

Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
“УНИВЕРСИТЕТ ИТМО”

Факультет Инфокоммуникационных Технологий

Образовательная программа 09.03.02

Направление подготовки (специальность) Мобильные сетевые технологии

О Т Ч Е Т

по курсовой работе

Тема задания: Реализация web-сервисов средствами Django REST framework, Vue.js, Muse-UI

Обучающийся Нурдино Ростислав Артурович, гр. K3340

Руководитель: Говоров А. И., ассистент факультета ИКТ Университета ИТМО

Оценка за курсовую работу ____

Подписи членов комиссии:

____ ()
(подпись)

____ ()
(подпись)

____ ()
(подпись)

Дата ____

Санкт-Петербург
20 20

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ.....	4
1.1. Описание предметной области	4
1.2. Описание функциональных требований.....	5
1.3. Выводы.....	5
2. ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ СЕРВИСА	6
2.1. Описание архитектуры сервиса.....	6
2.2. Модель данных.....	7
2.3. Выводы.....	8
3. РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ	9
3.1. Описание средств разработки серверной части.....	9
3.2. Реализация базы данных	9
3.3. Сериализация.....	10
3.4. Создание представлений	11
3.5. Настройка маршрутизации	13
3.6. Выводы.....	13
4. РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ	14
4.1. Описание средств разработки клиентской части.....	14
4.2. Разработанные интерфейсы	14
4.2.1. Вход в систему	14
4.3. Выводы.....	22
5. КОНТЕЙНЕРИЗАЦИЯ И ОРКЕСТРАЦИЯ	23
5.1. Описание средства контейнеризации и оркестрации.....	23
5.2. Контейнеризация проекта	24
5.3. Оркестрация проекта	25
5.4. Выводы.....	26
ЗАКЛЮЧЕНИЕ	27
СПИСОК ЛИТЕРАТУРЫ	28

ВВЕДЕНИЕ

Несмотря на то, что интернет давно и прочно вошел в нашу жизнь, многие предприниматели и даже крупные фирмы не понимают, что им даст создание собственного сайта, ведь есть другие хорошо зарекомендовавшие себя проверенные способы саморекламы: телевидение, радио, СМИ, баннеры, флайеры и тому подобное. У любой современной компании существует сайт. Это один из элементов престижа, ведь именно в Интернете потенциальные клиенты будут в первую очередь искать информацию о фирме. И если у нее нет хотя бы одностраничника с прайсом, это покажется подозрительным – насколько же это неуспешная фирма, если не может даже небольшой веб-ресурс создать?

Целью данной курсовой работы является разработка веб-сервиса согласно выбранному варианту. В рамках работы нужно было изучить и научиться применять средства создания веб-сервисов, которые используют в современных системах.

В ходе работы должны быть выполнены следующие задачи:

1. Изучение предметной области.
2. Анализ функциональных требований.
3. Проектирование архитектуры веб-сервиса.
4. Разработка серверной части системы.
5. Разработка клиентской части системы.
6. Контейнеризация проекта.

В первой главе проведен анализ предметной области и функциональных требований. Вторая глава посвящена проектированию архитектуры веб-сервиса. В третьей и четвертой главах рассмотрен подход к разработке серверной и клиентской частей системы соответственно. Пятая глава посвящена контейнеризации и оркестрации проекта.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

1.1. Описание предметной области

В качестве варианта был выбран вариант 11 – создание программной системы для биржи труда. Предметной областью является сама платформа. Основными пользователями являются администратор, соискатели и работодатели.

Информационная система для данной области представляет собой сервис, в котором реализованы стандартные процедуры для данной области: регистрация соискателей, добавления резюме и опыта работы, создание вакансий с предоставленными требованиями, обработка заявок, фильтрация и ранжирование вакансий. Такая система должна обеспечивать хранение сведений об имеющихся вакансиях, уровнях доступа различных групп, хранение информации, которую вводят соискатели. Каждый соискатель имеет личную информацию о себе, имеет определенное образование и навыки в той или иной области, может иметь несколько или не иметь совсем опыта работы. О каждом соискателе имеется следующая информация: фамилия, имя, отчество, адрес, дата рождения, номер телефона, тип образования, опыт работы, относящийся к конкретной профессии, стаж и описание. Работодатели имеют следующую информацию: фамилия, имя, отчество, почта, телефон, фирма. Так же работодатель может создавать вакансии, в которой описывает необходимые требования к должности, необходимое образование и стаж. Если работодатель удовлетворен заявкой, он может закрыть вакансию.

1.2. Описание функциональных требований

На основе анализа предметной области выявлены функциональные требования в разрабатываемой системе. В системе должна храниться вся имеющаяся информация о номерах, клиентах и работниках гостиницы.

Работа с системой предполагает получение информации регистрации соискателей и их данных, изменения этих данных, хранение информации о работодателях и их вакансиях, обработка заявок соискателей на определенную вакансию.

Администратор биржи труда должен иметь возможность выполнить следующие действия:

- Проверять, изменять, добавлять или удалять работодателей или соискателей.
- Отслеживать и изменять заявки.
- Редактировать профессии, их разряды, тип образования.
- Проверять, изменять, добавлять или удалять вакансии.

Служащий работник должен иметь возможность:

- Зарегистрироваться в системе.
- Добавить всю необходимую про него информацию
- Фильтровать вакансии
- Создавать заявку на интересующую его вакансию

1.3. Выводы

Проведен анализ предметной области и функциональных требований системы. В результате было улучшено понимание данной области и того, как должна работать разрабатываемая система.

2. ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ СЕРВИСА

2.1. Описание архитектуры сервиса

Для веб-сервиса, разрабатываемого в рамках курсовой работы, была выбрана архитектура «клиент-сервер», представленная на рис. 1. В данной архитектуре сетевая нагрузка распределена между поставщиками услуг, серверами, и заказчиками услуг, клиентами.

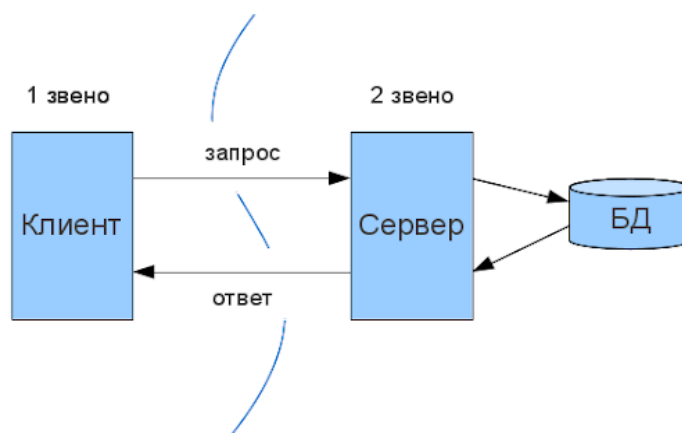


Рисунок 1 – Архитектура «Клиент-Сервер»

Сервером в данном случае считается абстрактная машина в сети, которая способна получить HTTP-запрос, обработать его и вернуть корректный ответ. Клиентом может считаться все, что способно сформировать и отправить HTTP-запрос. Сервер ожидает от клиента запрос и предоставляет свои ресурсы в виде данных или в виде сервисных функций.

База данных представляет собой третье звено архитектуры. Она нужна для того, чтобы информация могла сохраняться даже при падении и рестарте системы. Наличие базы данных гарантирует облегченный поиск по данным и их сохранность.

Перечислим преимущества такого подхода.

1. Сохранность информации.
2. Устойчивость к сбоям.
3. Масштабируемость.
4. Большая защищенность информации от несанкционированного доступа.
5. Меньшая нагрузка сети одним пользователем.
6. Гибкость системы.

2.2. Модель данных

В соответствии с вариантом задания и функциональными требованиями была создана модель данных, представленная в блоке exchange_engine на рис. 3.

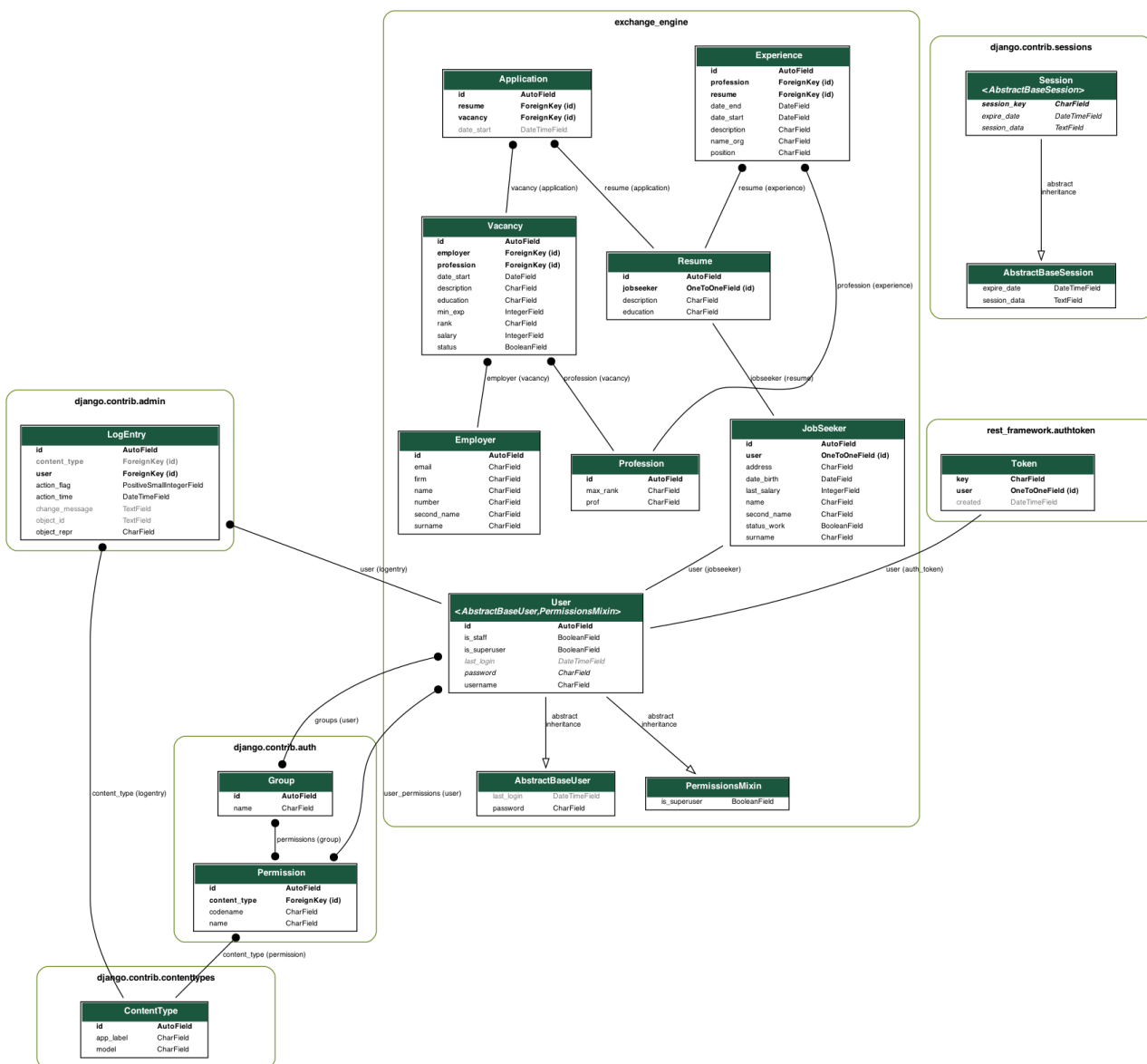


Рисунок 3 – Модель данных

Реализуемая модель содержит 10 сущностей:

- PermissionMixin – особые права для соискателя.
- AbstractBaseUser – вспомогательная сущность для расширения соискателей.
- User – наследуемая модель от AbstractBaseUser и PermissionMixin.
- Соискатели
- Профессии.
- Работодатели.
- Резюме.
- Вакансии

- Опыт работы
- Заявки

Сущности соединены между собой связями Один-к-Одному, Один-ко-Многим, Многие-ко-Многим. Основными из них являются: User, Jobseeker, Vacancy, Employer.

2.3. Выводы

В ходе проектирования архитектуры сервиса был выбран тип архитектуры «Клиент-сервер». На основе функциональных требований к системе была создана модель данных, которая позволяет реализовать требуемый функционал.

3. РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ

3.1. Описание средств разработки серверной части

Для реализации серверной части был использован фреймворк Django Rest, который является удобным инструментом, основанным на идеологии Django, для работы с rest. Так же необходимо было разработать способ аутентификации. Для этого был использован ареймворк `djangorestframework-simplejwt`.

Django – это высокоуровневая веб-инфраструктура языка Python, которая позволяет быстро создавать безопасные и поддерживаемые веб-сайты.

REST (сокр. англ. Representational State Transfer, «передача состояния представления») — стиль построения архитектуры распределенного приложения. Данные в REST должны передаваться в виде небольшого количества стандартных форматов (например HTML, XML, JSON). Сетевой протокол, как и HTTP, должен поддерживать кэширование, не должен зависеть от сетевого слоя, не должен сохранять информацию о состоянии между парами «запрос-ответ». Утверждается, что такой подход обеспечивает масштабируемость системы и позволяет ей эволюционировать с новыми требованиями. Кроме того, преимуществами использования REST являются надежность, производительность, прозрачность системы взаимодействия, простота интерфейсов и способность приспосабливаться к новым требованиям. [1]

JSON Web Token (JWT) — это открытый стандарт (RFC 7519) для создания токенов доступа, основанный на формате JSON. Как правило, используется для передачи данных для аутентификации в клиент-серверных приложениях. Токены создаются сервером, подписываются секретным ключом и передаются клиенту, который в дальнейшем использует данный токен для подтверждения своей личности.

3.2. Реализация базы данных

Представленная на рис. 3 модель данных была реализована с помощью СУБД PostgreSQL. PostgreSQL – это свободная объектно-реляционная система управления базами данных (СУБД). Преимуществами данной СУБД являются высокопроизводительные и надежные механизмы транзакций, расширенная система встроенных языков программирования, наследование и расширяемость [2].

Разработанная по представленной модели база данных содержит следующие таблицы:

- Jobseeker (соискатель) – таблица содержит информацию о соискателе.

- Employer (работодатель) – таблица содержит информацию о работодателе.
- Vacancy (вакансии) – таблица содержит информацию о вакансиях
- Profession (профессии) – таблица содержит информацию профессиях.
- Resume (резюме) – таблица содержит дополнительную информацию о соискателе.
- Experience (опыт) – таблица содержит опыт работы соискателей
- Application (заявки) – таблица содержит информацию о заявках.
- User (пользователь) – таблица содержит зарегистрированных пользователей.

3.3. Сериализация

В программировании сериализация представляет собой процесс перевода какой-либо структуры данных в последовательность битов. Другими словами, это процесс создания потокового представления данных, которые можно передавать по сети. Обратным процессом является десериализация.

Для разработки веб-приложения с Django Rest были созданы следующие сериализаторы:

- ResumeDetailSerializer - используется для получения, удаления и изменения данных резюме.
- ResumeListSerializer – используется для получения списка данных резюме.
- ResumeCreateSerializer – используется для создания резюме.
- UserSerializer – для создания аутентифицированных пользователей.
- JobSeekerListSerializer – используется для получения списка данных соискателей.
- JobSeekerDetailSerializer - используется для получения, удаления и изменения данных о соискателе.
- JobSeekerCreateSerializer - используется для создания соискателя.
- VacancyListSerializer - используется для получения списка данных вакансий.
- VacancyCreateSerializer
- – используется для получения данных о заселении клиента.
- VacancyDetailSerializer - используется для получения, удаления и изменения данных вакансии.
- ProfessionListSerializer - используется для получения списка данных профессий.
- ProfessionCreateSerializer - используется для создания профессий.
- ProfessionDetailSerializer - используется для получения, удаления и изменения данных профессии.
- EmployerListSerializer - используется для получения списка данных работодателей.
- EmployerCreateSerializer - используется для создания работников.
- EmployerDetailSerializer - используется для получения, удаления и изменения данных резюме.
- ExperienceListSerializer - используется для получения списка данных опытов.

- ExperienceCreateSerializer - используется для создания опыта работы.
- ExperienceDetailSerializer - используется для получения, удаления и изменения данных об опыте работы.
- ApplicationListSerializer - используется для получения списка данных заявок.
- ApplicationCreateSerializer - используется для создания заявки.
- ApplicationDetailSerializer - используется для получения, удаления и изменения данных о заявках.

Все сериализаторы имеют почти одинаковый вид. Из всех выделяется только один UserSerializer. Отличается он сохранением с предобработкой данных с помощью метода create и полем, которое не нужно явно объявлять в сериализаторе, с помощью параметра extra_kwargs.

На рис.4 приведен сериализатор для модели номера гостиницы.

```
class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = ['id', 'username', 'password']
        extra_kwargs = {
            'password': {'write_only': True}
        }

    def create(self, validated_data):
        return User.objects.create_user(
            username=validated_data['username'],
            password=validated_data['password']
        )
```

Рисунок 4 – Сериализатор номера гостиницы

3.4. Создание представлений

Представление (view) – это функция обработчик запросов, которая получает HTTP-запросы и возвращает ответы. View имеет доступ к данным через модели, которые определяют структуру данных приложения и предоставляют механизмы для управления базой данных.

Для каждого сериализатора был создан view в соответствии его назначения. Рассмотрим классы, которые были использованы при наследовании.

- generics.CreateAPIView – для создания объектов.
- generics.RetrieveUpdateDestroyAPIView – для получения, изменения или удаления объекта по ключу pk. Каждое из действий вызывается методом запроса (GET, POST, DELETE).
- generics.ListAPIView – для получения списка объектов.

Generics расширяет методы стандартного APIView, в котором реализованы основные методы запросов, таких как создание, получение списка или объекта, удаление, обновление

объектов. Однако во многих методах необходимо было доопределять методы. Рассмотрим несколько примеров.

- `get_queryset` – для создания фильтров. Переопределение этого метода позволяет настроить набор запросов так, чтобы они возвращались различными вариантами.
- `perform_create` – для переопределения создания объектов предварительно обработав поступившую на сервер информацию. Этот метод позволяет определять пользователя, который пользуется приложением, по его токену, не храня дополнительной информации.

Так же были использованы права доступа `permission`. `IsAuthenticated`, `IsAdmin` и `IsOwnerOrReadOnly`. С помощью них была разработана логика работы серверной части для различных групп пользователей.

Таким образом, с помощью выше описанных методов, можно реализовать специальные возможности администратора, чуть уменьшенные права пользователей, при этом всегда отличать данные в базе того, кто аутентифицирован на данный момент. Такой подход позволит получать нужные и необходимые данные сущностей, которые отдалены друг от друга, как например, сущность `user` и опыт работы.

На рис. 5 представлен пример готового View.

```
class VacancyListView(generics.ListAPIView):
    """Вывод списка вакансий"""
    serializer_class = VacancyListSerializer

    def get_queryset(self):
        queryset = Vacancy.objects.all()
        params = self.request.query_params

        profession = params.get('profession', None)
        from_s = params.get('from_s', None)
        to_s = params.get('to_s', None)
        min_exp = params.get('min_exp', None)

        if profession:
            queryset = queryset.filter(profession=profession)

        if from_s and to_s:
            queryset = queryset.filter(salary__range=(from_s, to_s))
        elif from_s:
            queryset = queryset.filter(salary__gte=from_s)
        elif to_s:
            queryset = queryset.filter(salary__lte=to_s)

        if min_exp:
            queryset = queryset.filter(min_exp__gte=min_exp)

        return queryset
```

Рисунок 5 – Представление для вывода и добавления работников

3.5. Настройка маршрутизации

Когда разработаны представления, нужно создать URL-адреса для того, чтобы система начала работать. В системе имеется список основных адресов, который включает адреса приложения и адреса для авторизации. URL-адреса приложения основаны на разработанных ранее представлениях. На рис. 6 представлен список URL-адресов веб-приложения.

```
urlpatterns = [

    path("jobseeker/list/", views.JobSeekerListView.as_view()),
    path("jobseeker/detail/<int:pk>/", views.JobSeekerRetrieveUpdateDeleteView.as_view()),
    path("jobseeker/create/", views.JobSeekerCreateView.as_view()),

    path("vacancy/list/", views.VacancyListView.as_view()),
    path("vacancy/detail/<int:pk>/", views.VacancyRetrieveUpdateDeleteView.as_view()),
    path("vacancy/create/", views.VacancyCreateView.as_view()),

    path("profession/list/", views.ProfessionListView.as_view()),
    path("profession/detail/<int:pk>/", views.ProfessionRetrieveUpdateDeleteView.as_view()),
    path("profession/create/", views.ProfessionCreateView.as_view()),

    path("employer/list/", views.EmployerListView.as_view()),
    path("employer/detail/<int:pk>/", views.EmployerRetrieveUpdateDeleteView.as_view()),
    path("employer/create/", views.EmployerCreateView.as_view()),

    path("experience/list/<int:pk>/", views.ExperienceListView.as_view()),
    path("experience/detail/<int:pk>/", views.ExperienceRetrieveUpdateDeleteView.as_view()),
    path("experience/create/", views.ExperienceCreateView.as_view()),

    path("resume/list/", views.ResumeListView.as_view()),
    path("resume/detail/<int:pk>/", views.ResumeRetrieveUpdateDeleteView.as_view()),
    path("resume/create/", views.ResumeCreateView.as_view()),

    path("application/list/", views.ApplicationListView.as_view()),
    path("application/detail/<int:pk>/", views.ApplicationRetrieveUpdateDeleteView.as_view()),
    path("application/create/", views.ApplicationCreateView.as_view()),
```

Рисунок 6 – Список url-адресов приложения

3.6. Выводы

Средствами фреймворка Django REST был разработан бэкенд биржи труда. Были созданы и описаны сериализаторы, представления, права доступа, url-адреса веб-приложения.

4. РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ

4.1. Описание средств разработки клиентской части

Для разработки клиентской части системы был использован фреймворк Vue.js и библиотека Muse UI.

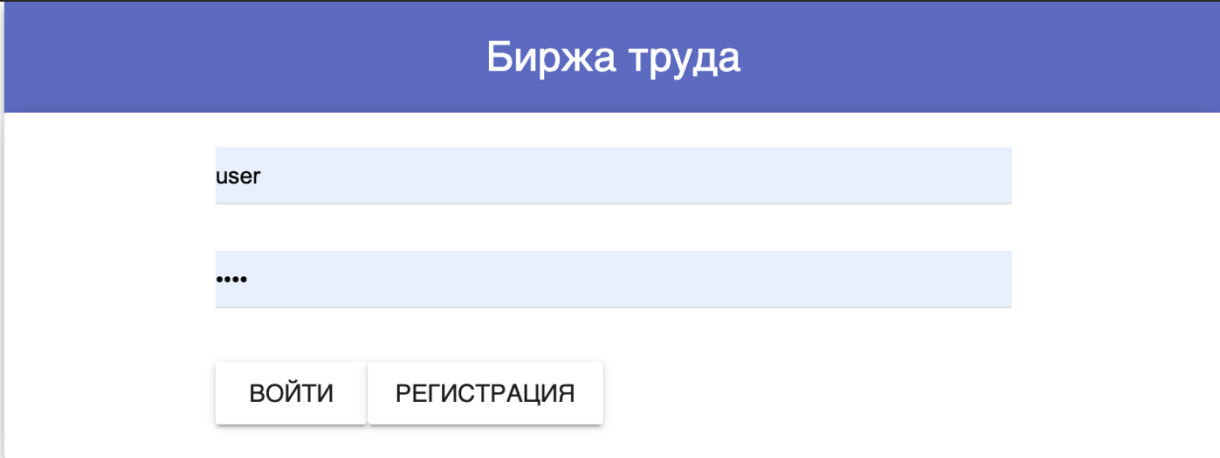
Vue.js — это прогрессивный фреймворк для создания пользовательских интерфейсов. В отличие от фреймворков-монолитов Vue создан пригодным для постепенного внедрения. Его ядро в первую очередь решает задачи уровня представления (view), что упрощает интеграцию с другими библиотеками и существующими проектами. С другой стороны, Vue полностью подходит и для создания сложных одностраничных приложений (SPA, Single-Page Applications), если использовать его совместно с современными инструментами и дополнительными библиотеками. [3]

Библиотека Muse UI представляет собой набор компонентов для Vue, которые используют Material Design [4]. Это фреймворк для быстрого создания и запуска пользовательского интерфейса с приятным и удобным дизайном.

4.2. Разработанные интерфейсы

4.2.1. Вход в систему

Для того, чтобы войти в свой аккаунт, нужно перейти на страницу login (рис. 7).



The image shows a web interface for a job market platform. At the top is a blue header with the text "Биржа труда" in white. Below the header is a white container with a light blue border. Inside this container, there are two light blue input fields. The first field is labeled "user" and the second field is for a password, indicated by four dots. Below these fields are two buttons: "ВОЙТИ" (Login) and "РЕГИСТРАЦИЯ" (Registration).

Рисунок 7 – Страница регистрации

Если же аккаунт отсутствует, то нажав на кнопку регистрации, вы попадаете на страницу авторизации, где можно завести нового пользователя (рис. 8).

Выполните вход

Логин

Пароль

Повторите пароль

РЕГИСТРАЦИЯ

Рисунок 8 – Страница регистрации

После регистрации пользователю предлагается заполнить свои личные данные. Поскольку база данных подразумевает разделение сущностей резюме и соискателя, то было принято решение дать пользователю заполнить данные на одной странице (рис. 9). После заполнения данных и нажатия кнопки “Зарегистрироваться”, сначала отправляется запрос на создание соискателя, после чего определяется его id и к этому id привязывается информация в резюме.

Имя Анна

Фамилия Волкова

Отчество Дмитриевна

Адрес home 44

Последняя зарплата 30000

Дата рождения 1995-04-12

Работает? ☒

Select Высшее (магистратура)

Описание Ищу работу

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок 9 – Пример заполненной формы личной информации

Пользователю предоставляется меню навигации с различными интерфейсами. (рис. 10).

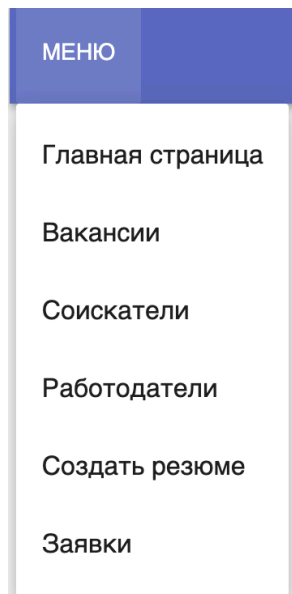


Рисунок 10 – Меню навигации

Интерфейс вакансий выводит список всех вакансий, которые были созданы работодателями в виде таблицы (рис. 11). Данный интерфейс предоставляет возможность увидеть краткие характеристики вакансий, возможность отсортировать данные по какому-то критерию, нажать на определенную вакансию и увидеть ее полное описание с возможностью подать соискателю заявку (рис. 12). Так же данный интерфейс имеет кнопку фильтр. Данная кнопка позволяет соискателю установить необходимые для него фильтры (Нужная вакансия, зарплаты меньше, больше или между введенными значениями, минимальный опыт работы). Для этого ему открывается форма, которую соискатель может заполнить и отправить нужный ему запрос (рис. 13).

Профессия	Дата публикации	Зарплата	Опыт работы	Статус
Уборщик	2020-07-09	60000	3	true
Электрик	2020-07-09	130000	6	true
Архитектор	2020-07-09	100000	3	true
Продавец	2020-07-09	33000	0	false
Инженер	2020-07-09	77000	1	true
Водитель	2020-07-09	100000	3	true
Копирайтер	2020-07-09	55555	1	true
Киповец	2020-07-09	150000	3	false
Сварщик	2020-07-09	175000	6	true
Грузчик	2020-07-09	32500	0	true
Программист	2020-07-09	60000	1	false
Электрик	2020-07-09	120000	3	true
Копирайтер	2020-07-09	77000	2	true

Рисунок 11 – Таблица вакансий

Электрик
2020-07-09

Описание

Нужен такой-то уборщик

Имя работодателя: Денис

Образование: Высшее (бакалавриат)

Необходимый разряд: 3 разряд

Минимальный опыт работы: 6

Зарплата: 130000

ПОДАТЬ ЗАЯВКУ

Рисунок 12 – Подробное описание вакансии

Фильтр

Фильтр по названию

Продавец

От

45500

До

98000

Минимальный
опыт работы

3

ЗАКРЫТЬ ОТФИЛЬТРОВАТЬ

Рисунок 13 – Форма фильтра

Интерфейс соискателей выводит список всех соискателей в таблице (рис. 14). Так же предоставляется посмотреть подробную информацию про соискателя (рис. 15) и (рис. 16). Здесь можно увидеть личные данные соискателя, его резюме, описание и опыт работы.

МЕНЮ	Биржа труда					LOGOUT
Фамилия	Имя	Отчество	Дата рождения	Статус	Последняя зарплата	
Алексеев	Алексей	Алексеевич	1990-07-09	true	100000	
Анатолийев	Анатолий	Анатольевич	1988-01-09	true	45000	
Михаилов	Михаил	Михайлович	2000-01-01	false	25000	
Генадьев	Генадий	Генадьевич	2001-02-12	true	33000	
Викторьевна	Виктория	Викторовна	1992-05-05	true	55000	
Денисов	Денис	Денисович	1988-04-03	false	43000	
Егор	Егоров	Егорович	1999-01-01	false	35000	
Петр	Петров	Петрович	1992-02-02	false	43133	

Рисунок 14 – Интерфейс вакансий.

Соискатель

Описание

Петр

Фамилия: Петр

Имя: Петров

Отчество: Петрович

Дата рождения: 1992-02-02

Адрес: home 123

Последняя зарплата: 13000

Резюме

Образование: Профессиональное

Описание: Работал электриком на севере в сложных природных условиях

Рисунок 15 – Подробная информация.

Опыт работы

Профессия	Организация	Позиция	Date start	Date end
Электрик	ООО Газпром	Младший электрик	2012-07-10	2012-07-10
Электрик	ООО Сибур	Электрик 3 разряда	2013-06-10	2020-06-10

Рисунок 15 – Список опытов работы.

Интерфейс заявок предоставляет возможность увидеть все поданные заявки. Связь устанавливается между вакансией и резюме. Тонизация позволяет определить текущего пользователя. Зная пользователя, можно определить id его резюме. Таким образом устанавливается связь. В таблице можно увидеть id резюме соискателя и id вакансии (рис. 16).

Биржа труда			
МЕНЮ			LOGOUT
ID	ID резюме	ID вакансии	Дата заявки
4	6	5	2020-07-09T23:31:13.041097Z
5	6	8	2020-07-09T23:31:17.407124Z
6	6	11	2020-07-09T23:31:22.568007Z
7	7	7	2020-07-09T23:31:27.737657Z
8	7	9	2020-07-09T23:31:34.818863Z
9	8	12	2020-07-09T23:31:39.371655Z
10	9	12	2020-07-09T23:31:45.480408Z
11	9	12	2020-07-09T23:31:50.311236Z
12	8	9	2020-07-09T23:31:55.460237Z
13	8	16	2020-07-09T23:32:02.488104Z
14	10	7	2020-07-09T23:32:07.515836Z
15	9	12	2020-07-09T23:32:11.661701Z

Рисунок 16 – Интерфейс заявок.

Интерфейс работодателей позволяет увидеть всех работодателей в таблице. На странице есть кнопки, которые позволяют добавить, удалить или изменить данные работодателя (рис. 17). Однако они доступны только для администратора, так как они ограничены правилами permissions. Поэтому, пользователю, который имеет привилегии пользователя не сможет отправить запрос в базу данных. Такой пользователь получит ошибку 403. То же самое касается и не авторизованных пользователей. Они получают 401 ошибку.

МЕНЮ		Биржа труда				LOGOUT	
Фамилия	Имя	Отчество	Email	Телефон	Фирма зарплата		
Денисов	Денис	Денисович	den@mail.ru	13014131123	ООО сибур	Удалить	Обновить
Генадьев	Генадий	Генадьевич	gena@mail.ru	812312312	SPAR	Удалить	Обновить
Сергеев	Сергей	Сергеевич	ser@mail.ru	298239423	ООО Газпром	Удалить	Обновить
Надеждова	Надежда	Сергеевна	nadya@mail.ru	294234892	Пятерочка	Удалить	Обновить
Борисов	Борис	Борисович	borya@mail.ru	22384728	ООО Газпром	Удалить	Обновить
Дарьева	Дарья	Алексеевна	dasha@mail.ru	29828346	ООО Электр осила	Удалить	Обновить

Добавить соискателя

СОЗДАТЬ

Рисунок 17 – Интерфейс работодателя.

Если администратору необходимо добавить или изменить работодателя, то при нажатии на соответствующие кнопки откроется одна и та же форма для заполнения данных. Однако, если это кнопка создания, то поля у формы будут пустые, кнопка обновления будет скрыта, вместо нее будет кнопка “Добавить” (рис. 18). Если же будет нажата кнопка “Обновить”, откроется такое же окно, но с заполненными данными работодателя, кнопка создания будет скрыта, вместо нее будет кнопка “Обновить” (рис. 19). Также имеется возможность посмотреть подробную информацию о работодателе, нажав на него (рис. 20).

Добавление работодателя

Фамилия

Имя

Отчество

Email

Номер
телефона

[ЗАКРЫТЬ](#) [ДОБАВИТЬ](#)

Рисунок 18 – Форма добавления работодателя

Добавление работодателя

Фамилия

Денисов

Имя

Денис

Отчество

Денисович

Email

den@mail.ru

Номер

телефона

13014131123

[ЗАКРЫТЬ](#) [ОБНОВИТЬ](#)

Рисунок 19 – Форма изменения работодателя

Фирма

ООО сибур

Фамилия: Денисов

Имя: Денис

Отчество: Денисович

Почта: den@mail.ru

Номер: 13014131123

Рисунок 20 – Подробная информация о работодателе

4.3. Выводы

Была разработана клиентская часть веб-сервиса. Разработаны интерфейсы для входа в систему, для работы администратора и для уборщика. Фреймворк Vue.js позволил создать логику на клиентской части. Аjaх позволил легко общаться с API. Muse UI предоставил большой, красивый и гибкий набор компонентов.

5. КОНТЕЙНЕРИЗАЦИЯ И ОРКЕСТРАЦИЯ

5.1. Описание средства контейнеризации и оркестрации

Использование контейнеров — один из наиболее активно развивающихся сегментов ИТ-рынка, ориентированный на корпоративное использование облачных услуг.

Контейнеры на рынке виртуализации появились относительно недавно. В основном благодаря технологии Docker, позволяющей запускать приложения в контейнере. Это похоже на обычную виртуальную машину, но гораздо лучше, поскольку достигается практически полная независимость от инфраструктуры и пониженное потребление ресурсами. Именно поэтому сейчас можно заметить смещение от традиционных виртуальных машин в сторону контейнеров.

Когда говорят о контейнерах, нельзя не упомянуть об оркестрации. Оркестрация — это координация взаимодействия нескольких контейнеров. Конечно, можно работать и без оркестрации — никто не запрещает создать контейнер, в котором будут запущены все необходимые процессы. Однако в этом случае вы будете лишены гибкости, масштабируемости, а также возникнут вопросы безопасности, поскольку запущенные в одном контейнере процессы не будут изолированы и смогут влиять друг на друга.

Оркестрация позволяет создавать информационные системы из множества контейнеров, каждый из которых отвечает только за одну определенную задачу, а общение осуществляется через сетевые порты и общие каталоги. При необходимости каждый такой контейнер можно заменить другим, что позволяет, например, быстро перейти на другую версию базы данных при необходимости. [5]

5.2. Контейнеризация проекта

Для контейнеризации проекта были созданы файлы Dockerfile в каждой из директорий для будущих контейнеров. Так как проект состоит из трех основных частей (база данных, фронтенд и бэкенд), то и контейнеров будет в итоге три.

Рассмотрим, как устроен проект (рис. 20)

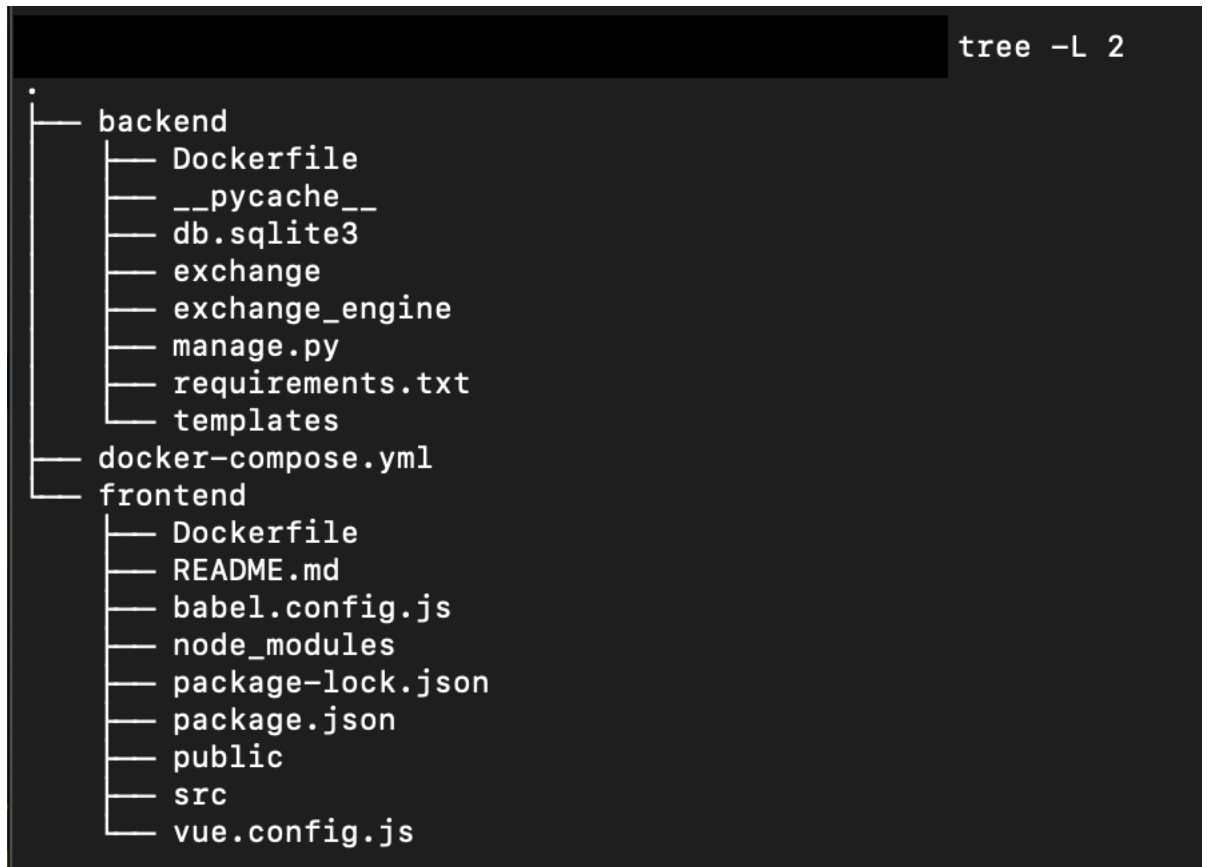


Рисунок 21 – Дерево проекта

На рисунке 22 представлен пример Dockerfile для серверной части проекта, которая лежит в директории backend. Файл серверной части включает в версию Python, указание рабочей директории и установку библиотек, необходимых для запуска проекта.

```
FROM python:3.8.2
RUN mkdir /backend
WORKDIR /backend
COPY . /backend
RUN pip install -r requirements.txt
```

Рисунок 22 – Пример файла Dockerfile для серверной части

На рис. 23 представлен пример файла Dockerfile для создания контейнера для клиентской части системы, который лежит в директории frontend. Он включает версию Node.js, указание рабочей директории и установку библиотек для запуска проекта.

```
FROM node:12

WORKDIR /client

COPY package*.json ./

RUN npm install --silent

CMD ["npm", "start"]

COPY . .
```

Рисунок 23 – Пример файла Dockerfile для клиентской части

База данных в данном случае не хранится в папке проекта, поэтому для нее не нужен отдельный файл Dockerfile. Однако контейнер для нее создается и прописывается в файле для оркестрации.

5.3. Оркестрация проекта

Для оркестрации проекта в корневой папке необходимо создать файл docker-compose.yml, который отвечает за оркестрацию. В нем описаны детали для запуска каждого контейнера, указаны порты и необходимые команды. Пример файла оркестрации представлен на рис. 25.

```

version: '3'

services:
  db:
    image: postgres
    ports:
      - "5436:5432"
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=123456
      - POSTGRES_DB=work
    volumes:
      - ./dbs/postgres-data:/var/lib/postgresql

  backend:
    container_name: backend_container
    build: ./backend
    command: bash -c "
      sleep 3 &&
      python3 manage.py makemigrations && python3 manage.py migrate &&
      python3 manage.py runserver --insecure 0.0.0.0:8000";
    volumes:
      - ./backend:/backend
    ports:
      - "8000:8000"
    depends_on:
      - db

  frontend:
    container_name: frontend_container
    build:
      context: ./frontend
      dockerfile: Dockerfile
    command: npm start --start;
    volumes:
      - ./frontend:/frontend
      - /frontend/node_modules
    ports:
      - "8080:8080"
    depends_on:
      - backend

```

Рисунок 25 – Часть файла docker-compose.yml

Первым этапом запуска проекта является сборка всех контейнеров с помощью команды *docker-compose build*. Затем необходимо выполнить команду *docker-compose up*, чтобы запустить проект.

5.4. Выводы

Проведена контейнеризация проекта для удобства запуска его в различных средах. Созданы необходимые файлы для оркестрации и запуска проекта в Docker.

ЗАКЛЮЧЕНИЕ

По итогам данной работы были выполнены следующие задачи: проанализирована предметная область и функциональные требования, создана архитектура проекта, разработана серверная и клиентская части системы, выполнена контейнеризация проекта. В результате реализована система биржи труда, которая соответствует требованиям и обладает необходимым функционалом. В системе реализовано:

- Вход и регистрация.
- Добавление личных данных и резюме соискателем.
- Просмотр, добавление, изменение и удаление работодателей.
- Фильтрация и сортировка вакансий.
- Создание заявки на вакансию соискателем.
- Просмотр опытов работы соискателей .
- Разграничение прав доступа по категориям

В рамках реализации задачи по созданию веб-сервиса были получены практические навыки работы с современными средствами разработки такими, как фреймворк Django REST, фреймворк Vue.js и библиотека Muse UI.

Для удобства запуска разработанного веб-сервиса в различных средах была использовала платформа Docker и выполнена контейнеризация проекта.

СПИСОК ЛИТЕРАТУРЫ

1. Документация Django Rest Framework [Электронный ресурс] — <https://www.django-rest-framework.org/topics/documenting-your-api/>. Дата обращения: 27.05.2020.
2. Документация PostgreSQL [Электронный ресурс] — <https://www.postgresql.org/docs/>. Дата обращения: 30.05.2020.
3. Документация Vue.js [Электронный ресурс] — <https://ru.vuejs.org/v2/guide/>. Дата обращения: 05.06.2020.
4. Документация Muse UI [Электронный ресурс] — <https://muse-ui.org/#/en-US>. Дата обращения: 06.06.2020.
5. Документация Docker [Электронный ресурс] — <https://docs.docker.com/engine/install/>. Дата обращения: 02.07.2020.