bytes | NOT NULL |

Таблица «Messages» хранит информацию о сообщениях в проекте. Структура таблицы представлена в таблице 2.6.

Таблица 2.6 – Таблица «Messages»

|  |  |  |  |
| --- | --- | --- | --- |
| Столбец | Тип данных | Размер | Ограничения |
| Id | UUID | 16 bytes | PRIMARY KEY |
| author\_id | UUID | 16 bytes | FOREIGN KEY (Users.id) ON DELETE CASCADE |
| project\_id | VARCHAR | varies | FOREIGN KEY (Projects.id) ON DELETE CASCADE |
| content | VARCHAR | varies | NOT NULL |
| created | TIMESTAMPTZ | 8 bytes | NOT NULL |

Таблица «Comments» хранит информацию о комментариях пользователей. Структура таблицы представлена в таблице 2.7.

Таблица 2.7 – Таблица «Comments»

|  |  |  |  |
| --- | --- | --- | --- |
| Столбец | Тип данных | Размер | Ограничения |
| Id | UUID | 16 bytes | PRIMARY KEY |
| author\_id | UUID | 16 bytes | FOREIGN KEY (Users.id) ON DELETE CASCADE |
| project\_id | VARCHAR | varies | FOREIGN KEY (Projects.id) ON DELETE CASCADE |
| content | VARCHAR | varies | NOT NULL |
| created | TIMESTAMPTZ | 8 bytes | NOT NULL |

Модели данных делятся на логические и физические. Логическая модель определяет структуру данных независимо от их физического хранения, включая сущности, атрибуты и отношения. Физическая модель определяет конкретные способы хранения данных на диске, такие как типы данных, индексы и организация файлов. Обе модели играют ключевую роль в разработке информационных систем, обеспечивая эффективное управление и использование данных.

Физическая модель данных разрабатываемого веб-приложения представлена на рисунке 2.4.

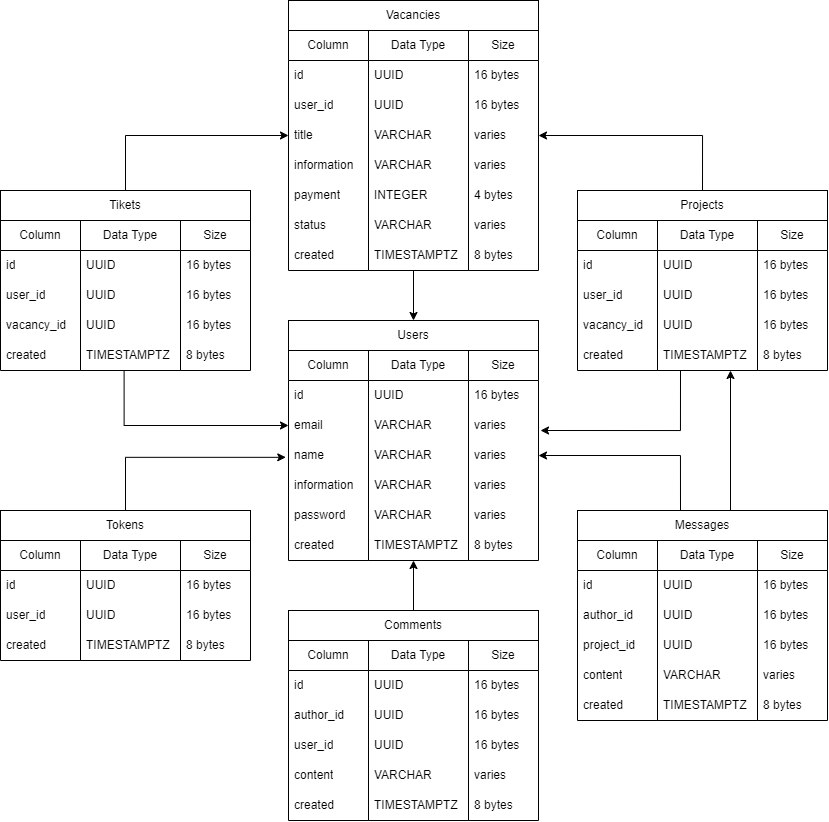


Рисунок 2.4 – Физическая модель базы данных

# Реализация веб-приложения

## Описание разделов веб-приложения

На главной странице представлено навигационное меню веб-приложения и форма для предоставления всей информации.

Навигационное меню – это область интерфейса, на которой представлены навигационные опции, которые позволяют пользователям находить контент веб-приложения.

Навигационное меню предоставляет удобный доступ к основным разделам приложения, обеспечивая ему возможность просмотра, добавления и редактирования информации.

Структура навигационного меню на главная странице представлена ниже:

– «Главная»;

– «Вакансиями»;

– «Мо вакансии»;

– «Аккаунт»;

– «Выйти».

Раздел «Главная» представляет собой основную страницу веб-приложения, с которой начинается работа пользователя. Здесь представлены диаграммы, которые показывают актуальную ситуацию по набору исполнителей. Главная страница служит отправной точкой для перехода к другим разделам.

В разделе «Вакансии» пользователи могут просматривать список всех доступных вакансий. Вакансии могут быть отсортированы по различным критериям, таким как дата размещения, категория, компания и т.д. Пользователь может ознакомиться с деталями каждой вакансии, такими как описание работы, требования и условия.

Раздел «Мои вакансии» предназначен для пользователей, которые разместили свои собственные вакансии. Этот раздел помогает пользователям следить за своими вакансиями.

В разделе «Аккаунт» пользователи могут управлять своими личными данными и настройками. Здесь можно просматривать профиль пользователя, а также оставлять и просматривать комментарии. Этот раздел предоставляет пользователю контроль над его учетной записью.

Раздел «Выйти» является интерактивной кнопкой, позволяющую пользователю выйти и сменить аккаунт.

## Разработка административной части приложения

Физическая структура – реальные компоненты и артефакты системы, такие как файлы, папки, базы данных. Это низкоуровневое представление системы, которое учитывает конкретные технологии, инфраструктуру и детали реализации. Физическая структура определяет, как компоненты системы фактически реализуются и взаимодействуют между собой в реальной среде.

Физическая структура административной части приложения представлена на рисунке 3.1.

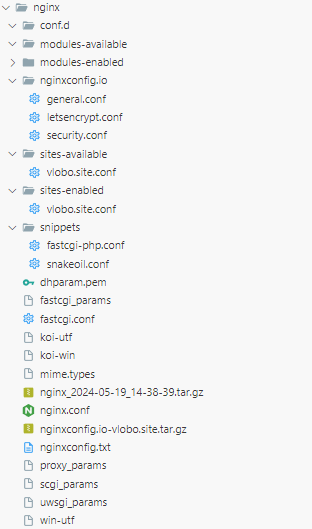


Рисунок 3.1 – Физическая структура административной части приложения

## Разработка клиентской части приложения

Клиентская часть веб-приложения представляет собой интерфейс, с которым взаимодействуют пользователи, обеспечивая доступ к функциональности и данным системы. Основное внимание уделяется удобству и интуитивности пользовательского интерфейса (UI), чтобы пользователи могли легко находить необходимую информацию и выполнять требуемые действия.

Навигационное меню должно быть интуитивно понятным и легкодоступным, чтобы пользователь мог быстро ориентироваться и переходить между различными страницами приложения. Оно предоставляет быстрый доступ к ко всем функциям приложения, позволяя пользователям эффективно управлять данными и осуществлять необходимые действия.

На главной странице приложения расположено навигационное меню и форма для предоставления информации, о актуальных метриках на сайте.

Навигационное меню предоставляет удобный доступ к основным разделам приложения, обеспечивая пользователям возможность быстро ориентироваться и переходить между различными страницами. Структура навигационного меню включает следующие элементы:

– главная страница, возвращает пользователя на стартовую страницу приложения;

– страница резюме, позволяет пользователям просматривать и управлять резюме;

– страница с вакансиями, отображает доступные вакансии, на которые пользователи могут откликаться;

– страница с «моими» вакансиями, показывает вакансии, на которые пользователь уже откликнулся или которые отметил;

– аккаунт, предоставляет доступ к настройкам и информации профиля пользователя;

– таблицы, отображает структурированные данные в виде таблиц для удобства анализа и работы.

– выйти, кнопка для выхода из учётной записи пользователя.

При разработке клиентской части веб-приложения особое внимание уделяется следующим аспектам:

– навигационное меню должно быть легко доступным и понятным, чтобы пользователь мог быстро находить нужные разделы;

– форма для ввода информации должна быть логично структурирована и включать необходимые подсказки для пользователя;

– визуальное оформление интерфейса должно быть привлекательным и соответствовать современным стандартам веб-дизайна;

– использование единой цветовой схемы и типографики для создания гармоничного внешнего вида приложения;

– все элементы интерфейса должны работать быстро и без задержек;

– реализация поиска и генерации отчётов в CSV-формате должна быть оптимизирована для быстрого выполнения запросов пользователя.

Разработка клиентской части приложения предполагает тесное взаимодействие с пользователями для получения обратной связи и проведения тестирования на всех этапах, что позволяет вносить необходимые изменения и улучшения для достижения наилучшего пользовательского опыта.

Физическая структура сайта для пользователя веб-приложения представлена на рисунке 3.1

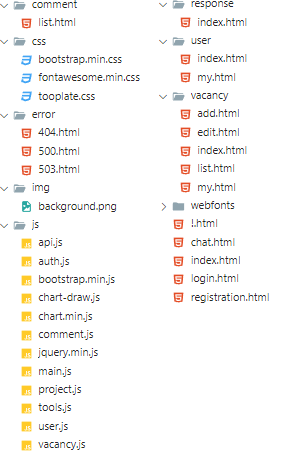


Рисунок 3.1 – Физическая структура сайта для пользователя

## Описание используемых функций и процедур

Для обеспечения функциональности веб-приложения был разработан набор JavaScript-функций и процедур. Ниже приведено описание каждой из этих функций:

Функция getCookie используется для извлечения значения cookie по его имени. Она принимает имя cookie в качестве аргумента и возвращает его значение. Если cookie с указанным именем не найден, функция возвращает null. Код функции представлен ниже.

function getCookie(name) {

var cookies = document.cookie.split(";"); // Разделяем строку куки на отдельные куки

for (var i = 0; i < cookies.length; i++) {

var cookie = cookies[i].trim(); // Удаляем лишние пробелы

if (cookie.startsWith(name + "=")) { // Проверяем, начинается ли текущая куки с искомого имени

return cookie.substring(name.length + 1); // Возвращаем значение куки (без имени)

}

}

return null; // Если куки с указанным именем не найдено

}

Функция deleteCookie удаляет cookie с указанным именем. Она устанавливает истекший срок действия (expires) для cookie на прошедшую дату, чтобы браузер мог удалить его. Код функции представлен ниже.

function deleteCookie(name) {

document.cookie = name + "=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;";

}

Функция setCookie устанавливает новый cookie с заданным именем, значением и сроком действия в днях. Если срок действия не указан, cookie будет действовать только в текущей сессии. Код функции представлен ниже.

function setCookie(name, value, days) {

var expires = "";

if (days) {

var date = new Date();

date.setTime(date.getTime() + (days \* 24 \* 60 \* 60 \* 1000));

expires = "; expires=" + date.toUTCString();

}

document.cookie = name + "=" + value + expires + "; path=/";

}

Функция sendSql отправляет SQL-запрос на сервер для выполнения. Она принимает строку SQL в качестве аргумента и возвращает данные, полученные в ответ на запрос. Код функции представлен ниже.

async function sendSql(sql) {

const response = await fetch("/api/sql", {

method: "POST",

headers: {

"Content-Type": "application/json"

},

body: JSON.stringify({ sql: sql })

});

const data = await response.json();

return data;

}

Функция deleteCookie удаляет cookie с указанным именем. Она устанавливает истекший срок действия (expires) для cookie на прошедшую дату, чтобы браузер мог удалить его. Код функции представлен ниже.

function deleteCookie(name) {

document.cookie = name + "=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;";

}

Функция getQueryParam извлекает значение параметра из строки запроса URL текущей страницы по его имени. Она принимает имя параметра в качестве аргумента и возвращает его значение. Код функции представлен ниже.

async function digestMessage(message) {

const msgUint8 = new TextEncoder().encode(message); // encode as (utf-8) Uint8Array

const hashBuffer = await crypto.subtle.digest("SHA-256", msgUint8); // hash the message

const hashArray = Array.from(new Uint8Array(hashBuffer)); // convert buffer to byte array

const hashHex = hashArray

.map((b) => b.toString(16).padStart(2, "0"))

.join(""); // convert bytes to hex string

return hashHex;

}

## Функциональное тестирование

Функциональное тестирование – это тестирование функций приложения на соответствие всем требованиям и проводится для выявления различных неполадок и недочетов программной реализации веб-приложения на этапе её сдачи в эксплуатацию. Представляет собой комплекс ключевых мероприятий по проверке программного обеспечения, по результатам которых устанавливается соответствие этого ПО исходным требованиям заказчика.

В таблице 3.1 представлены тест-кейсы для проведения функционального тестирования. Предварительное условие: запустить программу и находиться главной странице веб-приложения.

Таблица 3.1 – Тест-кейсы для проведения функционального тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль/Функция | Шаги воспроизведения | Результат |
| 1 | Регистрация пользователя | Перейти на страницу регистрации. Ввести данные: имя, электронная почта, пароль. Нажать кнопку "Зарегистрироваться". | Успешное отображение страницы регистрации. Пользователь успешно зарегистрирован. Перенаправление на страницу авторизации. Результат представлен на рисунке В.1 приложения В. |
| 2 | Авторизация пользователя | Перейти на страницу авторизации. Ввести данные: электронная почта, пароль. Нажать кнопку "Войти". | Успешное отображение страницы авторизации. Пользователь успешно авторизован. Перенаправление на домашнюю страницу.  Результат представлен на рисунке В.1 приложения В. |
| 3 | Добавление вакансии | Авторизоваться. Перейти на страницу добавления вакансии. Заполнить обязательные поля. Нажать кнопку "Сохранить". | Успешное отображение страницы добавления вакансии. Вакансия успешно добавлено. Перенаправление на список вакансий. Результат представлен на рисунке В.1 приложения В. |
| 4 | Добавление комментария | Авторизоваться. Перейти на страницу комментарий пользователя. Заполнить обязательные поля. Нажать кнопку "Отправить". | Успешное отображение комментария. Комментарий успешно добавлен. Результат представлен на рисунке В.1 приложения В. |
| 5 | Создание проекта | Авторизоваться. Перейти на страницу откликов вакансии. Выбрать исполнителя из списка. Нажать кнопку "Выбрать исполнителя". | Успешное отображение страницы вакансии. Чат для вакансии успешно создан. Перенаправление на страницу вакансии. Результат представлен на рисунке В.1 приложения В. |

# Применение

## Назначение веб-приложения

Веб-приложение разработано для облегчения процесса поиска исполнителей для реализации проектов и является важным инструментом для эффективного управления заданиями и услугами. Оно обеспечивает оперативное взаимодействие между заказчиками и исполнителями, способствуя быстрому подбору подходящих кандидатов и решению возникающих задач.

Целью разработки данного веб-приложения является автоматизация и упрощение процесса поиска исполнителей и управления проектами. Это позволяет сократить время на подбор исполнителей, улучшить качество выполнения работ и повысить удовлетворенность заказчиков.

Установка и запуск веб-приложения осуществляются на удаленном сервере без необходимости использования дополнительных веб-серверов или WSGI-серверов. Это обеспечивает простоту и гибкость в настройке и запуске приложения.

Программное обеспечение развернуто на сервере и включает в себя все необходимые компоненты для его функционирования, обеспечивая готовность к использованию без дополнительных конфигураций.

Веб-приложение предоставляет пользователям возможность регистрации, добавления и управления своими резюме или вакансиями, а также отклика на соответствующие запросы. Оно ориентировано на пользователей, ищущих возможности для сотрудничества в рамках различных проектов.

Однако следует отметить, что веб-приложение имеет ограниченный функционал и предназначено исключительно для управления поиском исполнителей и проектами. Доступ к приложению имеют только авторизованные пользователи, обеспечивая безопасность данных и контроль над использованием приложения.

## Программно-аппаратное обеспечение сервера и клиента

Для корректной работы веб-приложения на сервере требуется определенное программное обеспечение, которое включает в себя:

– процессор Intel (Skylake, IBRS) @ 3.792 ГГерц

– памяти 512 МиБ ОЗУ;

– памяти 15 ГиБ свободного места;

– операционная система Ubuntu 22.04.2;

– веб-сервер Nginx;

– подсистема инициализации и управления службами Systemd;

– система управления базами данных PostgreSQL.

После установки и настройки необходимого программного обеспечения на сервере веб-приложение может корректно функционировать и обрабатывать запросы от клиентов.

На стороне клиента требования более общие, которые представлены ниже:

– веб-браузер (Google Chrome, Mozilla Firefox, Safari, Microsoft Edge или Opera);

– стабильное интернет-соединение.

С учетом этих требований веб-приложение будет доступно для использования как на стороне сервера, так и на стороне клиента, обеспечивая стабильную и надежную работу для пользователей.

# Охрана труда и окружающей среды

## Правовые, нормативные, социально-экономические и организационные вопросы охраны труда.

В ООО «ВЭБ Технологии» разработано и функционирует Система управления охраной труда (СУОТ). СУОТ разработан в соответствии с требованиями нормативных правовых актов Республики Беларусь в области охраны труда и СТБ 45001-2020 «Системы менеджмента здоровья и безопасности при профессиональной деятельности. Требования и руководство по применению» и направлено на создание на предприятии условий, обеспечивающих сохранение жизни и здоровья работников в процессе трудовой деятельности, соблюдение законодательства в области охраны труда.

Для организации работы и осуществление контроля по охране труда на предприятии создана служба охраны труда в лице инженера по охране труда.

Основными задачами службы охраны труда являются:

– координация деятельности структурных подразделений организации по обеспечению здоровых и безопасных условий труда;

– совершенствование СУОТ;

– внедрение передового опыта и научных разработок по безопасности и гигиене труда, пропаганда охраны труда;

– информирование и консультирование работников организации, в том числе ее руководителя, по вопросам охраны труда;

– осуществление контроля за соблюдением требований актов законодательства, технических нормативных правовых актов и локальных нормативных правовых актов, содержащих требования охраны труда.

Главный инженер предприятия организует выполнение мероприятий по устранению недостатков и нарушений по охране труда, выявленных в ходе проведения контроля.

Организация, оперативное руководство и координация работы по охране труда, общее руководство управлением охраной труда осуществляется генеральным директором ООО «ВЭБ Технологии». Ответственным за реализацию функций СУОТ являются: главный инженер, заместитель генерального директора, начальники экспедиций, отделов и структурных подразделений, главные инженеры экспедиций.

Инженер по охране труда осуществляет организационно-методическую работу по управлению охраной труда, подготовку управленческих решений, контроль за их реализацией под руководством главного инженера.

Функции инженера по охране труда:

– анализ причин производственного травматизма и профессиональной заболеваемости в организациях, расположенных на подведомственной территории, разработка мер по их профилактике;

– участие в организации разработки и реализации территориальных целевых программ улучшения условий и охраны труда;

– организация обучения, повышения квалификации и проверки знаний по вопросам охраны труда руководителей и специалистов организаций, расположенных на подведомственной территории, которые не находятся в подчинении республиканских органов государственного управления, иных государственных организаций, подчиненных Правительству Республики Беларусь, и других организаций;

– организация и руководство работой кабинета охраны труда в соответствии с Типовым положением о кабинете охраны труда, утвержденным постановлением Министерства труда Республики Беларусь от 8 ноября 1999 г;

– подготовка государственной статистической отчетности по охране труда;

– пропаганда и распространение передового опыта в области охраны труда в организациях, расположенных на подведомственной территории, в том числе через средства массовой информации, организацию выставок, смотров-конкурсов, семинаров, совещаний и других мероприятий;

– участие в установленном законодательством порядке в расследовании несчастных случаев на производстве;

– информационное обеспечение организаций, расположенных на подведомственной территории, по вопросам охраны труда;

– консультирование, подготовка ответов на запросы организаций, рассмотрение в установленном порядке писем, обращений граждан по вопросам охраны труда.

В соответствии с требованиями ст.17 Закона от 23.06.2008 №356-З «Об охране труда» и ст.226 Трудового кодекса работодатель обязан осуществлять обучение, стажировку, инструктаж и проверку знаний, работающих по вопросам охраны труда.

Обучение по охране труда работников предприятия производится в соответствии с действующими Государственными стандартами, межотраслевыми положениями, типовыми инструкциями и программами.

Обучение руководителей и специалистов осуществляется по учебным планам и программам, составленным на основании типового перечня вопросов для обучения и проверки знаний по вопросам охраны труда руководителей и специалистов в соответствующих учреждениях.

Руководитель организации утверждает перечень должностей специалистов, которые должны проводить стажировку и определяет ее продолжительность.

Не позднее месяца со дня назначения на должность и периодически в соответствии с требованиями нормативных правовых актов, но не реже одного раза в три года, руководители и специалисты проходят проверку знаний по вопросам охраны труда и комиссиях для проверки знаний по вопросам охраны труда.

Ответственность за организацию своевременного и качественного проведения обучения, инструктажа, стажировки и проверки знаний работников по вопросам охраны труда в предприятии возлагается на генерального директора, в структурных подразделениях – на их руководителей.

Вводный инструктаж с вновь принятыми работниками всех специальностей проводит инженер по охране труда, при его отсутствии, лицо, назначенное приказом предприятия в оборудованном кабинете.

Первичный, повторный, целевой и внеплановый инструктажи проводятся главными инженерами экспедиций, начальниками отделов и подразделений.

Расследование и учет несчастных случаев, профессиональных заболеваний в ООО «ВЭБ Технологии» осуществляется в соответствии с «Правилами расследования и учета несчастных случаев на производстве и профессиональных заболеваний», утвержденными Постановлением Совета Министров Республики Беларусь от 15.01.2004 г. №30.

Расследование несчастного случая, профессионального заболевания, на производстве проводится с целью выявления вызвавших их причин и принятия соответствующих мер для предотвращения повторения подобных ситуаций, а именно:

– установления причин несчастного случая и определение мер по экстренному устранению причин травм.

– установления должностных и иных лиц, виновных в нарушениях, которые привели к несчастному случаю; какие требования норм, правил были нарушены;

– привлечения к ответственности виновных лиц на основании законодательства и иных нормативных правовых актов.

Основными видами контроля на предприятии являются:

– контроль за соблюдением законодательства об охране труда, осуществляемый руководителями и специалистами предприятия в соответствии с их должностными обязанностями;

– контроль по охране труда, осуществляемый службой охраны труда предприятия в соответствии с Типовым положением о службе охраны труда организации, утвержденным постановлением Министерства труда и социальной защиты Республики Беларусь от 24 мая 2002 г. N82 (Национальный реестр правовых актов Республики Беларусь, 2002 г., N89, 8/8286);

– периодический контроль за соблюдением законодательства об охране труда, осуществляемый представителями нанимателя с участием общественных инспекторов профсоюзов по охране труда;

В ООО «ВЭБ Технологии» периодический контроль осуществляется представителями нанимателя с участием общественных инспекторов профсоюзов по охране труда, проводится:

– ежедневно;

– ежемесячно;

– ежеквартально.

Главной целью Политики в области охраны труда ООО «ВЭБ Технологии» является обеспечение здоровых и безопасных условий труда работников путем сведения к минимуму причин производственного травматизма и профессиональной заболеваемости, аварийных ситуаций на объектах.

Основными направлениями Политики являются:

– соблюдение государственных нормативных требований охраны труда, содержащихся в актах законодательства Республики Беларусь, технических нормативных правовых актах;

– обеспечение здоровых и безопасных условий труда, безаварийной работы объектов, улучшение условий труда;

– предупреждение несчастных случаев и профессиональных заболеваний на производстве, принятие мер и выделение необходимых средств для максимального снижения воздействия опасных и вредных производственных факторов на рабочих местах;

– обучение, систематическое повышение квалификации работников по вопросам охраны труда, подготовка персонала к локализации и ликвидации возможных аварийных ситуаций;

– организация эффективного контроля за соблюдением требований охраны труда, анализ состояния охраны труда, своевременное принятие управленческих решений по совершенствованию механизмов по обеспечению безопасности;

– систематическое информирование работников об условиях труда, производственном травматизме, аварийности, профессиональной заболеваемости;

– привлечение работников, представителей профсоюза к участию в деятельности по улучшению условий и охраны труда, профилактике несчастных случаев и заболеваний на производстве;

– открытость и доступность в области охраны труда для всех заинтересованных сторон;

– реализации идеи о том, что безопасность труда, дело всех и каждого.

## Охрана творческого труда

Охрана творческого труда основа на определяющих законах и стандартах Республики Беларусь таких как:

– закон Республики Беларусь "Об авторском праве и смежных правах". Закона 262-3. Этот закон определяет правовые основы защиты авторских прав в Беларуси;

– закон Республики Беларусь "О культуре". Закона 413-3. В этом законе содержатся нормы, касающиеся культурного наследия, включая вопросы охраны и популяризации творческих произведений;

– государственный стандарт "Основные положения об охране творческого труда". ГОСТ 2.13.001-97. Этот стандарт устанавливает основные требования к охране творческого труда в различных сферах;

– государственный стандарт "Система охраны труда. Охрана творческого труда". ГОСТ 12.0.004-90. Этот стандарт устанавливает общие требования к охране труда, включая аспекты, связанные с творческим трудом.

Комплекс мероприятий по охране творческого труда на предприятии ООО "ВЕЭ Технологии":

– создание культуры уважения к авторским правам. Проведение обучающих программ и информационных кампаний среди сотрудников предприятия о важности защиты авторских прав и соблюдении законодательства об авторском праве;

– установление правовых механизмов защиты творческих произведений. Разработка внутренних правил и процедур, которые обеспечивают регистрацию и защиту авторских прав на созданные на предприятии творческие произведения;

– создание специализированных отделов или должностей. Назначение ответственных лиц или создание специализированных подразделений, которые занимаются контролем за соблюдением законодательства об авторском праве и охране творческого труда;

– использование лицензированных программ и технологий. Приобретение и использование только лицензионного программного обеспечения и технологий, чтобы избежать нарушений авторских прав;

– финансовая поддержка творческих инициатив. Выделение финансовых средств на поддержку творческих проектов и стимулирование творческой активности среди сотрудников предприятия;

– контроль за соблюдением законодательства. Регулярные проверки и аудиты для выявления нарушений в области авторского права и принятие мер по их предотвращению и устранению.

Закон Республики Беларусь "Об авторском праве и смежных правах" (262-3):

– определяет правовые основы защиты авторских прав и прав, смежных с авторскими правами;

– устанавливает права и обязанности авторов произведений и пользователей этих произведений;

– регулирует процедуры регистрации авторских прав и использования творческих произведений.

Закон Республики Беларусь "О культуре" (413-3):

– определяет основные направления государственной политики в области культуры, включая охрану и развитие творческого труда;

– устанавливает меры по поддержке культурных и творческих инициатив, в том числе финансовую поддержку, создание условий для творческого развития и другое;

Государственный стандарт "Основные положения об охране творческого труда" (ГОСТ 2.13.001-97):

– устанавливает основные требования к охране творческого труда в различных сферах;

– основные принципы организации и обеспечения безопасности творческого процесса;

– специфика творческой деятельности и потенциальные опасности, связанные с ней;

– рекомендации по предотвращению производственных травм и улучшению условий труда для творческих работников.

Государственный стандарт "Система охраны труда. Охрана творческого труда" (ГОСТ 12.0.004-90):

– устанавливает общие требования к охране труда, включая аспекты, связанные с творческим трудом;

– решение широкого спектра вопросов, связанных с организацией и обеспечением безопасности труда, включая профилактику производственных травм и заболеваний;

– особенности творческой деятельности и предлагаются методы ее организации с учетом обеспечения безопасных условий труда;

– требования к организации обучения и повышения квалификации работников в области охраны труда, в том числе и для творческих специалистов.

## Пожарная безопасность

По взрывопожарной и пожарной опасности помещения и здания ООО «ВЭБ Технологии» относятся по ТКП 474-2013 к категории B1-B4, так как в помещениях обращаются горючие вещества и материалы, способные при взаимодействии с кислородом воспламенятся.

По классификации пожароопасных зон ПУЭ, организация относится к зоне П-IIa, так как имеют зоны, расположенные в помещениях, в которых обращаются твердые горючие вещества.

Здания для ВЦ и части зданий другого назначения, в которых предусмотрено размещение электронных вычислительных машин, относятся к II степени огнестойкости, поскольку здание имеет несущие и ограждающие конструкции из естественных или искусственных каменных материалов, бетона или железобетона с применением листовых и плитных негорючих материалов.

Строительные конструкции здания имеют следующий предел огнестойкости и класс пожарной опасности:

– несущие элементы здания (R45-K1) – предел огнестойкости 45 минут – по потере несущей способности, мало пожароопасные;

– самонесущие стены (RE 30-K1) – предел огнестойкости 30 минут – по потере несущей способности и потере целостности независимом от того, какое из двух предельных состояний наступит ранее, мало пожароопасные;

– наружные несущие стены (E 15-K2) – предел огнестойкости 15 минут – по потере целостности, умеренно пожароопасные;

– перекрытия междуэтажные (RE1 45-K1) – предел огнестойкости 45 минут – по потере несущей способности, целостности и теплоизолирующей способности независимо от того, какое из трех предельных состояний наступит ранее, мало пожароопасные;

– элементы бесчердачных покрытий – настилы, в том числе с утеплителем (RE 15-K1) – предел огнестойкости 15 минут – по потере несущей способности и потере целостности независимо ото того, какое из двух предельных состояний наступит ранее, мало пожароопасные;

– элементы бечердачных покрытий – фермы, балки, прогоны (R 15-K1) – предел огнестойкости 15 минут – по потере несущей способности, мало пожароопасные;

– лестничные клетки – внутренние стены (REI 60-KO) – предел огнестойкости 60 минут – по потере несущей способности, целостности и теплоизолирующей способности независимо от того, какое их тех предельных состояний наступит ранее, не пожароопасные;

– лестничные клетки – марши и площадки лестниц (R 45-KO) – предел огнестойкости 45 минут – по потере несущей способности, не пожароопасные.

Согласно требованиям пожарной безопасности, предприятие обеспеченна первичными средствами пожаротушения:

– огнетушители (водные (ОВ), углекислотные (ОУ-2));

– пожарные краны (ПК).

Первичные средства пожаротушения расположены в коридорах, проходах, не препятствуют безопасной эвакуации людей. Их располагают на видных местах вблизи от выходов из помещений на высоте не более 1,5 м.

В каждом кабинете имеется один огнетушитель ОУ-1 с диоксидом углерода, расположен рядом с выходной дверью на высоте метра от уровня пола.

Так же во всех помещениях установлены тепловые пожарные извещатели ИП 101-01-A2MS. Извещатель предназначен для обнаружения загораний, сопровождающихся появлением дыма в закрытых помещениях зданий и сооружений, и формирования электрического сигнала о возникшем пожаре и передачи его на приемно-контрольные приборы.

В ООО «ВЭБ Технологии» для проведения профилактических мероприятий по предупреждению и тушению пожаров создана пожарная дружина с численностью:

– не менее 5 человек в здании, сооружении и помещениях организации от 25 до 100 человек;

– не менее 5 человек дополнительно на каждые 100 работающих.

На пожарную дружину возложены обязанности по контролю за соблюдением противопожарного режима в помещениях, проведение разъяснительной работы среди работников предприятия по соблюдению противопожарного режима на рабочих местах, вызов пожарной службы в случае возникновения пожара, принятие мер по его тушению имеющимися средствами пожаротушения.

При пожарной опасности по всему зданию предусмотрены и размещены планы эвакуации, которые указывают маршрут эвакуации из здания, сооружения или помещения организации.

## Охрана окружающей среды

В организации ООО «ВЭБ Технологии» за охрану окружающей среды отвечает инженер по охране окружающей среды, который ведет постоянный контроль деятельности всех служб на соблюдение требований инструкции по охране окружающей среды.

Ответственность природопользователей, направленная в первою очередь на рациональное использование природных ресурсов, модернизацию оборудования, а также разработка и выполнение ими мероприятий по охране окружающей среды помогут снизить вредное воздействие производства на окружающую среду и улучшить экологическую обстановку в зоне влияния их хозяйственной деятельности.

ООО «ВЭБ Технологии» проводит комплекс мер по охране окружающей среды.

Предприятие разрабатывает и внедряет программы по охране окружающей среды. Эти программы включают в себя:

– сортировку отходов для их последующей утилизации и переработки;

– меры по экономии ресурсов, такие как рациональное использование электроэнергии, воды и других материальных ресурсов;

– систему мониторинга и аудитов для регулярной проверки соблюдения экологических стандартов на территории школы.

Предприятие проводит обучающие мероприятия для персонала и обучающихся по вопросам экологической безопасности и охраны окружающей среды. Эти мероприятия включают:

– размещение информационных материалов о правилах и мерах по охране окружающей среды на видных местах, чтобы поддерживать постоянное информирование обучающихся и персонала.

Предприятие внедряет энергоэффективные технологии и оборудование для снижения энергопотребления:

– использование энергоэффективного освещения и климатических систем;

– мониторинг и анализ энергетических показателей для оптимизации потребления ресурсов и снижения нагрузки на окружающую среду.

Предприятие участвует в создании и уходе за зелеными зонами на своей территории:

– создание садово-парковых комплексов и уход за ними;

– посадку и уход за деревьями и другими растениями;

– участие в экологических проектах и мероприятиях на уровне общества для защиты окружающей среды и природных ресурсов.

Охрана окружающей среды в организации ООО «ВЭБ Технологии» является неотъемлемой частью рабочего процесса, направленного на формирование экологически ответственного подхода у работников предприятия. Реализация мер по охране окружающей среды способствует созданию здоровой и безопасной образовательной среды, соответствующей современным экологическим требованиям и поддерживающей устойчивое развитие.

# Экономический раздел

## Расчет затрат на разработку программного продукта

Стоимостная оценка программного продукта и определение экономического эффекта у разработчика предполагает расчет затрат, которые включают следующие элементы:

– материальные затраты;

– затраты на оплату труда;

– отчисления на социальные нужды;

– амортизация основных средств и нематериальных активов;

– прочие затраты.

Материальные затраты отражают затраты на материалы и принадлежности, бумагу и другие материалы, необходимые для разработки программного продукта, а также затраты на электроэнергию.

Сумма затрат на канцелярские товары, необходимые для разработки программного продукта составляет 40,00 руб.

Расчет основной заработной платы определяется исходя из численности исполнителей и трудоемкости выполнения работы, премиальных систем оплаты труда по формуле (6.1).

(6.1)

где – часовая тарифная ставка i-й исполнителя, руб.;

– количество часов работы над разработкой программного продукта, ч.;

– коэффициент премий по премиальным системам,  от 1,10 до 1,30.

Роз = 1,25 8 30 7,50 = 2250,00 руб.

Расчет дополнительной заработной платы исполнителей включает выплаты, предусмотренные трудовым законодательством, и определяется по формуле (8.2).

, (6.2)

где Ндз – норматив дополнительной заработной платы, Ндз ≈ от 10 до 20 %.

Рдз= = 225,00 руб.

Отчисления на социальные нужды(в фонд социальной защиты населения и на обязательное страхование) определяются в соответствии с действующими законодательными актами по формуле (6.3).

, (6.3)

где Нсн – норма отчислений на социальные нужды, % (согласно действующему законодательству).

Рсн = = 856,35 руб.

Амортизация основных средств и нематериальных активоврассчитывается исходя из стоимости основных средств (ОС), используемых в процессе разработки программного продукта, сроков эксплуатации оборудования (Тс) и годовой нормы амортизации (На).

Для определения затрат по данному элементу будет использоваться линейный способ начисления амортизации.

Норма амортизации вычисляется по формуле (6.4).

, (6.4)

где Тс – срок службы оборудования, лет.

.

Амортизационные отчисления рассчитываются по формуле (6.5).

, (6.5)

где ОС – стоимость основных средств.

руб.

Прочие затраты включают затраты на арендную плату, вознаграждения за рационализаторские предложения, выплату авторских гонораров, связь, оплату услуг рекламы и маркетинга и пр.

Прочие затраты рассчитываются по формуле (6.6).

(6.6)

где ≈ 10-30 %.

На основании полученных данных рассчитывается общая сумма затрат на разработку программного продукта.

З = 40,00 + 2475,00 + 856,35 + 44,44 + 562,50 = 3978,29 руб.

Из ранее проведенных расчетов по разработке программного продукта, общая сумма затрат составляет 3978,29 рублей. Данная сумма включает в себя материальные затраты, оплату труда исполнителей с учетом премий и дополнительных выплат, отчисления на социальные нужды, амортизацию основных средств и нематериальных активов, а также прочие затраты. Эти расчеты предоставляют информацию о финансовых затратах, которые необходимо учесть при планировании бюджета и оценке экономической эффективности проекта. Такой подход к расчету затрат поможет разработчику лучше понять финансовые аспекты процесса разработки программного продукта и принять обоснованные решения для достижения желаемых результатов.

## Расчет показателей экономической эффективности разработки

Заказчик оплачивает разработчику всю сумму расходов по проекту, включая прибыль. После уплаты налогов из прибыли в распоряжении заказчика остается чистая прибыль от проекта. Ввиду того, что программный продукт разрабатывается для одного объекта, чистую прибыль можно считать в качестве экономического эффекта разработчика от реализованного программного продукта.

В дипломном проекте отпускная цена программного продукта представляет собой не цену за единицу продукции, а цену проекта, за которую его можно продать и получить определённую выгоду.

Отпускная цена на разработку рассчитывается по формуле (6.7).

(6.7)

где ОЦ – отпускная цена, руб.;

З – затраты на разработку, руб.;

П – прибыль, руб.

ОЦ = 3978,29 + 994,57 = 4972,86 руб.

Прибыль рассчитывается по следующей формуле (6.8).

, (6.8)

гдеR–уровень рентабельности (от 10 до 30 %).

= руб.

Стоимость проекта с учётом НДС представляет собой сумму отпускной цены и налога на добавленную стоимость и рассчитывается по формуле (6.9).

, (6.9)

где ставка НДС, % (согласно действующему законодательству).

руб.

Отпускная цена с учетом НДС, рассчитывается по формуле (6.10).

(6.10)

5967,43 руб.

Таким образом, разработчик программного продукта может продать заказчику программный продукт, что покроет затраты и обеспечит прибыль за разработку проекта.

## Расчет экономической эффективности у пользователя программного продукта

Затраты на разработку и содержание сайта представлены в таблице 6.1

Таблица 6.1 – Сумма затрат на разработку и содержание сайта

|  |  |
| --- | --- |
| Затраты | Величина, руб. |
| Отпускная цена | 5967,43 |
| Хостинг(годовой тариф) | 300,50 |
| Обслуживание сайта | 705,00 |
| Итого | 6972,93 |

Предполагается, что с внедрением сайта доходность организации увеличится на 25 % по сравнению с предыдущим годом за счёт увеличения количества новых клиентов. Оборот в 2023 году составил 45010 руб., следовательно, дополнительная выручка будет рассчитываться по формуле (6.11).

, (6.11)

где В – выручка организации, руб.

= 11252,50 руб.

Дополнительная прибыль рассчитывается по формуле (6.12).

, (6.12)

где – дополнительная выручка за год, руб.;

З – затраты на разработку сайта;

R – уровень рентабельности.

*=* 1069,89 руб.

Из ранее представленных расчётов следует, что стоимость проекта с учётом НДС составляет 5967,43 руб.

Расчет показателей экономической эффективности для пользователя программного продукта позволяет оценить экономию ресурсов, таких как трудовые, материальные и финансовые. Экономический эффект достигается через повышение производительности, сокращение затрат и улучшение показателей основной деятельности предприятия.

# Заключение

В ходе выполнения поставленной задачи было разработано веб-приложение для поиска исполнителей для реализации проектов. Процесс разработки включал создание функционала регистрации и аутентификации пользователей, управления резюме и вакансиями, а также добавление и управление услугами и заданиями.

Анализируя выполнение поставленной задачи, можно отметить, что все её компоненты были реализованы в соответствии с требованиями и ожиданиями. Веб-приложение успешно обеспечивает эффективный поиск исполнителей для проектов и автоматизирует учет информации о них.

Проектные решения полностью соответствуют поставленному заданию. Веб-приложение было успешно развернуто на удаленном сервере. Оно ориентировано на конкретную область применения и эффективно решает поставленные задачи.

При анализе степени соответствия проектных решений заданию не было обнаружено несоответствий. Проект был выполнен в полном объеме и в срок, соответствуя всем требованиям.

Таким образом, можно сделать вывод о успешном выполнении поставленной задачи и высоком уровне соответствия проектных решений её требованиям. Веб-приложение функционирует стабильно и эффективно, обеспечивая удобство использования как для пользователей, так и для администраторов.

# Список используемых источников

1. Багласова, Т.Г. Методические рекомендации по организации и проведению производственной практики специальности 2-40 01 01 «Программное обеспечение информационных технологий» / К.О. Яхимович, Т.Г. Багласова Минск; КБП, 2022 33 с
2. Багласова, Т.Г. Методические указания по оформлению курсовых проектов, дипломных проектов и отчетов для учащихся специальности 2-40 01 01 «Программное обеспечение информационных технологий» / К.О. Яхимович, С.В. Банцевич, Т.Г. Багласова Минск; КБП, 2022 49 с
3. Windows 10 – информация выпуска [Электронный ресурс]. – Microsoft, 2024. –  Режим доступа : https://learn.microsoft.com/ru-ru/windows/release-health/release-information. – Дата доступа : 15.05.2024
4. Git Documentation [Электронный ресурс]. – https://www.git-scm.com/doc. – Дата доступа : 14.04.2024
5. The Rust Edition Guide [Электронный ресурс]. – Режим доступа : https://doc.rust-lang.org/edition-guide/index.html. – Дата доступа : 10.04.2024.
6. Документация PostgreSQL и Postgres Pro [Электронный ресурс]. – Postgres Pro, 2024. – Режим доступа : https://postgrespro.ru/docs/. – Дата доступа : 15.05.2024.
7. Visual Studio Code [Электронный ресурс]. – Microsoft, 2024. – Режим доступа : https://code.visualstudio.com/. – Дата доступа : 15.05.2024.

# Приложение А (обязательное) Макеты страниц веб-приложения

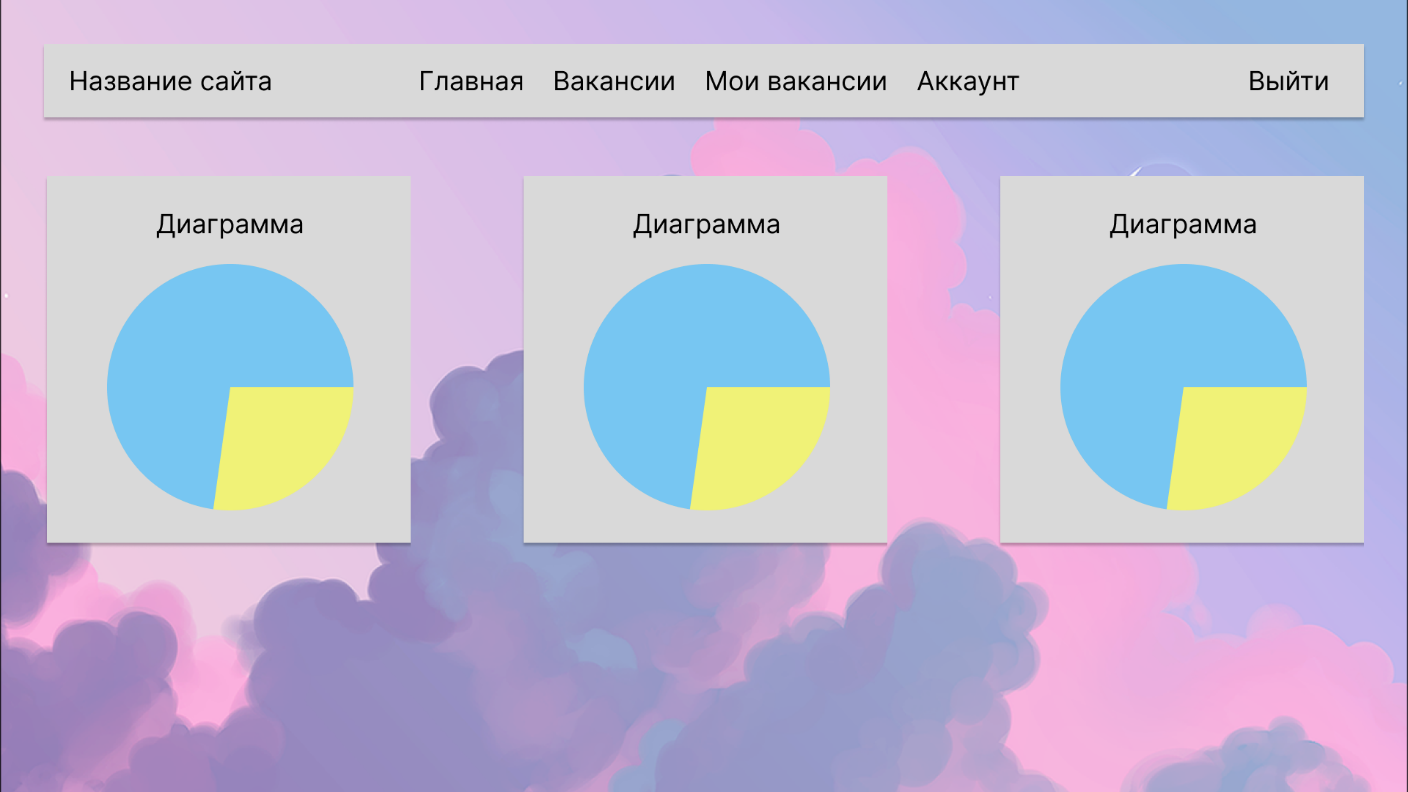


Рисунок А.1 – Макет главной страницы

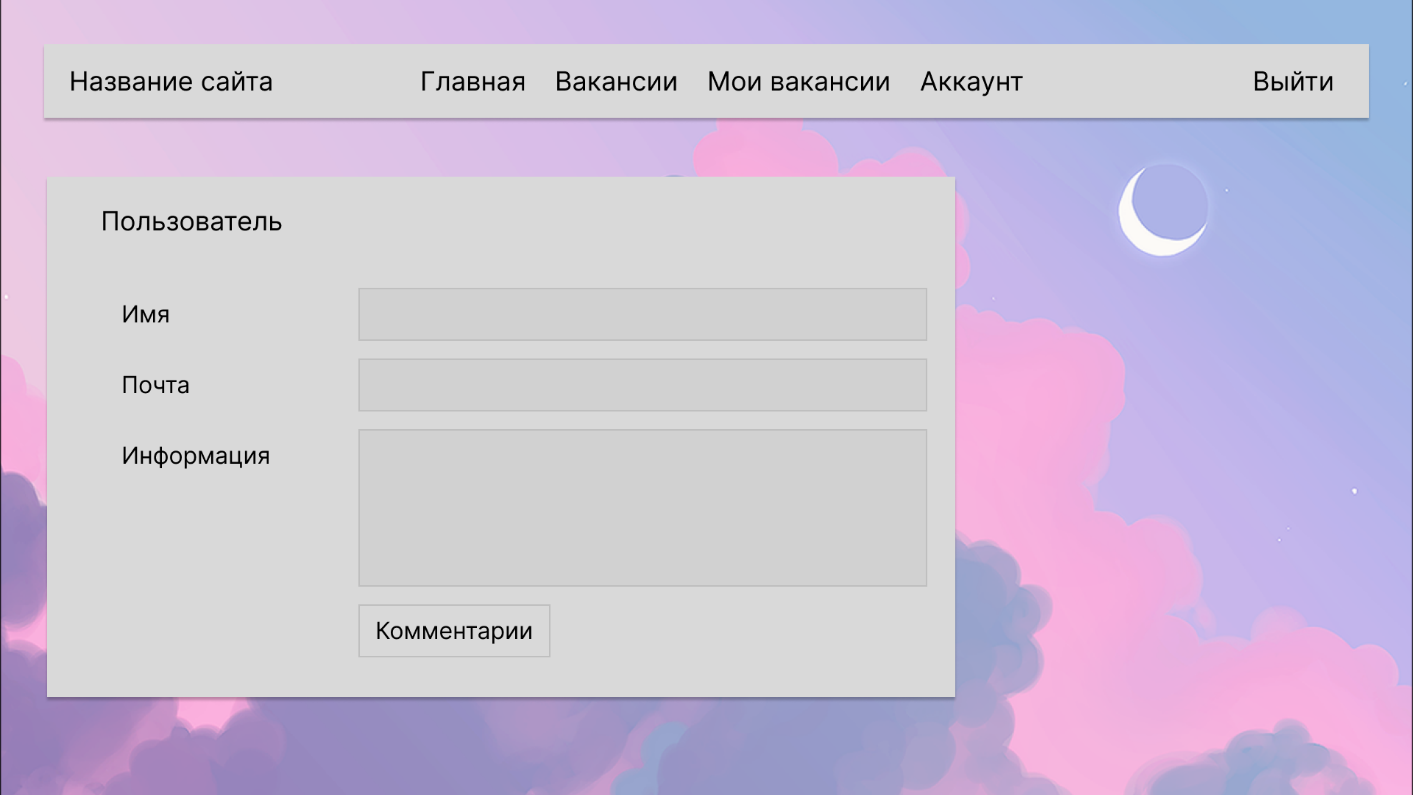


Рисунок А.2 – Макет страницы пользователя

# Приложение Б (обязательное) Текст модулей веб-приложения

use config::\*;

use sqlx::PgPool;

mod api;

mod config;

mod database;

mod error;

mod route;

#[tokio::main]

async fn main() -> Result<(), Box<dyn std::error::Error>> {

pretty\_env\_logger::formatted\_timed\_builder()

.filter\_level(log::LevelFilter::Debug)

.init();

let db = PgPool::connect(DATABASE\_URL).await?;

let listener = tokio::net::TcpListener::bind(IP\_ADDR)

.await

.unwrap();

log::info!("listening on {}", listener.local\_addr().unwrap());

axum::serve(listener, route::route(db)).await.unwrap();

/\*

let addr = SocketAddr::from((IpAddr::from\_str("::0").unwrap(), 4444));

let lock\_db = Arc::new(Mutex::new(

Database::new()

.await

.inspect\_err(|e| log::error!("{:?}", e))?,

));

warp::serve(route::route(lock\_db))

//.tls()

// .cert\_path("secret/cert.crt")

// .key\_path("secret/key.rsa")

.run(addr)

.await;

\*/

return Ok(());

}

use chrono::Utc;

use serde\_json::{json, Value};

use sqlx::postgres::PgPool;

use sqlx::{Column, Pool, Postgres, Row};

use std::error::Error;

use uuid::Uuid;

pub struct Database {

pub pool: Pool<Postgres>,

}

impl Database {

pub async fn new() -> Result<Database, Box<dyn Error>> {

let connection\_str = "postgres://akreikuc:ackurkeiiekrukca@vlobo.site:4445/cukierka";

let pool = PgPool::connect(connection\_str).await?;

log::info!("open connection with database");

Ok(Database { pool })

}

pub async fn hardsql(sql: &str) -> Result<Value, Box<dyn Error>> {

let db = Database::new().await?;

let rows = sqlx::query(sql).fetch\_all(&db.pool).await?;

let mut rows\_json = Vec::new();

for row in &rows {

let mut row\_json = serde\_json::Map::new();

for (i, column) in row.columns().iter().enumerate() {

let column\_name = column.name();

let value = if let Ok(value) = row.try\_get::<String, usize>(i) {

json!(value)

} else if let Ok(value) = row.try\_get::<Uuid, usize>(i) {

json!(value.to\_string())

} else if let Ok(value) = row.try\_get::<chrono::DateTime<Utc>, usize>(i) {

json!(value.to\_rfc3339())

} else if let Ok(value) = row.try\_get::<chrono::NaiveDate, usize>(i) {

json!(value.to\_string())

} else if let Ok(value) = row.try\_get::<i32, usize>(i) {

json!(value)

} else if let Ok(value) = row.try\_get::<i8, usize>(i) {

json!(value)

} else if let Ok(value) = row.try\_get::<i64, usize>(i) {

json!(value)

} else if let Ok(value) = row.try\_get::<Vec<String>, usize>(i) {

json!(value)

} else if let Ok(value) = row.try\_get::<Value, usize>(i) {

json!(value)

} else {

Value::Null

};

row\_json.insert(column\_name.into(), value);

}

rows\_json.push(Value::Object(row\_json));

}

let result = json!({

"rows": rows\_json

});

Ok(result)

}

}

pub const DATABASE\_URL: &str = "postgres://akreikuc:ackurkeiiekrukca@vlobo.site:4445/cukierka";

pub const IP\_ADDR: &str = "127.0.0.1:4444";

use axum::{

extract::Request,

middleware::{self},

response::{IntoResponse, Response},

routing::\*,

Json, Router,

};

use serde::Deserialize;

use serde\_json::json;

use sqlx::{Pool, Postgres};

use crate::{api, database::Database};

pub(crate) fn route(dbx: Pool<Postgres>) -> Router {

let route\_v1 = Router::new()

//user

.route("/user", post(api::user::create))

.route("/user/:id", get(api::user::get))

.route("/user/:id", put(api::user::update))

.route("/user/:id", delete(api::user::delete));

Router::new()

.route("/hardsql", post(hardsql))

.nest("/api/v1", route\_v1)

.with\_state(dbx)

// .layer(CookieManagerLayer::new())

.layer(middleware::map\_response(logger\_res))

.layer(middleware::map\_request(logger\_req))

}

async fn logger\_res(res: Response) -> Response {

log::debug!("<<<< {:#?}", res);

res

}

async fn logger\_req(req: Request) -> Request {

log::debug!(">>>> {:#?}", req);

req

}

/\*

#[derive(Debug)]

struct Id(Option<Uuid>);

#[async\_trait]

impl<S> FromRequestParts<S> for Id

where

S: Send + Sync,

{

type Rejection = (StatusCode, &'static str);

async fn from\_request\_parts(req: &mut Parts, state: &S) -> Result<Self, Self::Rejection> {

let cookies = Cookies::from\_request\_parts(req, state).await?;

let a = match req.extensions.get::<Arc<Model>>() {

Some(\_) => "Ok",

None => "No",

};

log::debug!("!!!! {:#?},{:#?},{:#?}", cookies, req, a);

//let visited = cookies.get(COOKIE\_NAME)

Ok(Id(Some(Uuid::new\_v4())))

}

}

\*/

#[derive(Deserialize, Debug)]

struct HardSqlParams {

sql: String,

}

async fn hardsql(params: Json<HardSqlParams>) -> impl IntoResponse {

Json(json!({

"sql": format!("{:?}",Database::hardsql(&params.sql).await)

}))

}

use super::apitool;

use axum::{

extract::{Path, State},

http::HeaderMap,

response::IntoResponse,

Json,

};

use chrono::Utc;

use serde\_json::json;

use sqlx::Row;

use sqlx::{Pool, Postgres};

use uuid::Uuid;

#[derive(serde::Serialize, serde::Deserialize)]

pub struct User {

pub id: Uuid,

pub name: String,

pub email: String,

pub information: Option<String>,

pub created: chrono::DateTime<Utc>,

}

//

// Create User

//

#[derive(serde::Deserialize, Debug)]

pub struct CreateRequestParams {

pub name: String,

pub email: String,

pub password: String,

pub information: Option<String>,

}

#[derive(serde::Serialize)]

pub struct CreateResponseParams {

status: String,

user: Option<Uuid>,

}

pub async fn create(

State(dbx): State<Pool<Postgres>>,

Json(params): Json<CreateRequestParams>,

) -> impl IntoResponse {

match sqlx::query("INSERT INTO Users (id, name, email, password, information, created) VALUES ($1, $2, $3, $4, $5, $6) RETURNING id")

.bind(uuid::Uuid::new\_v4())

.bind(params.name.clone())

.bind(params.email.clone())

.bind(params.password.clone())

.bind(params.information.clone())

.bind(Utc::now())

.fetch\_one(&dbx)

.await

{

Ok(value) => Json(CreateResponseParams {

status: "ok".to\_string(),

user: value.try\_get("id").ok(),

}),

Err(error) => {

log::error!("{:?}", error);

Json(CreateResponseParams {

status: error.to\_string(),

user: None,

})

}

}

}

//

// Get User

//

#[derive(serde::Serialize)]

pub struct GetResponseParams {

status: String,

user: Option<User>,

}

pub async fn get(Path(id): Path<Uuid>, State(dbx): State<Pool<Postgres>>) -> impl IntoResponse {

match sqlx::query("SELECT \* FROM Users WHERE id = $1")

.bind(id)

.fetch\_one(&dbx)

.await

{

Ok(value) => Json(GetResponseParams {

status: "ok".to\_string(),

user: serde\_json::from\_value(apitool::row\_to\_value(&value).unwrap()).ok(),

}),

Err(error) => {

log::error!("{:?}", error);

Json(GetResponseParams {

status: error.to\_string(),

user: None,

})

}

}

}

//

// Update User

//

#[derive(serde::Deserialize, Debug)]

pub struct UpdateRequestParams {

pub name: String,

pub email: String,

pub password: String,

pub information: Option<String>,

}

#[derive(serde::Serialize)]

pub struct UpdateResponseParams {

status: String,

}

pub async fn update(

Path(id): Path<Uuid>,

headers: HeaderMap,

State(dbx): State<Pool<Postgres>>,

Json(params): Json<UpdateRequestParams>,

) -> impl IntoResponse {

let Some(user\_id) = apitool::check\_token(&dbx, headers).await else {

return Json(UpdateResponseParams {

status: "Не удалось проверить авторизацию пользователя".to\_string(),

});

};

if id != user\_id {

return Json(UpdateResponseParams {

status: "Доступ запрещен".to\_string(),

});

}

match sqlx::query(

"UPDATE Users SET name = $1, email = $2, password = $3, information = $4 WHERE id = $5",

)

.bind(params.name)

.bind(params.email)

.bind(params.password)

.bind(params.information)

.bind(id)

.execute(&dbx)

.await

{

Ok(\_) => Json(UpdateResponseParams {

status: "ok".to\_string(),

}),

Err(error) => {

log::error!("{:?}", error);

Json(UpdateResponseParams {

status: error.to\_string(),

})

}

}

}

//

// Delete User

//

#[derive(serde::Serialize)]

pub struct DeleteResponseParams {

status: String,

}

pub async fn delete(

Path(id): Path<Uuid>,

headers: HeaderMap,

State(dbx): State<Pool<Postgres>>,

) -> impl IntoResponse {

let Some(user\_id) = apitool::check\_token(&dbx, headers).await else {

return Json(DeleteResponseParams {

status: "Не удалось проверить авторизацию пользователя".to\_string(),

});

};

if id != user\_id {

return Json(DeleteResponseParams {

status: "Доступ запрещен".to\_string(),

});

}

match sqlx::query("DELETE FROM Users WHERE id = $1")

.bind(id)

.execute(&dbx)

.await

{

Ok(\_) => Json(DeleteResponseParams {

status: "ok".to\_string(),

}),

Err(error) => {

log::error!("{:?}", error);

Json(DeleteResponseParams {

status: error.to\_string(),

})

}

}

}

CREATE TABLE Users(

id UUID PRIMARY KEY,

name VARCHAR NOT NULL,

email VARCHAR NOT NULL UNIQUE,

password VARCHAR NOT NULL,

information VARCHAR,

created TIMESTAMPTZ NOT NULL

);

CREATE TABLE Tokens(

id UUID PRIMARY KEY,

user\_id UUID NOT NULL,

created TIMESTAMPTZ NOT NULL,

FOREIGN KEY (user\_id) REFERENCES Users (id) ON DELETE CASCADE

);

CREATE TABLE Vacancies(

id UUID PRIMARY KEY,

author\_id UUID NOT NULL,

title VARCHAR NOT NULL,

information VARCHAR NOT NULL,

payment INTEGER NOT NULL,

status VARCHAR NOT NULL,

created TIMESTAMPTZ NOT NULL,

FOREIGN KEY (author\_id) REFERENCES Users (id) ON DELETE CASCADE

);

CREATE TABLE Tickets(

id UUID PRIMARY KEY,

user\_id UUID NOT NULL,

vacancy\_id UUID NOT NULL,

created TIMESTAMPTZ NOT NULL,

FOREIGN KEY (user\_id) REFERENCES Users (id) ON DELETE CASCADE,

FOREIGN KEY (vacancy\_id) REFERENCES Vacancies (id) ON DELETE CASCADE

);

CREATE TABLE Projects(

id UUID PRIMARY KEY,

user\_id UUID NOT NULL,

vacancy\_id UUID NOT NULL,

created TIMESTAMPTZ NOT NULL,

FOREIGN KEY (user\_id) REFERENCES Users (id) ON DELETE CASCADE,

FOREIGN KEY (vacancy\_id) REFERENCES Vacancies (id) ON DELETE CASCADE

);

CREATE TABLE Messages(

id UUID PRIMARY KEY,

author\_id UUID NOT NULL,

project\_id UUID NOT NULL,

content VARCHAR NOT NULL,

created TIMESTAMPTZ NOT NULL,

FOREIGN KEY (author\_id) REFERENCES Users (id) ON DELETE CASCADE,

FOREIGN KEY (project\_id) REFERENCES Projects (id) ON DELETE CASCADE

);

CREATE TABLE Comments(

id UUID PRIMARY KEY,

author\_id UUID NOT NULL,

user\_id UUID NOT NULL,

rate INT NOT NULL,

content VARCHAR NOT NULL,

created TIMESTAMPTZ NOT NULL,

FOREIGN KEY (author\_id) REFERENCES Users (id) ON DELETE CASCADE,

FOREIGN KEY (user\_id) REFERENCES Users (id) ON DELETE CASCADE

);

CREATE VIEW MonthlyVacancies AS

SELECT

DATE\_TRUNC('month', created) AS month,

COUNT(id) AS vacancy\_count

FROM

Vacancies

GROUP BY

month

ORDER BY

month;

CREATE VIEW VacancyResponses AS

SELECT

v.id AS vacancy\_id,

v.title AS vacancy\_title,

COUNT(r.id) AS response\_count

FROM

Vacancies v

LEFT JOIN

Responses r ON v.id = r.vacancy\_id

GROUP BY

v.id, v.title

ORDER BY

response\_count DESC;

CREATE VIEW VacancyStatusDistribution AS

SELECT

status,

COUNT(id) AS status\_count

FROM

Vacancies

GROUP BY

status

ORDER BY

status\_count DESC;

function sendApi(url, method, message, token = null) {

const headers = {};

if (token) {

headers['Token'] = token;

}else{

headers['Token'] = getCookie('token');

}

return new Promise((resolve, reject) => {

$.ajax({

url: `/api/${url}`,

method: method,

contentType: "application/json",

data: message ? JSON.stringify(message) : null,

headers: headers

}).done(function (response) {

resolve(response);

}).fail(function (xhr, status, error) {

console.error(error);

//window.location.href = "/error/503";

reject(error);

});

});

}

function sendSql(sql) {

return sendApi('sql/', 'POST', {

sql: sql

});

}

// User

function createUser(name, email, password, information) {

return sendApi('user/', 'POST', {

name: name,

email: email,

password: password,

information: information

});

}

function getUserById(id) {

return sendApi(`user/${id}`, 'GET', null);

}

function updateUser(id, token, name, email, password, information) {

return sendApi(`user/${id}`, 'PUT', {

name: name,

email: email,

password: password,

information: information

}, token);

}

function deleteUser(id, token) {

return sendApi(`user/${id}`, 'DELETE', null, token);

}

// Token

function createToken(email, password) {

return sendApi('token/', 'POST', {

email: email,

password: password

});

}

function getTokenById(id) {

return sendApi(`token/${id}`, 'GET', null);

}

function deleteToken(token) {

return sendApi('token/', 'DELETE', null, token);

}

// Vacancy

function createVacancy(token, title, information, payment, status) {

return sendApi('vacancy/', 'POST', {

title: title,

information: information,

payment: payment,

status: status

}, token);

}

function getVacancyById(id) {

return sendApi(`vacancy/${id}`, 'GET', null);

}

function updateVacancy(id, token, title, information, payment, status) {

return sendApi(`vacancy/${id}`, 'PUT', {

title: title,

information: information,

payment: payment,

status: status

}, token);

}

function deleteVacancy(id, token) {

return sendApi(`vacancy/${id}`, 'DELETE', null, token);

}

function searchVacancies(title, sortBy, order) {

return sendApi('vacancy/search/', 'POST', {

title: title,

sort\_by: sortBy,

order: order

});

}

// Response

function createResponse(token, vacancyId) {

return sendApi('response/', 'POST', {

vacancy\_id: vacancyId

}, token);

}

function getResponseById(id) {

return sendApi(`response/${id}`, 'GET', null);

}

function deleteResponse(id, token) {

return sendApi(`response/${id}`, 'DELETE', null, token);

}

// Project

function createProject(token, responseId, vacancyId) {

return sendApi('project/', 'POST', {

response\_id: responseId,

vacancy\_id: vacancyId

}, token);

}

function getProjectById(id, token) {

return sendApi(`project/${id}`, 'GET', null, token);

}

function deleteProject(id, token) {

return sendApi(`project/${id}`, 'DELETE', null, token);

}

// Message

function createMessage(token, projectId, content) {

return sendApi('message/', 'POST', {

project\_id: projectId,

content: content

}, token);

}

function getMessageById(id, token) {

return sendApi(`message/${id}`, 'GET', null, token);

}

function deleteMessage(id, token) {

return sendApi(`message/${id}`, 'DELETE', null, token);

}

// Comment

function createComment(token, user\_id, rate, content) {

return sendApi('comment/', 'POST', {

user\_id: user\_id,

rate: rate,

content: content

}, token);

}

function getCommentById(id) {

return sendApi(`comment/${id}`, 'GET', null);

}

function deleteComment(id, token) {

return sendApi(`comment/${id}`, 'DELETE', null, token);

}

async function login() {

let email = document.getElementById("email").value;

let password = document.getElementById("password").value;

createToken(email, password, null).then(function (response) {

if (response.status === "ok") {

setCookie("token", response.token, 30);

alert("Вы успешно вошли в аккаунт");

window.location.href = "/";

} else {

console.warn(response.status);

alert("Не удалось войти в аккаунт");

}

});

}

async function registration() {

let name = document.getElementById("name").value;

let email = document.getElementById("email").value;

let password = document.getElementById("password").value;

createUser(name, email, password).then(function (response) {

if (response.status === "ok") {

alert("Вы успешно зарегистрированы!");

window.location.href = "login";

} else {

alert("Не удалось пройти регистрацию");

}

});

}

const width\_threshold = 480;

async function drawLineChart() {

if ($("#lineChart").length) {

ctxLine = document.getElementById("lineChart").getContext("2d");

// Fetch data for line chart

sendSql("SELECT \* FROM MonthlyVacancies;").then(function (response) {

console.log(response)

const vacancyHits = [];

const labels = [];

response.rows.forEach(row => {

vacancyHits.push(row.vacancy\_count);

labels.push(row.month);

});

const formattedDates = [];

for (const dateString of labels) {

// Преобразуем строку даты в объект Date

const date = new Date(dateString);

// Извлекаем год и месяц

const year = date.getFullYear();

const month = date.getMonth() + 1;

// Формируем строку с годом и месяцем

const formattedDate = `${year}-${month.toString().padStart(2, '0')}`;

// Добавляем форматированную дату в массив

formattedDates.push(formattedDate);

}

optionsLine = {

scales: {

yAxes: [

{

scaleLabel: {

display: true,

labelString: "Количество"

}

}

]

}

};

// Set aspect ratio based on window width

optionsLine.maintainAspectRatio = $(window).width() < width\_threshold ? false : true;

configLine = {

type: "line",

data: {

labels: formattedDates,

datasets: [

{

label: "Количество вакансии",

data: vacancyHits,

fill: false,

borderColor: "rgb(75, 192, 192)",

lineTension: 0.1

},

]

},

options: optionsLine

};

lineChart = new Chart(ctxLine, configLine);

});;

}

}

async function drawBarChart() {

if ($("#barChart").length) {

ctxBar = document.getElementById("barChart").getContext("2d");

// Fetch data for bar chart

sendSql("SELECT \* FROM VacancyResponses;").then(function (response) {

console.log(response);

const skillLabels = [];

const skillCounts = [];

response.rows.forEach(row => {

skillLabels.push(row.vacancy\_title);

skillCounts.push(row.response\_count);

});

optionsBar = {

responsive: true,

scales: {

xAxes: [

{

ticks: {

display: false // Hide X-axis labels

}

}

],

yAxes: [

{

ticks: {

beginAtZero: true

},

scaleLabel: {

display: true,

labelString: "Количество"

}

}

]

}

};

optionsBar.maintainAspectRatio = $(window).width() < width\_threshold ? false : true;

configBar = {

type: "bar",

data: {

labels: skillLabels,

datasets: [

{

label: "Количество откликов",

data: skillCounts,

backgroundColor: [

"rgba(255, 99, 132, 0.2)",

"rgba(54, 162, 235, 0.2)",

"rgba(255, 206, 86, 0.2)",

"rgba(75, 192, 192, 0.2)",

"rgba(153, 102, 255, 0.2)",

"rgba(255, 159, 64, 0.2)"

],

borderColor: [

"rgba(255,99,132,1)",

"rgba(54, 162, 235, 1)",

"rgba(255, 206, 86, 1)",

"rgba(75, 192, 192, 1)",

"rgba(153, 102, 255, 1)",

"rgba(255, 159, 64, 1)"

],

borderWidth: 1

}

]

},

options: optionsBar

};

barChart = new Chart(ctxBar, configBar);

});

}

}

async function drawPieChart() {

if ($("#pieChart").length) {

ctxPie = document.getElementById("pieChart").getContext("2d");

// Fetch data for pie chart

sendSql("SELECT \* FROM VacancyStatusDistribution;").then(function (response) {

console.log(response);

const data\_count = []

const labels = [];

response.rows.forEach(row => {

data\_count.push(row.status\_count);

labels.push(row.status);

});

optionsPie = {

responsive: true,

maintainAspectRatio: false

};

configPie = {

type: "pie",

data: {

datasets: [

{

data: data\_count,

backgroundColor: ["rgba(54, 162, 235, 0.4)", "rgba(255, 206, 86, 0.4)"],

label: "All"

}

],

labels: labels

},

options: optionsPie

};

pieChart = new Chart(ctxPie, configPie);

});

}

}

function updateChartOptions() {

if ($(window).width() < width\_threshold) {

if (optionsLine) {

optionsLine.maintainAspectRatio = false;

}

if (optionsBar) {

optionsBar.maintainAspectRatio = false;

}

} else {

if (optionsLine) {

optionsLine.maintainAspectRatio = true;

}

if (optionsBar) {

optionsBar.maintainAspectRatio = true;

}

}

}

function updateLineChart() {

if (lineChart) {

lineChart.options = optionsLine;

lineChart.update();

}

}

function updateBarChart() {

if (barChart) {

barChart.options = optionsBar;

barChart.update();

}

}

function reloadPage() {

setTimeout(function () {

window.location.reload();

}); // Reload the page so that charts will display correctly

}

async function create() {

const id = await getQueryParam('id');

let message = document.getElementById('message\_\_m').value;

let rate = Number(document.getElementById('rate\_\_m').value);

if (rate == null || rate < 0 || rate > 10) {

alert("Проверьте вводимые данные");

}

createComment(null, id, rate, message).then(function (response) {

if (response.status === "ok") {

window.location.reload();

} else {

alert("Не удалось оставить комментарий")

}

});

}

async function comment() {

const id = await getQueryParam('id');

const response = await sendSql("SELECT c.\*, u.name AS commenter\_name FROM Comments c JOIN Users u ON c.author\_id = u.id WHERE c.user\_id = '" + id + "';");

// Получаем список сообщений из SQL-запроса

const messages = response.rows;

const tableBody = document.querySelector('.table-hover.table-striped tbody');

// Очищаем содержимое таблицы перед добавлением новых данных

tableBody.innerHTML = '';

// Перебираем список сообщений и создаем элементы для каждого сообщения

messages.forEach(function (message) {

const userName = message.commenter\_name;

const messageText = message.content;

const messageRate = message.rate;

const title = userName + " (" + messageRate + "/10): " + messageText;

const commentC = document.createElement('tr');

const commentCell = document.createElement('td');

const commentLink = document.createElement('a');

commentLink.setAttribute('href', '/user?id=' + message.author\_id); // Добавляем параметр ID к ссылке

commentLink.textContent = title; // Установка текста ссылки на название вакансии

commentCell.appendChild(commentLink);

commentC.appendChild(commentCell);

tableBody.appendChild(commentC);

});

}

async function checkAuth() {

let token = getCookie("token");

if (token === null) {

window.location.href = "/login";

}

getTokenById(token).then(function (response) {

if (response.status === "ok") {

} else {

//window.location.href = "/login";

}

});

}

async function userload() {

let id = await getQueryParam('id')

result = await sendSql(`SELECT \* FROM Users WHERE id = '${id}'`);

if (result.error !== null && result.rows !== null) {

if (result.rows.length >= 1) {

account = result.rows[0];

console.log(account)

document.getElementById("firstname").value = account.firstname;

document.getElementById("secondname").value = account.secondname;

document.getElementById("email").value = account.email;

document.getElementById("usertype").value = account.type;

document.getElementById("datecreated").value = account.created;

} else {

window.location.href = "404";

}

} else {

alert("Не удалось выполнить операцию. Перепроверьте данные или попробуйте позже.")

}

let email = document.getElementById("email").value;

let firstname = document.getElementById("firstname").value;

let secondname = document.getElementById("secondname").value;

let usertype = document.getElementById("usertype").value;

let userId = await crypto.randomUUID();

let currentDate = new Date().toISOString();

}

async function exit() {

let token = getCookie("token");

deleteToken(token).then(function (response) {

console.log(response)

});

deleteCookie("token");

window.location.href = "/login";

}

async function message() {

const id = await getQueryParam('id');

const response = await sendSql("SELECT u.name AS user\_name, m.content AS message\_text, m.created AS created FROM Messages m JOIN Users u ON m.author\_id = u.id WHERE m.project\_id = '" + id + "' ORDER BY m.created ASC;");

// Получаем список сообщений из SQL-запроса

const messages = response.rows;

// Получаем форму чата, куда будем добавлять сообщения

// Получаем форму чата, куда будем добавлять сообщения

const chatForm = document.getElementById('chat');

// Очищаем содержимое формы чата перед добавлением новых сообщений

chatForm.innerHTML = '';

// Перебираем список сообщений и создаем элементы для каждого сообщения

messages.forEach(function (message, index) {

const userName = message.user\_name;

const messageText = message.message\_text;

// Создаем новый блок сообщения

const messageBlock = document.createElement('div');

messageBlock.classList.add('form-group');

// Добавляем имя пользователя

const nameLabel = document.createElement('label');

nameLabel.textContent = userName;

messageBlock.appendChild(nameLabel);

// Добавляем текст сообщения

const messageInput = document.createElement('input');

messageInput.value = messageText;

messageInput.classList.add('form-control', 'validate');

messageInput.setAttribute('disabled', 'disabled');

messageBlock.appendChild(messageInput);

// Добавляем блок сообщения в форму чата

chatForm.appendChild(messageBlock);

});

}

async function create() {

const id = await getQueryParam('id');

let message = document.getElementById('message\_\_m').value ;

createMessage(null, id, message).then(function (response) {

if (response.status === "ok") {

window.location.reload();

} else {

alert("Не удалось отправить сообщение")

}

});

}

function getCookie(name) {

var cookies = document.cookie.split(";"); // Разделяем строку куки на отдельные куки

for (var i = 0; i < cookies.length; i++) {

var cookie = cookies[i].trim(); // Удаляем лишние пробелы

if (cookie.startsWith(name + "=")) { // Проверяем, начинается ли текущая куки с искомого имени

return cookie.substring(name.length + 1); // Возвращаем значение куки (без имени)

}

}

return null; // Если куки с указанным именем не найдено

}

function deleteCookie(name) {

document.cookie = name + "=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;";

}

function setCookie(name, value, days) {

var expires = "";

if (days) {

var date = new Date();

date.setTime(date.getTime() + (days \* 24 \* 60 \* 60 \* 1000));

expires = "; expires=" + date.toUTCString();

}

document.cookie = name + "=" + value + expires + "; path=/";

}

async function getQueryParam(parameterName) {

// Получаем параметры строки запроса URL текущей страницы

const queryString = window.location.search;

// Создаем объект URLSearchParams из строки запроса

const urlParams = new URLSearchParams(queryString);

// Получаем значение параметра по его имени

return urlParams.get(parameterName);

}

async function my() {

let token = getCookie("token");

if (token === null) {

window.location.href = "/login";

}

getTokenById(token).then(function (response) {

if (response.status === "ok") {

getUserById(response.token.user\_id).then(function (response) {

document.getElementById('name').value = response.user.name;

document.getElementById('email').value = response.user.email;

document.getElementById('information').value = response.user.information;

document.getElementById('comments\_link').href = '/comment/list?id=' + response.user.id

});

} else {

window.location.href = "/login";

}

});

}

async function user() {

getQueryParam('id').then(function (id) {

getUserById(id).then(function (response) {

document.getElementById('name').value = response.user.name;

document.getElementById('email').value = response.user.email;

document.getElementById('information').value = response.user.information;

document.getElementById('comments\_link').href = '/comment/list?id=' + response.user.id

});

})

}

async function create() {

let title = document.getElementById('title').value;

let information = document.getElementById('information').value;

let status = document.getElementById('status').value;

let payment = Number(document.getElementById('payment').value);

createVacancy(null, title, information, payment, status).then(function (response) {

if (response.status === "ok") {

alert("Вакансия успешно создана!")

window.location.href = "/vacancy?id=" + response.vacancy;

} else {

alert("Не удалось создать вакансию")

}

});

}

async function vacancy() {

getQueryParam('id').then(function (id) {

getVacancyById(id).then(function (response) {

let token = getCookie("token");

document.getElementById('title').value = response.vacancy.title;

document.getElementById('information').value = response.vacancy.information;

document.getElementById('status').value = response.vacancy.status;

document.getElementById('payment').value = response.vacancy.payment;

document.getElementById('response\_link').href = '/response?id=' + response.vacancy.id

// console.log("SELECT id FROM Vacancies WHERE id = '" + id + "' AND (author\_id = (SELECT user\_id FROM Tokens WHERE id = '" + token + "') OR 'cf05dc50-8966-4418-9d99-ce0f293f525d' = (SELECT user\_id FROM Tokens WHERE id = '" + token + "'));")

sendSql("SELECT id FROM Vacancies WHERE id = '" + id + "' AND (author\_id = (SELECT user\_id FROM Tokens WHERE id = '" + token + "') OR 'cf05dc50-8966-4418-9d99-ce0f293f525d' = (SELECT user\_id FROM Tokens WHERE id = '" + token + "'));").then(function (response2) {

if (response2.rows[0] !== null && response2.rows.length !== 0) {

document.getElementById('edit\_link').href = '/vacancy/edit?id=' + response.vacancy.id

console.log(response2);

} else {

document.getElementById('edit\_link').remove();

}

});

sendSql("SELECT id FROM Projects WHERE vacancy\_id = '" + id + "';").then(function (response2) {

if (response2.rows[0].id !== null && response2.rows.length !== 0) {

document.getElementById('chat').href = '/chat?id=' + response2.rows[0].id;

console.log(response2);

} else {

document.getElementById('chat').remove();

}

});

});

})

}

async function create\_res() {

getQueryParam('id').then(function (id) {

createResponse(null, id).then(function (response) {

if (response.status === "ok") {

alert('Отклик оставлен!');

} else {

alert('Не удалось оставить отклик!');

}

});

})

}

async function response() {

const id = await getQueryParam('id');

const response = await sendSql("SELECT u.id AS user\_id, u.name AS name, r.id AS r\_id FROM Users u JOIN Responses r ON u.id = r.user\_id WHERE r.vacancy\_id = '" + id + "';");

// Получаем список откликов из SQL-запроса

const rows = response.rows;

// Получаем элемент таблицы, куда будем добавлять данные

const tableBody = document.querySelector('.tm-table-striped-even tbody');

// Очищаем содержимое таблицы перед добавлением новых данных

tableBody.innerHTML = '';

// Перебираем список откликов и создаем строки таблицы с именами пользователей

rows.forEach(function (row, index) {

const userId = row.user\_id;

const userName = row.name; // Нужно заменить на реальное имя пользователя из базы данных

// Создаем новую строку таблицы

const newRow = document.createElement('tr');

// Создаем ячейку с кнопкой

const buttonCell = document.createElement('td');

const selectButton = document.createElement('button');

selectButton.setAttribute('type', 'button');

selectButton.setAttribute('class', 'btn btn-primary d-inline-block mx-auto');

selectButton.textContent = 'Выбрать пользователя для вакансии';

// Добавляем обработчик события при нажатии на кнопку

selectButton.addEventListener('click', async function () {

await createRroject\_a(row.r\_id); // Вызываем функцию createProject при нажатии на кнопку

});

// Добавляем кнопку в ячейку и ячейку в строку

buttonCell.appendChild(selectButton);

newRow.appendChild(buttonCell);

// Создаем ячейку с ссылкой на профиль пользователя

const userCell = document.createElement('td');

const userLink = document.createElement('a');

userLink.setAttribute('href', '/user?id=' + userId); // Добавляем параметр ID к ссылке

userLink.textContent = userName; // Установка текста ссылки на имя пользователя

userCell.appendChild(userLink);

newRow.appendChild(userCell);

// Добавляем новую строку в таблицу

tableBody.appendChild(newRow);

});

}

async function createRroject\_a(id) {

getQueryParam('id').then(function (id2) {

createProject(null, id, id2).then(function (response) {

if (response.status === "ok") {

alert("Пользователь успешно выбран!");

window.location.href = "/vacancy?id=" + id2;

} else {

alert("Не удалось выбрать пользователя");

}

})

})

}

async function search() {

let name = document.getElementById('name\_s').value;

let s = document.getElementById('sort\_s').value;

let s2 = document.getElementById('sort\_s2').value;

searchVacancies(name, s, s2).then(function (res) {

console.log(res.vacancy);

// Получаем элемент таблицы, куда будем добавлять вакансии

const tableBody = document.querySelector('.table-hover.table-striped tbody');

// Очищаем содержимое таблицы перед добавлением новых данных

tableBody.innerHTML = '';

(async function go(i) {

if (i < res.vacancy.length) {

const vacancy = await getVacancyById(res.vacancy[i]);

const title = vacancy.vacancy.title + " (" + vacancy.vacancy.status + ")";

// Создаем новую строку таблицы

const newRow = document.createElement('tr');

// Создаем ячейку с ссылкой на вакансию

const vacancyCell = document.createElement('td');

const vacancyLink = document.createElement('a');

vacancyLink.setAttribute('href', '/vacancy?id=' + vacancy.vacancy.id); // Добавляем параметр ID к ссылке

vacancyLink.textContent = title; // Установка текста ссылки на название вакансии

vacancyCell.appendChild(vacancyLink);

newRow.appendChild(vacancyCell);

// Добавляем новую строку в таблицу

tableBody.appendChild(newRow);

go(i + 1);

console.log(i);

}

})(0);

})

}

async function my() {

let token = getCookie("token");

if (token === null) {

window.location.href = "/login";

return;

}

// Пример SQL-запроса для получения списка вакансий и их идентификаторов

const response = await sendSql("SELECT v.id FROM Vacancies v WHERE v.author\_id = (SELECT user\_id FROM Tokens WHERE id = '" + token + "') UNION SELECT v.id FROM Vacancies v JOIN Responses r ON v.id = r.vacancy\_id WHERE r.user\_id = (SELECT user\_id FROM Tokens WHERE id = '" + token + "');");

// Получаем список вакансий из SQL-запроса

const vacancies = response.rows;

// Получаем элемент таблицы, куда будем добавлять вакансии

const tableBody = document.querySelector('.table-hover.table-striped tbody');

// Очищаем содержимое таблицы перед добавлением новых данных

tableBody.innerHTML = '';

// Перебираем список вакансий и создаем строки таблицы с названиями вакансий и ссылками

vacancies.forEach(async function (v\_id) {

const vacancy = await getVacancyById(v\_id.id);

const title = vacancy.vacancy.title + " (" + vacancy.vacancy.status + ")";

// Создаем новую строку таблицы

const newRow = document.createElement('tr');

// Создаем ячейку с ссылкой на вакансию

const vacancyCell = document.createElement('td');

const vacancyLink = document.createElement('a');

vacancyLink.setAttribute('href', '/vacancy?id=' + vacancy.vacancy.id); // Добавляем параметр ID к ссылке

vacancyLink.textContent = title; // Установка текста ссылки на название вакансии

vacancyCell.appendChild(vacancyLink);

newRow.appendChild(vacancyCell);

// Добавляем новую строку в таблицу

tableBody.appendChild(newRow);

});

}

async function vacancy\_e() {

getQueryParam('id').then(function (id) {

getVacancyById(id).then(function (response) {

let token = getCookie("token");

document.getElementById('title').value = response.vacancy.title;

document.getElementById('information').value = response.vacancy.information;

document.getElementById('status').value = response.vacancy.status;

document.getElementById('payment').value = response.vacancy.payment;

document.getElementById('response\_link').href = '/response?id=' + response.vacancy.id

});

})

}

async function create\_e() {

getQueryParam('id').then(function (id) {

let title = document.getElementById('title').value;

let information = document.getElementById('information').value;

let status = document.getElementById('status').value;

let payment = Number(document.getElementById('payment').value);

updateVacancy(id, null, title, information, payment, status).then(function (response) {

if (response.status === "ok") {

alert("Вакансия успешно обновлена!")

window.location.href = "/vacancy?id=" + response.vacancy;

} else {

alert("Не удалось обновить вакансию")

}

});

})

}

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Главная</title>

<!--

Template 2108 Dashboard

http://www.tooplate.com/view/2108-dashboard

-->

<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Open+Sans:300,400,600">

<!-- https://fonts.google.com/specimen/Open+Sans -->

<link rel="stylesheet" href="/css/fontawesome.min.css">

<!-- https://fontawesome.com/ -->

<link rel="stylesheet" href="/css/bootstrap.min.css">

<!-- https://getbootstrap.com/ -->

<link rel="stylesheet" href="/css/tooplate.css">

</head>

<body id="reportsPage">

<div class="" id="home">

<div class="container">

<div class="row">

<div class="col-12">

<nav class="navbar navbar-expand-xl navbar-light bg-light">

<a class="navbar-brand" href="#">

<i class="fas fa-3x fa-circle-notch tm-site-icon"></i>

<h1 class="tm-site-title mb-0">Cukierka</h1>

</a>

<button class="navbar-toggler ml-auto mr-0" type="button" data-toggle="collapse"

data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"

aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav mx-auto">

<li class="nav-item">

<a class="nav-link active" href="/">Главная</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/vacancy/list">Вакансии</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/vacancy/my">Мои вакансии</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/user/my">Мои аккаунт</a>

</li>

</ul>

<ul class="navbar-nav">

<li class="nav-item">

<a class="nav-link d-flex" onclick="exit()">

<i class="far fa-user mr-2 tm-logout-icon"></i>

<span>Выйти</span>

</a>

</li>

</ul>

</div>

</nav>

</div>

</div>

<!-- row -->

<div class="row tm-content-row tm-mt-big">

<div class="tm-col tm-col-big">

<div class="bg-white tm-block h-100">

<h2 class="tm-block-title">Статистика за неделю</h2>

<canvas id="lineChart"></canvas>

</div>

</div>

<div class="tm-col tm-col-big">

<div class="bg-white tm-block h-100">

<h2 class="tm-block-title">Популярные вакансии</h2>

<canvas id="barChart"></canvas>

</div>

</div>

<div class="tm-col tm-col-small">

<div class="bg-white tm-block h-100">

<h2 class="tm-block-title">Статистика состояний вакансий</h2>

<canvas id="pieChart" class="chartjs-render-monitor"></canvas>

</div>

</div>

</div>

<footer class="row tm-mt-small">

<div class="col-12 font-weight-light">

<p class="d-inline-block tm-bg-black text-white py-2 px-4">

Авторские права &copy; 2024 Владислав Черняков

</p>

</div>

</footer>

</div>

</div>

<script src="/js/jquery.min.js"></script>

<script src="js/api.js"></script>

<script src="js/tools.js"></script>

<script src="/js/chart.min.js"></script>

<script src="/js/chart-draw.js"></script>

<script src="/js/main.js"></script>

<script>

let ctxLine,

ctxBar,

ctxPie,

optionsLine,

optionsBar,

optionsPie,

configLine,

configBar,

configPie,

lineChart;

barChart, pieChart;

// DOM is ready

$(function () {

updateChartOptions();

drawLineChart(); // Line Chart

drawBarChart(); // Bar Chart

drawPieChart(); // Pie Chart

checkAuth();

$(window).resize(function () {

updateChartOptions();

updateLineChart();

updateBarChart();

reloadPage();

});

})

</script>

</body>

</html>

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Вакансии</title>

<!--

Template 2108 Dashboard

http://www.tooplate.com/view/2108-dashboard

-->

<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Open+Sans:300,400,600">

<!-- https://fonts.google.com/specimen/Open+Sans -->

<link rel="stylesheet" href="/css/fontawesome.min.css">

<!-- https://fontawesome.com/ -->

<link rel="stylesheet" href="/css/bootstrap.min.css">

<!-- https://getbootstrap.com/ -->

<link rel="stylesheet" href="/css/tooplate.css">

</head>

<body id="reportsPage">

<div class="" id="home">

<div class="container">

<div class="row">

<div class="col-12">

<nav class="navbar navbar-expand-xl navbar-light bg-light">

<a class="navbar-brand" href="#">

<i class="fas fa-3x fa-circle-notch tm-site-icon"></i>

<h1 class="tm-site-title mb-0">Cukierka</h1>

</a>

<button class="navbar-toggler ml-auto mr-0" type="button" data-toggle="collapse"

data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"

aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav mx-auto">

<li class="nav-item">

<a class="nav-link" href="/">Главная</a>

</li>

<li class="nav-item">

<a class="nav-link active" href="/vacancy/list">Вакансии</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/vacancy/my">Мои вакансии</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/user/my">Мои аккаунт</a>

</li>

</ul>

<ul class="navbar-nav">

<li class="nav-item">

<a class="nav-link d-flex" onclick="exit()">

<i class="far fa-user mr-2 tm-logout-icon"></i>

<span>Выйти</span>

</a>

</li>

</ul>

</div>

</nav>

</div>

</div>

<!-- row -->

<div class="row tm-content-row tm-mt-big">

<div class="col-xl-8 col-lg-12 col-md-12 col-sm-12">

<div class="bg-white tm-block h-100">

<div class="row">

<div class="col-md-8 col-sm-12">

<h2 class="tm-block-title d-inline-block">Вакансия</h2>

</div>

<div class="col-md-4 col-sm-12 text-right">

<a href="/vacancy/add" class="btn btn-small btn-primary">Создать вакансию</a>

</div>

</div>

<div class="row mt-4 tm-edit-product-row">

<div class="col-xl-12 col-lg-12 col-md-12">

<table class="table table-hover table-striped mt-3">

<tbody>

</tbody>

</table>

</div>

</div>

</div>

</div>

<div class="col-xl-4 col-lg-12 tm-md-12 tm-sm-12 tm-col">

<div class="bg-white tm-block">

<h2 class="tm-block-title d-inline-block">Поиск</h2>

<form id="se" class="tm-login-form">

<div class="input-group">

<label for="name"

class="col-xl-4 col-lg-4 col-md-4 col-sm-5 col-form-label">Название</label>

<input name="name" type="text" class="form-control validate" id="name\_s">

</div>

<div class="input-group mt-3">

<label for="sort\_s" class="col-xl-4 col-lg-4 col-md-4 col-sm-5 col-form-label">Тип

сортировки</label>

<select class="custom-select col-xl-9 col-lg-8 col-md-8 col-sm-7" id="sort\_s"

required>

<option value="created">Дате создания</option>

<option value="status">Статусу</option>

<option value="payment">Цене</option>

</select>

</div>

<div class="input-group mt-3">

<label for="sort\_s2" class="col-xl-4 col-lg-4 col-md-4 col-sm-5 col-form-label">Вид

сортировка</label>

<select class="custom-select col-xl-9 col-lg-8 col-md-8 col-sm-7" id="sort\_s2"

required>

<option value="ASC">По возрастания</option>

<option value="DESC">По убывания</option>

</select>

</div>

<div class="input-group mt-3">

<button type="submit" class="btn btn-primary d-inline-block mx-auto">Искать</button>

</div>

</form>

</div>

</div>

</div>

</div>

<!-- row -->

<footer class="row tm-mt-small">

<div class="col-12 font-weight-light">

<p class="d-inline-block tm-bg-black text-white py-2 px-4">

Авторские права &copy; 2024 Владислав Черняков

</p>

</div>

</footer>

</div>

</div>

<script src="/js/jquery.min.js"></script>

<script src="/js/api.js"></script>

<script src="/js/tools.js"></script>

<script src="/js/main.js"></script>

<script src="/js/vacancy.js"></script>

<script>

$(function () {

search();

})

document.addEventListener("DOMContentLoaded", function () {

document.getElementById("se").addEventListener("submit", function (event) {

event.preventDefault();

search();

});

});

</script>

</body>

</html>

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Вакансия</title>

<!--

Template 2108 Dashboard

http://www.tooplate.com/view/2108-dashboard

-->

<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Open+Sans:300,400,600">

<!-- https://fonts.google.com/specimen/Open+Sans -->

<link rel="stylesheet" href="/css/fontawesome.min.css">

<!-- https://fontawesome.com/ -->

<link rel="stylesheet" href="/css/bootstrap.min.css">

<!-- https://getbootstrap.com/ -->

<link rel="stylesheet" href="/css/tooplate.css">

</head>

<body id="reportsPage">

<div class="" id="home">

<div class="container">

<div class="row">

<div class="col-12">

<nav class="navbar navbar-expand-xl navbar-light bg-light">

<a class="navbar-brand" href="#">

<i class="fas fa-3x fa-circle-notch tm-site-icon"></i>

<h1 class="tm-site-title mb-0">Cukierka</h1>

</a>

<button class="navbar-toggler ml-auto mr-0" type="button" data-toggle="collapse"

data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"

aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav mx-auto">

<li class="nav-item">

<a class="nav-link" href="/">Главная</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/vacancy/list">Вакансии</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/vacancy/my">Мои вакансии</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/user/my">Мои аккаунт</a>

</li>

</ul>

<ul class="navbar-nav">

<li class="nav-item">

<a class="nav-link d-flex" onclick="exit()">

<i class="far fa-user mr-2 tm-logout-icon"></i>

<span>Выйти</span>

</a>

</li>

</ul>

</div>

</nav>

</div>

</div>

<!-- row -->

<div class="row tm-content-row tm-mt-big">

<div class="col-xl-8 col-lg-10 col-md-12 col-sm-12">

<div class="bg-white tm-block h-100">

<div class="row">

<div class="col-12">

<h2 class="tm-block-title d-inline-block">Вакансия</h2>

</div>

</div>

<div class="row mt-4 tm-edit-product-row">

<div class="col-xl-12 col-lg-12 col-md-12">

<form action="" class="tm-edit-product-form">

<div class="input-group mb-3">

<label for="title"

class="col-xl-4 col-lg-4 col-md-4 col-sm-5 col-form-label">Заголовок

</label>

<input placeholder="Заголовок вакансии" id="title" type="title"

class="form-control validate col-xl-9 col-lg-8 col-md-8 col-sm-7" disabled>

</div>

<!-- row

<div class="input-group mb-3">

<label for="author"

class="col-xl-4 col-lg-4 col-md-4 col-sm-5 col-form-label">Автор

</label>

<input placeholder="Автор вакансии" id="author" type="text"

class="form-control validate col-xl-9 col-lg-8 col-md-8 col-sm-7" disabled>

</div>

-->

<div class="input-group mb-3">

<label for="information"

class="col-xl-4 col-lg-4 col-md-4 col-sm-5 col-form-label">Информация</label>

<textarea class="form-control validate col-xl-9 col-lg-8 col-md-8 col-sm-7"

rows="3" placeholder="Краткая информация о вакансии" id="information"

disabled></textarea>

</div>

<div class="input-group mb-3">

<label for="status"

class="col-xl-4 col-lg-4 col-md-4 col-sm-5 col-form-label">Статус</label>

<select class="custom-select col-xl-9 col-lg-8 col-md-8 col-sm-7" id="status"

disabled>

<option value="open" selected>Открыто</option>

<option value="close">Закрыто</option>

<option value="inwork">В процессе</option>

</select>

</div>

<div class="input-group mb-3">

<label for="payment"

class="col-xl-4 col-lg-4 col-md-4 col-sm-5 col-form-label">Цена

</label>

<input placeholder="Средняя цена" id="payment" type="text"

class="form-control validate col-xl-9 col-lg-8 col-md-8 col-sm-7" disabled>

</div>

<div class="input-group mb-3">

<div class="ml-auto col-xl-8 col-lg-8 col-md-8 col-sm-7 pl-0">

<a class="btn btn-primary" id="response\_link" href="/response">Отклики</a>

</div>

</div>

<div class="input-group mb-3">

<div class="ml-auto col-xl-8 col-lg-8 col-md-8 col-sm-7 pl-0">

<button type="button" id="resp\_btn"

class="btn btn-primary d-inline-block mx-auto">Оставить отклик</button>

</div>

</div>

<div class="input-group mb-3">

<div class="ml-auto col-xl-8 col-lg-8 col-md-8 col-sm-7 pl-0">

<a class="btn btn-primary" id="edit\_link"

href="/vacancy/edit">Редактировать</a>

</div>

</div>

<div class="input-group mb-3">

<div class="ml-auto col-xl-8 col-lg-8 col-md-8 col-sm-7 pl-0">

<a class="btn btn-primary" id="chat" href="/chat">Чат</a>

</div>

</div>

<!-- row

<div class="input-group mb-3">

<div class="ml-auto col-xl-8 col-lg-8 col-md-8 col-sm-7 pl-0">

<button type="button" class="btn btn-primary">Редактировать

</button>

</div>

</div>

-->

</form>

</div>

</div>

</div>

</div>

</div>

<!-- row -->

<footer class="row tm-mt-small">

<div class="col-12 font-weight-light">

<p class="d-inline-block tm-bg-black text-white py-2 px-4">

Авторские права &copy; 2024 Владислав Черняков

</p>

</div>

</footer>

</div>

</div>

<script src="/js/jquery.min.js"></script>

<script src="/js/api.js"></script>

<script src="/js/tools.js"></script>

<script src="/js/main.js"></script>

<script src="/js/vacancy.js"></script>

<script>

$(function () {

vacancy();

})

document.addEventListener("DOMContentLoaded", function () {

document.getElementById("resp\_btn").addEventListener("click", function (event) {

event.preventDefault();

create\_res();

});

});

</script>

</body>

</html>

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Пользователь</title>

<!--

Template 2108 Dashboard

http://www.tooplate.com/view/2108-dashboard

-->

<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Open+Sans:300,400,600">

<!-- https://fonts.google.com/specimen/Open+Sans -->

<link rel="stylesheet" href="/css/fontawesome.min.css">

<!-- https://fontawesome.com/ -->

<link rel="stylesheet" href="/css/bootstrap.min.css">

<!-- https://getbootstrap.com/ -->

<link rel="stylesheet" href="/css/tooplate.css">

</head>

<body id="reportsPage">

<div class="" id="home">

<div class="container">

<div class="row">

<div class="col-12">

<nav class="navbar navbar-expand-xl navbar-light bg-light">

<a class="navbar-brand" href="#">

<i class="fas fa-3x fa-circle-notch tm-site-icon"></i>

<h1 class="tm-site-title mb-0">Cukierka</h1>

</a>

<button class="navbar-toggler ml-auto mr-0" type="button" data-toggle="collapse"

data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"

aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav mx-auto">

<li class="nav-item">

<a class="nav-link" href="/">Главная</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/vacancy/list">Вакансии</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/vacancy/my">Мои вакансии</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/user/my">Мои аккаунт</a>

</li>

</ul>

<ul class="navbar-nav">

<li class="nav-item">

<a class="nav-link d-flex" onclick="exit()">

<i class="far fa-user mr-2 tm-logout-icon"></i>

<span>Выйти</span>

</a>

</li>

</ul>

</div>

</nav>

</div>

</div>

<!-- row -->

<div class="row tm-content-row tm-mt-big">

<div class="col-xl-8 col-lg-10 col-md-12 col-sm-12">

<div class="bg-white tm-block h-100">

<div class="row">

<div class="col-12">

<h2 class="tm-block-title d-inline-block">Пользователь</h2>

</div>

</div>

<div class="row mt-4 tm-edit-product-row">

<div class="col-xl-12 col-lg-12 col-md-12">

<form action="" class="tm-edit-product-form">

<div class="input-group mb-3">

<label for="name" class="col-xl-4 col-lg-4 col-md-4 col-sm-5 col-form-label">Имя

</label>

<input placeholder="Полное имя пользователя" id="name" type="text"

class="form-control validate col-xl-9 col-lg-8 col-md-8 col-sm-7" disabled>

</div>

<div class="input-group mb-3">

<label for="email"

class="col-xl-4 col-lg-4 col-md-4 col-sm-5 col-form-label">Почта

</label>

<input placeholder="Электронная почта" id="email" type="email"

class="form-control validate col-xl-9 col-lg-8 col-md-8 col-sm-7" disabled>

</div>

<div class="input-group mb-3">

<label for="information"

class="col-xl-4 col-lg-4 col-md-4 col-sm-5 col-form-label">Информация</label>

<textarea class="form-control validate col-xl-9 col-lg-8 col-md-8 col-sm-7"

rows="3" placeholder="Краткая информация о пользователе" id="information"

disabled></textarea>

</div>

<div class="input-group mb-3">

<div class="ml-auto col-xl-8 col-lg-8 col-md-8 col-sm-7 pl-0">

<a class="btn btn-primary" id="comments\_link" href="/comment/list">Комментарии</a>

</div>

</div>

<!-- row

<div class="input-group mb-3">

<div class="ml-auto col-xl-8 col-lg-8 col-md-8 col-sm-7 pl-0">

<button type="button" class="btn btn-primary">Редактировать

</button>

</div>

</div>

-->

</form>

</div>

</div>

</div>

</div>

</div>

<!-- row -->

<footer class="row tm-mt-small">

<div class="col-12 font-weight-light">

<p class="d-inline-block tm-bg-black text-white py-2 px-4">

Авторские права &copy; 2024 Владислав Черняков

</p>

</div>

</footer>

</div>

</div>

<script src="/js/jquery.min.js"></script>

<script src="/js/api.js"></script>

<script src="/js/tools.js"></script>

<script src="/js/main.js"></script>

<script src="/js/user.js"></script>

<script>

$(function () {

user();

})

</script>

</body>

</html>

# Приложение В (обязательное) Результаты работы приложения

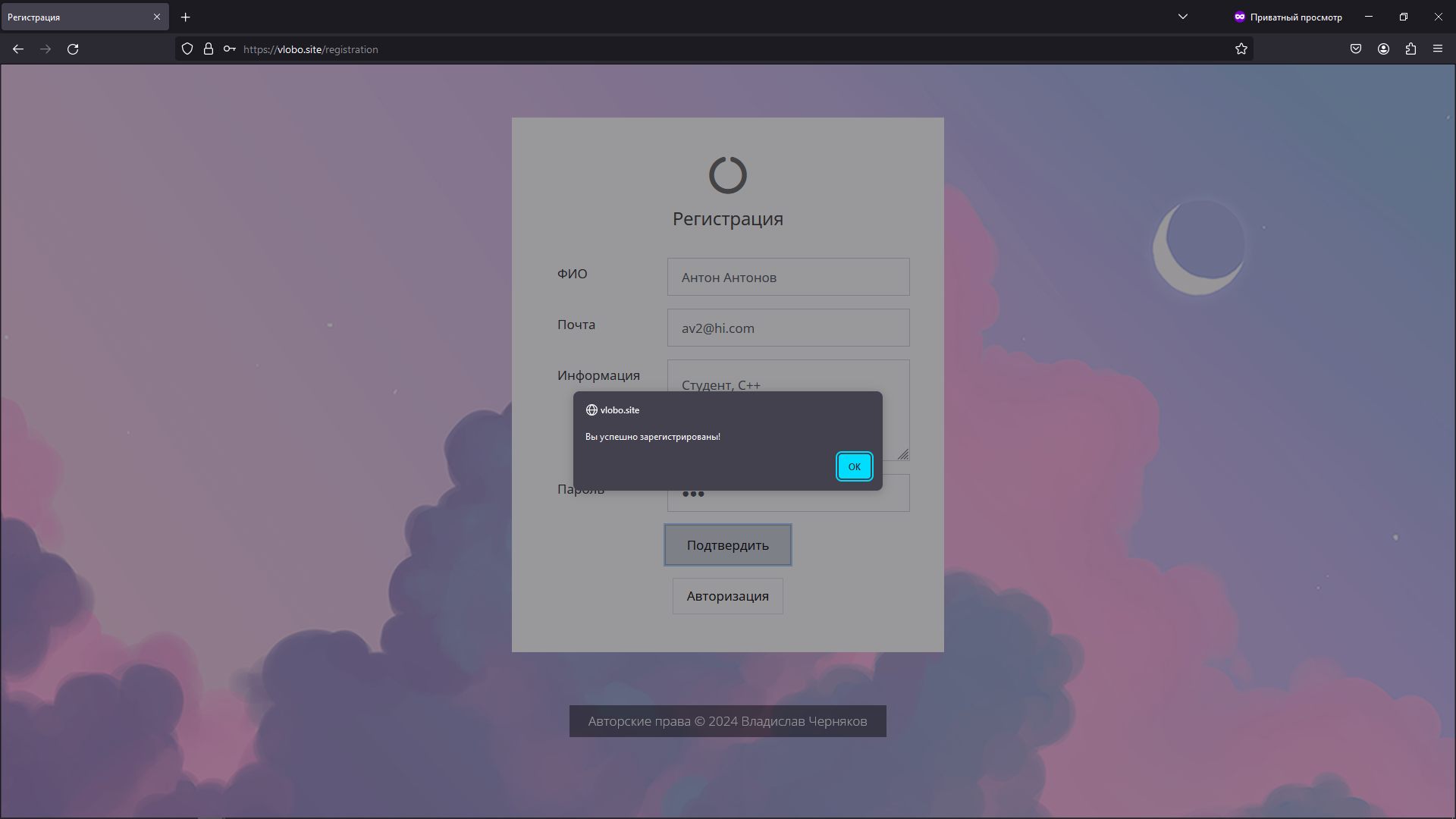


Рисунок В.1 – Результат регистрации пользователя

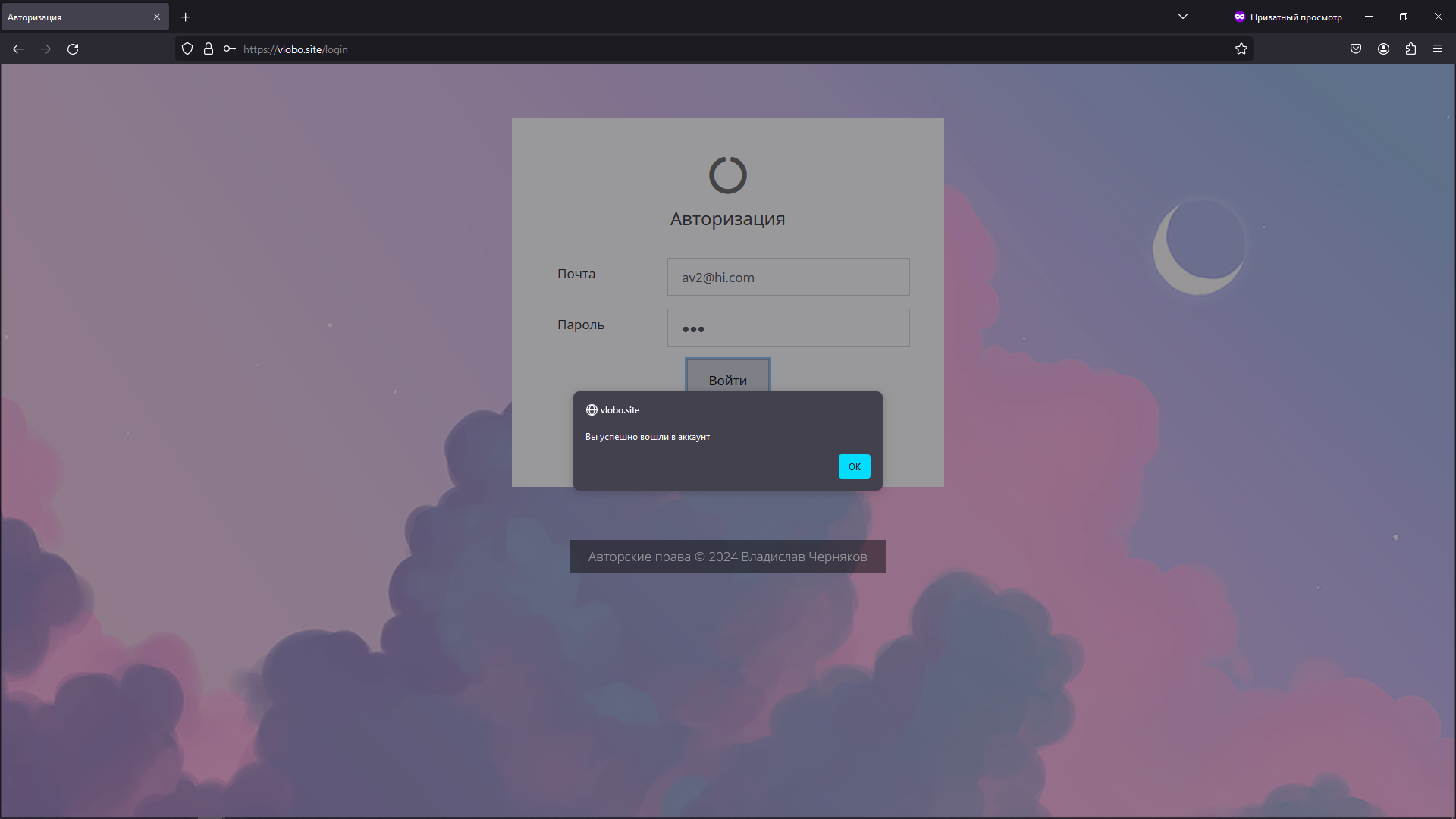


Рисунок В.2 – Результат авторизации пользователя

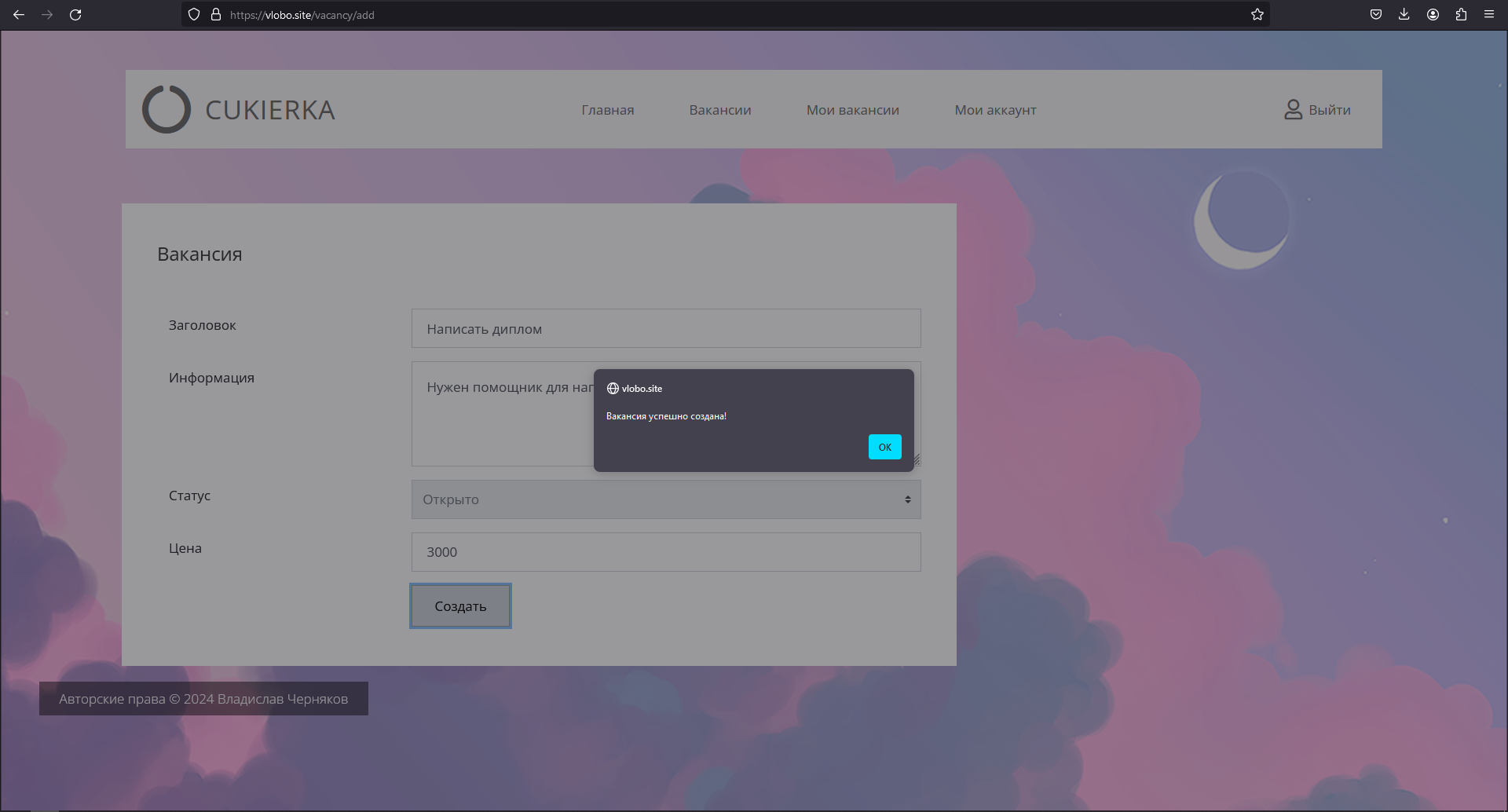


Рисунок В.3 – Результат создания вакансии

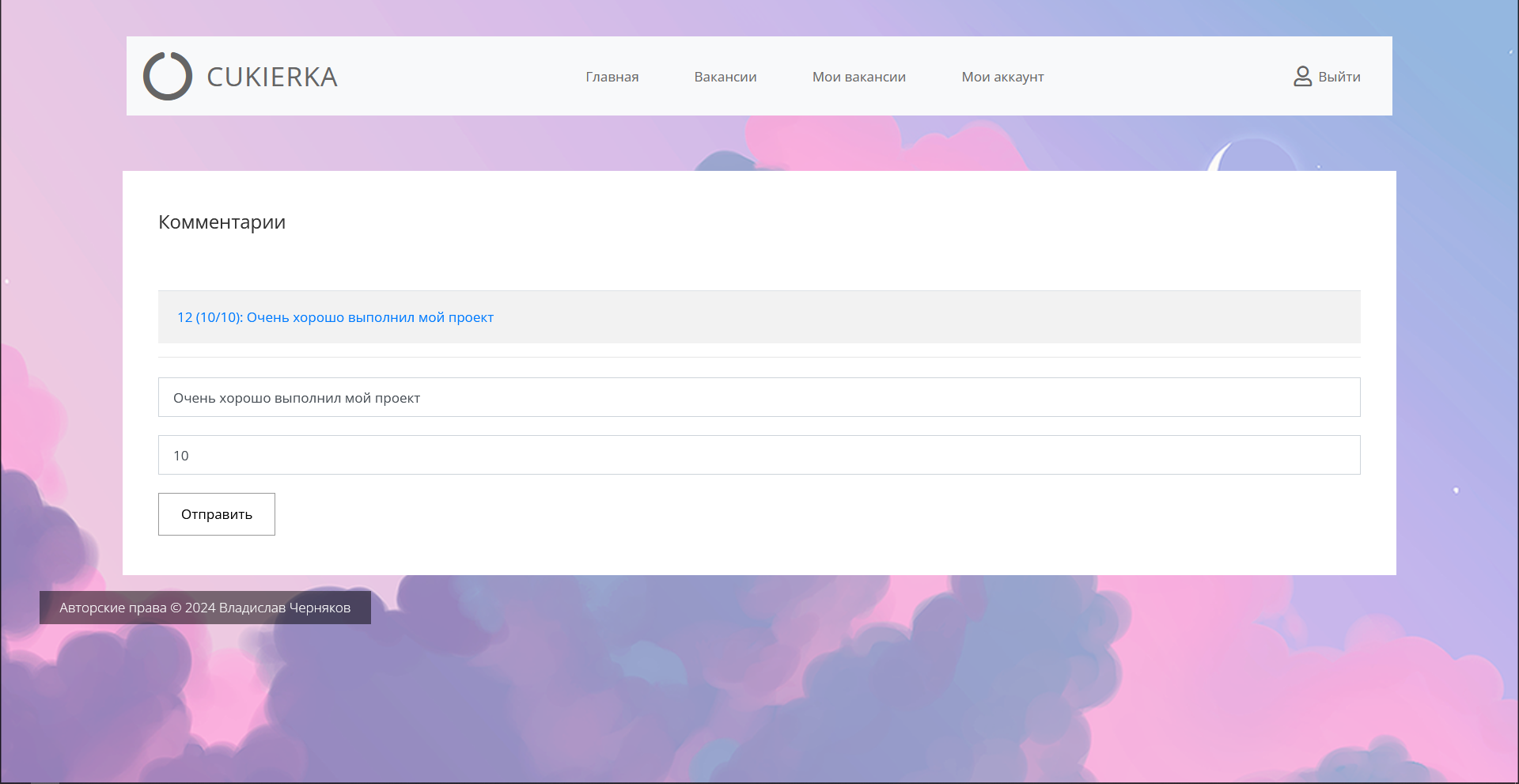


Рисунок В.4 – Результат создания комментария

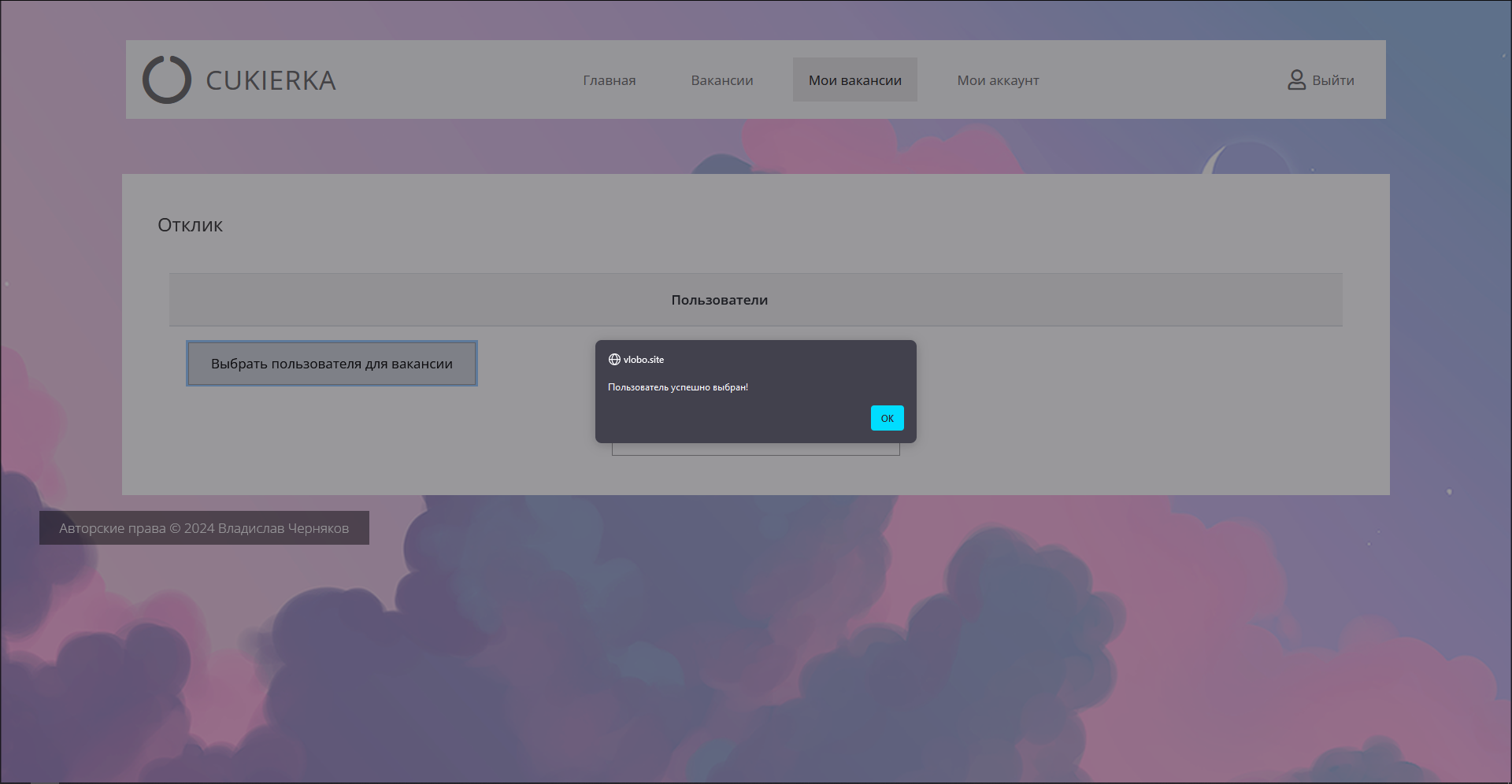


Рисунок В.5 – Результат создания проекта