TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Инженерная школа информационных технологий и робототехники
Направление подготовки: 09.03.04 «Программная инженерия»
Отделение информационных технологий

БАКАЛАВРСКАЯ РАБОТА

Тема работы
Проектирование и разработка веб-приложения для службы по трудоустройству

УДК 004.774-047.84:331.53

Студенты

Группа	ФИО	Подпись	Дата
8К51	Немнов Роман Ильич		
8К51	Руденцов Данил Павлович		

Руководитель ВКР

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ ИШИТР ТПУ	Соколова Вероника Валерьева	к.т.н.		

КОНСУЛЬТАНТЫ ПО РАЗДЕЛАМ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОСТН ШБИП ТПУ	Подопригора Игнат Валерьевич	к.э.н.		

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ООД ШБИП ТПУ	Винокурова Галина Федоровна	к.т.н.		

ДОПУСТИТЬ К ЗАЩИТЕ:

Руководитель ООП	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ ИШИТР ТПУ	Чердынцев Евгений Сергеевич	к.т.н.		

Томск – 2019 г.

Планируемые результаты обучения по направлению 09.03.04

«Программная инженерия»

Код результата	Результат обучения (выпускник должен быть готов)
Р1	Применять базовые и специальные естественнонаучные и математические знания в области информатики и вычислительной техники, достаточные для комплексной инженерной деятельности.
Р2	Применять базовые и специальные знания в области современных информационных технологий для решения инженерных задач.
Р3	Ставить и решать задачи комплексного анализа, связанные с созданием аппаратно-программных средств информационных и автоматизированных систем, с использованием базовых и специальных знаний, современных аналитических методов и моделей.
Р4	Разрабатывать программные и аппаратные средства (системы, устройства, блоки, программы, базы данных и т. п.) в соответствии с техническим заданием и с использованием средств автоматизации проектирования.
Р5	Проводить теоретические и экспериментальные исследования, включающие поиск и изучение необходимой научно-технической информации, математическое моделирование, проведение эксперимента, анализ и интерпретация полученных данных, в области создания аппаратных и программных средств информационных и автоматизированных систем.
Р6	Внедрять, эксплуатировать и обслуживать современные программно-аппаратные комплексы, обеспечивать их высокую эффективность, соблюдать правила охраны здоровья, безопасность труда, выполнять требования по защите окружающей среды.
Р7	Использовать базовые и специальные знания в области проектного менеджмента для ведения комплексной инженерной деятельности.
Р8	Владеть иностранным языком на уровне, позволяющем работать в иноязычной среде, разрабатывать документацию, презентовать и защищать результаты комплексной инженерной деятельности.
Р9	Эффективно работать индивидуально и в качестве члена группы, состоящей из специалистов различных направлений и квалификаций, демонстрировать ответственность за результаты работы и готовность следовать корпоративной культуре организации.
Р10	Демонстрировать знания правовых, социальных, экономических и культурных аспектов комплексной инженерной деятельности.
Р11	Демонстрировать способность к самостоятельной к самостоятельному обучению в течение всей жизни и непрерывному самосовершенствованию в инженерной профессии.

Федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Инженерная школа информационных технологий и робототехники
Направление подготовки: 09.03.04 «Программная инженерия»
Отделение информационных технологий

УТВЕРЖДАЮ:
Руководитель ООП
_____
(Подпись) (Дата) (Ф.И.О.)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

В форме:

бакалаврской работы

Студентам:

Группа	ФИО
8К51	Немнову Роману Ильичу
8К51	Руденцову Данилу Павловичу

Тема работы:

Проектирование и разработка веб-приложения для службы по трудоустройству	
Утверждена приказом директора (дата, номер)	№1513/с от 26.02.2019

Срок сдачи студентом выполненной работы:	

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Исходные данные к работе *(наименование объекта исследования или проектирования; производительность или нагрузка; режим работы (непрерывный, периодический, циклический и т. д.); вид сырья или материал изделия; требования к продукту, изделию или процессу; особые требования к особенностям функционирования (эксплуатации) объекта или изделия в плане безопасности эксплуатации, влияния на окружающую среду, энергозатратам; экономический анализ и т. д.).*	Объектом проектирования в исследовательской работе является веб-приложение для службы по трудоустройству. Режим работы: клиент-сервер. Особые требования к продукту: независимость развертывания веб-приложения от конкретного хостинга; поддержка всех современных браузеров; модульная архитектура.
Перечень подлежащих исследованию, проектированию и разработке вопросов *(аналитический обзор по литературным источникам с целью выяснения достижений мировой науки техники в рассматриваемой области; постановка задачи исследования, проектирования, конструирования; содержание процедуры исследования, проектирования, конструирования; обсуждение результатов выполненной работы; наименование дополнительных разделов, подлежащих разработке; заключение по работе).*	1. Исследование предметной области и проектирование веб-приложения 2. Разработка веб-приложения 3. Анализ результатов разработки веб-приложения 4. Финансовый менеджмент 5. Социальная ответственность

Перечень графического материала *(с точным указанием обязательных чертежей)*	1. Модель работы веб-приложения (диаграммы в нотациях IDEF0, IDEF3) 2. Диаграмма EPC 3. Диаграмма BPMN процесса обработки и визуализации данных 4. Диаграмма вариантов использования 5. Диаграммы деятельности и активности 6. Рисунки, демонстрирующие результаты 7. Диаграмма Ганта

Консультанты по разделам выпускной квалификационной работы
(с указанием разделов)

Раздел	Консультант
Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	Подопригора Игнат Валерьевич
Социальная ответственность	Винокурова Галина Федоровна
Названия разделов, которые должны быть написаны на русском и иностранном языках:	

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	

Задание выдал руководитель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ ИШИТР ТПУ	Соколова Вероника Валерьевна	к.т.н.		

Задание приняли к исполнению студенты:

Группа	ФИО	Подпись	Дата
8К51	Немнов Роман Ильич		
8К51	Руденцов Данил Павлович		

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Инженерная школа информационных технологий и робототехники
Направление подготовки: 09.03.04 «Программная инженерия»
Уровень образования бакалавриат
Отделение информационных технологий
Период выполнения осенний/весенний семестр 2018/2019 учебного года

Форма представления работы:

бакалаврская работа

КАЛЕНДАРНЫЙ РЕЙТИНГ-ПЛАН
выполнения выпускной квалификационной работы

Срок сдачи студентом выполненной работы:	

Дата контроля	Название раздела (модуля) / вид работы (исследования)	Максимальный балл раздела (модуля)
	Глава 1. Исследование предметной области и проектирование веб-приложения	25
	Глава 2. Разработка веб-приложения	25
	Глава 3. Анализ результатов разработки веб-приложения	20
	Глава 4. Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	15
	Глава 5. Социальная ответственность	15

Составил преподаватель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ ИШИТР ТПУ	Соколова Вероника Валерьевна	к.т.н.		

СОГЛАСОВАНО:

Руководитель ООП	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ ИШИТР ТПУ	Чердынцев Евгений Сергеевич	к.т.н.		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ»

Студентам:

Группа	ФИО
8К51	Немнову Роману Ильичу
8К51	Руденцову Данилу Павловичу

Школа	ИШИТР	Отделение школы (НОЦ)	ОИТ
Уровень образования	Бакалавриат	Направление/специальность	09.03.04 Программная инженерия

Исходные данные к разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»:	
1. *Стоимость ресурсов научного исследования (НИ): материально-технических, энергетических, финансовых, информационных и человеческих* 2. *Нормы и нормативы расходования ресурсов* 3. *Используемая система налогообложения, ставки налогов, отчислений, дисконтирования и кредитования*	*Амортизационные затраты на спецоборудование – 27 222рубля;* *Затраты на основную и дополнительную з/п – 476569+ 74525 рублей;* *Затраты на отчисление во внебюджетные фонды – 153455 рублей;* *Накладные расходы – 102 022 рубля.*
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. *Оценка коммерческого потенциала, перспективности и альтернатив проведения НИ с позиции ресурсоэффективности и ресурсосбережения*	*Описание потенциальных потребителей;* *Анализ технических конкурентных решений;* *SWOT-анализ.*
2. *Планирование и формирование бюджета научных исследований*	*Структура работ в рамках научного исследования* *Определение трудоемкости выполнения работ и разработка графика проведения научного исследования* *Бюджет проекта*
3. *Определение ресурсной (ресурсосберегающей), финансовой, бюджетной, социальной и экономической эффективности исследования*	*Определение интегрального финансового показателя разработки* *Определение интегрального показателя ресурсоэффективности разработки* *Определение интегрального показателя эффективности*
Перечень графического материала *(с точным указанием обязательных чертежей)***:**	
1. *Оценка конкурентоспособности технических решений* 2. *Матрица SWOT* 3. *График проведения и бюджет НИ* 4. *Оценка ресурсной, финансовой и экономической эффективности НИ*	

Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОСТН ШБИП ТПУ	Подопригора Игнат Валерьевич	Кандидат экономических наук		

Задание принял к исполнению студенты:

Группа	ФИО	Подпись	Дата
8К51	Немнов Роман Ильич		
8К51	Руденцов Данил Павлович		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА «СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Студентам:

Группа	ФИО
8К51	Немному Роману Ильичу
8К51	Руденцову Данилу Павловичу

Школа	ИШИТР	Отделение (НОЦ)	ОИТ
Уровень образования	Бакалавриат	**Направление/специальность**	09.03.04 Программная инженерия

Тема ВКР:

Проектирование и разработка веб-приложения для службы по трудоустройству	
Исходные данные к разделу «Социальная ответственность»:	
1. Характеристика объекта исследования (вещество, материал, прибор, алгоритм, методика, рабочая зона) и области его применения	Объект исследования – веб-приложение, автоматизирующее процессы интернет-рекрутинга. Рабочее место – рабочий стол с персональным компьютером в учебной аудитории.
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Правовые и организационные вопросы обеспечения безопасности: – специальные (характерные при эксплуатации объекта исследования, проектируемой рабочей зоны) правовые нормы трудового законодательства; – организационные мероприятия при компоновке рабочей зоны.	– Рабочее место при выполнении работ сидя регулируется ГОСТом 12.2.032 –78 – Организация рабочих мест с электронно-вычислительными машинами регулируется СанПиНом 2.2.2/2.4.1340 – 03 – Использование персональных данных пользователей регулируется Федеральным законом №152 «О персональных данных»
2. Производственная безопасность: 2.1. Анализ выявленных вредных и опасных факторов 2.2. Обоснование мероприятий по снижению воздействия	– Отклонение показателей микроклимата – Отсутствие или недостаток естественного света – Недостаточная освещенность рабочей зоны – Повышенный уровень электромагнитных излучений
3. Экологическая безопасность:	Анализ негативного воздействия на окружающую природную среду: утилизация компьютеров, смартфонов, оргтехники.
4. Безопасность в чрезвычайных ситуациях:	Возможные чрезвычайные ситуации: ● Пожар ● Землетрясение
Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ООД ШБИП ТПУ	Винокурова Галина Федоровна	Кандидат технических наук, доцент		

Задание приняли к исполнению студенты:

Группа	ФИО	Подпись	Дата
8К51	Немнов Роман Ильич		
8К51	Руденцов Данил Павлович		

Реферат

Выпускная квалификационная работа содержит 168 страниц, 76 рисунков, 21 таблицу, 30 источников и 4 приложения.

Ключевые слова: система трудоустройства, микросервисная архитектура, дни карьеры, электронное резюме, электронная вакансия, Docker, React.

Объектом исследования является веб-приложение для службы по трудоустройству.

Цель работы – спроектировать и разработать веб-приложение для службы по трудоустройству.

В процессе исследования проводился опрос среди студентов ТПУ с целью выяснения конкурирующих программных продуктов в сфере интернет-рекрутинга с последующим аналитическим обзором

В результате исследования было выявлено, что существующие программные продукты не предоставляют дополнительного функционала, такого как, оповещения о «Днях карьеры» с помощью SMS-, E-mail или push-уведомлений, для молодых специалистов или учащихся старших курсов при трудоустройстве.

Экономическая эффективность работы заключается в актуальности поставленной задачи, применению современных программных средств, реализации дополнительных функциональных возможностей для молодых специалистов.

В дальнейшем планируется настроить интеграцию с дополнительными сервисами, позволяющими соискателям подтвердить уровень владения каким-либо навыком, расширить возможности клиентского интерфейса, в том числе, реализовать возможность использования веб-приложения иностранными студентами.

Оглавление

Перечень терминов и условных обозначений

1. **UML** – унифицированный язык моделирования, позволяющий раскрывать информацию об устройстве предметной области в удобном графическом представлении с помощью стандартизованных нотаций;
2. **Веб-сервер** – сервер, который использует протокол HTTP для передачи статического и динамического контента по запросу пользователей;
3. **Аутентификация** – процедура сравнения введенных данных с записанными на внешнее хранилище с целью идентификации пользователя;
4. **Логирование** – процедура записи параметров входящих запросов на внешнее хранилище;
5. **HTTP** – протокол передачи данных между клиентом и сервером в виде гипертекстовых документов формата HTML;
6. **OSI** – семиуровневая модель взаимодействия различных устройств в сети Интернет;
7. **Прокси-сервер** – сервер, выступающий в роли посредника между клиентом и обрабатывающем его запросы сервером;
8. **A/B тестирование** – вид маркетингового исследования, при котором определенной части пользователей выдается более новая версия сервиса или графического интерфейса и высчитываются метрики;
9. **URL** – унифицированный указатель ресурса, определяющий его конкретное местоположение;
10. **Виртуальная машина** – программная среда, эмулирующая работу какой-либо операционной системы, не отличимая с точки зрения запущенных процессов от обычной ОС;
11. **AMQP** – основанный на очередях открытый протокол передачи и приема сообщений между компонентами в децентрализованной системе;
12. **CRUD** – акроним для описания типовых операций с данными, от англ. – Create, Remove, Update, Delete;

13. **SQL** – язык структурированных запросов к реляционным базам данных для выполнения операций реляционной алгебры, в том числе CRUD;
14. **СУБД** – программное обеспечение, позволяющее создавать и управлять базами данных;
15. **ORM** – программное решение, позволяющее связать схему базы данных с представлениями объектов в объектно-ориентированных языках программирования;
16. **Транзакция** – атомарная с точки зрения СУБД последовательность операций, переводит базу из одного устойчивого состояния в другое;
17. **JSON** – формат обмена данными, запись которых идентична представлению объектов в JavaScript;
18. **Кэш** – буфер с высокоскоростным доступом, содержащий наиболее часто используемую информацию;
19. **DOM** – представление содержимого веб-страницы в виде иерархической структуры;
20. **Фреймворк** – программное обеспечение, представляющее шаблон для программной платформы и облегчающее процесс разработки;
21. **Транспиляция** – процесс преобразования кода, написанного на новых версиях языка в стандартный, поддерживаемый вариант с сохранением функциональности;
22. **Оркестрация –** автоматическая координация, размещение и управление над сложными проектами;
23. **GUID –** идентификатор длиной 128 бит, является статистически уникальным;
24. **Webpack –** современная система сборки клиентского кода, поддерживающая возможности гибкой настройки параметров;
25. **NPM** – распространенный менеджер пакетов, входящий в состав Node.js.

Введение

За последние 10 лет средний по стране уровень безработицы в России не опускался ниже 5 %, при этом в отдельных регионах этот показатель может быть значительно выше – 11 % в Северо-Кавказском Федеральном округе и 6,8 % в Сибирском, по этой причине безработица является важной социальной проблемой [1]. Ещё одним негативным фактором в сфере трудовой занятости является невозможность трудоустройства небольшой доли экономически активного населения по специальности, полученной в учебных заведениях.

Особенно остро эта проблема касается учащихся старших курсов и молодых специалистов, окончивших обучение, ввиду отсутствия трудового опыта, который является одним из основных критериев для работодателей при трудоустройстве.

Наличие у большинства граждан производительных персональных компьютеров или мобильных устройств делает возможным создание веб-приложения, способного автоматизировать процессы интернет-рекрутинга, что позволит работодателям и соискателям в электронном виде быстрее и точнее составлять резюме и вакансии, а также более оперативно устанавливать между собой деловые контакты.

При реализации дополнительных возможностей для упрощения процесса трудоустройства молодым специалистам и внедрении приложения в системы образовательных учреждений, можно ожидать увеличения количества трудоустроенных граждан, и как следствие, снижение уровня безработицы, что благоприятно скажется на общем социальном фоне в Российской Федерации.

Глава 1. Исследование предметной области и проектирование веб-приложения

1.1. Общая информация

Разрабатываемое веб-приложение предназначено для автоматизации процессов в сфере интернет-рекрутинга, где основными действующими лицами являются соискатели и работодатели. Принимая во внимание тот факт, что веб-приложение ориентировано на интеграцию с учебными заведениями, следует отдельно выделить дополнительное действующее лицо – менеджера учебного заведения. Основными документами, которыми оперируют участники процесса трудоустройства, являются резюме и вакансии, дополнительным документом, с которым работает менеджер учебного заведения, является информационная брошюра так называемых «Дней карьеры».

Основными полями резюме является:

- ФИО соискателя.
- Контакты – электронная почта и номер мобильного телефона.
- Общая информация – возраст, пол, место проживания.
- Ключевые навыки соискателя, в том числе владение иностранными языками, имеющиеся категории прав.
- Информация об образовании и прошлом опыте работы.
- Рецензии, отзывы, сертификаты и прочие достижения.
- Предпочтения соискателя – желаемый график работы, готовность к переездам и пр.

Основными полями вакансии являются:

- Предлагаемая должность.
- Условия и график работы.
- Минимальная и максимальная планка оклада труда.
- Требования к соискателю.
- Контактная информация.

- Описание обязанностей.
- Дополнительная информация.

Информационная брошюра дней карьеры содержит:

- Время и место проведения мероприятия.
- Список участвующих компаний.
- Краткое описание программы проведения.
- Дополнительную информацию.

Результатом процессов, происходящих в рассматриваемой предметной области, является трудоустройство соискателя. На рисунке 1 представлена диаграмма в нотации IDEF0, которая позволяет понять входы, выходы, механизмы и элементы управления, задействованные в этом процессе при использовании веб-приложения.

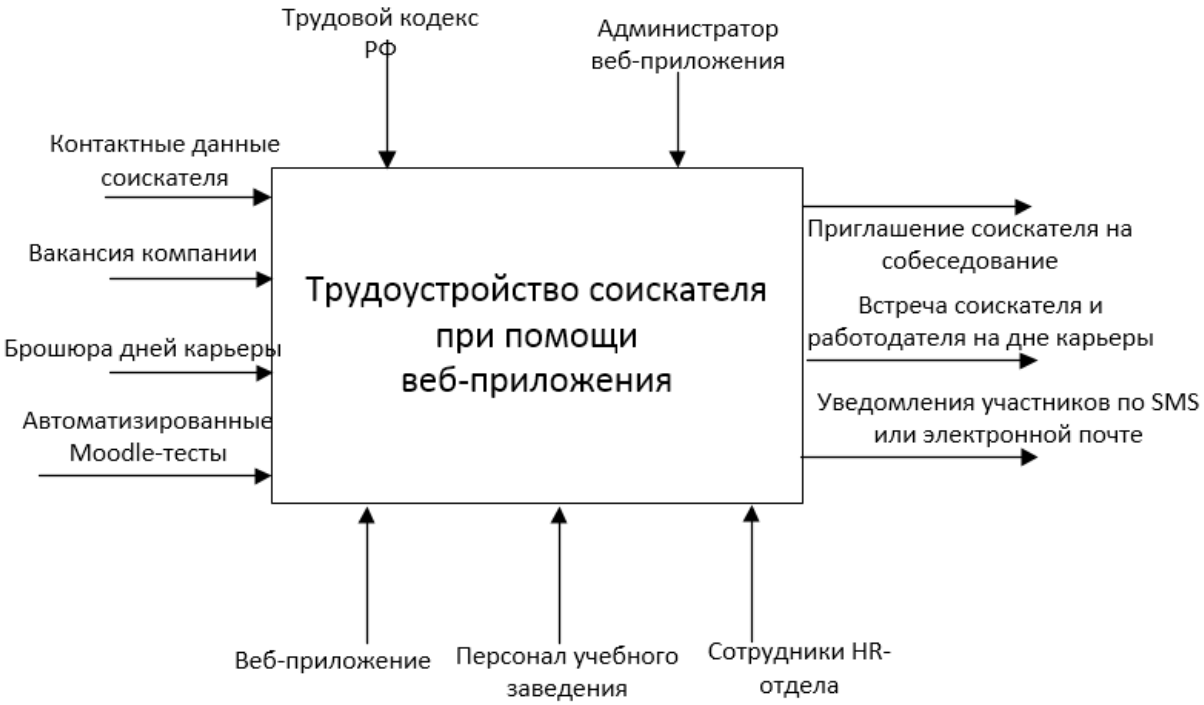

Рисунок 1. Процесс трудоустройства представлен в виде контекстной диаграммы

Как видно из диаграммы выше, процесс трудоустройства может завершиться двумя способами – соискатель откликается на резюме, и менеджер компании приглашает его на собеседование, или встреча соискателя и работодателя может произойти на «Дне карьеры», благодаря

участию в процессе менеджера учебного заведения. Второй способ является дополнительной возможностью для трудоустройства молодых специалистов.

1.2. Объекты предметной области и их взаимодействие с системой

К выявленному ранее списку участников процесса трудоустройства следует добавить ещё одного, специфичного для программной реализации веб-приложения – неавторизованного пользователя.

Таким образом, любой пользователь системы в определенный момент времени может занимать одну из следующих ролей:

- Неавторизованный пользователь – обладает ограниченными функциональными возможностями, стандартная роль для клиента веб-приложения.
- Соискатель – добавляет, изменяет и удаляет резюме.
- Менеджер компании – добавляет, изменяет и удаляет вакансии.
- Менеджер учебного заведения – добавляет, изменяет и удаляет информацию о днях карьеры.
- Администратор веб-приложения – обладает всеми возможностями соискателя, менеджера учебного заведения и менеджера компании.

1.2.1. Наследование функциональных возможностей

Некоторые возможности пользователей по взаимодействию с веб-приложением должны быть доступны вне зависимости от занимаемой ими роли, при этом неавторизованным пользователям доступны несколько возможностей, которыми не может воспользоваться ни один авторизованный пользователь. Для демонстрации наследования функциональных возможностей пользователей веб-приложения на рисунке 2 изображена соответствующая иерархия.

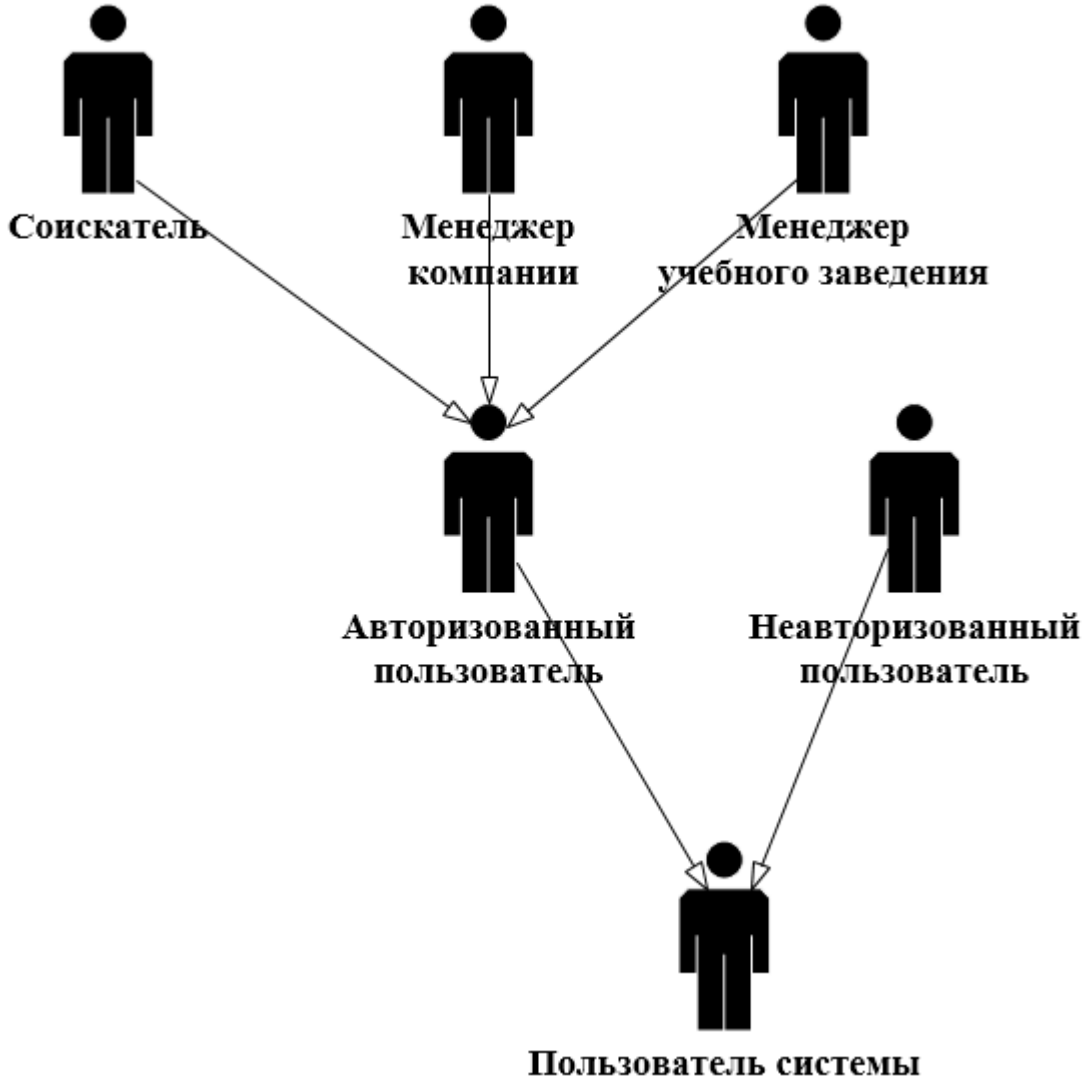

Рисунок 2. Общие функциональные возможности главных пользователей веб-приложения

Следует учитывать, что роли авторизованного пользователя и пользователя системы недоступны для клиентов веб-приложения и являются служебными.

Диаграмма вариантов использования, показанная рисунке 3, содержит перечисление возможностей, доступным всем авторизованным и неавторизованным пользователям соответственно.

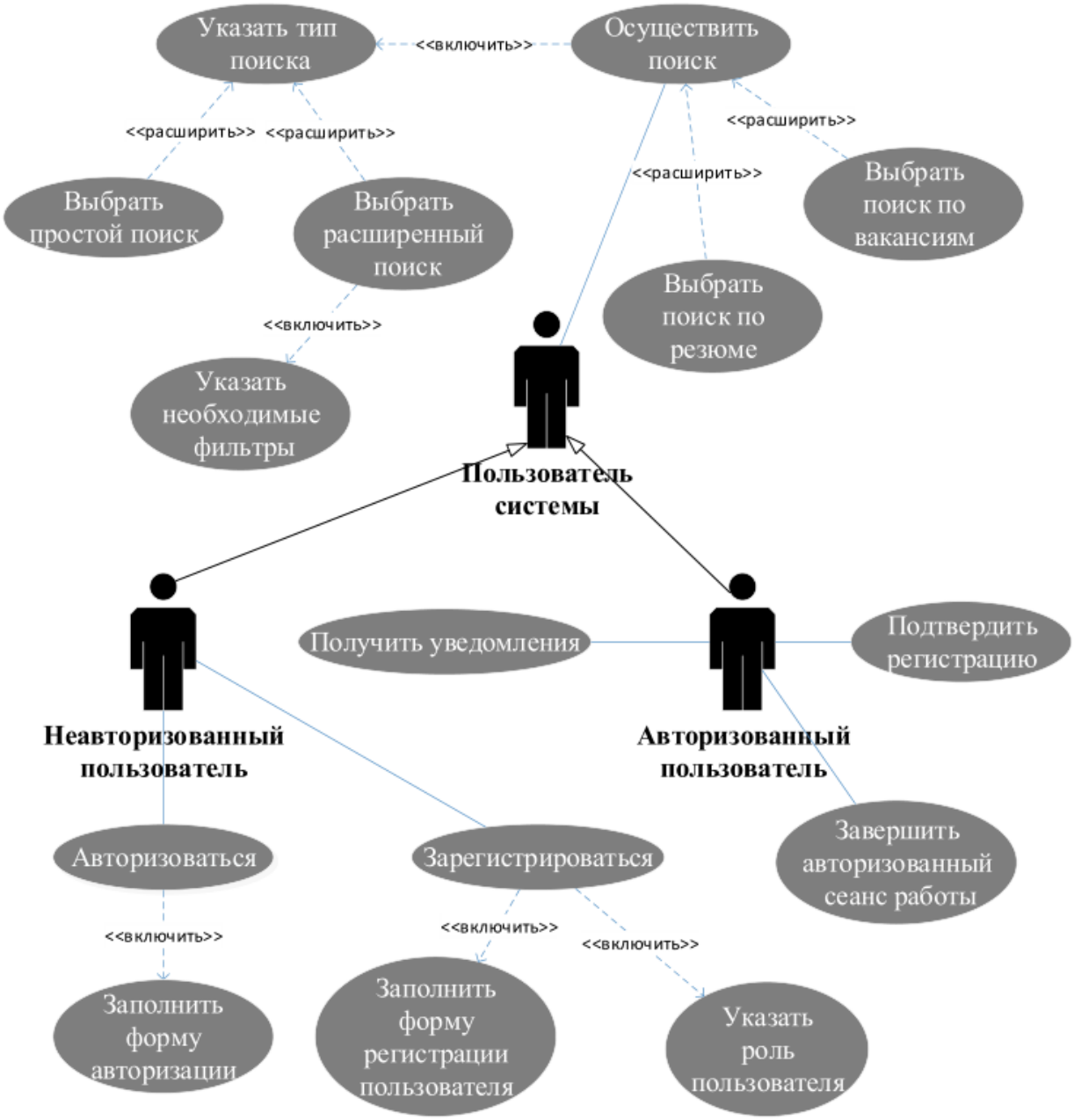

Рисунок 3. Функциональные возможности, доступные сразу нескольким ролям пользователей

1.2.2. Функциональные возможности соискателя

Основными возможностями соискателя в веб-приложении являются:

- Произвести CRUD-операции со списком резюме.
- Редактировать профиль.
- Оставить отклик на вакансию.
- Просмотреть статус отклика.
- Настроить список предпочитаемых специализаций для дней карьеры.

- Пройти автоматизированные тесты, прикрепленные к вакансии.
- Получить статистику просмотров резюме.
- Получить уведомление о «Днях карьеры».

Диаграмма вариантов использования, показанная на рисунке 4, позволяет увидеть все возможности соискателя по взаимодействию с веб-приложением.

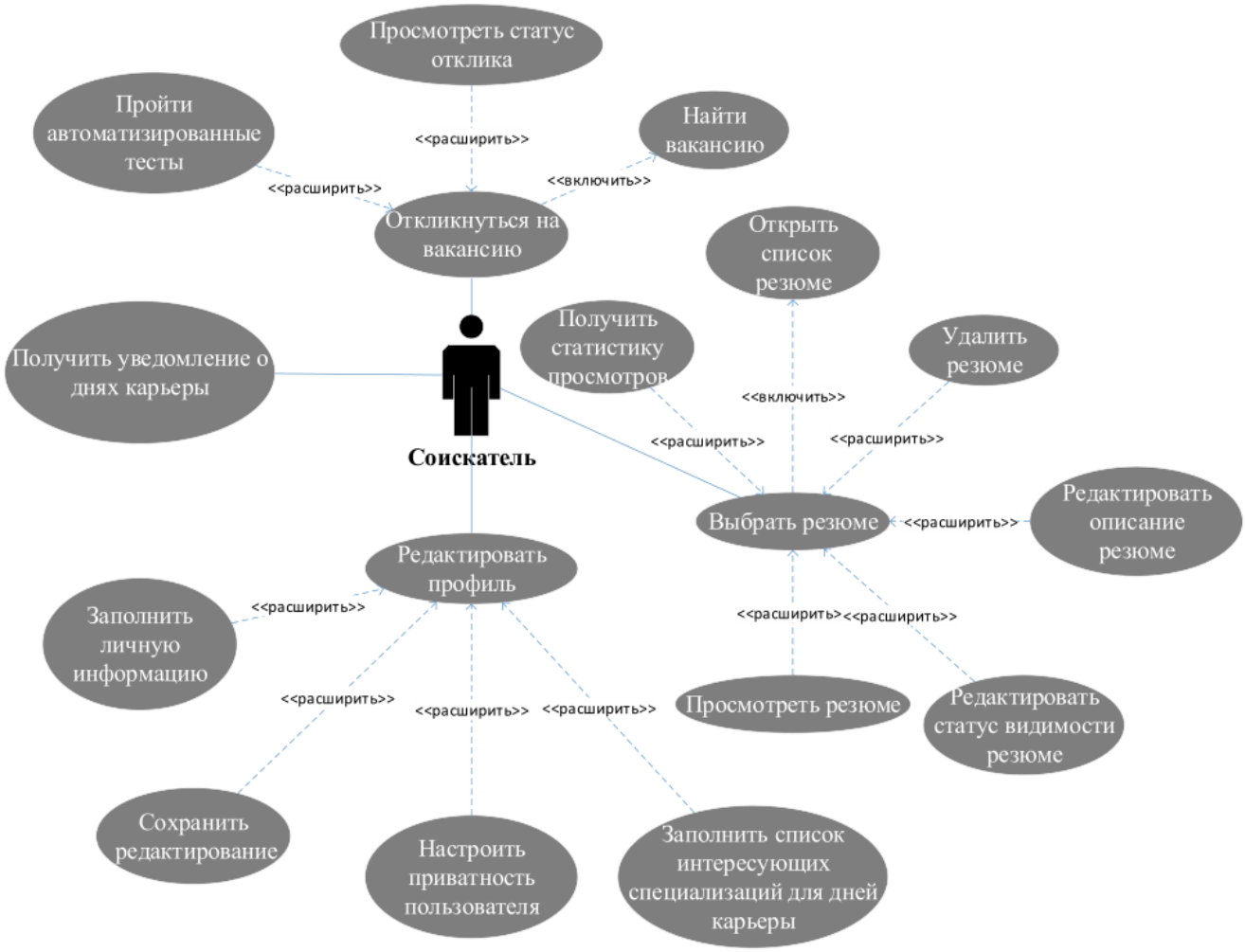

Рисунок 4. Сценарии использования приложения соискателем

Как видно из диаграммы, веб-приложение позволяет пользователю гибко редактировать как каждое резюме по отдельности, так и весь профиль целиком.

1.2.3. Функциональные возможности менеджера компании

Основными возможностями менеджера компании в веб-приложении являются:

- Произвести CRUD-операции со списком вакансий.
- Редактировать профиль компании.
- Составить автоматизированные тесты для соискателей через платформу Moodle.
- Просмотреть подробную информацию о результатах прохождения тестов.
- Ответить на отклик соискателя.
- Воспользоваться премиум-услугами для продвижения вакансии.

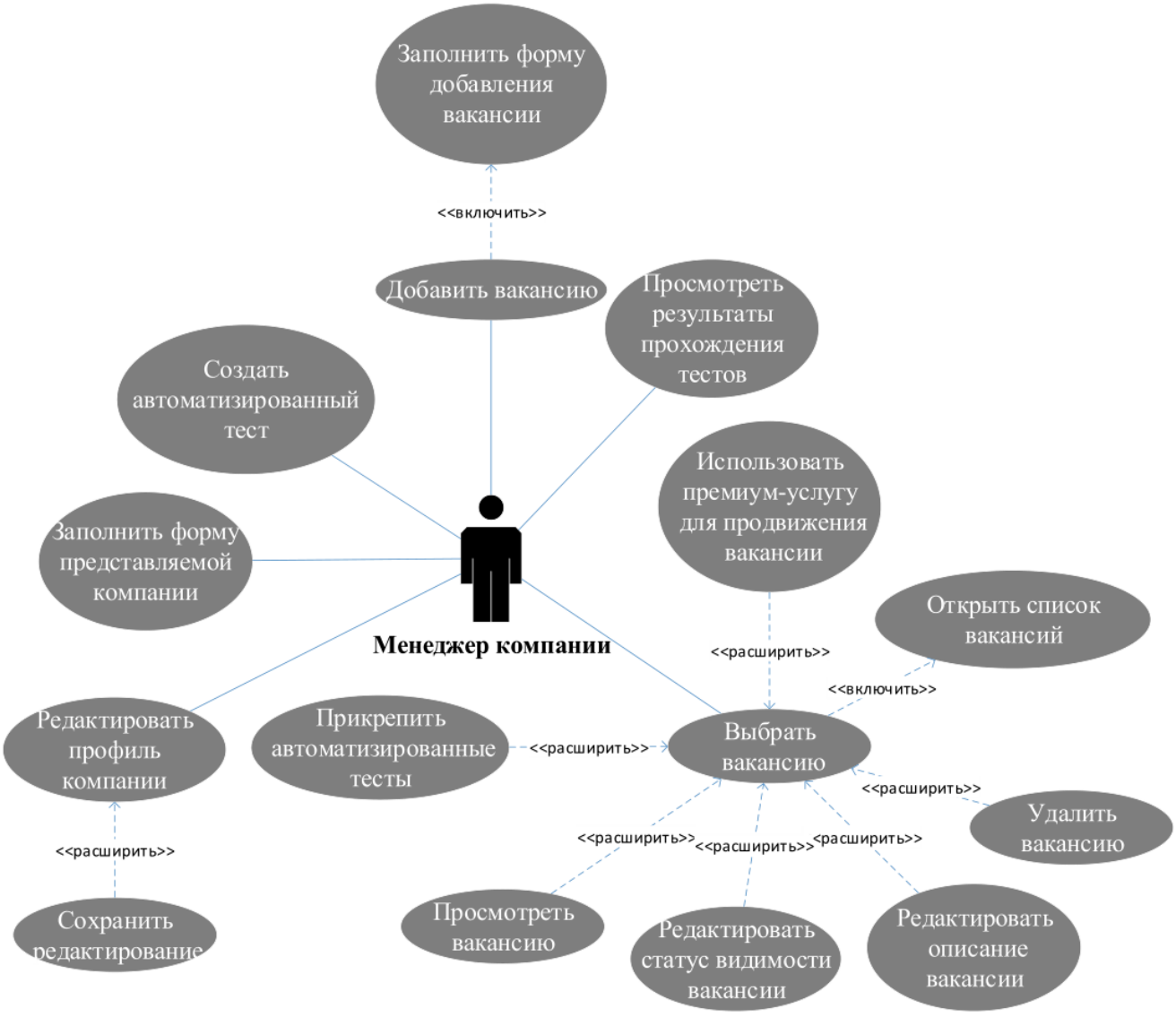

Рисунок 5. Сценарии использования приложения менеджером компании

Из диаграммы выше можно понять, что веб-приложение предлагает менеджеру компании гибкие настройки по редактированию информации отдельно взятой вакансии и компании в целом.

1.2.4. Функциональные возможности менеджера учебного заведения

Возможности этого пользователя системы ограничены CRUD-операциями с днями карьеры и опциональным добавлением списка специализаций, изучая которые, у студентов будет интерес для посещения этого мероприятия. Более детально это показано на рисунке 6, содержащей диаграмму вариантов использования веб-приложения менеджером учебного заведения.

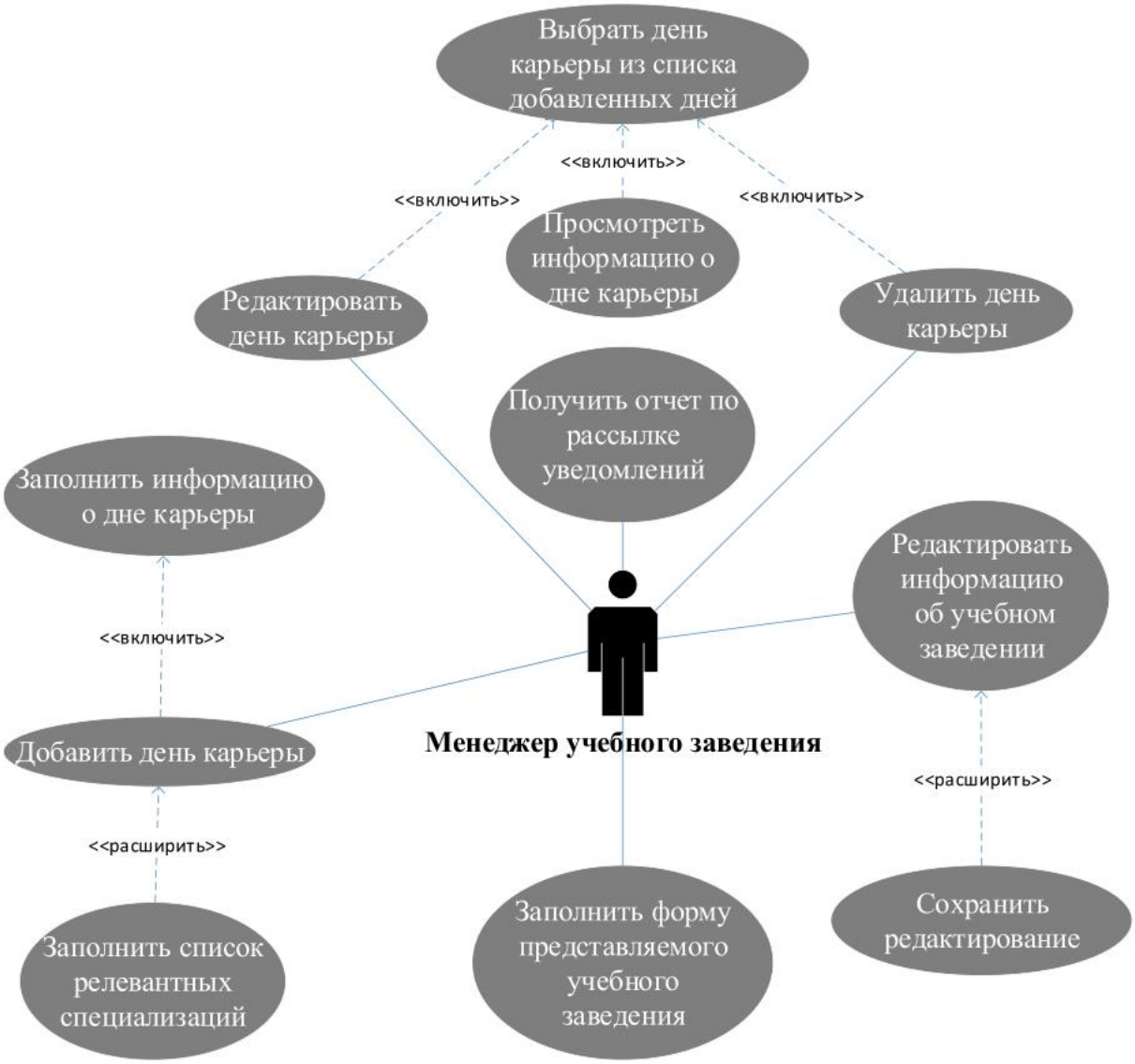

Рисунок 6. Сценарии использования приложения менеджером учебного заведения

1.3. Обзор конкурирующих программных продуктов

Средняя доля экономически активного населения в странах мира составляет более 60 % [2], что приводит к огромному количеству потенциальных клиентов систем, автоматизирующих процессы трудоустройства. По этой причине на момент написания работы существуют десятки крупных сервисов по трудоустройству, в том числе специализированные социальные сети, например, «LinkedIn» [3].

1.3.1. Выявление списка конкурентов

Разработка веб-приложения велась в предположении интеграции в системы учебных заведений, и одним из первых таких заведений стал бы Томский Политехнический Университет. Чтобы сузить круг конкурентов и выявить ключевые качества, которыми должно обладать приложение, чтобы быть более предпочтительным для молодых специалистов и студентов, было принято решение провести опрос среди учащихся ТПУ старших курсов и выяснить, какие системы трудоустройства знакомы им. По результатам опроса, наиболее известными системами стали Headhunter[4] и веб-сайт Зарплата.ру [5]. Далее необходимо проанализировать этих конкурентов и сравнить с разрабатываемым веб-приложением (рабочее название – JobObserver).

1.3.2. Сравнение JobObserver с Headhunter

Headhunter существует на рынке более 19 лет, и за это время его база накопила более 30 млн. резюме [4]. Поимо сайта, компания поддерживает также и мобильные приложения под популярные мобильные операционные системы. Численность штаба работников и техническая оснащенность компании значительно превышает аналогичные показатели у JobObserver, что приводит к затруднительной конкуренции за среднестатического клиента приложений по трудоустройству. Однако если обратить внимание на долю клиентов, приходящихся на молодых специалистов, можно заметить, что разрабатываемое приложение выгодно отличает возможность

автоматического оповещения о предстоящих днях карьеры. Более подробное сравнение по различным критериям представлено в главе, посвященной оценке коммерческого потенциала.

1.3.3. Сравнение JobObserver с сайтом Зарплата.ру

Зарплата.ру – сайт, основанный в 2015 году, база резюме составляет около 6.5 млн. [5]. Несмотря на широкую распространенность сайта в Сибири и Урале, этот конкурент представляет меньшую угрозу, чем Headhunter, так как его мобильное приложение работает только под Android, а сама система ориентирована только на взаимодействие работодателей и соискателей, обеспечивая минимальный функционал. Против этого конкурента у разрабатываемого приложения есть два ключевых преимущества – оповещение учащихся о днях карьеры и возможность менеджерам компании создавать пользовательские тесты на платформе Moodle, что позволит быстрее идентифицировать соискателей, чей уровень подготовки не соответствует требованиям компании.

1.4. Отличительные особенности разрабатываемой системы

Ввиду того, что конкуренты существуют на рынке несколько лет и обладают богатым техническим оснащением, единственным способом для эффективной конкуренции остается реализация в веб-приложении дополнительных возможностей.

1.4.1. Возможность оповещения учащихся об университетских «Днях карьеры»

1.4.1.1. Обоснование реализации возможности оповещения

Составление и отправка резюме заинтересованным работодателям является не единственным способом трудоустройства, молодым специалистам также следует обратить внимание и на другой метод получения работы – участие в днях карьеры, проводимых учебными

заведениями. Однако, анализ посещаемости таких мероприятий свидетельствует о низкой явке учащихся на дни карьеры.

Из-за низкой посещаемости студентов компаниям становится сложно подобрать подготовленных учащихся на открытые вакансии, что снижает их мотивацию на дальнейшее участие. Одной из основных причин низкой явки является использование устаревших технологий для информирования учащихся о мероприятиях – из-за учебной нагрузки студент может забыть о предстоящих днях карьеры или вовсе не знать о мероприятии, если оно проводится в другом учебном заведении.

За счет внедрения современных технологий в систему информирования студентов о предстоящих мероприятиях можно повысить посещаемость дней карьеры и, как следствие, количество составляемых на них резюме.

1.4.1.2. Особенности проведения дней карьеры без веб-приложения

Низкая явка на дни карьеры – комплексная проблема, для понимания которой следует разбить её на составляющие и установить причинно-следственные связи.

Декомпозировав исходную проблему, можно получить следующие её составляющие:

- Ограниченный список участвующих компаний.
- Коммуникационные сложности.
- Слабая мотивация учащихся.
- Неосведомленность учащихся.

У каждой проблемы есть причины возникновения, которые показаны на рисунке 7 в виде диаграммы Исикавы.

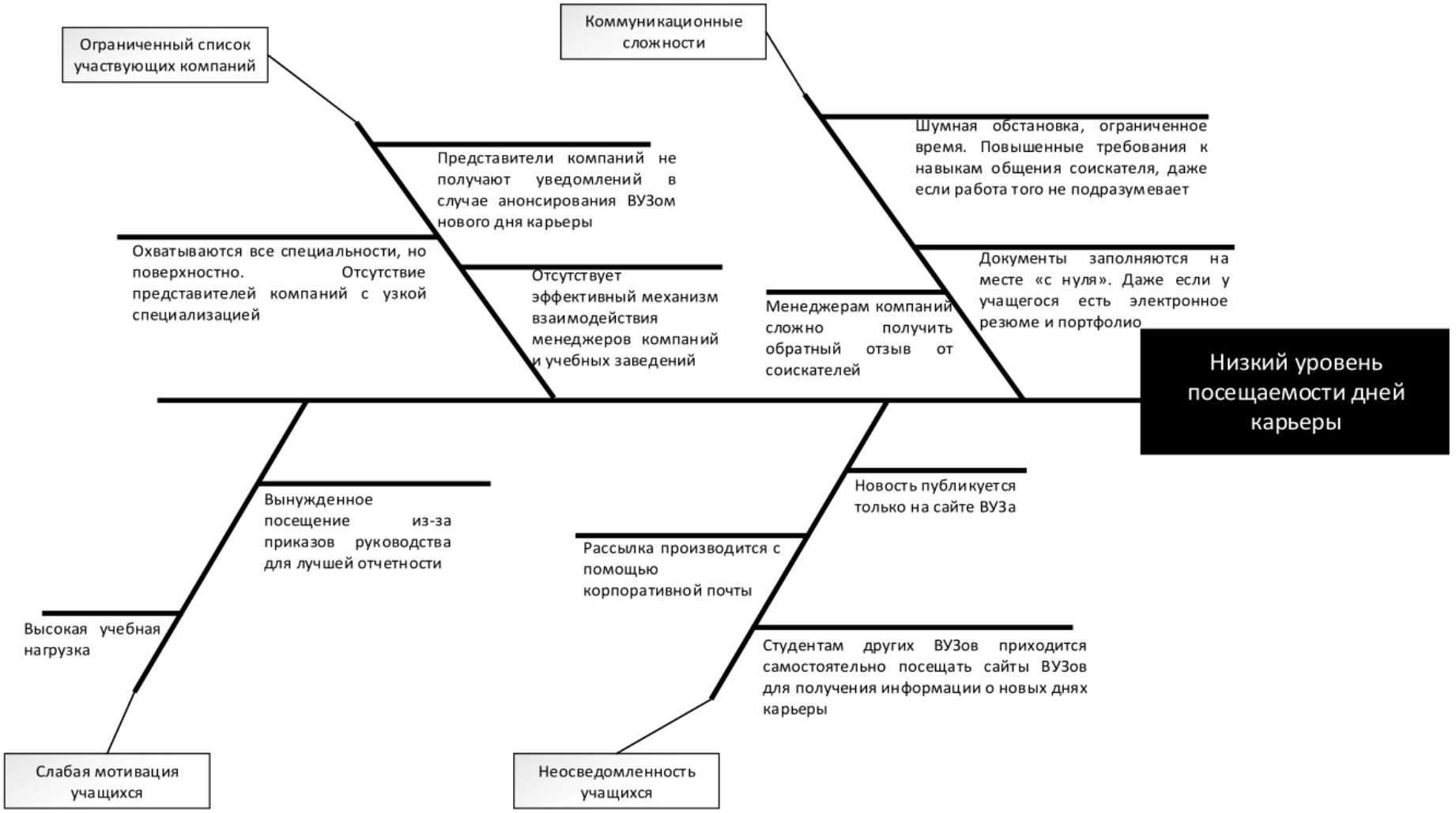

Рисунок 7. Диаграмма Исикавы для выявления причинно-следственных связей

Разрабатываемое веб-приложение призвано решить проблему низкой посещаемости, снизив неосведомленность учащихся. Необходимо произвести функциональную декомпозицию процесса организации «Дней карьеры» с целью выяснения конкретного функционального блока, который приводит к недостаточной осведомленности учащихся. На рисунке 8 исходный процесс показан в виде контекстной диаграммы.

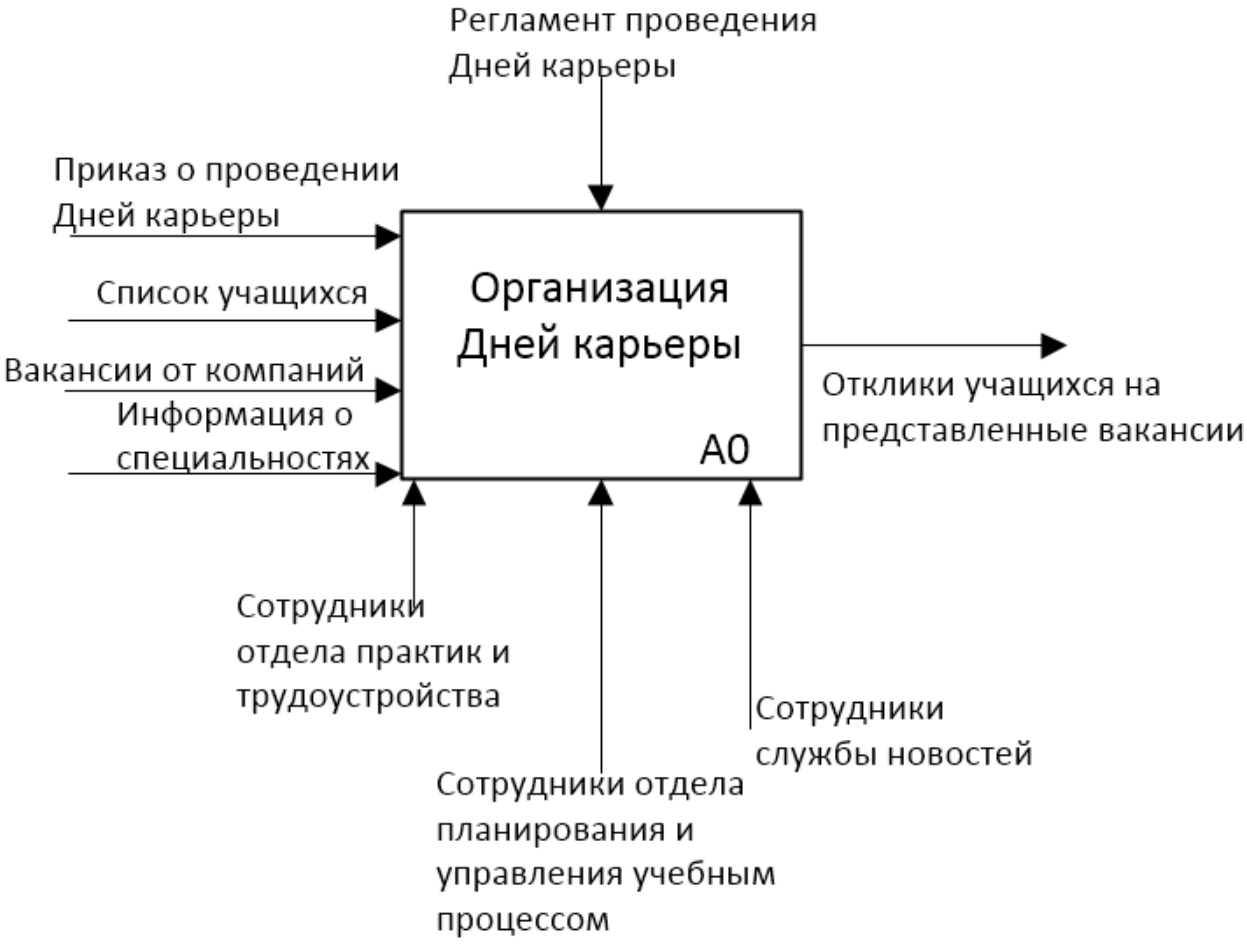

Рисунок 8. Процесс организации дней карьеры представлен в виде контекстной диаграммы

Как видно, исходный процесс организации дней карьеры не содержит никаких специальных механизмов, задействованы только сотрудники учебного заведения.

Рисунок 9 содержит декомпозицию контекстной диаграммы, с помощью которой можно проанализировать входы, выходы, механизмы и управляющие элементы каждого составляющего блока.

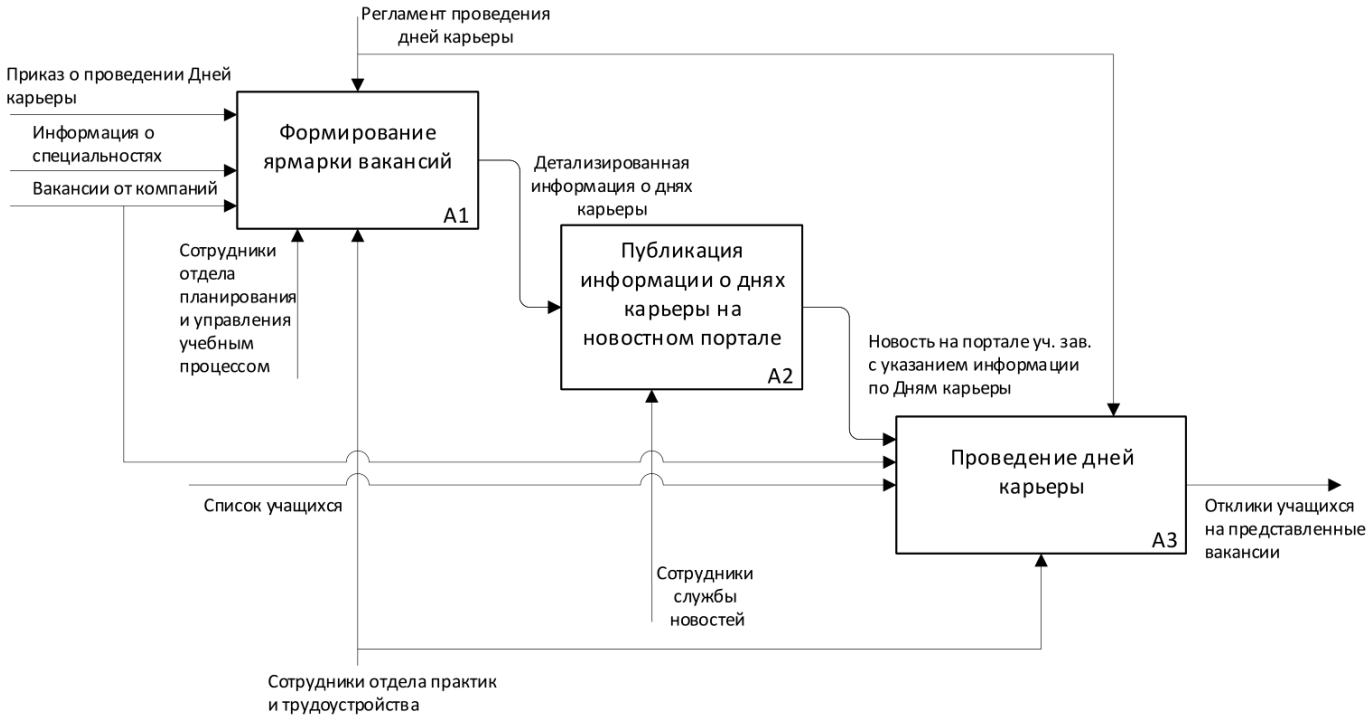

Рисунок 9. Процесс организации дней карьеры разбит на 3 функциональных блока

Очевидно, что блоки *A*1 и *A*3 полностью регламентируются учебным заведением и никак не могут быть улучшены с помощью приложения. Блок *A*2 несет ответственность за осведомленность учащихся, единственным механизмом этого блока являются сотрудники службы новостей учебного заведения, дополнительные программно-технические средства не задействуются. Внедрение веб-приложения в систему учебных заведений может существенно изменить ход процессов, происходящих на этом этапе, и повысить результирующую осведомленность.

В результате анализа особенностей проведения «Дней карьеры» с использованием только корпоративной почты и новостного портала, было принято решение реализовать модуль подписки учащихся таким образом, чтобы на вход поступала информация о планируемых «Днях карьеры».

1.4.1.3. Планируемый процесс проведения дней карьеры при использовании веб-приложения

Продемонстрировать изменения в процессе можно с помощью диаграммы IDEF3, представленной на рисунке 10.

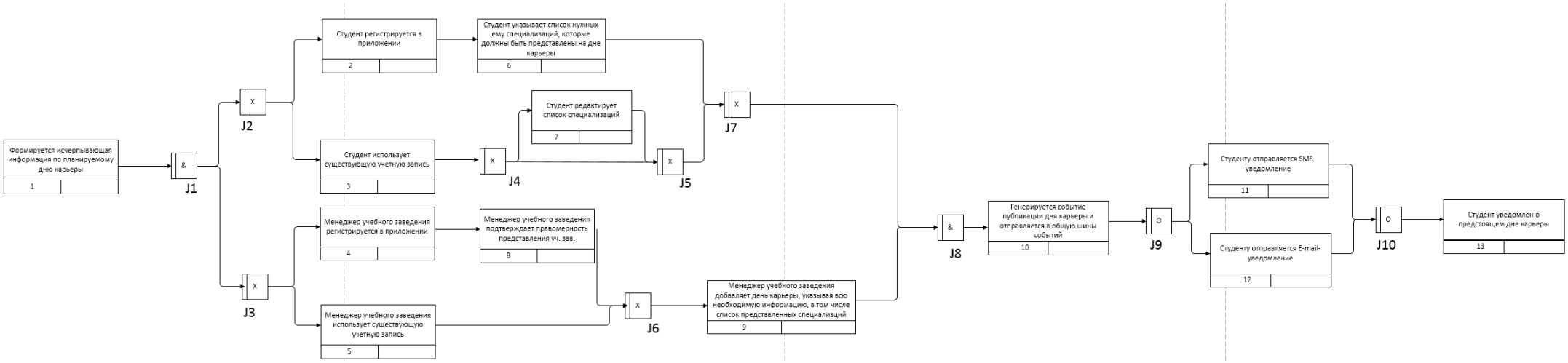

Рисунок 10. Декомпозиция процесса уведомления учащихся о днях карьеры

Как видно, модуль позволяет оповещать учащегося как по SMS, так и через электронную почту. Пара операторов логического «И» *J*1 и *J*9 означает, что студент получит уведомление, только если он указал список интересующих его специализаций, а менеджер учебного заведения опубликовал информацию о новых «Днях карьеры».

Диаграмма BPMN этого же процесса, продемонстрированная на рисунке 11, раскрывает информацию о том, какой именно пользователь веб-приложения выполняет действия по созданию условий для уведомления учащегося о днях карьеры.

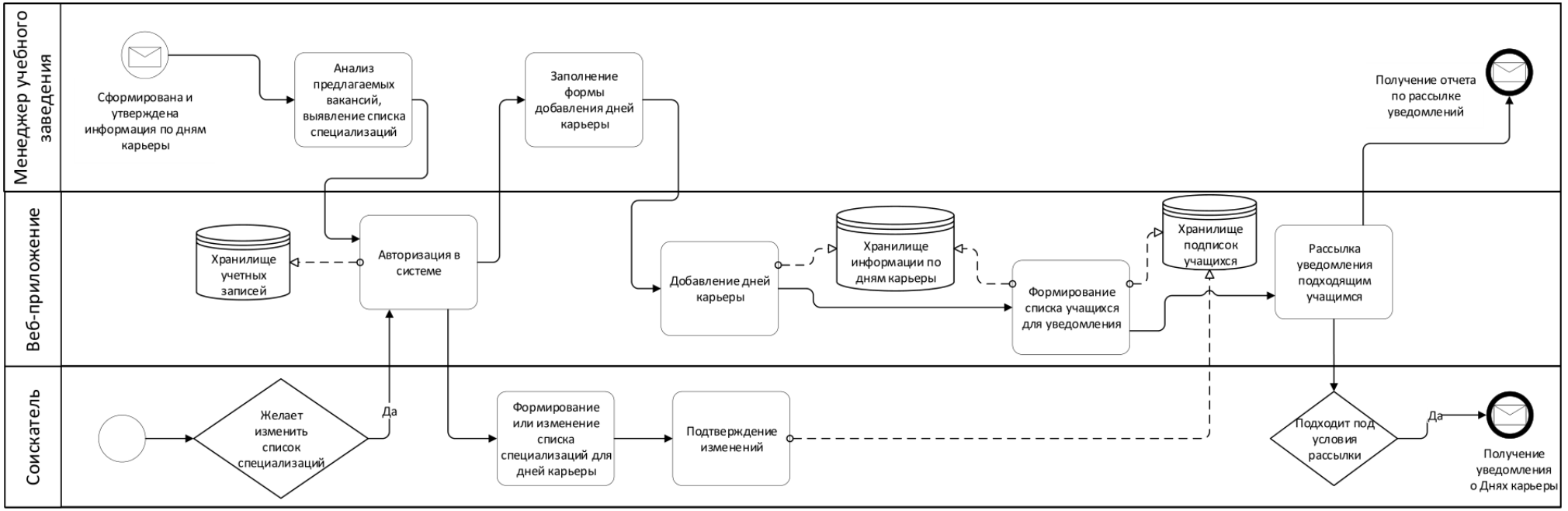

Рисунок 11. Взаимодействие участников процесса по уведомлению учащихся о «Днях карьеры»

Как видно из диаграммы моделирования процесса, перед отправкой уведомления молодому специалисту или учащемуся (представленному в приложении лице соискателя) производится проверка на соблюдение условия рассылки. Событийная цепочка процессов, представленная на рисунке 12, раскрывает детали этой проверки.

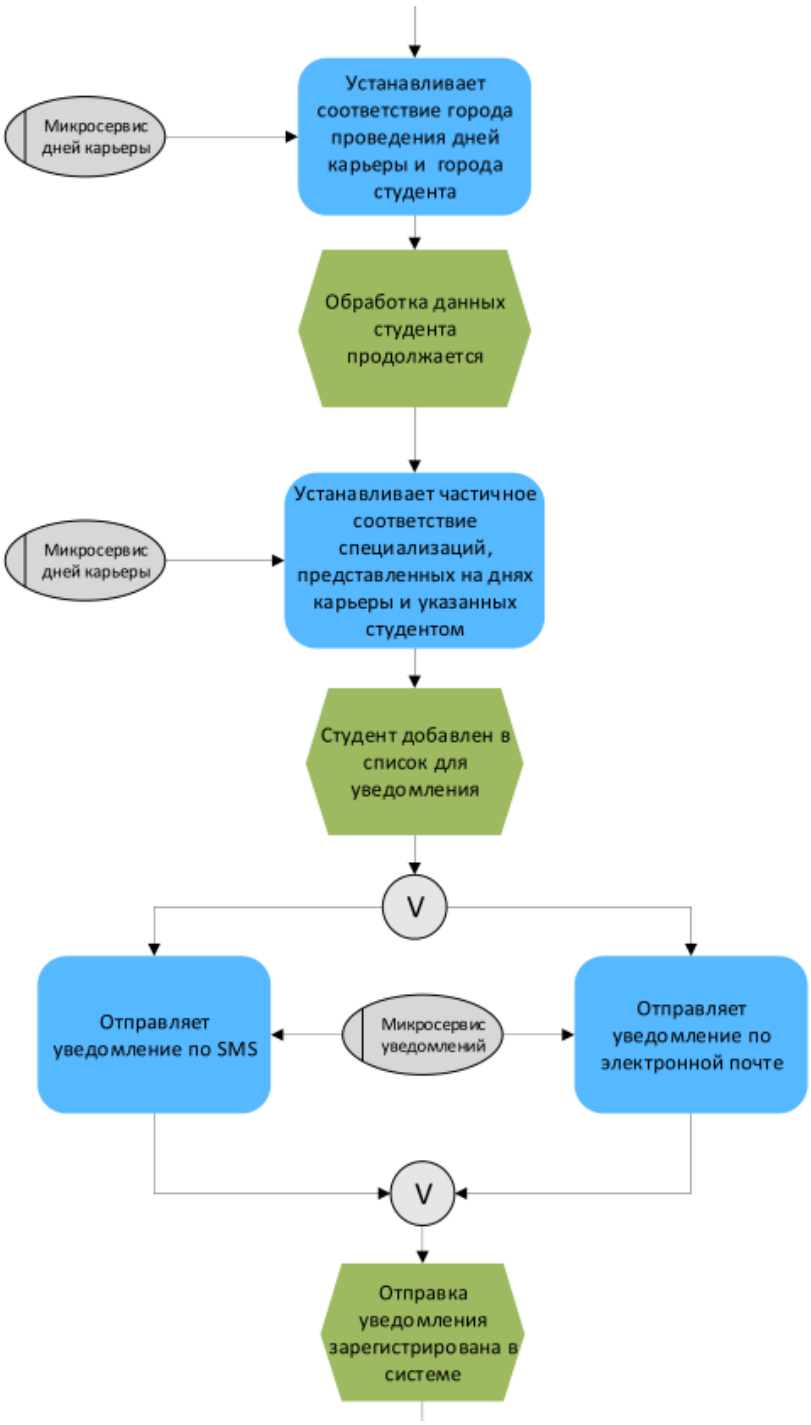

Рисунок 12. Фрагмент событийной цепочки процессов

Таким образом, рисунок 12 позволяет понять совершаемые в веб-приложении процессы для оповещения соискателя о «Днях карьеры».

1.4.1.4. Проектирование модуля оповещения

Функционал модуля полностью опирается на микросервисы «Дней карьеры» и оповещений, при этом данные микросервисы являются лишь составной частью всех микросервисов веб-приложения. Диаграмма потоков

данных, продемонстрированная на рисунке 13, позволяет понять взаимодействие этих микросервисов с остальной частью приложения.

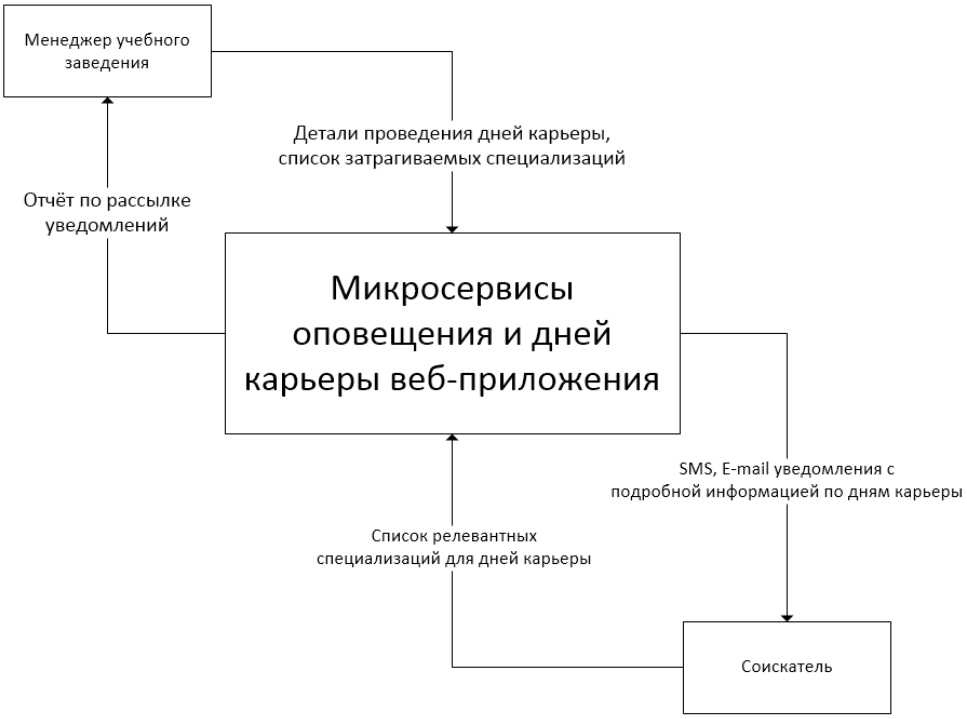

Рисунок 13. Контекстная диаграмма данных модуля оповещения

Очевидно, что возможность оповещения соискателей о днях карьеры затрагивает не только самого соискателя, но и также менеджера учебного заведения, в возможности которого входит создание информационной брошюры о днях карьеры. Предлагается рассмотреть эту возможность отдельно со стороны соискателя и менеджера, что представлено соответственно на рисунках 14 и 15.

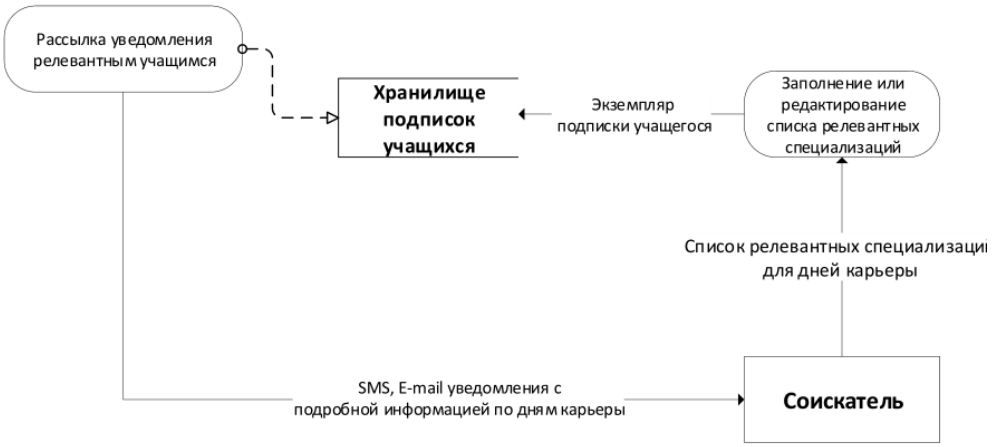

Рисунок 14. Потоки данных, связанных с возможностью оповещения о днях карьеры, со стороны соискателя

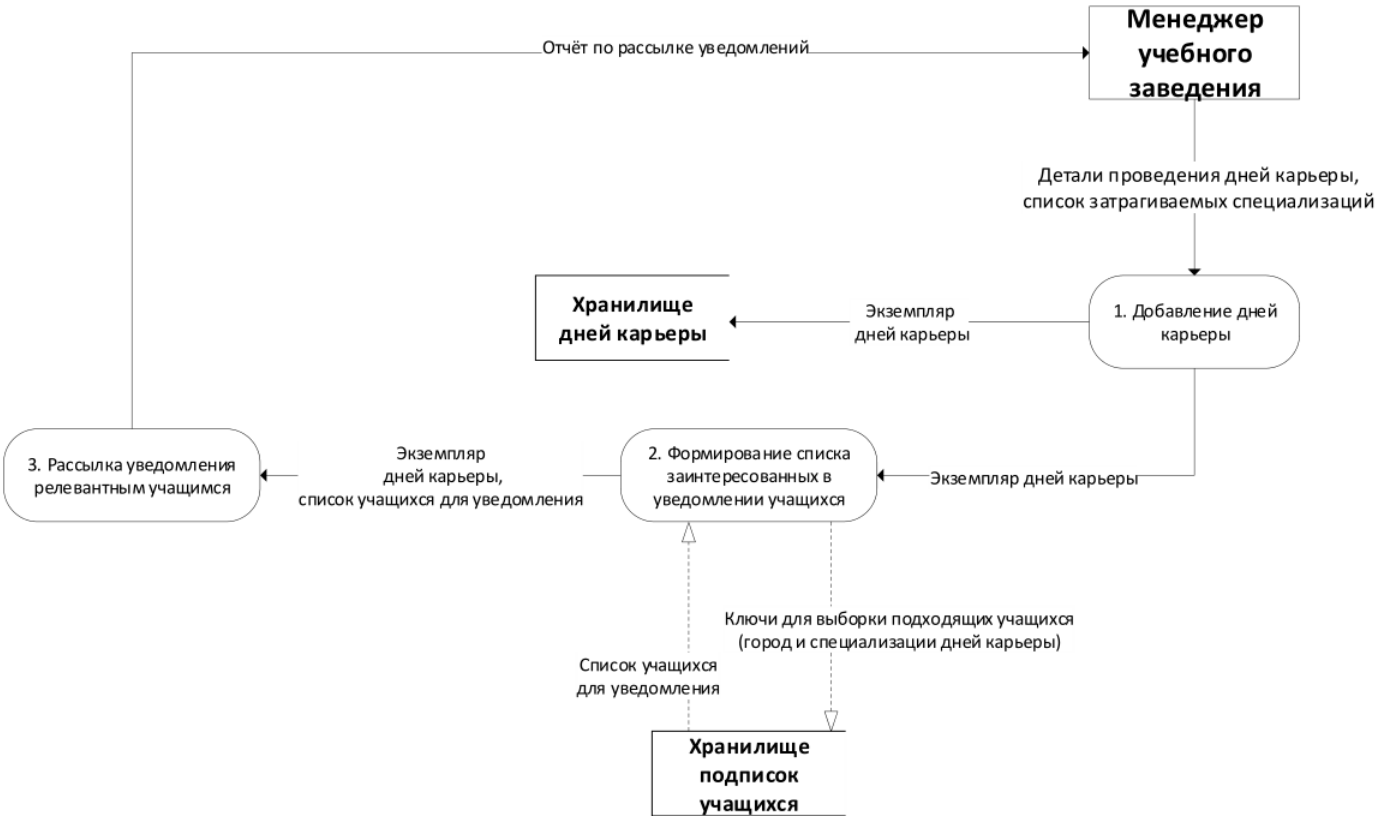

Рисунок 15. Потоки данных, связанных с возможностью оповещения о днях карьеры, со стороны соискателя

После того, как были показаны потоки данных, возникающих в результате взаимодействия модуля подписки с основной частью веб-приложения, следует спроектировать внутреннее устройство модуля. Одним из наиболее удобных для восприятия способов является построение ER-диаграммы, позволяющей раскрыть концептуальный уровень модели данных.

При проектировании на этом уровне, следует учесть, что соискатель может быть заинтересован в нескольких направлениях, на одном мероприятии дней карьеры также может быть представлено множество специализаций. С построенной диаграммой можно ознакомиться на рисунке 16.

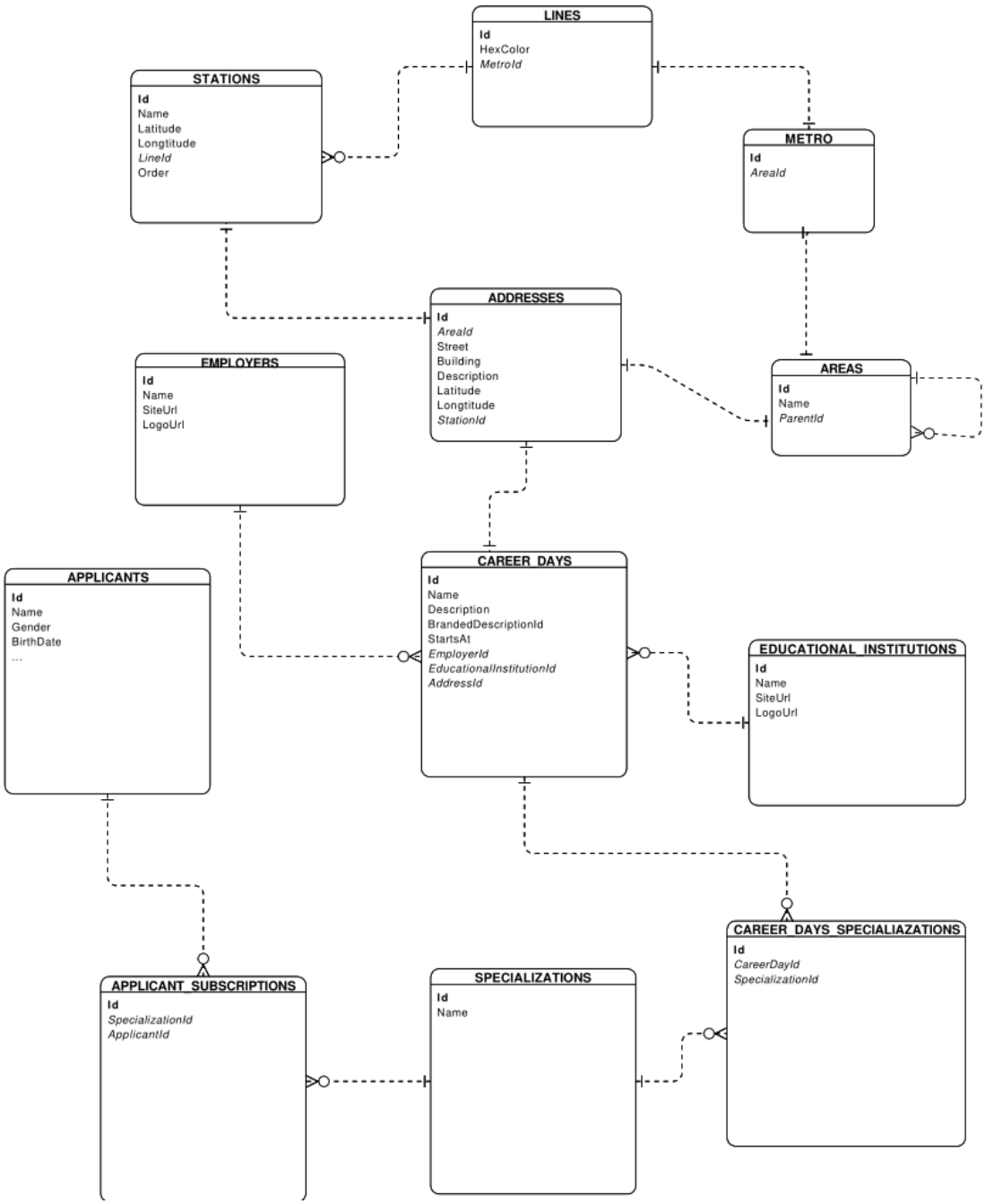

Рисунок 16. Концептуальный уровень данных модуля

Основными сущностями, позволяющими модулю оповещения выполнять свои обязанности, являются `APPLICANT_SUBSCRIPTIONS` (специализации, в которых заинтересован соискатель), `SPECIALIZATIONS` и `CAREER_DAYS_SPECIALIZATIONS` (представленные на «Днях карьеры» специализации).

1.4.2. Возможность работодателям генерировать тестовые задания

1.4.2.1. Обоснование реализации

Модуль предназначен для возможности проведения автоматического тестирование соискателей.

Проверка компетентности соискателей может занимать длительное время в связи с необходимостью личной встречи, а также отсутствием удобного взаимодействия с пользователем в рамках процесса удаленного выполнения тестового задания. Автоматизация данного процесса позволит компании взаимодействовать с соискателем в режиме реального времени, отслеживать процесс выполнения тестового задания и производить поверхностный отбор соискателей на более ранних этапах (до собеседования).

За счет внедрения системы генерации текстовых заданий планируется добиться автоматизации процесса тестирования.

1.4.2.2. Особенности проверки компетентности соискателей без использования веб-приложения

Большие временные затраты на проверку компетентности соискателя – комплексная проблема, для понимания которой следует разбить её на подпроблемы и установить причинно-следственные связи. Для проведений этой процедуры разработана диаграмма Исикавы, которую можно увидеть на рисунке 17.

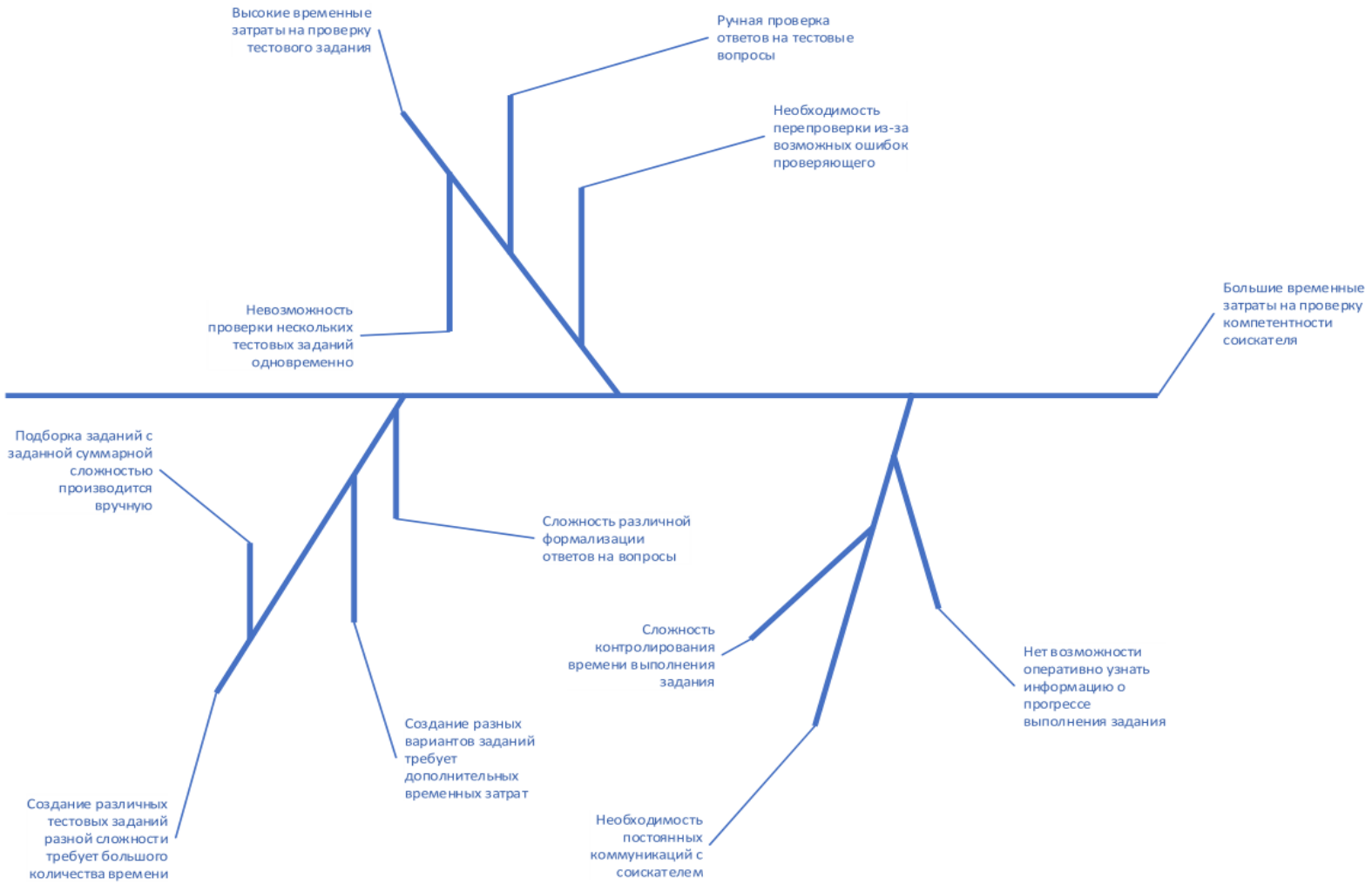

Рисунок 17. Диаграмма Исикавы для выявления причинно-следственных связей

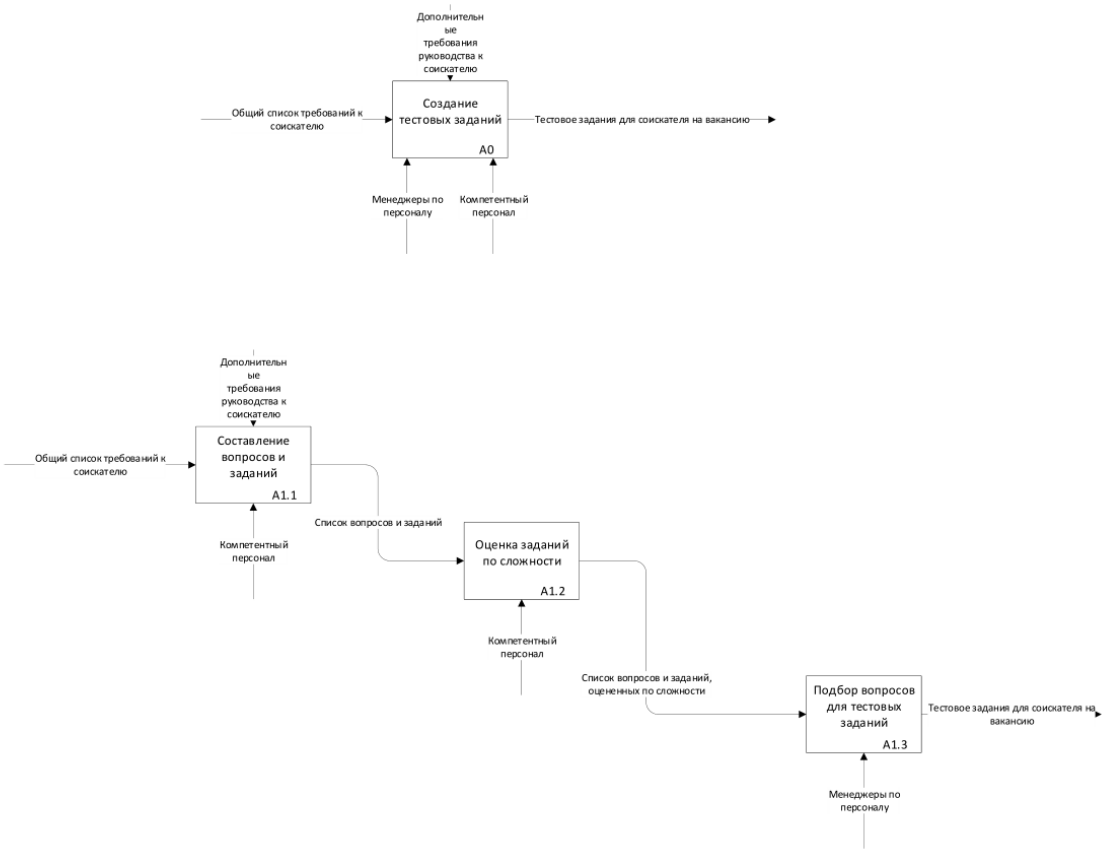

Рисунок 18. Диаграммы IDEF0 для исходного процесса

Наибольшее число изменений затронет функциональный блок *A*1.2, поэтому для него следует провести подробную декомпозицию, представленную на рисунке 19.

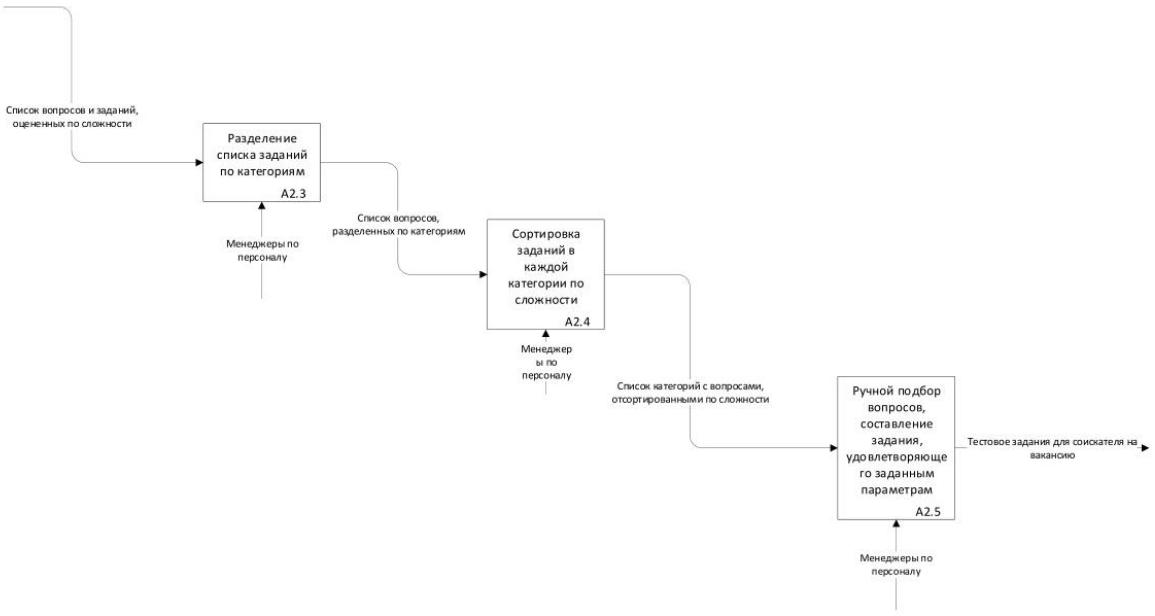

Рисунок 19. Декомпозиция функционального блока

1.4.2.3. Описание изменений в процессе проверки компетентности

С помощью контекстной диаграммы в нотации IDEF3 подробно описан процесс проверки компетентности в веб-приложении.

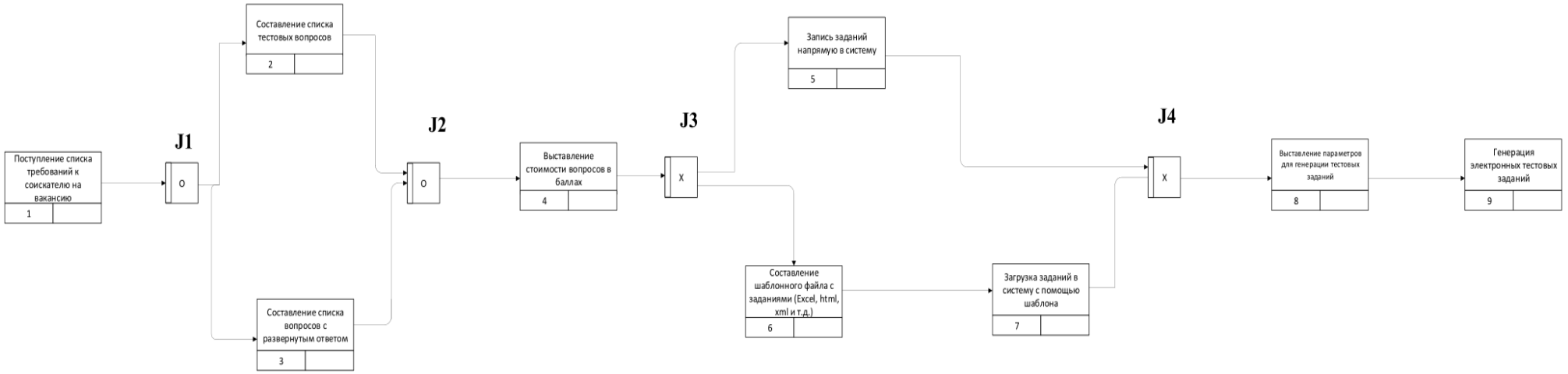

Рисунок 20. Усовершенствованный процесс проверки компетентности с использованием веб-приложения

Построенная на рисунке 21 диаграмма BPMN позволяет понять детали взаимодействия пользователей с веб-приложением при использовании возможности по созданию и добавлению автоматизированных тестов.

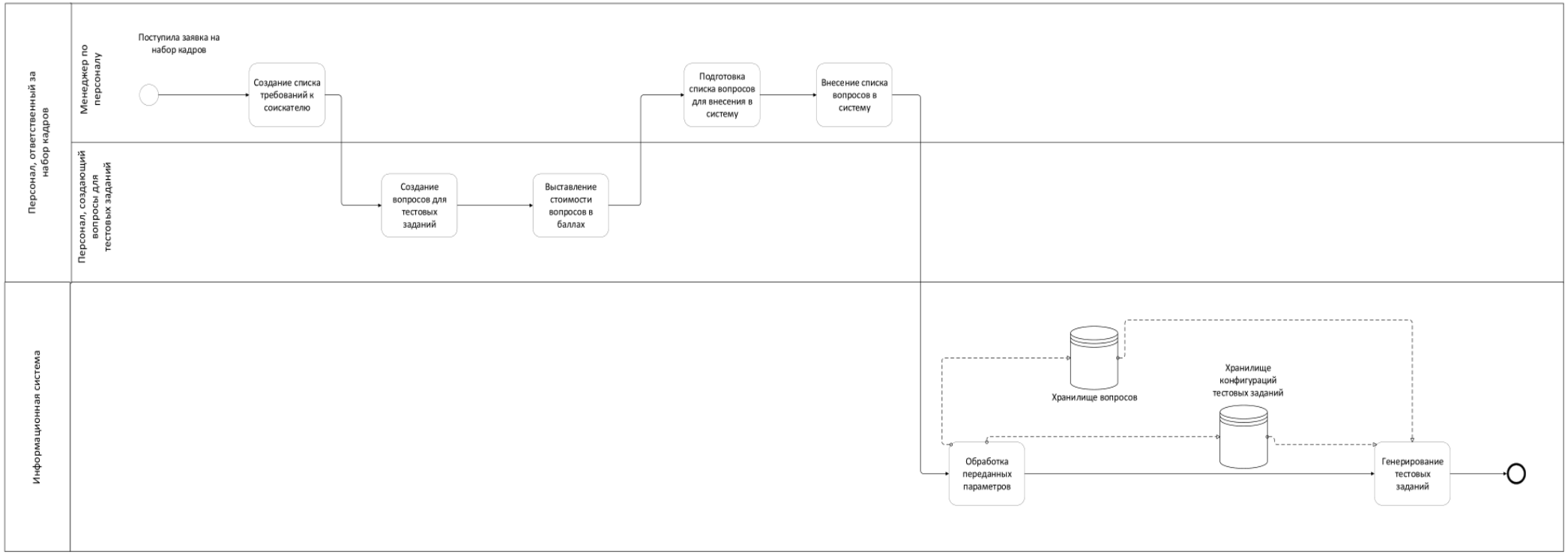

Рисунок 21. Взаимодействие пользователей с веб-приложением при использовании возможности создания автоматизированных тестов

Наибольшее количество обязанностей возложено на менеджера по персоналу, это единственный пользователь, который взаимодействует с веб-приложением в рамках рассматриваемой возможности.

1.4.2.4. Проектирование модуля

Для проработки архитектуры модуля следует изучить его поток данных, что удобно сделать с помощью DFD-диаграммы, показанной на рисунке 22.

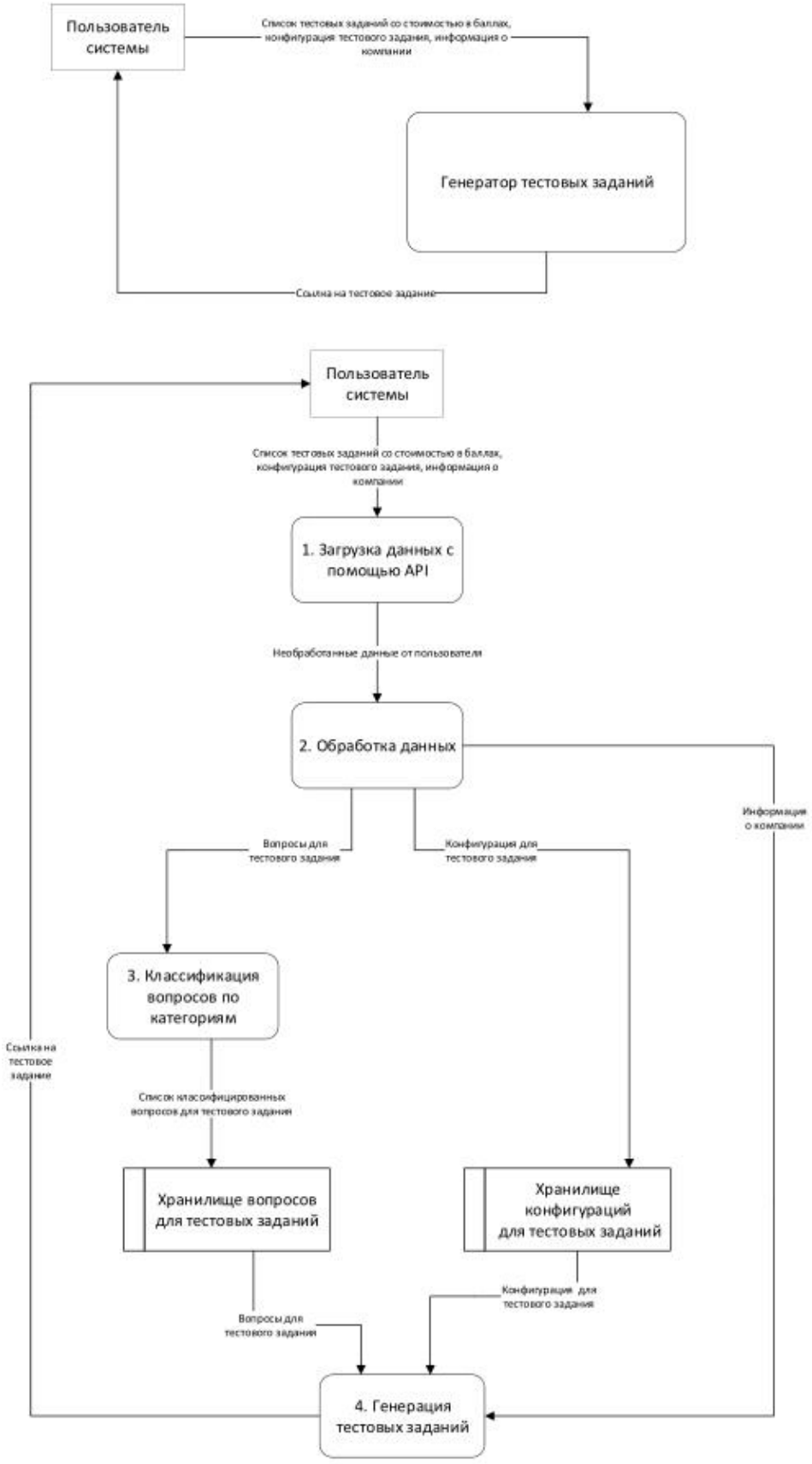

Рисунок 22. Потоки данных модуля по созданию автоматизированных тестов

Как видно из рисунка, для реализации хранения текстовых заданий в веб-приложении будет задействовано два хранилища – для вопросов текстовых заданий и конфигурации.

1.5. Выбор программно-технических средств для реализации веб-приложения

1.5.1. Выбор веб-сервера

Как известно, веб-сервер является сложной программной оболочкой, позволяющей выполнять целый ряд нижеперечисленных задач:

- Получение запросов пользователей.
- Формирование и отправка ответа обратно пользователям.
- Использование защищенных протоколов передачи данных (таких как HTTP Security – HTTPS).
- Ведение журнала логирования.
- Аутентификация пользователей.
- Возможность использования различных протоколов прикладного уровня модели OSI (например, FTP, SMTP).
- Возможность использования различных протоколов транспортного уровня модели OSI (TCP/UDP).
- Возможность использования веб-сервера в качестве прокси- или обратного прокси-сервера.

Ввиду особенности выбранной архитектуры серверной части разрабатываемого веб-приложения, которая будет раскрыта в последующей главе, особую ценность веб-сервера играет возможность его использования в качестве обратного прокси.

У обратного прокси существует множество сценариев полезного использования, ниже перечислены наиболее полезные особенности для разрабатываемого приложения:

- Балансировка нагрузки на сервер – оптимальное распределение нагрузки на доступные вычислительные ресурсы.

- Уменьшение нагрузки на сервер и ускорение выдачи ответов на запросы за счет использования кеширования статического и динамического контента.
- Выступает для клиентов фасадом серверной части приложения.
- Возможность проведения A/B тестирования.

На рисунке 23 представлена концептуальная схема взаимодействия клиент-сервер с использованием прокси-сервера.

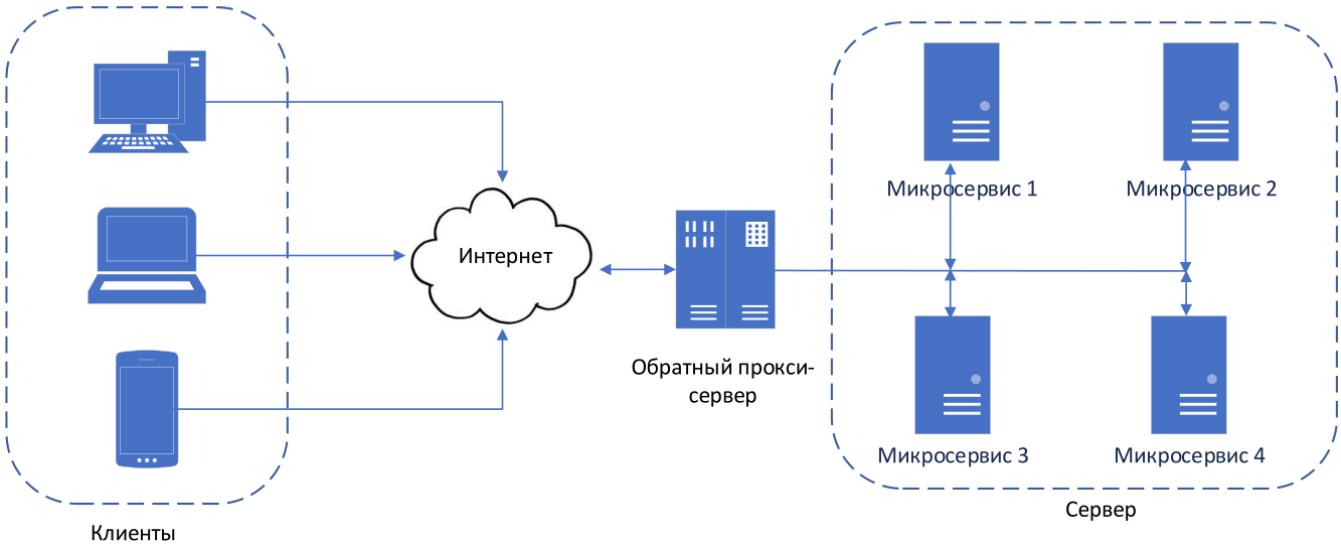

Рисунок 23. Принцип работы обратного прокси-сервера

При использовании такого подхода клиентская часть приложения становится более гибкой и готовой к изменениям, к тому же ее разработка существенно упрощается, так теперь клиент не знает конкретную структуру серверной части и, соответственно, не зависит от нее, посылая при этом запросы на единый URL-адрес. Диаграмма активности, представленная на рисунке 24, демонстрирует процесс обеспечения безопасного соединения.

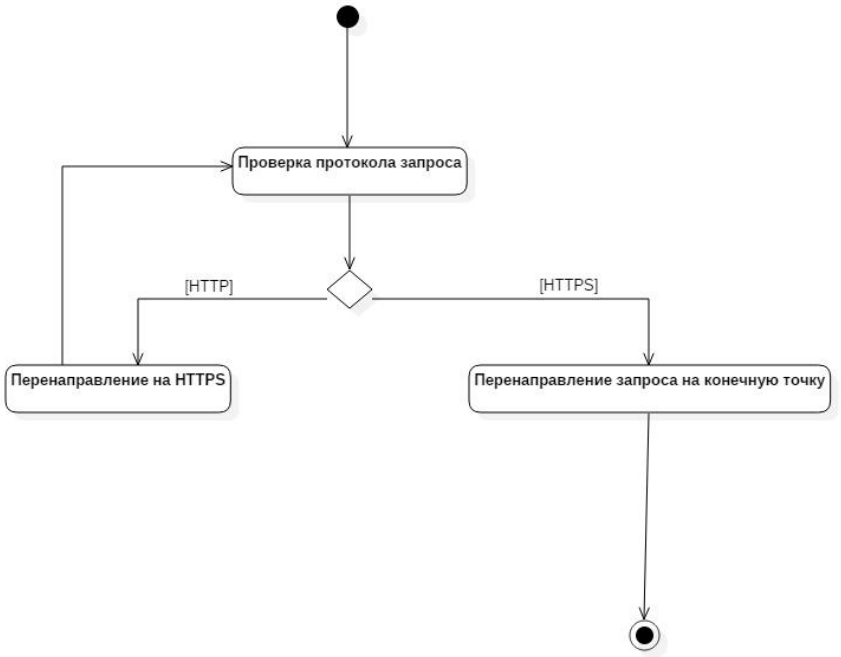

Рисунок 24. Обеспечение безопасного соединения с веб-сервером

На данный момент существует несколько готовых программных решений, позволяющих развернуть веб-сервер, наиболее часто используемые из них – Apache, IIS, Nginx.

Выявленные командой разработчиков требования к веб-серверу, как и численная оценка вышеперечисленных решений показаны в таблице 1.

Здесь и далее при оценивании программных решений в скобках после наименования критерия указывается его максимальное значение. Для лучшей детализации у некоторых программных решений помимо количественной оценки также может быть дано дополнительное пояснение.

Таблица 1. Критериальное оценивание веб-серверов

Критерий/Веб-сервер	Nginx	Apache	IIS
Возможность контейнеризации (10)	10 – есть образ в Docker	10 – есть образ в Docker	3 – только виртуальная машина
Поддержка протокола HTTP2 (3)	3	3	3
Стоимость программного решения (8)	8 – есть бесплатная версия	8 – есть бесплатная версия	1 – проприетарное ПО, поставляется вместе с Windows
Возможность установления безопасного	6 – SSL	6 – SSL	6 – SSL

Критерий/Веб-сервер	Nginx	Apache	IIS
соединения (6)			
Защита от сетевых атак (4)	4	4	3
Производительность при работе со статическим контентом (5)	5	4	4
Накладные расходы на системные процессы (7)	7	5	4
ИТОГО	43	40	24

Как видно из таблицы, веб-сервер Nginx лучше всего удовлетворяет требованиям разрабатываемого приложения, при этом стоит отметить, что такой критерий как возможность установления безопасного соединения получает высокий максимальный балл (6) ввиду того, что с каждым днем все больше сайтов и веб-приложений используют защищенный HTTPS-протокол обмена данными, и пользователи все меньше доверяют незащищенному соединению и могут отказаться от услуг сервиса, даже если от них не требуется ввода данных платежных средств.

Сертификат SSL подтверждает, что сайт действительно является тем, за кого себя выдает, и возможно установление безопасного соединения. Проверка подлинности становится возможной благодаря использованию криптографических вычислений (как правило, это алгоритм SHA-256): генерируются два связанных между собой ключа, публичный и приватный, информация о веб-узле зашифровывается с помощью приватного ключа, клиент же с помощью публичного ключа может расшифровать информацию и сравнить ее с той, что декларирует узел [6].

1.5.2. Выбор языка программирования для реализации серверной части

Одним из преимуществ микросервисной архитектуры является возможность использования различных технологий для реализации каждого микросервиса, поэтому командой разработчиков было принято решение определить список подходящих языков программирования для серверной

разработки исходя из личного опыта: C# (.NET Framework и .NET Core) и Node.js. Следует заметить, что в случае необходимости любой микросервис можно реализовать с использованием других технологий менее чем за две недели из-за ограниченности функционала, который может быть обеспечен одним микросервисом, что значительно снижает цену ошибку при выборе [7]. В таблице 2 приведены критерии и количественная оценка предложенных выше решений.

Главными критериями при выборе технологии серверной части являются: возможность контейнеризации, производительность и готовая реализация протокола AMQP.

Таблица 2. Критериальное оценивание программных решений для разработки серверной части

Критерий/Решение	.NET Core	.NET Framework	Node.JS
Возможность контейнеризации (10)	10 – есть образ SDK в Docker	6 – требуются windows-контейнеры	10 – есть образ SDK в Docker
Производительность решения (9)	8 – могут возникнуть дополнительные накладные расходы на архитектуре x86	9	4 – интерпретируемый, а не компилируемый язык
Возможность использования многопоточности (7)	7	7	6 – поддерживают не все версии
Поддержка протокола AMQP (9)	8 – требуется установка NuGet – пакета	8 – требуется установка NuGet – пакета	7 – требуется установка нескольких NPM-пакетов
Автоматическое нахождение некоторых трудноуловимых ошибок (6)	6 – используется статическая типизация	6 – используется статическая типизация	1 – используется динамическая слабая типизация
ИТОГО	40	36	28

Из вышеприведенной таблицы видно, что среди предложенных технологий разработки серверной части наиболее подходящим под потребности приложения является .NET Core.

1.5.3. Выбор брокера для управления потоком сообщений

Управлять потоком сообщений в событийно-ориентированной системе можно с помощью реализации протокола AMQP или использования реплицированного журнала фиксаций. Существует несколько реализаций протокола AMQP (RabbitMQ, Apache Qpid, Apache ActiveMQ), однако различия между ними столь несущественны, что было принято решение выбрать конкретную реализацию протокола, основываясь на клиентах той или иной реализации [8]. С таким подходом к выбору наиболее удачным решением станет использование RabbitMQ, который используют крупные компании, выпускающие высоконагруженные приложения – Google, MS Azure service bus, NASA, Red Hat). В таблице 3 представлено сравнение RabbitMQ с Apache Kafka – брокером, использующим для хранения и распределения сообщений журнал фиксаций.

Таблица 3. Критериальное оценивание брокеров потока сообщений

Критерий/Брокер	RabbitMQ	Apache Kafka
Горизонтальное масштабирование (4)	3 – производительность растет не линейно	4
Производительность (7)	6	7
Простота использования (9)	9 – требуется только подписаться на брокера	3 – реализован лишь базовый функционал
Гарантия доставки (9)	9 – есть возможность использовать флаг подтверждения	1 – требуется реализовывать вручную
Возможность контейнеризации (10)	10 – есть образ Docker	10 – есть образ Docker
Накладные расходы – CPU, RAM (6)	2	6
Вариативность режимов работы (5)	5 – шина событий, RPC, Round-Robin	3 – только шина событий
ИТОГО	39	31

Результаты таблицы показывают, что использование брокера RabbitMQ на базе протокола AMQP является более предпочтительным для приложения ввиду готовой реализации большинства аспектов обмена сообщениями между сервисами. При этом возрастают накладные расходы, незначительно

уменьшается производительность и эффект от горизонтального масштабирования, однако эти негативные факторы становятся заметными только после преодоления отметки в 45,000 публикуемых сообщений в секунду, что недостижимо на данный момент развития приложения [9].

1.5.4. Выбор модели базы данных

Принято выделять SQL и NoSQL базы данных. Использование SQL баз данных – это традиционный подход к организации внешнего хранилища приложения, где данные на концептуальном уровне хранятся в виде связанных таблиц.

Связь между таблицами может быть одной из следующих типов:

- Один к одному – одной записи в таблице А соответствует ровно одна запись в таблице Б.
- Один ко многим – одной записи в таблице А соответствует одна или больше записей в таблице Б.
- Многие ко многим – одна или несколько записей в таблице А связана с одной или несколькими записями в таблице Б.
- Рекурсивная – запись таблицы ссылается на другую запись в этой же таблице.

Большинство приложений использует объектно-ориентированный подход, то становится удобным использование реляционных СУБД и ORM-решений для обеспечения согласованности используемых модели данных. Процесс декомпозиции сущности базы данных на несколько связанных должен проходить с соблюдением определенных правил и называется нормализацией. Цель нормализации базы данных – устранение избыточности хранимых данных и обеспечение корректности выполнения CRUD-операций.

Однако для отображения информации в удобном виде для конечного пользователя обработчики графического интерфейса ожидают, как правило, денормализованные данные, получить которые из нормализованных можно с помощью операции *join* – объединение данных из нескольких таблиц по

заданным условиям. Нормализация сущностей реального приложения обычно быстро увеличивает количество используемых приложением таблиц и сильно усложняет построение запросов к СУБД для получения необходимых денормализованных данных, к тому же объединение нескольких таблиц является дорогостоящей с точки зрения вычислительной мощности операции.

Перечисленные выше причины являются далеко не единственными, способствующими активному внедрению NoSQL баз данных в последнее время. Такие СУБД отказываются от использования структурированного языка запросов и применяют иные концепции для хранения данных.

Основными типами таких баз являются:

- базы типа ключ-значение,
- семейство столбцов,
- документарные,
- графовые.

В таблице 4 приведено более подробное сравнение СУБД, использующей язык структурированных запросов и NoSQL.

Таблица 4. Детальное сравнение SQL и NoSQL СУБД

Параметр сравнения/СУБД	SQL СУБД	NoSQL СУБД
Возможность расширения	Только вертикальное	Горизонтальное и вертикальное
Схема хранения данных	Задается при создании, не рекомендуется изменение	Опциональна, можно хранить произвольные данные
Производительность CRUD операций	Высокая	Зависит от направленности СУБД. Для специализированной СУБД – максимальная.
Безопасность изменения данных	Полностью поддерживаются транзакции	Ограничения теоремы CAP – полная поддержка транзакций невозможна
Возможность хранения реляционных данных	Поддерживается весь необходимый инструментарий	Не рекомендуется из-за медленных операций *join* и необходимости ручной ее

Параметр сравнения/СУБД	SQL СУБД	NoSQL СУБД
		реализации
Сложность взаимодействия с базой данных	Требуется знание SQL или ORM-решений	Требуются базовые знания JavaScript и формата JSON

Из таблицы 4 следует, что для хранения реляционных данных предпочтительнее использовать SQL СУБД из-за возможности применения ORM и механизма транзакций, для остальных слабо связанных данных, которые требуются читать гораздо чаще, чем изменять – NoSQL СУБД, так как такие решения легче масштабируются и готовы к изменению схемы хранения данных. Так как сущности системы трудоустройства естественным образом взаимодействуют между собой и ссылаются друг на друга, то есть, степень связанности (реляционности) сущностей высока, предпочтительней будет использование SQL СУБД для основных сервисов приложения.

Однако, стоит учитывать, что приложение использует Nginx в качестве обратного прокси, поддерживающего возможность кеширования ответов сервера. Использование реляционной СУБД для хранения кэша является нецелесообразным ввиду отсутствия реляционных данных. Использование NoSQL базы данных типа ключ-значение позволит веб-серверу существенно быстрее взаимодействовать с кэшированными ресурсами. Следовательно, необходимо выбрать подходящее решение как для SQL, так и для NoSQL СУБД.

1.5.5. Выбор SQL СУБД

Первые реляционные СУБД появились более 30 лет назад, за столь длительный срок было реализовано множество программных решений, наиболее поддерживаемые из которых: MS SQL, MySQL, Oracle. В таблице 5 представлены критерии, по которым каждая из этих СУБД была оценена.

Таблица 5. Критериальное оценивание различных СУБД

Критерий/СУБД	MS SQL	MySQL	Oracle
Возможность контейнеризации (10)	10 – есть образ Docker	10 – есть образ Docker	10 – есть образ Docker
Интеграция с Visual Studio (8)	8	4 – требуется установка NuGet пакета	2 – требуется установка NuGet пакета и JDBC-драйвера
Стоимость использования (10)	10 – наличие бесплатной версии	10 – наличие бесплатной версии	1 – только коммерческая версия
Накладные вычислительные расходы (6)	4	6	5
Доступный функционал (6)	4 – доступен базовый функционал	4 – доступен базовый функционал	6 – доступны самые новые возможности
Интеграция с ORM Entity Framework (8)	8	1 – модуль интеграции необходимо реализовать вручную	1 – модуль интеграции необходимо реализовать вручную
ИТОГО	44	35	25

Исходя из приведенных в таблице данных, для реализации системы следует использовать Microsoft Service SQL, ввиду высокого уровня интеграции с другими продуктами Microsoft и наличию бесплатно распространяемой версии продукта.

1.5.6. Выбор NoSQL СУБД

NoSQL СУБД планируется использовать для одной конкретной задачи – кэширование ответов сервера, то для нее формируются следующие требования:

- Возможность развертывания в контейнере с помощью Docker.
- Высокая скорость извлечения данных.
- Подтип СУБД – ключ-значение.
- Низкие накладные расходы на поддержание работоспособности.

Согласно исследованиям [10] о быстродействии различных NoSQL СУБД Redis показывает наивысшую скорость чтения и при этом является очень легковесным решением, которое можно быстро развернуть в Docker,

поэтому было принято решение использовать СУБД Redis в качестве хранилища кэшированных ресурсов.

1.5.7. Выбор формата обмена данными

Наиболее распространенными и поддерживаемыми программными средствами форматами данных являются JSON и XML. Несмотря на гибкость разрабатываемого решения и независимость работы серверной части от конкретного формата данных, необходимо выбрать формат, поддержку которого следует организовать в первую очередь. В таблице 6 указаны критерии, по которым проводилось сравнение JSON и XML.

Таблица 6. Критериальное оценивание различных форматов передачи и хранения данных

Критерий/Формат данных	JSON	XML
Избыточная информация (7)	6 – необходимость использования кавычек	2 – название каждого атрибута дублируется
Поддерживаемые архитектурные стили (5)	3 – только REST	5 – REST, SOA
Степень поддержки JavaScript’ом (6)	6 – полностью поддерживается	3 – требуется использовать сторонние библиотеки
Удобство представления для пользователя (3)	2 – легко читается, но требуется знание JavaScript	1 – множество тегов, требуется знание HTML
ИТОГО	17	11

JSON обладает лучшей поддержкой и не требует установки дополнительных программных средств или какой-либо настройки, это формат по умолчанию для многих сервисов, поэтому было принято решение использовать JSON в качестве основного формата данных.

1.5.8. Выбор технологий для реализации клиентской части

Технологии представления клиентской части веб-приложений за последние 10 лет кардинально изменились. Клиентские интерфейсы стали включать значительно больше элементов управления, данные в блоках веб-

страницы могут частично дублироваться, контроллеры интерактивных элементов стали включать значительно усложненную логику изменения содержимого других элементов. Поэтому сложность добавления новых возможностей возрастает нелинейно, для упрощения процесса разработки необходимо использовать более современный подход – концепцию *реактивных данных*. При такой концепции разработчики более не работают напрямую с реальным DOM, его состояние инкапсулирует виртуальный DOM. Реактивные фреймворки отслеживают изменения в содержимом виртуального DOM и автоматически определяют компоненты, состояние которых необходимо изменить и перестроить [11]. Такой подход позволяет разработчику внедрять новые возможности в интерфейс пользователя, абстрагируясь от взаимодействия существующих компонентов. На рисунке 25 продемонстрировано взаимодействие между родительскими и дочерними компонентами в классическом реактивном приложении.

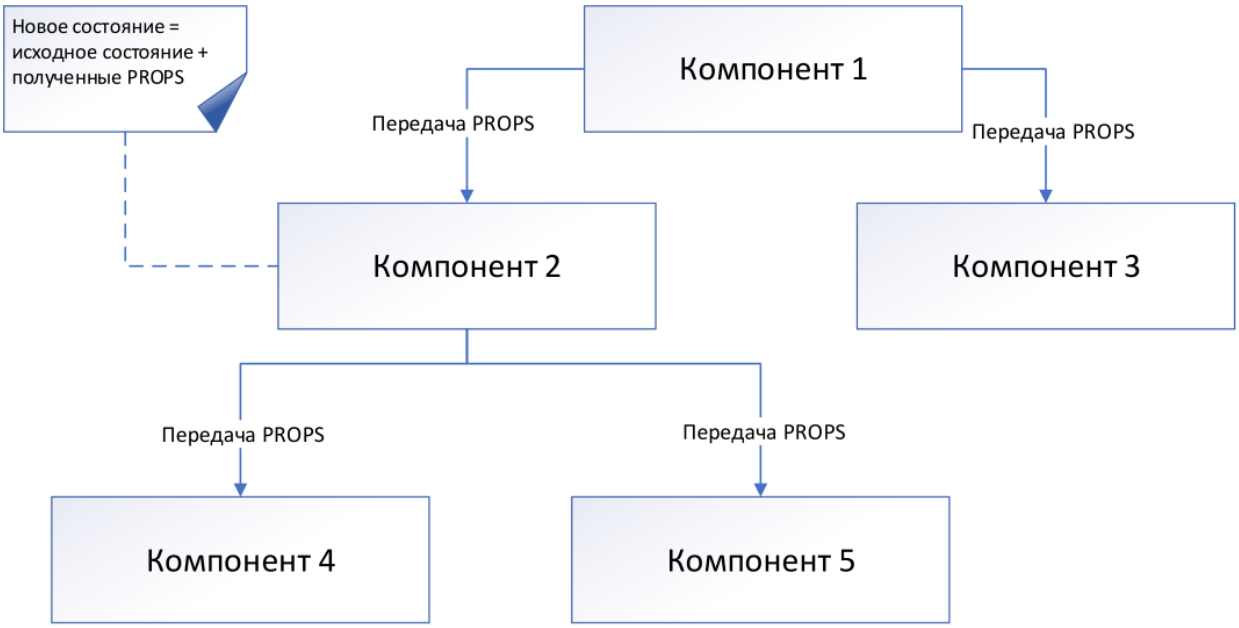

Рисунок 25. Передача данных между реактивными компонентами

Пользователь взаимодействует с корневым компонентом, изменяя его состояние, и тот реактивно распространяет изменения на дочерние компоненты, передавая им новую порцию данных (`props`) из которой будет

строиться их новое состояние. Если их состояние изменится, процесс повторится для компонентов 4 и 5.

На данный момент доступно несколько решений для реализации реактивности данных в клиентской части веб-приложений – Angular, React, Vue. Команда разработчиков выделила показанные в таблице 7 критерии оценки реактивных фреймворков исходя из особенностей работы разрабатываемой системы.

Таблица 7. Критериальное оценивание реактивных фреймворков

Критерий/Фреймворк	React	Angular	Vue
Количество языковых конструкций, не включенных в стандартный синтаксис JS (8)	7	2 – требуется изучение TypeScript	5 – новые конструкции для вывода коллекции
Возможность определения разметки компонента в основной функции (3)	3	1	1
Поддержка и улучшение продукта (6)	6 – поддерживается Facebook’ом	4 – отток комьюнити, поддерживается Google’ом	1 – широко распространено только в Китае
Производительность решения (7)	7 – оптимизация отрисовки функциональных компонентов	5 – больше инструкций в собранном скрипте из-за транспиляции	7
Возможности по анимации компонентов (5)	3	3	5
ИТОГО	26	15	23

Группой разработчиков было принято решение использовать реактивный фреймворк React для реализации клиентского интерфейса из-за высокой поддержки программного продукта и использованию парадигм функционального программирования, призванного сделать программный код более декларативным, читаемым и предсказуемым.

Выводы по главе

В данном разделе, посвященном исследованию предметной области и проектированию веб-приложения, были установлены границы

рассматриваемой предметной области, выявлены главные пользователи приложения и их возможности. Был произведен анализ существующих на рынке конкурирующих продуктов, выявлены ключевые отличительные особенности разрабатываемого веб-приложения по сравнению с конкурентами, выбраны серверные, клиентские и прочие программные продукты для реализации приложения, обоснована необходимость их использования.

Глава 2. Разработка веб-приложения

2.1. Распределение задач между участниками проекта

Численность научно-исследовательской группы составляет три человека, что связано с особенностями предметной области, сложным взаимодействиям между предметными сущностями, большим количеством функциональных возможностей, доступных для пользователей системы. Для разработки были задействованы эффективные средства коммуникации, контроля исходного кода и распределения задач, поэтому каждый участник разработки мог вести независимую работу над каким-либо компонентом системы.

Обязанности участников разработки были разделены следующим образом:

- Немнов Р.И. – дизайнер и ведущий разработчик клиентской части веб-приложения, реализовал около 70 % графических компонентов и адаптировал все компоненты под мобильные устройства, технический писатель раздела клиентской части.
- Руденцов Д.П. – архитектор системы, ведущий разработчик серверной части веб-приложения, реализовал около 75 % микросервисов системы и настроил автоматизированное развертывание, технический писатель раздела серверной части.
- Трусов Е.Д. – `full-stack` разработчик, реализовал оставшиеся графические компоненты и микросервисы, настроил брокер сообщений, совместно с Руденцовым Д.П. интегрировал сервис Moodle, технический писатель раздела проектирования.

Для мониторинга процесса выполняемых задач использовался сервис Trello, позволяющий создавать виртуальные доски заданий [12]. Пример задач, находящихся на различных этапах выполнения, продемонстрирован на рисунке 26.

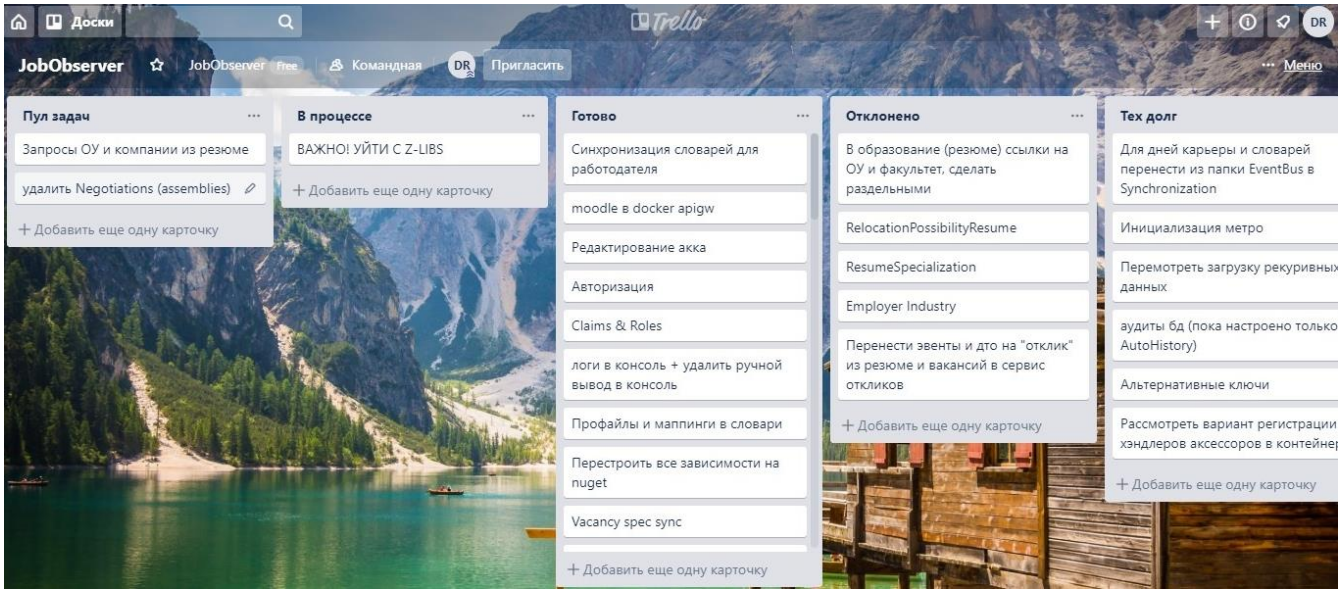

Рисунок 26. Использование виртуальной доски заданий Trello

Задачи на виртуальной доске этого сервиса разделяются на несколько категорий:

- Пул задач – содержит перечень задач, которые необходимо решить, но исполнитель ещё не назначен.
- В процессе – содержит перечень задач, решаемых в данный момент временим каким-либо разработчиком.
- Готовые – содержит задачи, решенные за последние две недели.
- Отклоненные – содержит перечень задач, от решения которых по каким-либо причинам отказались.
- Тех долг – содержит перечень задач, которые группа разработчиков не успела решить за предыдущий спринт (две недели).

2.2. Архитектура серверной части

Под архитектурой программного обеспечения понимают совокупность принятых решений о деталях организации программной системы, основные шаблоны архитектуры ПО:

- Многослойная.
- Событийно-ориентированная.
- Сервисно-ориентированная.
- Конвейерная.

- Микросервисная.
- Пространственно-ориентированная.

Классическим способом организации архитектуры является построение многослойного монолита, как это показано на рисунке 27.

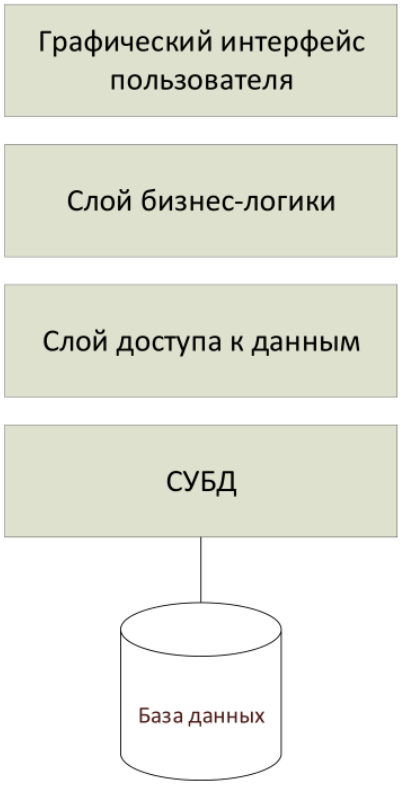

Рисунок 27. Монолитное приложение представлено в виде отдельных слоев

При такой организации, пользователи никак не зависят от конкретной базы данных.

Основными достоинствами монолитных приложений являются:

- Высокая производительность, так как «общение» между слоями происходит с помощью вызова локальных методов.
- Интеграционные тесты не требуют интернет-соединения.
- Приложение легко развернуть на удаленном хостинге.

Однако монолитные приложения не лишены недостатков:

- В случае повышенной нагрузки на приложение придется развернуть дополнительный экземпляр всего приложения целиком, продублировав в том числе невостребованный функционал.

- При возникновении критической проблемы в одном из слоев приложение целиком становится недоступным.
- В процессе развития приложения разным модулям могут потребоваться несовместимые между собой версии стороннего программного обеспечения.

Альтернативой монолитным приложениям стали приложения, основанные на микросервисах. Отличительными чертами микросервиса являются:

- Сфокусированность – микросервис решает ровно одну бизнес-задачу.
- Слабая связанность – микросервис может быть автономно развернут, зависимости сведены к минимуму.
- Высокая согласованность – не содержит лишней инфраструктуры.

Микросервисная архитектура обеспечивает следующие преимущества:

- Возможность обработать повышенное количество запросов на определенную часть системы, точечно увеличив ее производительность.
- Дублирование определенного микросервиса для проведения A/B тестирования.
- Отказ одного микросервиса не приводит к отказу всей системы.
- Реализацию любого микросервиса можно полностью заменить за короткое время (менее 2 недель).

К недостаткам систем с такой архитектурой можно отнести:

- Повышенные требования к квалификации разработчиков.
- Сложность развертывания – требуется использовать дополнительное ПО.
- Сложное и долгое интеграционное тестирование.
- Производительность в нормальных условиях работы ниже, чем у монолитных приложений.
- Сложности при определении границ микросервиса.

2.3. Планомерное изменение архитектуры серверной части

Первые версии разрабатываемого приложения базировались на монолитной многослойной архитектуре, но чем больше функционала серверной части было реализовано, тем яснее становились границы контекста между модулями и нарастало понимание того, что текущая архитектура имеет множество критичных недостатков, таких как:

- Сложность горизонтального масштабирования. Монолитная архитектура имеет малую гибкость в плане горизонтального масштабирования, т.к. не позволяет создать дополнительный экземпляр сервиса для распределения нагрузки.
- Сложность вертикального масштабирования. Разработчикам необходимо осваивать большую кодовую базу для реализации нового функционала в рамках текущей архитектуры.
- Быстрое увеличение связей в кодовой базе проекта.
- Сложности в области управления проектом на старте. Наработки разработчиков частично пересекаются, имеется ряд сложностей при распределении задач на этапе создания проекта.
- Привязан к конкретной платформе, сложность публикации на сервере.

Самодостаточные части приложения затем группировались в отдельные микросервисы, подключались к собственным базам данных и отсоединялись от монолита. Повторив эту итерационную процедуру несколько раз, удалось разбить монолитный сервис на 9 микросервисов, схематично изображенных на рисунке 28.

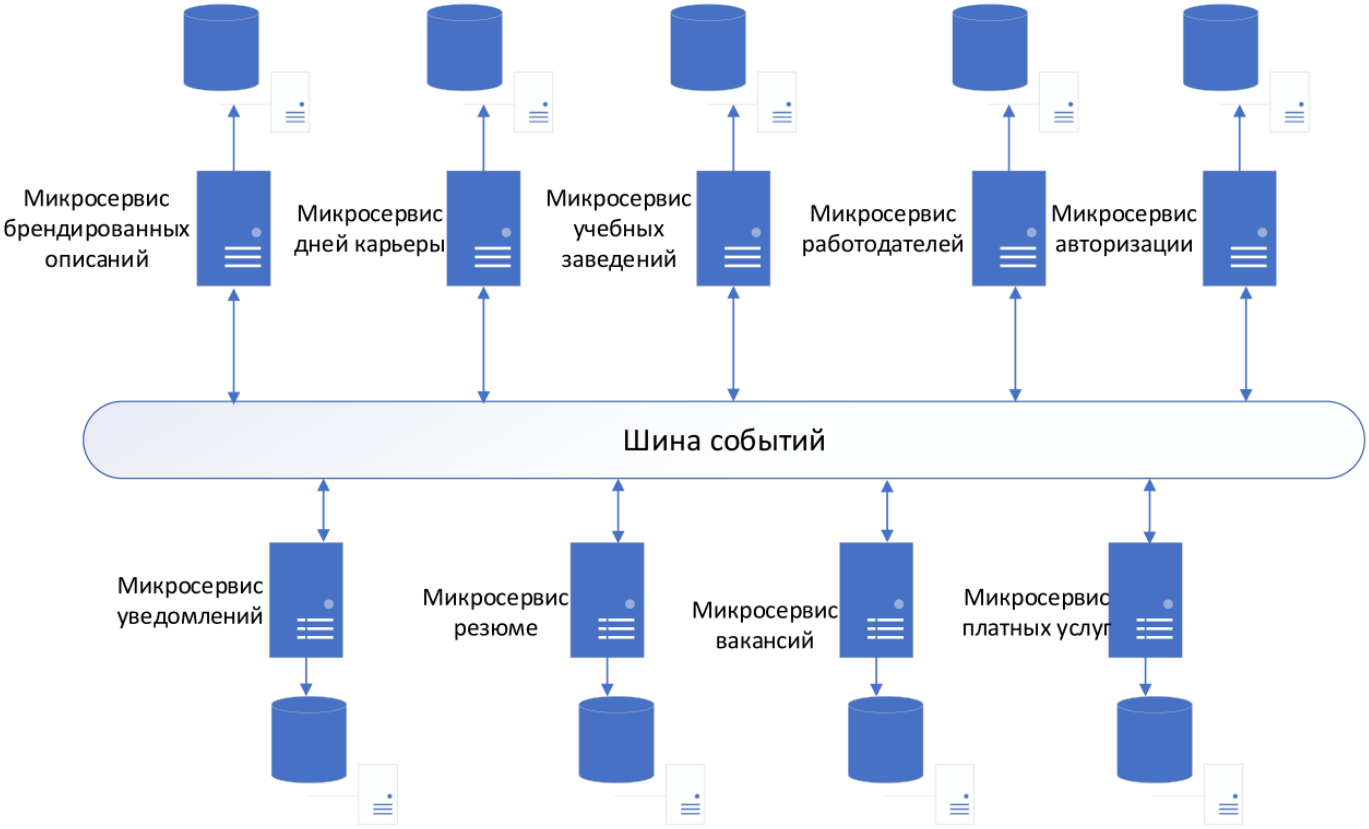

Рисунок 28. Микросервисы, образованные в результате декомпозиции монолитного приложения

Переход на микросервисную архитектуру позволил устранить многие недостатки монолитного подхода. Переход на новую архитектуру включил в себя несколько этапов:

- Выделение бизнес-потребностей. Бизнес-потребностью называется минимальная цельная единица бизнес-процесса. Каждый микросервис должен полностью покрывать одну бизнес-потребность.
- Итеративное выделение микросервисов из монолита в соответствии с бизнес-потребностями.
- Создание архитектуры взаимодействия микросервисов.
- Контейнеризация микросервисного приложения.
- Настройка рабочего окружения с помощью оркестратора контейнеров либо систем управления многоконтейнерными приложениями.

Основным недостатком микросервисного подхода к архитектуре веб-приложения заключалась в необходимости больших по сравнению с монолитом трудозатрат на старте проекта для настройки системы взаимодействия микросервисов. Архитектура серверной части проекта на высоком уровне абстракции показана на рисунке 29.

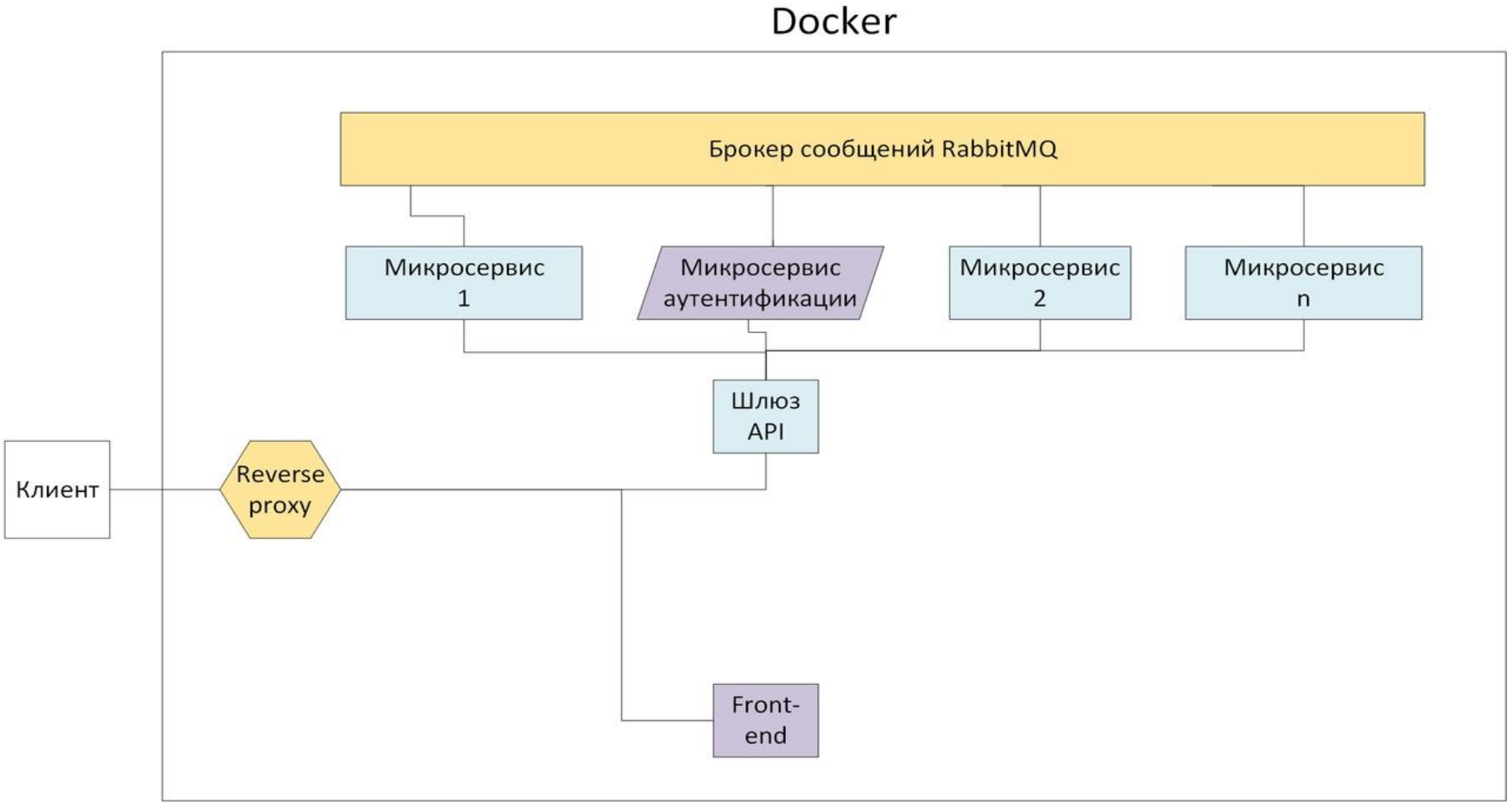

Рисунок 29. Высокоуровневое представление серверной архитектуры

Как видно из высокоуровневого описания, веб-клиент представляет собой отдельный процесс, запущенный в среде Docker, все запросы к серверной части проходят через обратный прокси и шлюз API прежде, чем обратиться к какому-либо микросервису, которые в свою очередь коммуницируют по выделенной шине событий.

Переход на новую архитектуру позволил разработчикам абстрагироваться от конкретной платформы, переведя всю инфраструктуру в Docker-контейнеры, эффективно разделить пул работ, увеличить уровень масштабируемости системы, упростить непрерывную интеграцию и непрерывную доставку модулей. Также данный подход снизил издержки на настройку рабочего окружения, так как все вспомогательное программное обеспечение было также переведено в контейнеры и не требовало настройки на конкретной машине. Кодовая база проекта разделилась между микросервисами, следовательно, разработчик не должен вникать в логику всего проекта для реализации функциональности отдельно взятого модуля.

2.4. Использование средства контейнеризации Docker для развертывания микросервисов

Для упрощения организации всех этапов процесса разработки, вся инфраструктура проекта была перенесена в Docker.

Docker – программное обеспечение для автоматизации процессов управления программными продуктами и их развертывания в системах, поддерживающих контейнеризацию [13].

Docker предоставляет возможность «упаковать» программный продукт и все его зависимости в изолированную среду и перенести на любую систему с поддержкой системы контейнеризации. Исходя из этого, следует, что данная система позволяет разработчикам абстрагироваться от разработки под конкретную платформу и поставлять готовый продукт в виде Docker-образов, что упрощает процесс развертывания, т.к. данный процесс стандартизирован для всех платформ. Также Docker нивелирует потребность в настройке

рабочего окружения и упрощает масштабируемость приложений с помощью систем оркестрации контейнеров.

На рисунках 30 и 31 представлены отличия изоляции процессов с помощью виртуальных машин и с помощью системы контейнеризации.

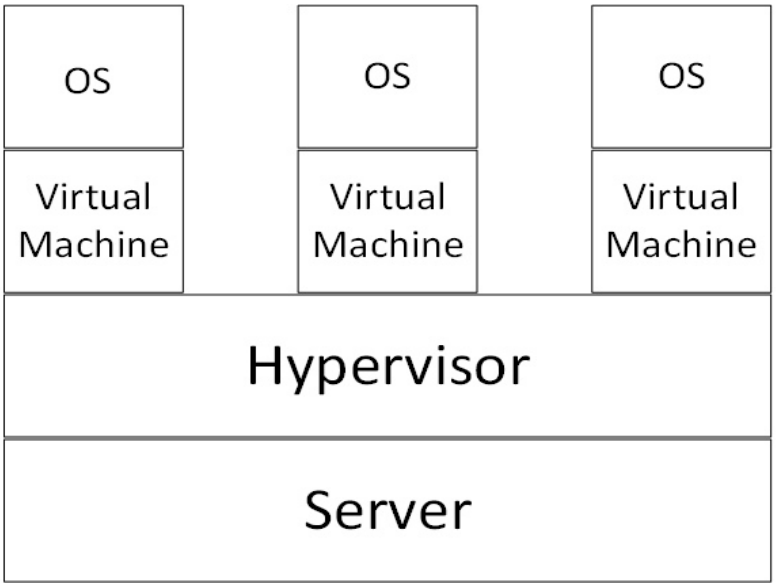

Рисунок 30. Организация изоляции процессов с помощью виртуальных машин

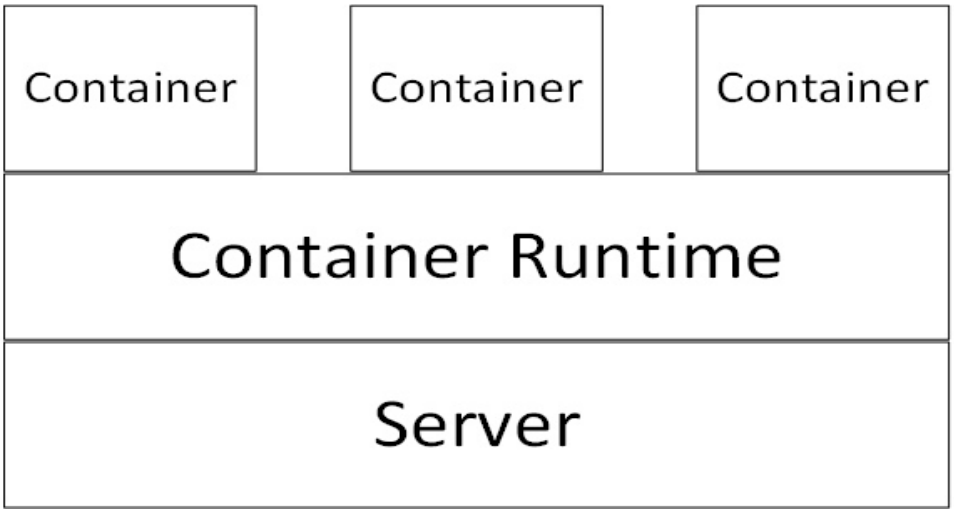

Рисунок 31. Организация изоляции процессов с помощью системы контейнеризации

Из рисунков выше следует, что для организации процессов с помощью Docker требуется меньше промежуточных слоев, чем для организации с помощью виртуальных машин.

Основные минусы виртуальных машин, способствующие переходу на Docker:

1. Размер. Образ виртуальной машины может быть довольно большим, что доставляет неудобства. Любое изменение в образе внутри виртуальной машины требует скачать весь образ заново.
2. Сложное управление совместным использованием вычислительных ресурсов. Не все виртуальные машины поддерживают совместное использование памяти и ресурсов процессора, а те, что поддерживают, требуют тонкой настройки.
3. Большие трудозатраты на организацию инфраструктуры. Создание кластера виртуальных машин требует больше времени и навыков разработчика для корректной настройки.

Основные понятия системы контейнеризации Docker:

1. Образ (Image). Образ является шаблоном, доступным только для чтения, из которого создается контейнер. Каждый образ состоит из множества уровней. Docker позволяет им прозрачно накладываться, создавая корректную фаловую систему. Уровни позволяют образам модифицироваться без пересборки всего образа, уровень просто добавляется или обновляется. Образы поддерживают наследование, что говорит о том, что в основе почти всех образов лежит другой образ, чаще всего образ какого-либо дистрибутива операционной системы Linux.

2. Контейнер (Container). Контейнер состоит из операционной системы, пользовательских данных и метаданных. Каждый контейнер создается из образа. Образ сообщает Docker содержимое контейнера и какой процесс запустить, когда запускается контейнер и другие данные для конфигурации. Контейнер создает уровень чтения/записи поверх образа, в котором может быть запущен программный продукт.

3. Система оркестрации контейнеров. Оркестрация – координация взаимодействия между контейнерами. Данная система позволяет строить инфраструктуру из множества независимых контейнеров,

осуществлять общение между ними через сетевые порты и общие директории, а также централизовано настраивать рабочее окружение внутри контейнеров.

4. Репозиторий образов. Система централизованного хранения образов контейнеров. Официальным реестром является Docker Hub.

5. Docker-daemon. Первая часть ядра Docker, работающая на хост-машине. Предоставляет возможность скачивать и отправлять в реестр образы, управлять, запускать и следить за контейнерами, а также собирать логи и настраивать сетевое взаимодействие между контейнерами.

6. Docker CLI. Вторая часть ядра Docker, предоставляющая интерфейс для управления системой контейнеризации.

7. Тома (Volumes). Инструмент для привязки некоторых частей файловой системы контейнера к файловой системе хост-машины.

8. Сети (Networks). Позволяет Docker организовывать свои сети между контейнерами, используя различные сетевые драйверы.

На рисунке 32 представлена схема организации Docker Engine (ядра Docker).

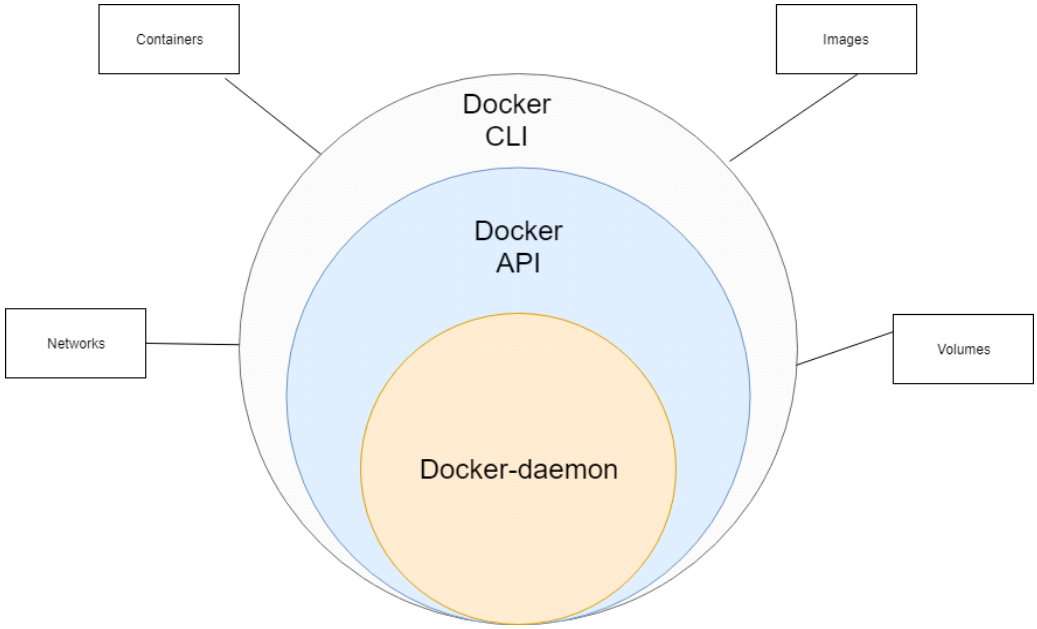

Рисунок 32. Устройство ядра Docker

На рисунке 33 представлена общая схема работы системы с использованием Docker Engine.

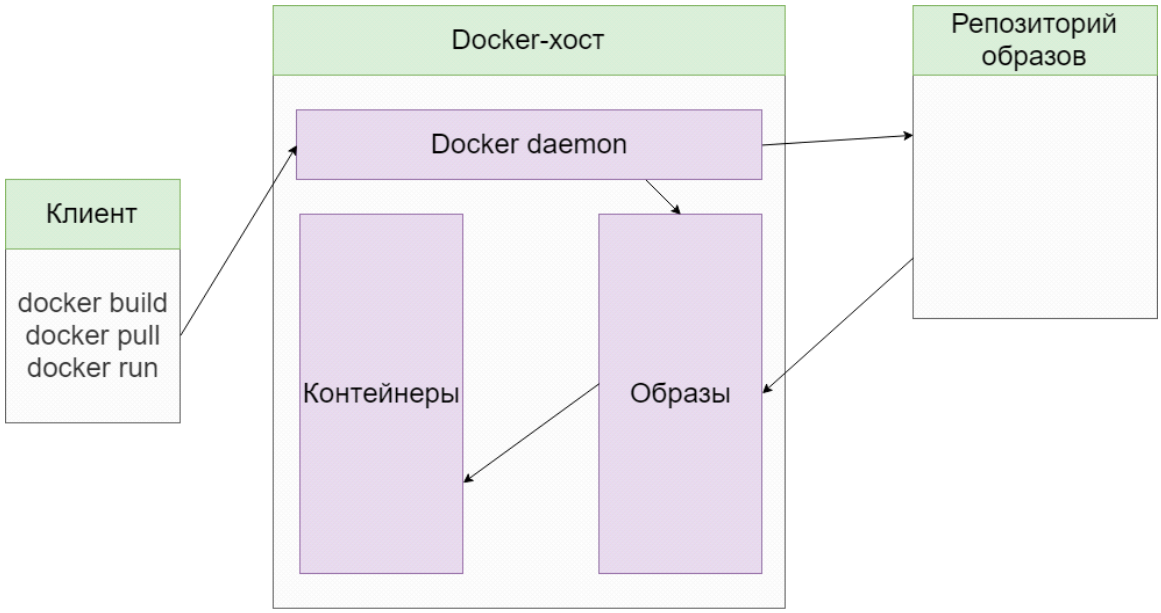

Рисунок 33. Общая схема работы системы с Docker Engine

Из предоставленной информации следует вывод, что Docker является гибкой системой для переноса образов между различными хостами. Кроме того, благодаря своей гибкости он позволяет с минимальными трудозатратами настроить взаимодействия в распределенной системе, а также быстро развернуть на хост-машине вспомогательное программное обеспечение из репозитория образов, такое как CI/CD системы, системы логгирования и т.д.

2.5. Строительные блоки для микросервисов

Одним из основных недостатков микросервисной архитектуры является целенаправленное дублирование похожего, либо абсолютно идентичного, программного кода в различных микросервисах. Целью данного подхода является снижение связности между компонентами приложения для достижения большей автономности и изолированности. Микросервисная архитектура предполагает сведение к минимуму общих библиотек и сборок, т.к. это препятствует возможности изменения одного из компонентов, не затрагивая при этом работоспособность других. Любые

изменения в общем коде требуют повторной сборки и публикации всех микросервисов, что нивелирует саму суть данного подхода.

Учитывая все вышесказанное, следует вывод, что в идеальном случае все микросервисы должны иметь свою кодовую базу, не пересекающуюся с другими сервисами. Необдуманное следование данному принципу приводит, с одной стороны, как к полной независимости кодовой базы компонентов друг от друга, так и к переизбытку дублирующегося кода, что усложняет его поддержку и отладку. Для решения описанных ранее проблем, используется подход, при котором в общую кодовую базу выносится программный код, который единым образом используют все компоненты, либо большая их часть. Полученные сборки называют строительными блоками (Building Blocks) микросервисов.

Строительные блоки представляют собой кодовую базу, с которой единым образом взаимодействуют все микросервисы. Если в таком блоке происходят какие-то изменения, требуется изменить взаимодействие с ним для всех модулей. Для того, чтобы снизить издержки на решение подобных задач, необходимо, чтобы строительные блоки имели слабую связанность с конкретными сервисами. В общем случае они представляют собой абстрактные сущности, которые не навязывают разработчику никаких конкретных реализаций, что полностью основано на принципе инверсии зависимостей.

Примером строительного блока является набор сборок, отвечающих за взаимодействие с асинхронной шиной событий по протоколу AMQP. Базовая сборка содержит в себе список абстракций для реализации взаимодействия по данному протоколу:

- Менеджер подключений к брокеру сообщений. Представляет собой абстракцию, описывающие методы установления связи между приложением и шиной.
- Абстракция, описывающая интерфейс взаимодействия приложения с асинхронным брокером сообщений.

- Менеджер подписок, связывающий между собой типы событий, передающихся через шину и подписчиков на данные события, получающие и обрабатывающие события, переданные им брокером.
- Базовая реализация события, передающегося по шине.
- Базовая реализация обработчика событий.

В данном случае имеется сборка, описывающая в общем виде взаимодействие с какой-либо асинхронной шиной событий. Все микросервисы, реализующие взаимодействие с брокером будут использовать абстракции из данной сборки. Вторым строительным блоком будет конкретная реализация описанных в предыдущей сборке абстракций. В проекте веб-приложения по трудоустройству в качестве шины событий используется брокер сообщений RabbitMQ. Вторая сборка в строительных блоках будет описывать взаимодействие с данным брокером, для чего необходимо реализовать:

- Менеджер подключений к брокеру сообщений RabbitMQ.
- Интерфейс взаимодействия с данной шиной событий.

Микросервисы не используют напрямую реализации из данной сборки, конкретные реализации абстрактного взаимодействия внедряются с помощью контейнеров инверсии зависимостей, что отвязывает модули от конкретного брокера и предоставляет более гибкое взаимодействие.

2.6. Взаимодействие микросервисов

Микросервисы являются автономными и изолированными друг от друга приложениями, их взаимодействие через общую кодовую базу является невозможным. Взаимодействие между модулями осуществляется с помощью протоколов, таких как HTTP, AMQP, RPC и т.д.

Выделяется два основных вида взаимодействия:

1. Синхронное взаимодействие. При данном подходе микросервисы обмениваются данными в режиме, предусматривающем ожидание ответа от конечной точки, на которую был отправлен запрос.

Примером протокола для реализации синхронного взаимодействия является HTTP. Данный подход используется, когда микросервис не может продолжить свою работу без данных, полученных от другого компонента. Использование описанного подхода в общем случае указывает на ошибки при организации архитектуры приложения, так как указывает на сильную связанность между модулями, пример такого взаимодействия показан на рисунке 34.

2. Асинхронное взаимодействие. При использовании данного подхода микросервис не ждет ответа от других конечных точек, так как не имеет информации о наличии и количестве других модулей. Пример протокола: AMQP. Данный вид взаимодействия, показанный на рисунке 35, является предпочтительным, т.к. уменьшает до возможного минимума связь между компонентами программной системы и упрощает горизонтальное и вертикальное масштабирование системы.

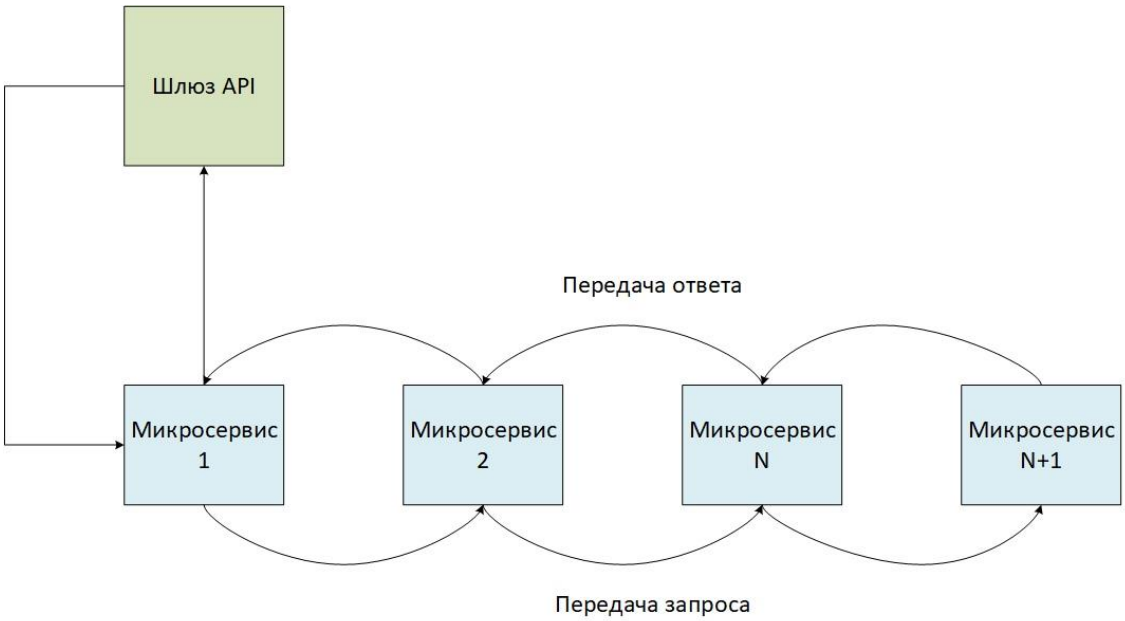

Рисунок 34. Синхронное взаимодействие нескольких микросервисов

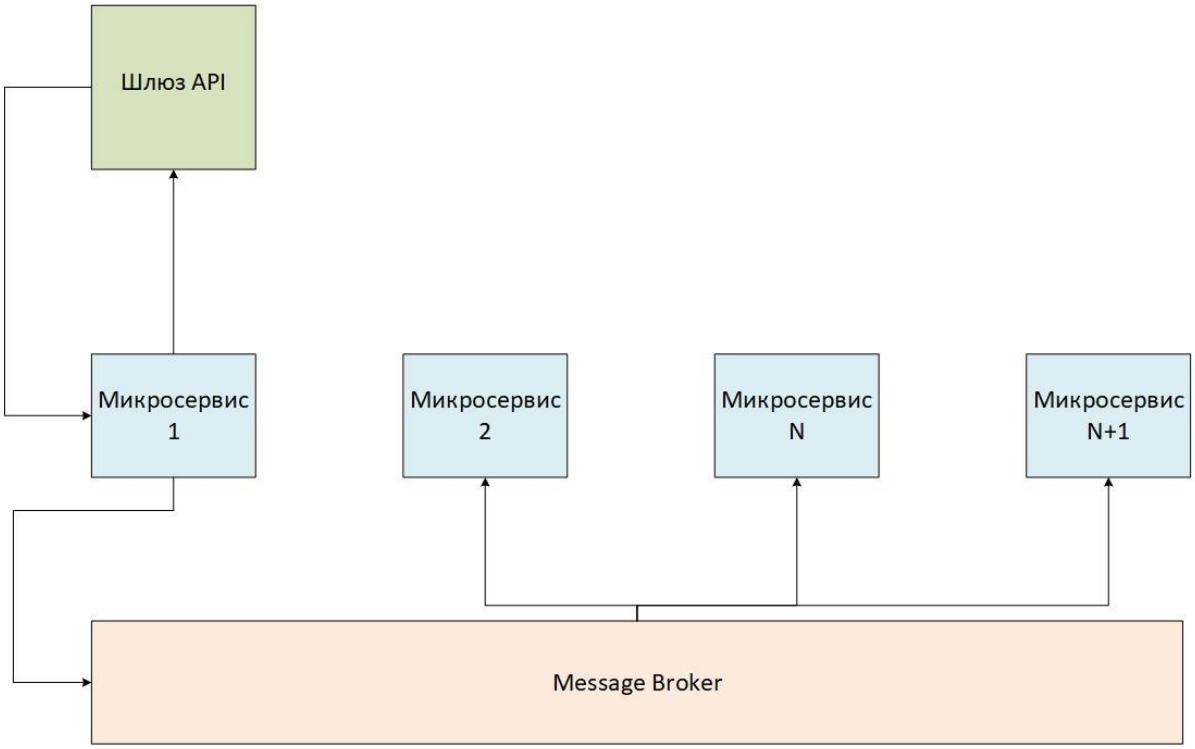

Рисунок 35. Асинхронное взаимодействие микросервисов

Микросервис 1 не знает о том, какие именно микросервисы получат его сообщение.

Веб-приложение по трудоустройству является полностью событийно-ориентированной (`event-driven`) и осуществляет взаимодействие модулей только по асинхронным протоколам, таким как AMQP, используя брокер сообщений RabbitMQ.

После того архитектура серверной была рассмотрена на высоком уровне, предлагается рассмотреть детализацию структуры основных микросервисов.

2.7. Подробное описание реализации микросервисов приложения

2.7.1. Микросервис «Дней карьеры»

Микросервис предназначен для хранения данных о предстоящих и прошедших «Днях карьеры». Сущность «Дня карьеры» хранит информацию о связанным с ним образовательным учреждением через внешний ключ `EducationInstitutaionId`, а с участвующими работодателями – через внешний ключ `EmployerId`. Место проведения мероприятия состоит из

нескольких связанных сущностей: адреса, области (`Areas`), метро, линии метро, станции метро. Причем сущность `Areas` является рекурсивной, это связано с вложенностью описания географического места. К примеру, город ссылается на область, в которой он содержится, область – на регион, регион – на страну.

На рисунке 36 представлена ER-диаграмма для микросервиса «Дней карьеры».

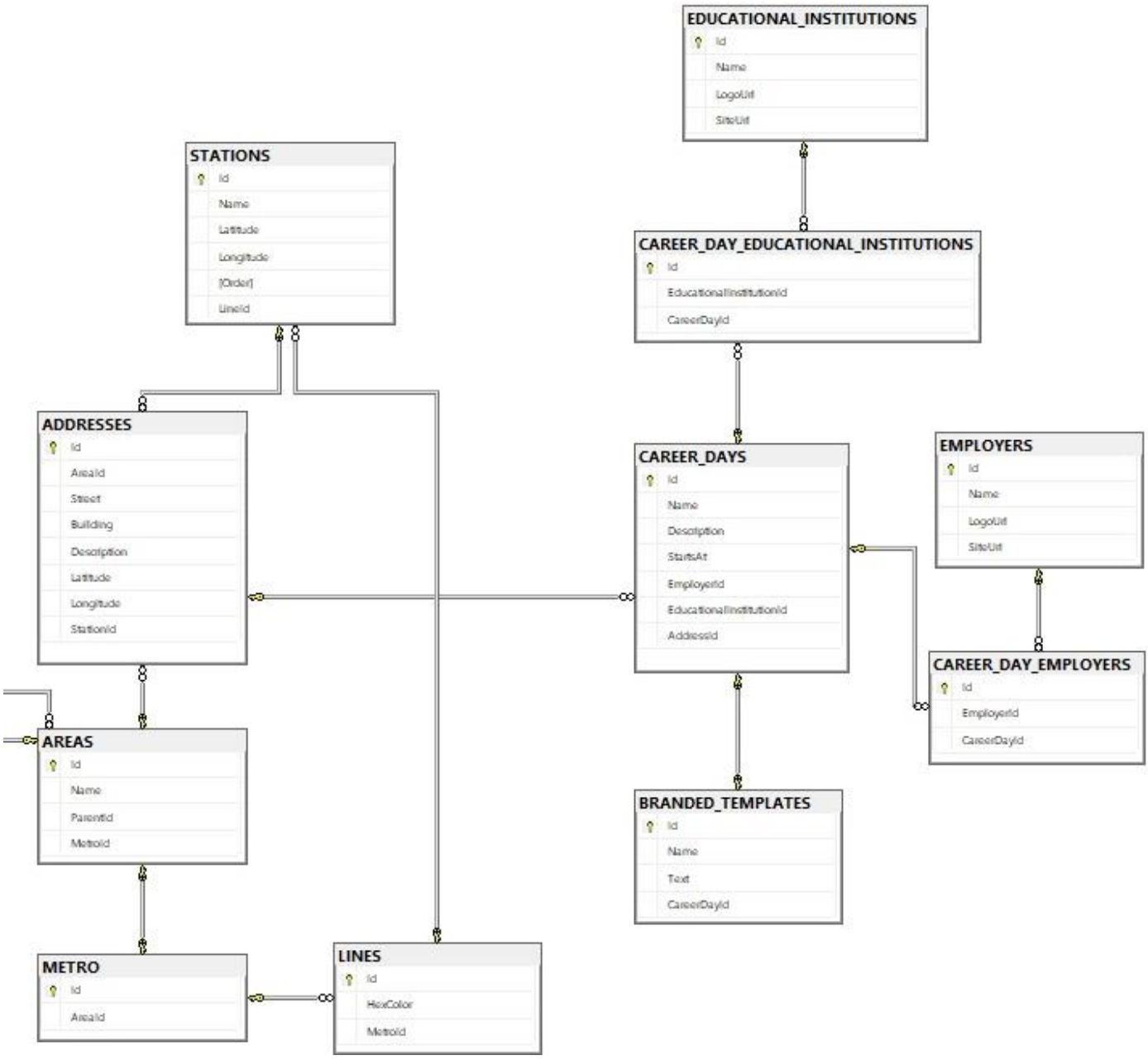

Рисунок 36. Концептуальная схема базы данных микросервиса «Дней карьеры»

2.7.2. Микросервис образовательных учреждений

Микросервис предназначен для хранения и предоставления данных по образовательным учреждениям. При реализации микросервиса было учтено, что у одного образовательного учреждения может быть несколько компаний-партнеров. Студенты проходят обучение на одном из факультетов, каждый факультет может иметь несколько специализаций.

На рисунке 37 представлена ER-диаграмма для микросервиса образовательных учреждений.

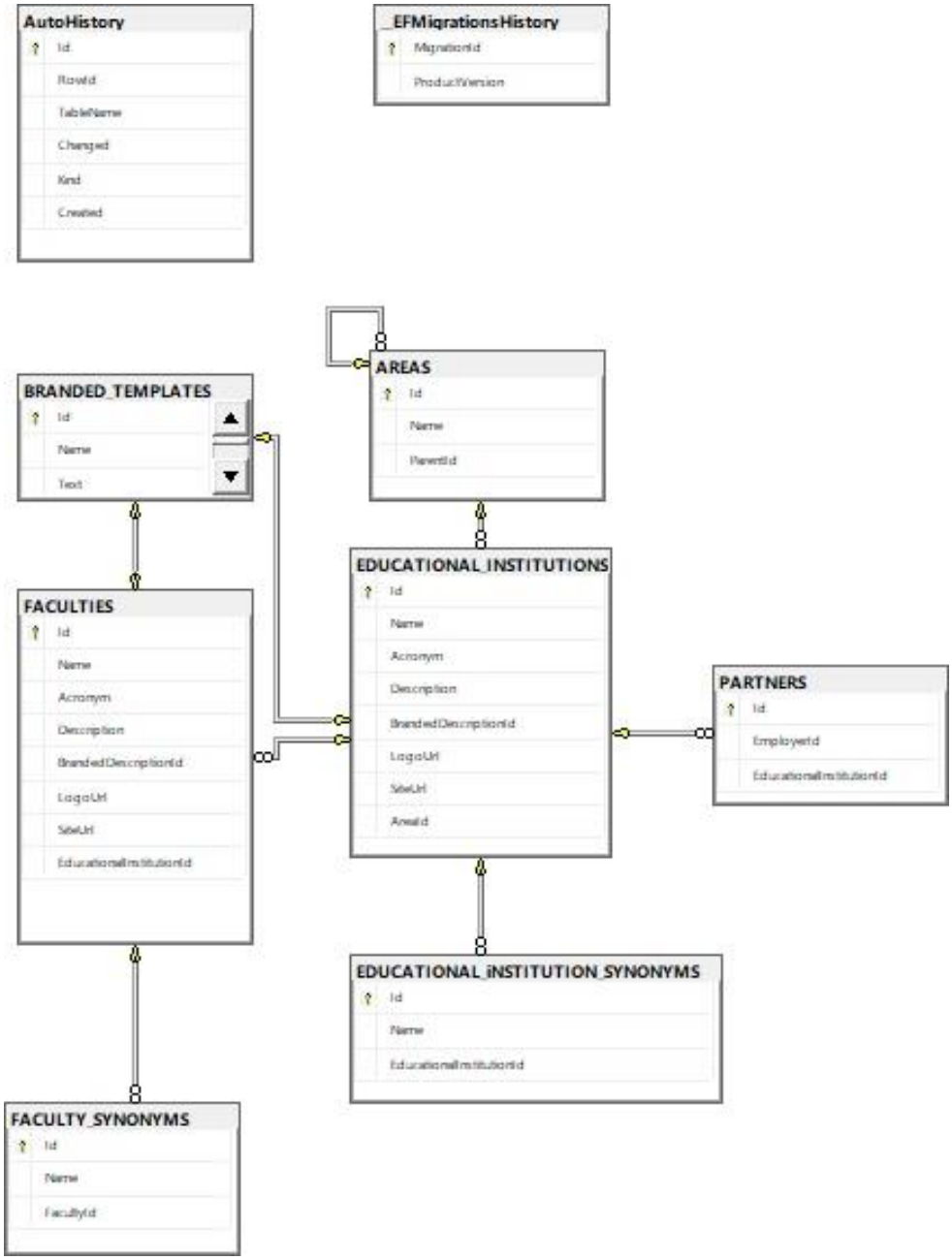

Рисунок 37. Концептуальная схема базы данных микросервиса образовательных учреждений

2.7.3. Микросервис работодателей

Микросервис предназначен для хранения и предоставления информации по работодателям, общей информации о них, для вакансий предусмотрен отдельный микросервис. Веб-приложение допускает возможность наличия у работодателя нескольких партнеров, синонимичных названий и наличия отделений, каждое отделение может иметь собственный адрес.

На рисунке 38 представлена ER-диаграмма для микросервиса работодателей.

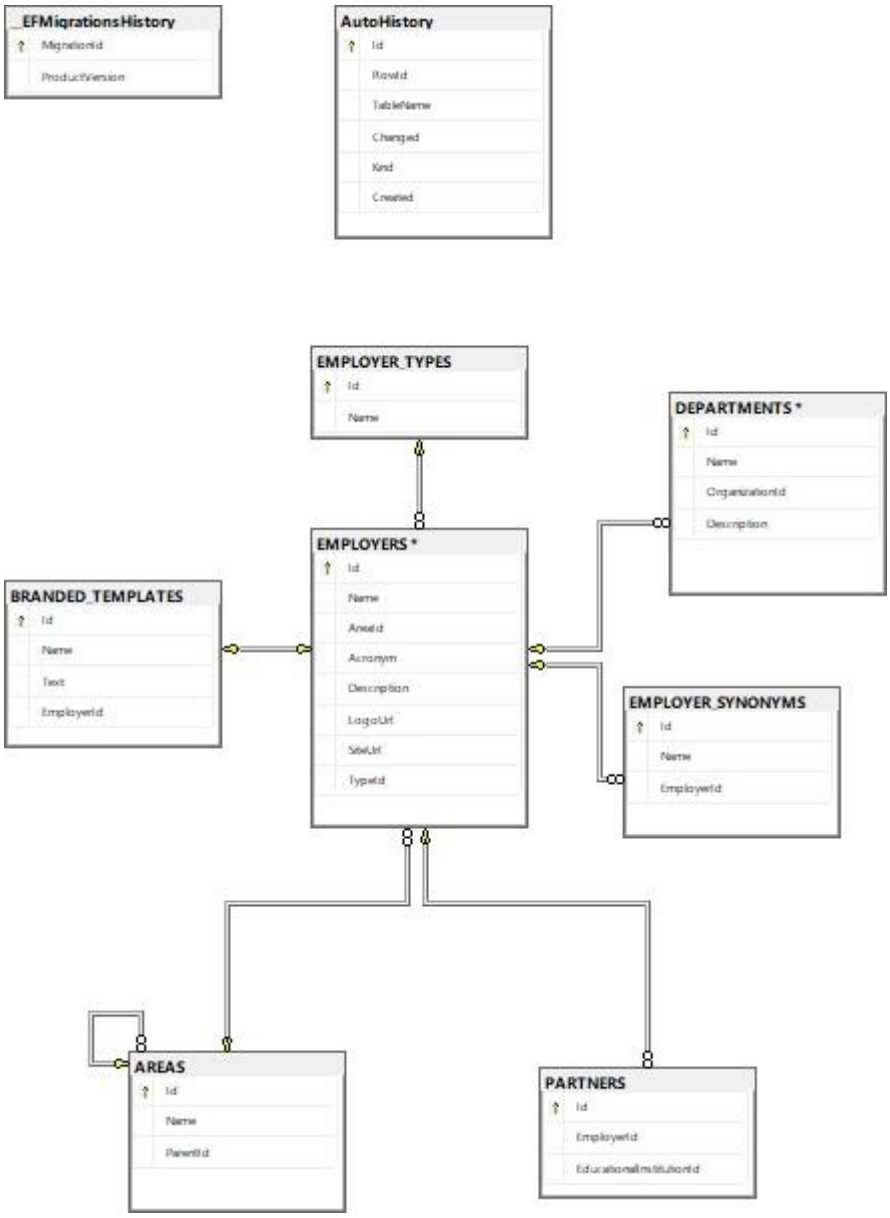

Рисунок 38. Концептуальная схема базы данных микросервиса работодателей

2.7.4. Микросервис авторизации

Микросервис позволяет войти в систему неавторизованным пользователям при вводе корректных учетных данных. Сущность `AspNetUsers` содержит поля и данные, которыми наделен каждый пользователь системы. Для хранения дополнительной информации о менеджерах учебных заведений и компаний предусмотрены соответствующие сущности – `Educational_Institution_Manager_Attributes` и `Employer_Manager_Attributes`. Рисунок 39 содержит ER-диаграмму микросервиса авторизации.

Сущность `AspNetUsers` связана с другими сущностями следующим образом:

- с `Areas` – через `AreaId`,
- с `Educational_Institution_Manager_Attributes` – через `EducationalId`,
- с `Employer_Manager_Attributes` – через `OrganizationId`,
- с `Contacts` – через `ContactsId`,
- с `Site_Types` – через `SiteTypeId`,
- с `Genders` – через `GenderId`.

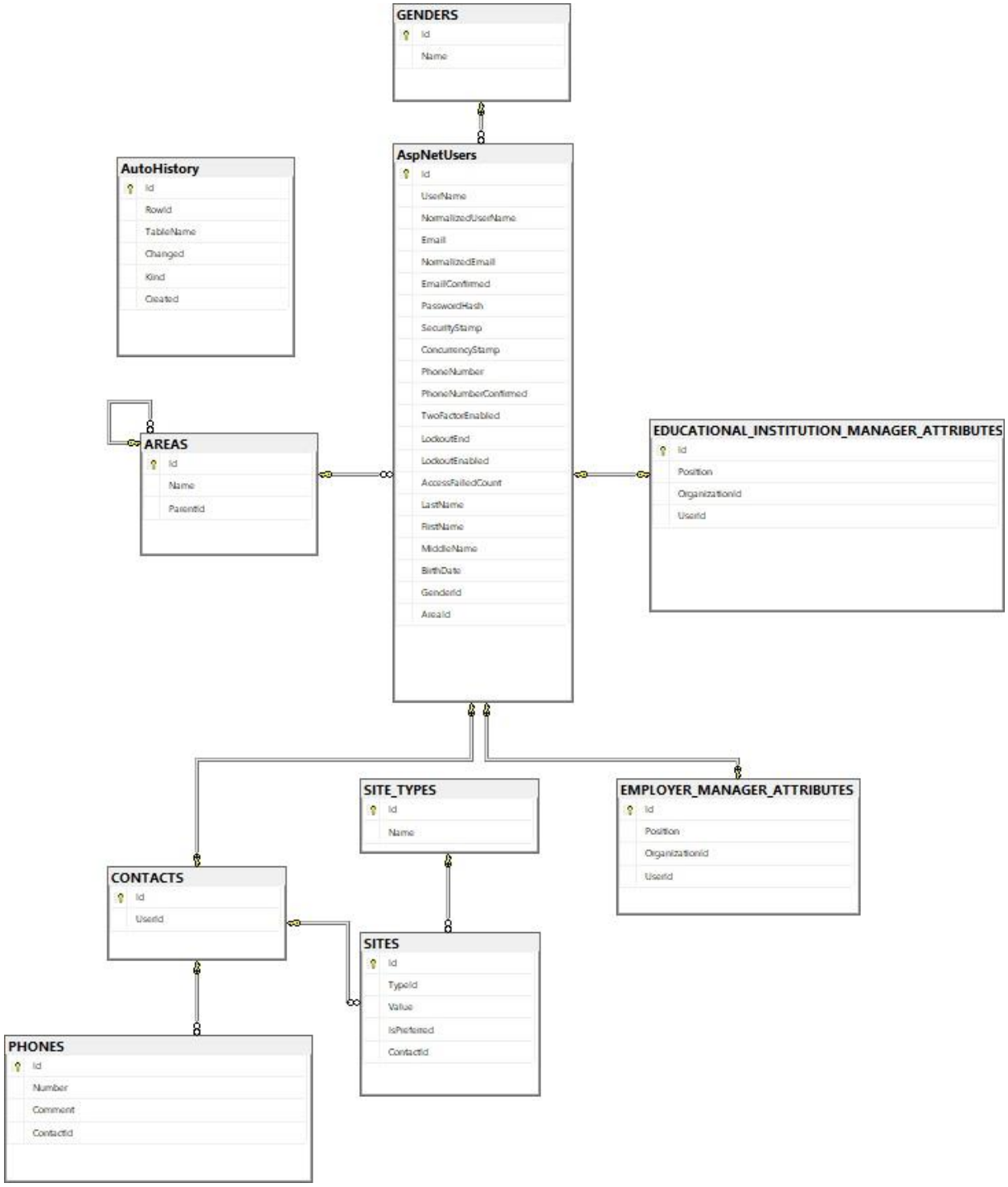

Рисунок 39. Концептуальная схема базы данных микросервиса авторизации

2.7.5. Микросервис платных сервисов

Микросервис предназначен для хранения и предоставления информации о платных услугах, доступных работодателям, соискателям или менеджерам учебных заведений. Примером платной услуги является продвижение в поисковой выдаче резюме или вакансии. Эти функциональные возможности были изначально заложены в архитектуру

веб-приложения на ранних стадиях. Оценка коммерческого потенциала, представленная в 4 главе работы, показала, что одним из ключевых недостатков главного конкурента, Headhunter, является наличие платных возможностей. Система, лишенная такого недостатка, стала бы значительно более конкурентоспособной. Более того, полноценное использование платных сервисов требует подключения платежного шлюза, что займет непозволительно много времени в рамках проведения научно-исследовательской работы. Учитывая это, командой разработчиков было принято решение вынести функционал прототипа платных услуг в отдельный микросервис и временно прекратить его развитие. На рисунке 40 можно увидеть ER-диаграмму микросервиса платных сервисов.

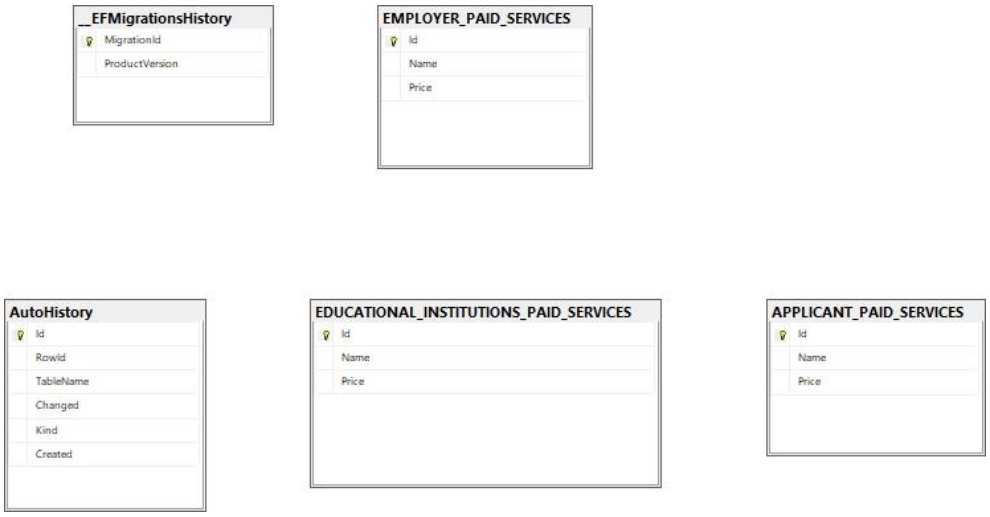

Рисунок 40. Концептуальная схема базы данных микросервиса платных сервисов

2.7.6. Микросервис резюме

Микросервис предназначен для возможности по созданию, удалению и редактированию резюме. Резюме – один из двух основных документов предметной области, поэтому его диаграмма сущностей и связей, представленная на рисунке 41, содержит 39 сущностей, сложным образом связанных между собой. У одного соискателя может быть несколько резюме.

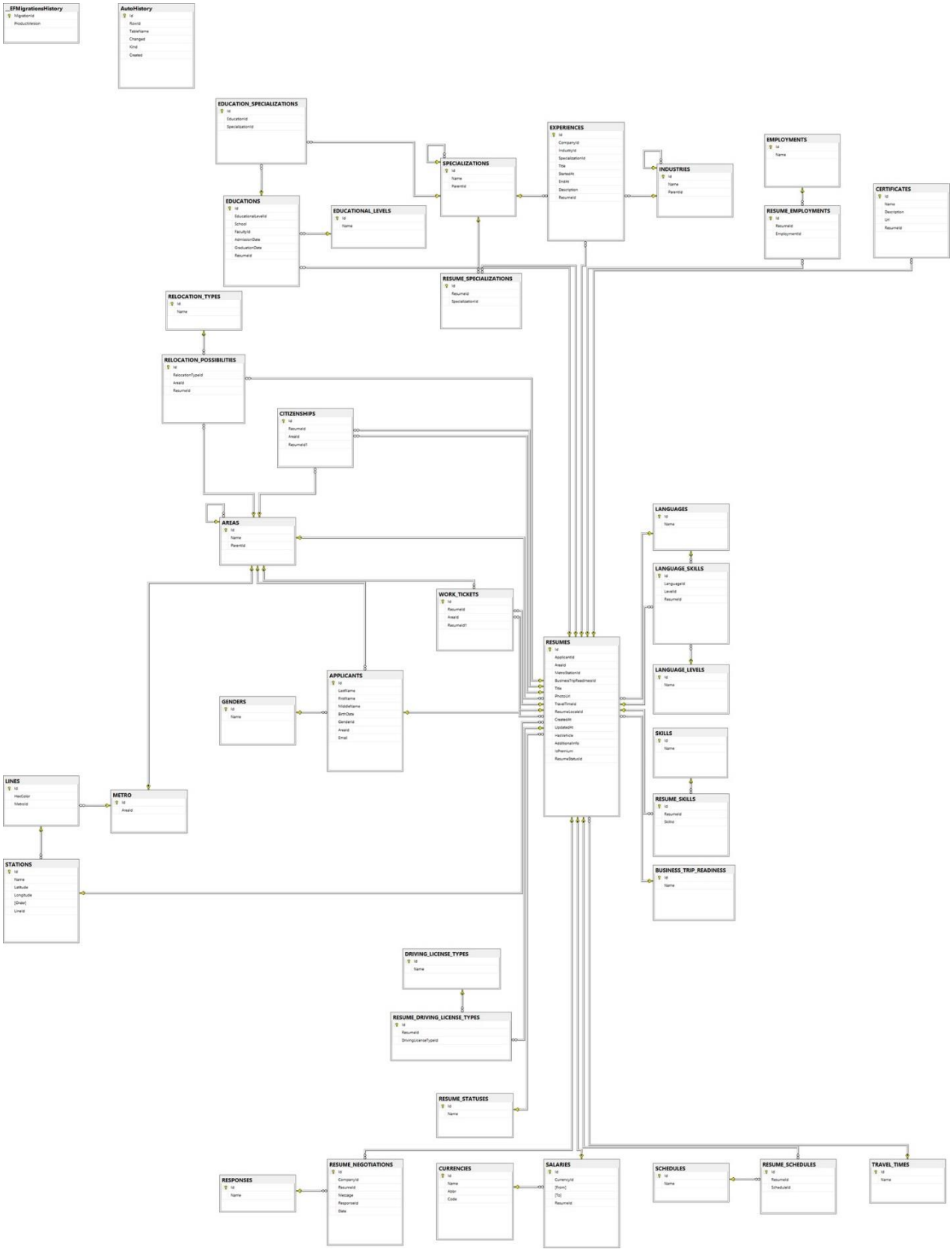

Рисунок 41. Концептуальная схема базы данных микросервиса резюме

2.7.7. Микросервис вакансий

Микросервис предназначен для возможности по созданию, удалению и редактированию вакансий. Вакансия также является основным документом в

сфере трудоустройства, поэтому для её реализации необходимо обеспечить связность большого числа сущностей, что показано на рисунке 42 в виде ER-диаграмма микросервиса вакансий. У одной компании может быть несколько вакансий.

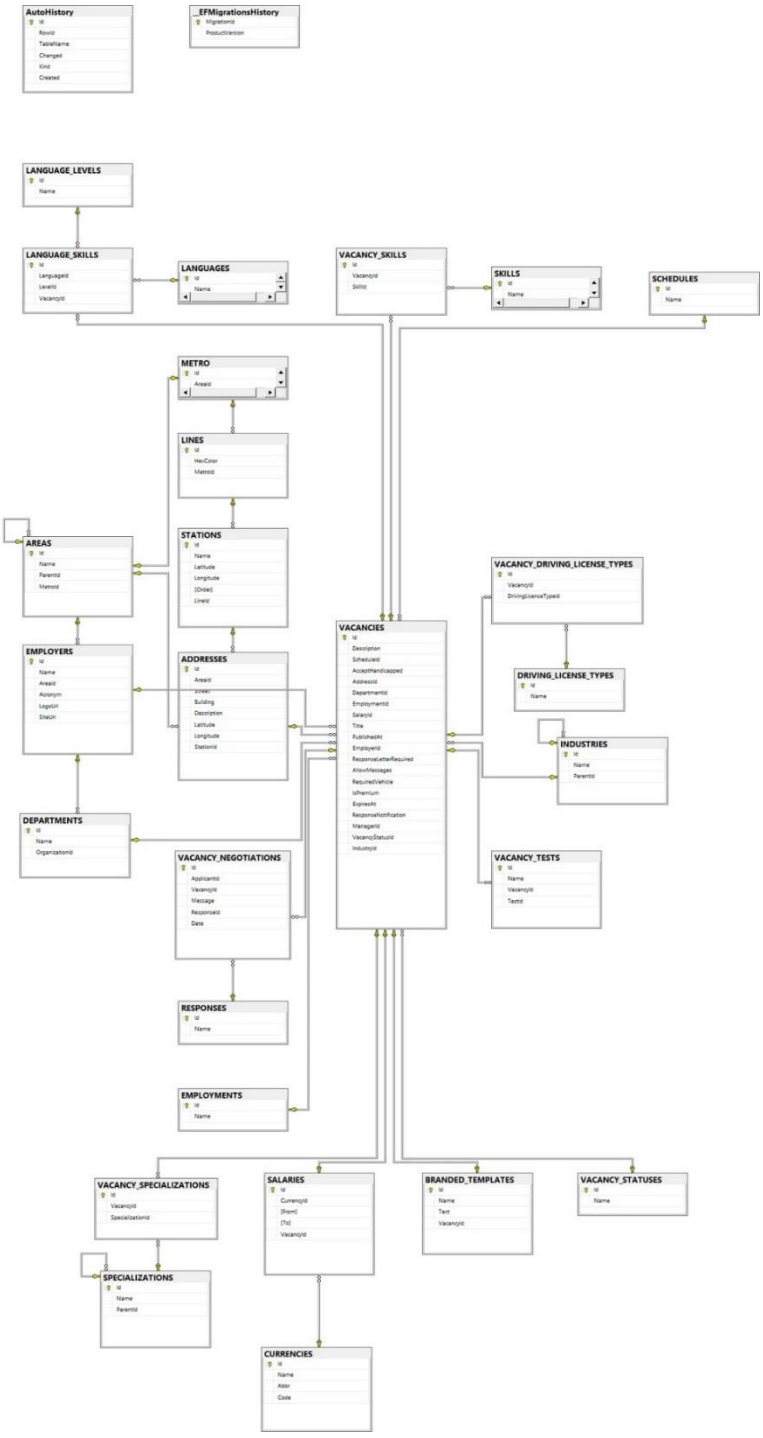

Рисунок 42. Концептуальная схема базы данных микросервиса вакансий

2.8. Архитектура клиентской части

Разрабатываемое веб-приложение по трудоустройству требует тщательного подхода к выбору технологий и методологий для разработки клиентской и серверной части. Это обусловлено рядом причин, одна из которых формируется из поставленной задачи: необходимо организовать клиентскую часть так, чтобы для пользователей с разными ролями оставался один и тот же собранный файл – `bundle`, использовался единый дизайн, но при этом был реализован различный функционал. Как пример можно привести то, что во вкладке «вакансии» для соискателя должны отображаться вакансии компаний с фильтрами, на эти вакансии он может оставить свой отклик, а для менеджера компании в той же вкладке необходимо отображать вакансии компании, которую представляет текущий менеджер, возможность их редактирования, удаления, добавления новых и т.д.

Данная задача побуждает не только к созданию нестандартной архитектуры веб-приложения, но и к разработке системы запросов, приспособленной к тому, чтобы для разных ролей пользователей реализовывать уникальный функционал, при этом не вызывая дублирования кода, что является анти-паттерном программирования.

Также разрабатываемое веб-приложения должно соответствовать всем современным требованиям к разработке клиентской части, т.е. являться легковесным, реактивным, производительным, легко модифицируемым в соответствии с внезапно появляющимися бизнес-требованиями.

Для решения задачи с гибкостью приложения было решено реализовать модульную архитектуру. Данный подход позволяет в кратчайшие сроки менять структуру веб-приложения, добавлять новый функционал, при надобности избавляться от уже существующего функционала, а также модифицировать его.

В клиентской части веб-приложения была реализована структура директорий, содержащая WebPack, папку `src` и `node_modules`.

Директория `webpack` содержит в себе файлы настроек с расширением `.js`. Данные файлы позволяют настроить сборки проекта для различных целей и указывают сборщику, как собирать проекты в версии для клиентов и как во время разработки; какие применять настройки для сборки браузерной версии веб-приложения и какие для сборки мобильной версии. В файлах настроек указано, какие загрузчики необходимо применять для файлов из структуры проекта с различными расширениями. В конечном итоге задача данных файлов настроек заключается в том, чтобы превратить структуру проекта, удобную для разработки, в три файла: `index.html`, `index.js`, `index.css`. Три файла браузеры пользователей будут гораздо быстрее загружать с сервера, также сборщик сжимает код, что делает общий вес приложения в разы меньше исходного. Также система сборки, руководствуясь файлами настроек, выполняет одну из своих главных задач – транспиляцию кода.

При этом в результате клиент получит файл, использующий только язык JavaScript. Причем этот файл будет использовать исключительно синтаксис старых версий языка на `EcmaScript 5`, что позволит работать веб-приложению со всеми браузерами, в том числе и считающимися устаревшими, а также на любых устройствах, начиная с мобильных телефонов и заканчивая персональными компьютерами.

Директория `node_modules` необходима для хранения библиотек, используемых в приложении. Процесс установки библиотек и фреймворков происходит посредством *NPM*. *NPM* позволяет использовать большое количество современных библиотек, что дает возможность подобрать необходимый набор инструментов разработчика при разработке конкретного приложения. Установка пакетов происходит посредством команды `npm install <имя пакета>`. После запуска данной команды в командной строке, NPM автоматически подгружает в проект указанную библиотеку или иной инструмент разработчика. Также есть возможность установить пакет определенной версии в том случае, если вам необходимо обеспечить

совместимость тех или иных средств разработки. Помимо этого, если была найдена достойная альтернатива используемой библиотеке, или в результате разработки было выявлено, что используемая библиотека не является необходимой в данном веб-приложении, с помощью NPM можно легко удалить эту библиотеку, используя команду `npm uninstall <имя пакета>`. Это позволит уменьшить общий вес приложения, что повысит его производительность.

В директории `node_modules` разработчик может посмотреть исходный код любой загруженной библиотеки и, при желании, даже изменить его. Данное преимущество чаще используется в целях изучения работы тех или иных библиотек, а также получения каких-либо знаний в области программирования. Это связано с тем, что большинство библиотек пишутся опытными, умелыми программистами, которые часто используют не совсем популярные, но иногда очень полезные подходы к программированию, о которых не всегда можно найти информацию в сети.

Непосредственно сам программный код веб-приложения располагается в директории `src`. Она предназначена для построения архитектуры приложения, размещения модулей и компонентов. Данная директория включает в себя:

- `Assets`,
- `Components`,
- `Features`,
- `Modules`,
- `Shared`,
- `configureStore.js`,
- `index.js`,
- `indexMobile.js`,
- `createRouter.js`.

В директории Assets находятся всего два файла: index.html и indexMobile.html. Это единственные файлы с расширением «HTML» во всей структуре проекта. Они являются отправной точкой для сборщика проекта, и в них в конечном итоге запишется вся верстка проекта, реализованная с использованием синтаксиса JSX, предоставляемого библиотекой React. В данных файлах внутри тега <body> находится всего один элемент с классом root, он служит для того, чтобы в дальнейшем с помощью функции React.render() все содержимое проекта поместилось внутрь данного элемента.

Следующая директория – components. Ее содержимое соответствует названию, то есть в ней лежат простые компоненты. Зачастую, это переиспользуемые компоненты, такие как: Input, Button, SwitchBox и т.д. Но также в ней встречаются и уникальные на странице веб-приложения компоненты, такие как: Header, Footer.

Особенностью компонентов данной директории является то, что они не несут в себе какого-либо функционала, связанного с redux-хранилищем. Они служат преимущественно для того, чтобы помочь реализовать функционал в более крупных компонентах. Также большинство таких компонентов являются гибко настраиваемыми, то есть позволяют программисту возможность посредством передачи определенных параметров типа props задавать им необходимые цвет, размер, тип и т.д.

Данные компоненты, как и все остальные в проекте, написаны с использованием методологии BEM, что позволяет использовать их в любом месте в проекте, независимо от уровня вложенности и особенностей родительского компонента [14]. Также они являются полностью независимыми от контекстного окружения, что дает возможность при надобности переносить их из одной части проекта в другую, не теряя зависимостей при импортировании. Все компоненты данной директории являются функциональными (stateless), что повышает общую производительность кода.

Следующая директория features является одной и самых важных в проекте: именно она содержит наиболее важные компоненты, обеспечивающие предоставление пользователю внедренного функционала.

В ней располагаются крупные компоненты, такие как: Resume, Vacancy, Profile и т.д. Особенностью данных компонентов является не только прямая передача функционала веб-приложения юзеру, но и то, что каждый из них является крупной, независимой частью веб-приложения, имеющего свой redux-state, actions, reducers, поддиректорию с данными, необходимыми для корректной работы компонента. Также, большинство компонентов имеют свои дочерние компоненты, позволяющие им реализовывать свой функционал. Иными словами говоря, если компоненты из директории components были управляемыми, то компоненты данной директории являются управляющими. Они имеют свое окружение, используют иные компоненты, управляют данными посредством redux.

Также стоит упомянуть, что многие из компонентов директории features являются statefull components, это обусловливается тем, что в подобных компонентах зачастую необходимо использование методов жизненного цикла, таких как: componentWillMount(), componentWillRecieveProps(), componentWillUpdate(), componentWillUnmount() и т.д. Связано это напрямую с тем, что данные компоненты нуждаются в получении данных с сервера в момент своего рендеринга. Следовательно, в начале своего жизненного цикла компонент должен запросить данные, для работы с которыми он и предназначен. В некоторых ситуациях необходимо вызвать определенные actions.

Собственные redux-хранилище, action, reducers, необходимы в первую очередь для того, чтобы снабдить компонент всем необходимым функционалом и данными, а также для абстрагирования его от остальных частей приложения. Используя подобный подход, можно добиться полной

модульности приложения: появляется удалять, добавлять и изменять компоненты, не нарушая его работоспособности. Это означает, что, удалив любой компонент, мы не сломаем остальные, так как у них собственное окружение, и они не нуждаются в наличие остальных компонентов в проекте.

В `actions` также вызываются запросы к API, что позволяет загружать данные с сервера и записывать их в `redux-хранилище` компонента для дальнейшего использования. Там же вызываются и запросы для отдачи данных на сервер и иные. Более подробно архитектура веб-приложений, использующих Redux, показана на рисунке 43.

Рисунок 43. Взаимодействие представления с хранилищем посредством использования `Reducers`

Следующая важная директория – `modules`. Данная директория является довольно простой по своей структуре, но выполняет крайне важные задачи: обеспечивает модульность и маршрутизацию в веб-приложении. В каждой папке данной директории находится отдельный модуль, который определяет по каким путям данного модуля отображать какие компоненты. Каждый такой модуль возвращает класс с методом `getRoutes()`, который передает все указанные в модуле маршруты в приложение. Также компоненты передают некоторые `match-параметры` в вызываемые компоненты, что позволяет расширить функционал маршрутизации в приложении.

Помимо модулей в данном приложении находятся еще два файла: desktop.js и mobile.js. Данные файлы импортируют в себя необходимые модули и затем экспортируют их дальше в проект.

Следует детально ознакомиться с файлами, находящимися в корневой директории src: index.js, indexMobile.js и createRouter.js. В первых двух файлах из директории модулей импортируются все модули для браузерной и мобильной версии соответственно. Там же вызывается метод React.render(), в котором в HTML передается компонент App. Данный компонент имеет стандартное в библиотеке React поле children. В этом поле как раз и прокидываются все маршруты веб-приложения. Чтобы получить весь набор маршрутов из модулей, вызывается функция, описанная в createRouter.js. Они принимает в себя экземпляры классов модулей, вызывает у них циклически метод getRoutes() и формирует один большой список маршрутизации приложения. Данный прием позволяет на этапе программирования удобно разделять логику работы каждого модуля в отдельности, но в собранном проекте использовать уже совокупность модулей в веб-приложении.

Тут же лежит не менее важный файл configureStore.js. В нем происходит одна из самых важных частей любого проекта, использующего методологию разработки Flask, в частности Redux, а именно создание глобального хранилища приложения. Как уже упоминалось выше, каждый важный компонент имеет свое хранилище. Именно в этом файле происходит слияние хранилищ каждого такого компонента в одно глобальное хранилище. Это реализовано посредством метода, встроенного в библиотеку react-redux createStore().

В данный метод передается объект, получившийся посредством объединения всех хранилищ, промежуточных слоев и дополнительных параметров [15]. В приложении используется промежуточный слой (middleware) thunk, предоставляемый библиотекой redux-thunk. Данная библиотека позволяет использовать асинхронность в actions,

которая является необходимой при работе с запросами к серверу. С помощью данной легковесной, содержащей около 15 строчек кода, библиотеки появляется возможность использовать `dispatch()` для вызова различных reducers, а также асинхронной работы с запросами.

Теперь предлагается рассмотреть дополнительные параметры. Простыми словами, это объекты или функции, доступ к которым можно получить в любом `action`. Передав их один раз в момент создания хранилища, открывается возможность использовать эти параметры во всех действиях без регулярного импортирования. В данном проекте в дополнительные параметры передается экземпляр класса API, благодаря чему, в любом действии появляется возможность обращения к системе запросов приложения. Это является весьма удобным проектным решением, так как для получения доступа к системе запросов в любом действии достаточно лишь извлечь данный параметр из параметров, приходящих в функцию по умолчанию.

Последней же ключевой директорией в структуре проекта является директория `shared`. Данная директория очень функциональна, поэтому стоит перечислить по порядку, что в ней есть и для чего это предназначено.

Одним из важнейших аспектов данной директории является наличие в ней папки API, в которой описана вся система запросов веб-приложения. В качестве библиотеки для запросов было решено использовать `axios`, так как они обладает высоким процентом положительных отзывов IT-сообщества и занимает крайне мало места. При малом весе обладает такой же функциональностью, что и любая другая библиотека запросов, но при этом еще и имеет элементарный интуитивно-понятный синтаксис. С помощью данной библиотеки в проекте описан класс `BaseApi`, который имеет один единственный метод `sendQuery()`. Данный метод принимает следующие параметры: тип запроса, URL запроса, тело запроса, заголовки запроса, конвертер для успешного запроса, конвертер для неудачного запроса. После чего данный метод вызывает необходимые методы из библиотеки `axios` и

передает в него нужные параметры. Далее после получения ответа с сервера, к полученным данным применяется переданный ранее конвертер (чистая функция) в зависимости от успешности запроса.

На выходе после использования любого запроса к API функции возвращают ответ, в котором хранится информация об успешности или неуспешности запроса, сконвертированные данные, полученные от сервера, и сообщение об ошибке в том случае, если запрос прошел неуспешно.

От класса `BaseApi` в дальнейшем наследуются все остальные конкретные API (например, `ResumeApi`, `ProfileApi` и т.д.), и эти классы имеют доступ к методу базового класса, что позволяет не писать больше количество отдельных запросов. Таким образом, для любого класса API вызовется один и тот же метод `sendQuery()` из базового класса, но в него будут переданы разные параметры функции.

Также в директории `shared` хранится языковая система. В веб-приложении по трудоустройству языковая система реализована следующим образом. В проекте есть объект `locale`, который для каждого ключа языка содержит определенный объект словарей к этому языку. В каждый словарь импортируются меньшие объекты-словари, которые для каждого крупного компонента свои. В дальнейшем при запуске приложения вызывается действие, которое по текущему языку пользователя загружает в глобальное хранилище объект словарей для конкретного языка по ключу из объекта `locale`. В случае смены языка в глобальное хранилище загружаются новые словари, что обеспечивает смену языка.

Сами словари каждый компонентом, которому они необходимы, получает в функции `mapStateToProps()` из глобального хранилища по ключу, который отдает словарь, необходимый данному компоненту. Таким образом, для каждого компонента, которому необходимо наличие словаря, этот словарь передается в `props`, откуда компонент может его извлекать, использовать и при надобности передавать по виртуальному DOM-дереву дальше.

Помимо этого, в директории `shared` располагается папка `styles`. В ней хранятся глобальные настройки стилей для всего проекта, для мобильной версии и для браузерной версии. Так как в качестве языка стилей в данном проекте используется SCSS, то это дает возможность в данных глобальных файлах задавать глобальные цветовые переменные, писать медиазапросы для изменения шрифтов, а также описывать глобально часто встречающиеся в проекте анимации [16]. В дальнейшем достаточно в любом другом компоненте в файле стилей импортировать данные глобальные файлы и доступ к подобных глобальным переменным будет получен. С помощью подобного подхода легко реализуется переиспользуемость дизайна, так как при необходимости можно в одной переменной изменить цвет, и он изменится во всем проекте.

Последней особенностью директории `shared` является папка `utils`, она хранит в себе различные данные, необходимые для функционирования приложения. Примером таких данных могут быть некоторые функции, модели, словари для меню, элементы навигации, иконки и т.д.

Выше была описана общая структура клиентской части веб-приложения по трудоустройству. Стоит отметить, что с помощью подобной структуры достигается независимость и модульность архитектуры проекта. Данный подход позволяет во время разработки рассматривать каждую его часть, как независимую единицу, и в то же время в конечной сборке получать всего лишь три файла, которые будут обладать той же функциональностью, что и все эти единицы в совокупности.

Далее стоит отметить, что в клиентской части веб-приложения по трудоустройству использовалась идиома функционального программирования. Для этого есть несколько причин.

JavaScript поддерживает функциональный язык. Это значит, что к написанию кода на JavaScript применима в первую очередь парадигма функционального программирования. Данный язык предоставляет все инструменты для написания хорошего программного кода через реализацию

функций, как простых, так и высшего порядка. В JavaScript любая функция является объектом, что позволяет передавать функции, как параметры, и использовать их в любом месте проекта. Также в данном языке понятно реализован функционал функций обратного вызова и обратный поток данных через подобные функции. Также функции в JavaScript работают быстрее, чем классы, так как классы в данном языке программирования по сути являются обычными функциями с применением дополнительных оберточных слоев.

Второй важной причиной является факт того, что React постепенно развивается в сторону функционального программирования, так как это позволяет использовать все преимущества новых версий `EcmaScript` и его особенности синтаксиса. Разработчики React открыто заявляют о том, что рекомендуют использовать функциональные компоненты, так как в ближайших версиях библиотеки планируется полностью отказаться от классов и перейти только на функциональные компоненты. Это связано в первую очередь с тем, что функциональные компоненты работают быстрее классов, а также имеют более понятный синтаксис. В свою очередь, разработчики React с каждой новой версией предоставляют программистам, использующим данную библиотеку, все больше инструментов для работы с функциональными компонентами. Таким образом, можно сделать вывод о том, что в ближайшем будущем функциональные компоненты будут иметь возможность предоставлять функционал классов, но имея повышенную в разы производительность.

И третьей, не менее важной причиной, является то, что функциональное программирование обеспечивает декларативных подход к разработке программного кода. Как известно, декларативный подход обеспечивает хорошую читаемость кода, а также простоту в изменении функционала, обеспеченного этим кодом. Декларативный подход ставит перед собой задачу решить вопрос: что необходимо сделать, в то время как императивный подход отвечает на вопрос: как необходимо решить задачу. Из этого следует, что написание функционального кода улучшает

поддерживаемость кода, а также упрощает для программистов, которые в дальнейшем будут работать с этим кодом, понимание написанного.

Из всего вышесказанного следует, что функциональное программирование является правильным решением при написании клиентской части веб-приложения по трудоустройству. В данном веб-приложении функциональность кода достигается в первую очередь за счет использования функциональных компонентов в рамках инструментов, предоставляемых разработчиками библиотеки. Также все аспекты веб-приложения, отвечающие за работу с данными, также строятся на функциях, как простых, так и высшего порядка. В данном веб-приложении использовались только `чистые` функции, то есть они принимают в себя некоторые параметры и работают только с ними, не меняя свое окружение [17]. При необходимости изменить окружение используются функции высшего порядка, которые вызывают внутри себя функции, которые передаются как аргументы. Также данный подход позволяет решить одну из сложнейших задач данного веб-приложения – изменять функционал в зависимости от ролей пользователя, не внося серьезных изменений в читаемую структуру проекта.

Данный результат был достигнут следующим программным решением. Вызванный модуль забирает из глобального хранилища роль пользователя, после чего вызывает не сам компонент, а декоратор над данным компонентом. В `декоратор` в качестве аргумента передается роль пользователя. Далее в `декораторе` текущая роль используется как ключ, значением к которому является необходимый компонент, который реализует функционал, необходимый данному пользователю [18].

Таким образом, достигается поставленная цель без изменений структуры проекта и ее усложнения. Данное решение позволяет, используя функциональное программирование, переопределять функционал для пользователей разных ролей, применяя декораторы. Таким образом, общий программный объем проекта практически не изменяется, но

производительность возрастает в разы по сравнению с тем, если бы был использован любой иной подход, не построенный на функциональном программировании.

Также в веб-приложении по трудоустройству активно используются новейшие возможности, предоставляемые React 16.8, такие как `hooks`. Использование `useState()`, `useEffect()`, `useReduce()` позволяет добиваться того же функционала, что и ранее компоненты, реализованные классами. Открываются возможности для использования локальных состояний, методов жизненного цикла компонентов и даже собственных хранилищ компонентов. Все эти преимущества достигаются благодаря функциям и их активному развитию и популяризации в *EcmaScript* и *React*.

2.9. Словари данных

Для обеспечения функционирования серверной части веб-приложения необходимо было реализовать 29 словарей данных, показанных на рисунке 44.

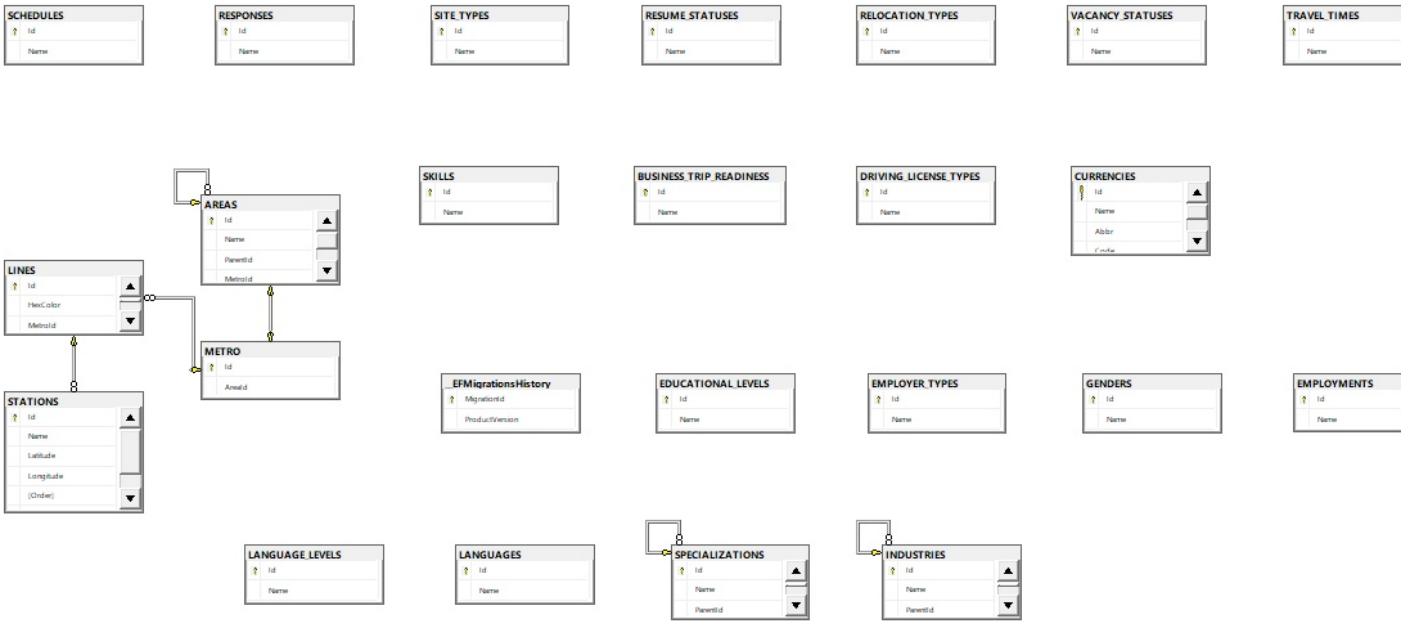

Рисунок 44. Используемые словари данных

Как видно все словари, кроме тех, что относятся к местоположению, являются простыми структурами, в состав которых входит от 2 до 5 атрибутов.

2.9.1. Словарь «Areas»

Словарь содержит иерархическую структуру вложенных географических объектов. Пример записи в такой структуре представлен на рисунке 45.

Рисунок 45. Фрагмент структуры справочника `Areas`

2.9.2. Словарь валют

Словарь содержит перечень валют с кодовым обозначением и аббревиатурой, в поле `rate` содержится отношение одного российского рубля к единице валюты. Пример записи в такой структуре представлен на рисунке 46. Значение `in_use` является логическим флагом и отражает возможность использования этой валюты для указания заработных плат. На данном этапе развития веб-приложения поддерживаются только доллары и российские рубли.

```
"id": "7",
    "code": "RUR",
    "abbr": "руб.",
    "name": "Рубли",
    "default": true,
    "rate": 1,
    "in_use": true
},
{
"id": "8",
    "code": "UAH",
    "abbr": "грн.",
    "name": "Гривны",
    "default": false,
    "rate": 0.419651,
    "in_use": false
},
{
"id": "9",
    "code": "USD",
    "abbr": "USD",
    "name": "Доллары",
    "default": false,
    "rate": 0.015584,
    "in_use": true
```


Рисунок 46. Фрагмент структуры данных справочника валют

2.9.3. Словарь сферы промышленности

Словарь содержит иерархическую структуру индустрии современного производства и входящие в них направления. Пример записи в такой структуре представлен на рисунке 47.


```
{
   "id":1,
   "name":"Перевозки, логистика, склад, ВЭД",
   "industries":[
      {
         "id":2,
         "name":"Автомобильные перевозки",
         "parent_id":1
      },
      {
         "id":3,
         "name":"Морские, речные перевозки",
         "parent_id":1
      },
      {
         "id":4,
         "name":"Транспортно-логистические комплексы
         "parent_id":1
      },
      {
         "id":5,
         "name":"Авиаперевозки",
         "parent_id":1
      },
      {
```


Рисунок 47. Фрагмент структуры справочника сферы промышленности

2.9.4. Словарь специализаций

Словарь содержит иерархическую структуру различных направлений бизнеса и входящих в них специализаций. Пример записи в такой структуре представлен на рисунке 48.


```
"id":60,
"name":"Продажи",
"specializations":[
    {
        "id":61,
        "name":"Электротехническое оборудование, Светотехника",
        "laboring":false,
        "parent_id":60
    },
    {
        "id":62,
        "name":"Химическая продукция",
        "laboring":false,
        "parent_id":60
    },
    {
        "id":63,
        "name":"Сертификация",
        "laboring":false,
        "parent_id":60
    },
```


Рисунок 48. Фрагмент структуры справочника сферы промышленности

2.9.5. Инициализация словарей

Процесс инициализации словарей требует последовательного обращения к разным компонентам серверной части веб-приложения, что подробно показано в диаграмме последовательности, представленной на рисунке 49.

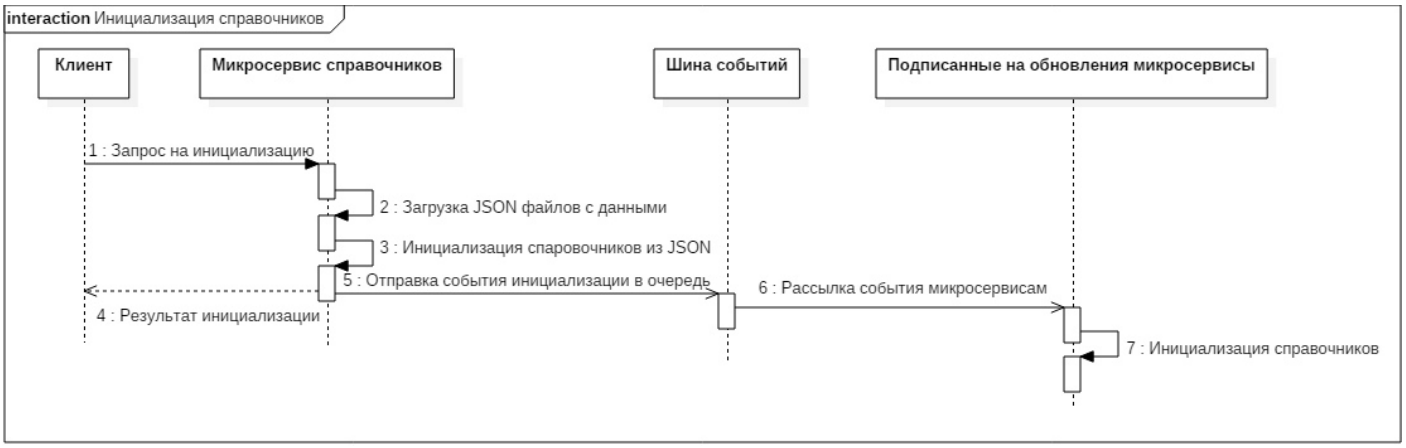

Рисунок 49. Инициализации справочников

2.10 Экспорт пользовательских данных

Веб-приложение имеет возможность экспортировать данные о резюме и вакансиях в файлы формата PDF. Для реализации данной функциональности используется сервис `jsreport`, который имеет возможность обрабатывать HTML-страницы и генерировать на их основе PDF-файлы [19].

Экспорт резюме или вакансии состоит из следующих этапов:

1. Клиент отправляет запрос на экспорт.
2. Подгружается шаблон формата CSHTML, отображающий структуру конечного PDF-файла.
3. Подгружается требуемая вакансия/резюме.
4. Шаблон cshtml заполняется данными о вакансии/резюме.
5. Полученный шаблон преобразуется в код в формате HTML.
6. Полученный шаблон отправляется сервису `jsreport` для экспорта.
7. Jsreport возвращает результат обработки в виде потока байтов.
8. В специальный словарь добавляется запись в формате «уникальный идентификатор файла – поток для скачивания».
9. Клиенту отправляется уникальный идентификатор для скачивания файла.
10. Клиент отправляет запрос на скачивание файла по идентификатору.
11. Из словаря по идентификатору подгружается поток для скачивания файла и отправляется клиенту.
12. Клиент скачивает итоговый файл.

Разбиение процесса на 2 запроса реализовано с целью повышения отзывчивости пользовательского интерфейса. Запрос на экспорт может быть асинхронным, что позволить пользовательскому интерфейсу реагировать на действия пользователя в течение всего процесса экспорта. Этап загрузки файла требует синхронного запроса, поэтому запрос отправляется уже после полной готовности PDF-файла к загрузке. Для понимания компонентов,

задействованных в процессе экспорта, была построена диаграмма компонентов, представленная на рисунке 50.

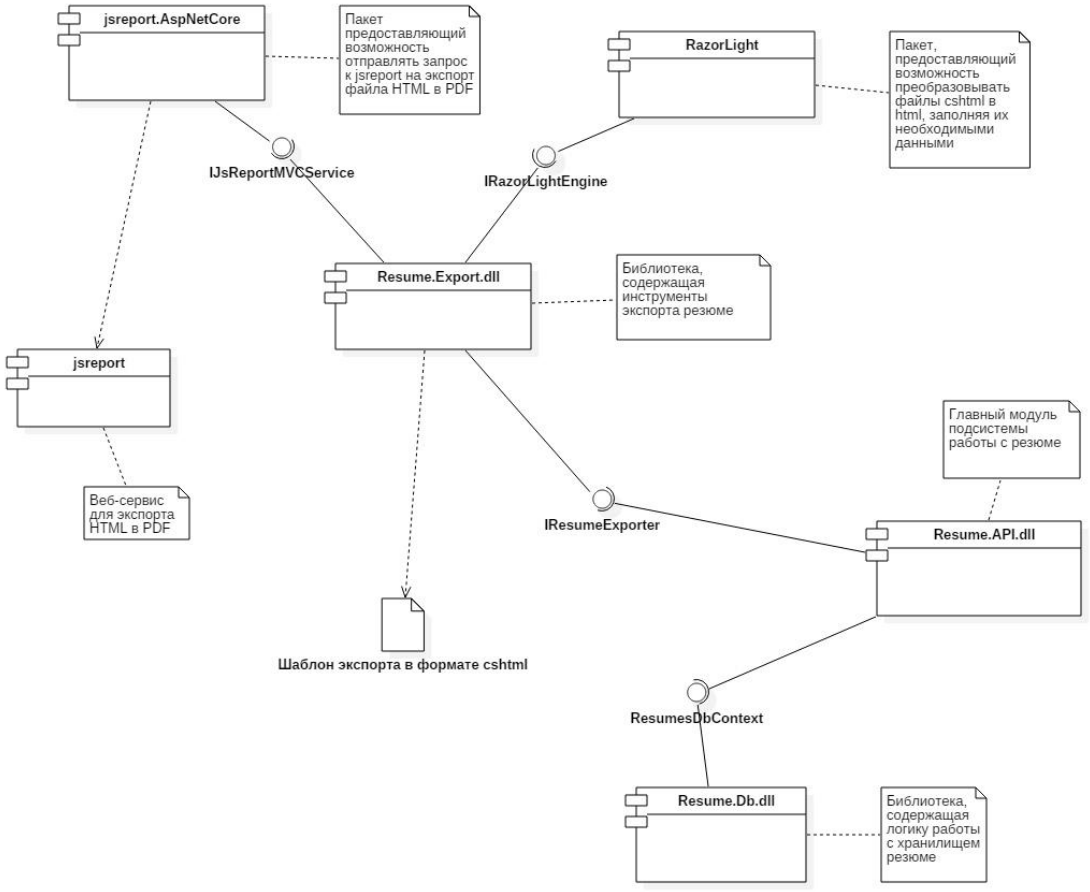

Рисунок 50. Процесс экспорта резюме

Учитывая большое количество задействованных для экспорта компонентов, будет полезным построить диаграмму, описывающую потоки данных между этими компонентами. Соответствующая DFD-диаграмма представлена на рисунке 51.

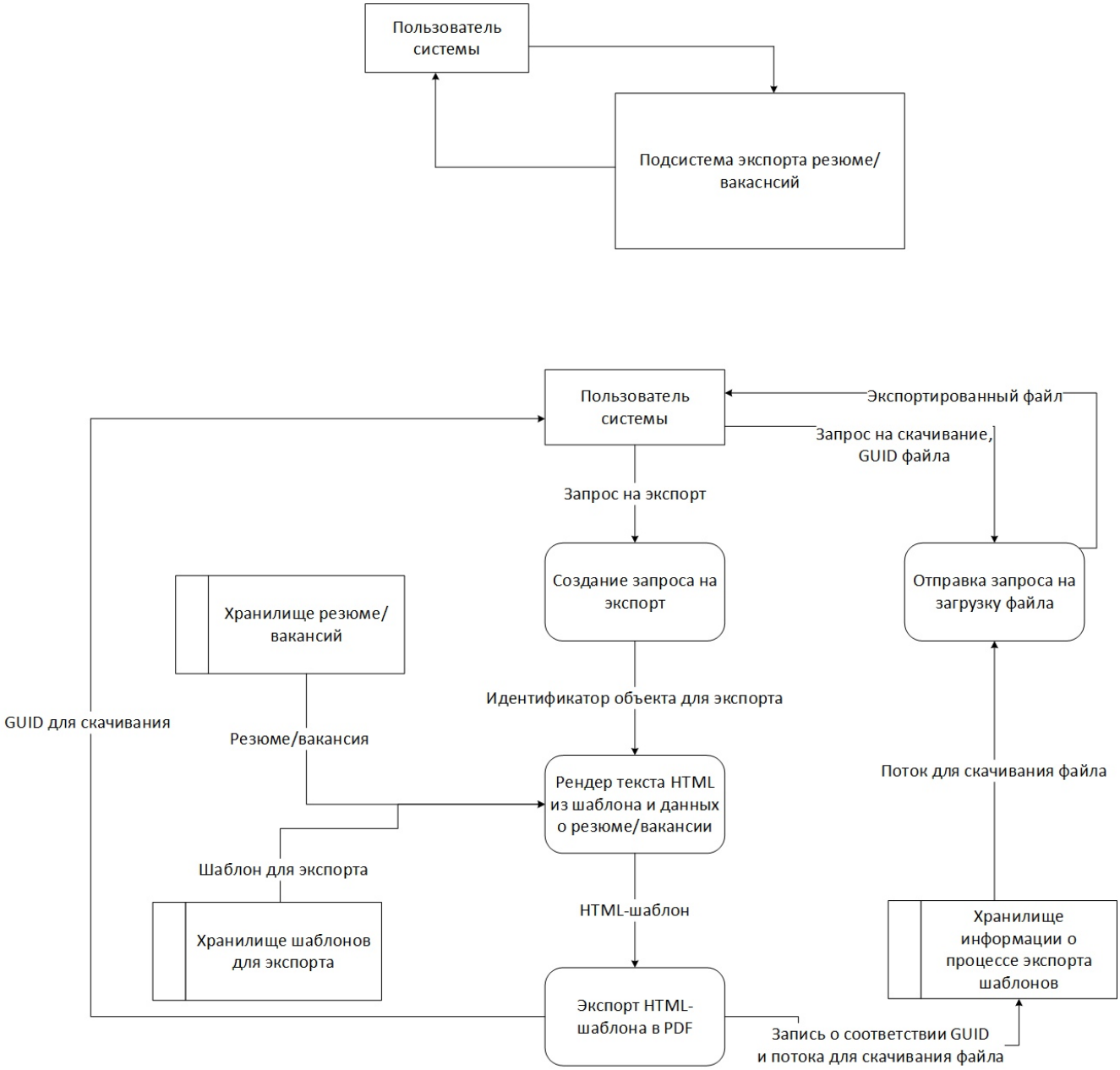

Рисунок 51. Потоки данных между компонентами веб-приложения при экспорте

Выводы по главе

В данном разделе, посвященному разработке веб-приложения, были распределены задачи между участниками разработки, детально описана архитектура серверной и клиентской части, в том числе описание процесса формирования и перехода к микросервисам. Каждый микросервис приложения был подробно описан с указанием используемых программных сущностей, дано пояснение по возможности экспорта данных.

Глава 3. Анализ результатов разработки веб-приложения

3.1. Метрики программного кода

Встроенные в Visual Studio средства анализа кода были применены к исходному коду серверной части, в результате чего были получены следующие данные:

- Содержит 172976 строк кода.
- Индекс удобства поддержки – 91,97 %.
- Сложность организации циклов – 2,86.
- Глубина наследования – 2,05.
- Взаимозависимость классов – 8,04.

Около 30 % кодовой базы составил служебный код, генерирующийся автоматически. Объем кода обусловлен сложностью предметной области, большим количеством сложно взаимодействующих программных сущностей, особенностями микросервисной архитектуры. Индекс удобства поддержки составил более 90%, что является крайне высоким показателем, означающим, что полученная система является гибкой и готовой к модификациям и расширению. Этого удалось достичь благодаря использованию паттернов проектирования и использования гибкой архитектуры.

3.2. Средства непрерывной интеграции

TeamCity является многофункциональным сервером непрерывной интеграции и непрерывной доставки модулей. Позволяет создавать конфигурации для автоматизированной сборки и публикации проекта на конечных серверах. Имеет встроенную поддержку триггеров, реагирующих на изменения в ветках git-репозиториев и запускающих процесс сборки и публикации в автоматическом режиме, на рисунке 52 показан процесс срабатывания триггера автоматической сборки.

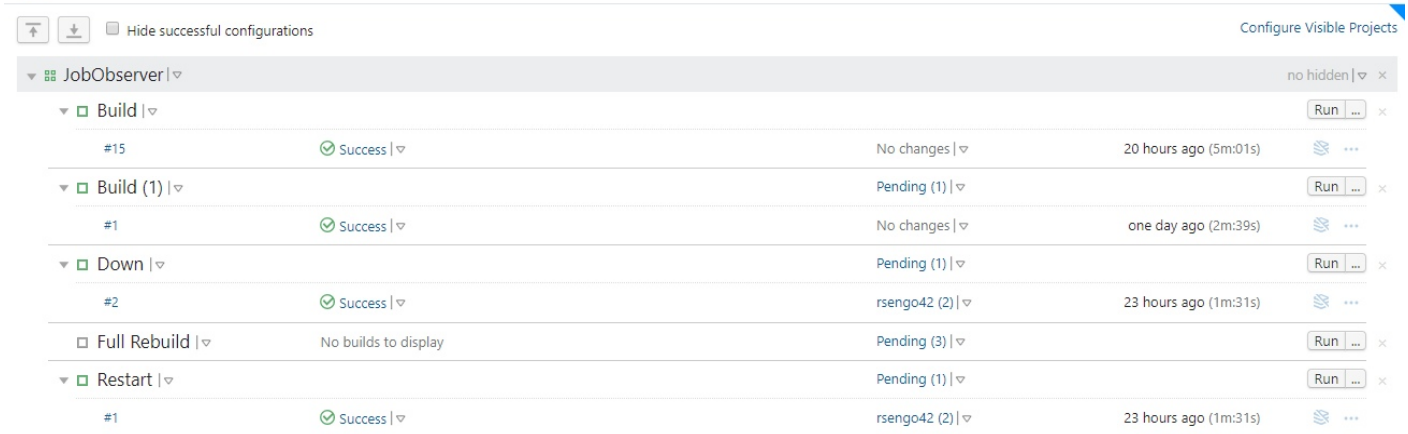

Рисунок 52. Веб-интерфейс TeamCity

Как видно из рисунка выше, при внесении изменений в исходный код была осуществлена автоматическая пересборка проекта.

3.3. Средства управления контейнерами

Portainer – это система управления контейнерами Docker, предоставляющая графический интерфейс для мониторинга состояния контейнеров, управления их жизненным циклом, привязками контейнеров к реальному хосту и мониторинга состояния хоста, на котором используется система Docker-контейнеров. На рисунке 53 показаны несколько запущенных контейнеров.

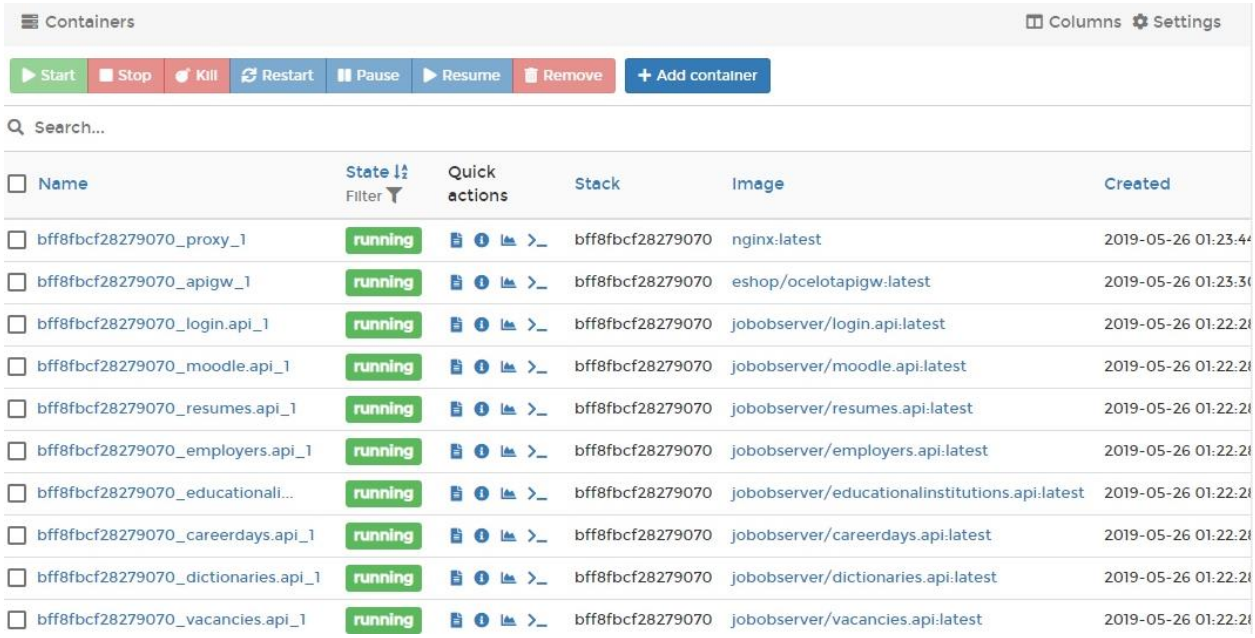

Рисунок 53. Веб-интерфейс Portainer

Как видно из рисунка выше, было успешно запущено 10 контейнеров, в каждом из которых работает отдельный микросервис.

3.4. Средства логирования

Seq – это система централизованного структурированного логирования, используемая вместе с Serilog [20]. Представляет собой REST-сервис, хранящий данные в встроенной NoSQL СУБД. Используется на локальных машинах и серверах с низкими вычислительными мощностями, так как менее требователен к ресурсам. На рисунке 54 можно ознакомиться с логами веб-приложения, созданными системой логирования Seq.

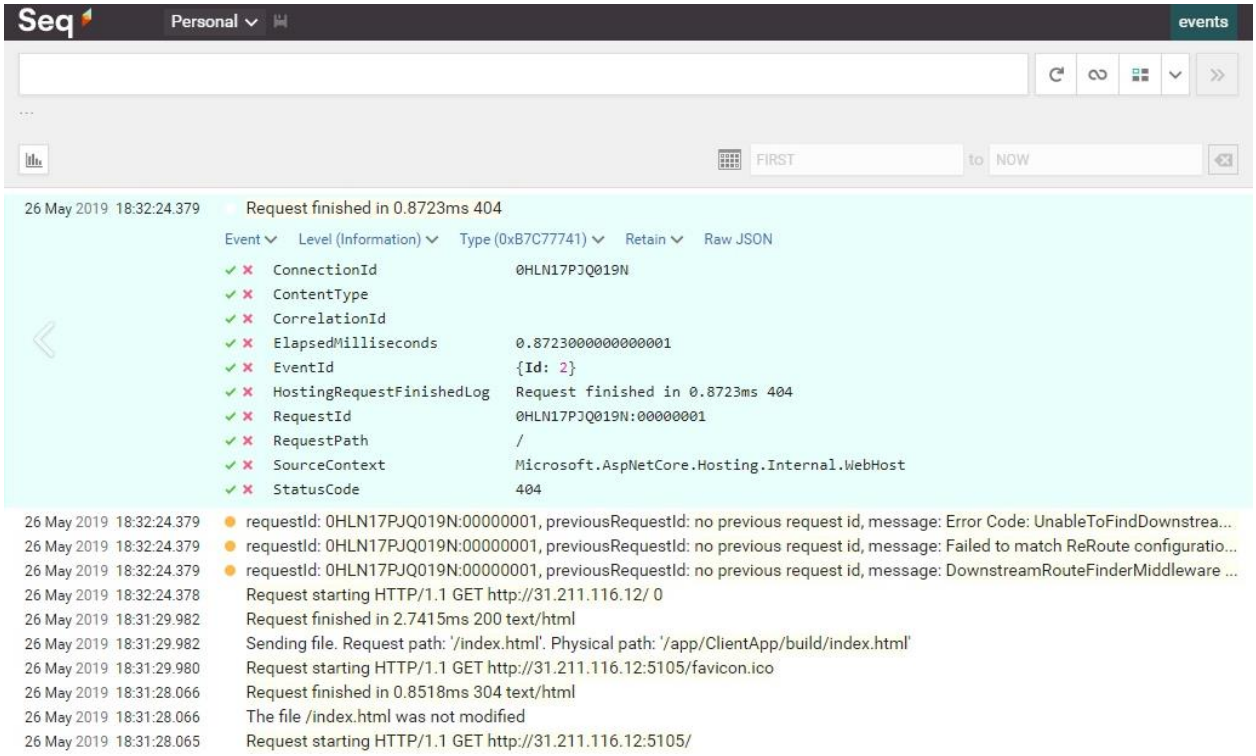

Рисунок 54. Журнал системы логирования Seq

3.5. Средства управления брокером сообщений

Использующийся в приложении брокер сообщений RabbitMQ имеет реализацию веб-клиента, позволяющего в реальном времени увидеть запущенные очереди, сообщения в очередях, их статус и другую информацию. На рисунке 55 в веб-клиенте RabbitMQ показаны несколько запущенных очередей.

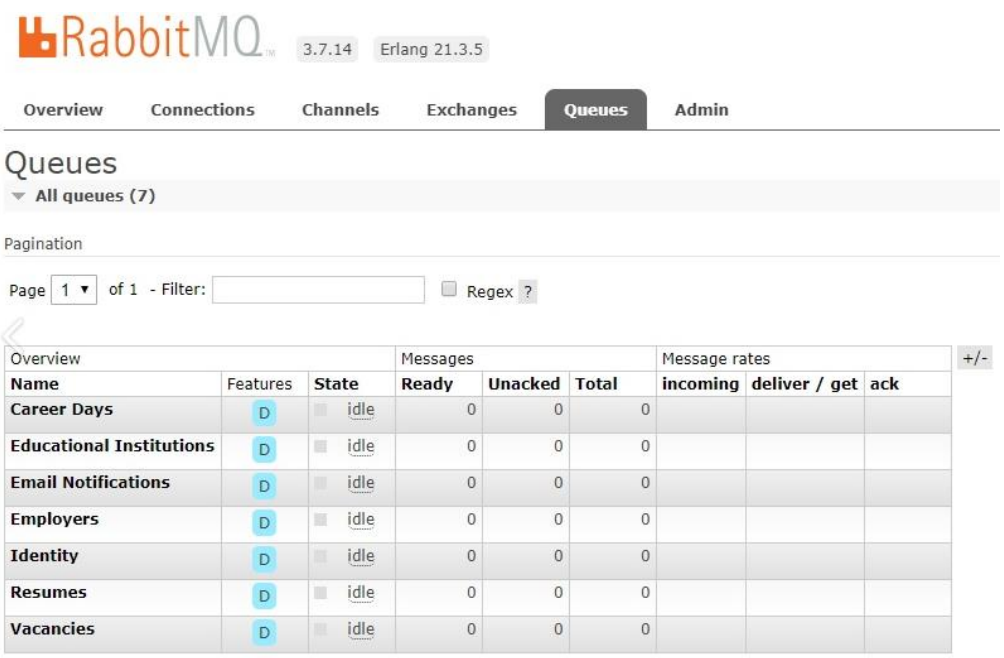

Рисунок 55. Веб-интерфейс RabbitMQ

Как видно из рисунка выше, для каждого из 7 микросервисов запущена отдельная очередь сообщений.

3.6. Экспорт пользовательских данных в формат PDF

Одной из возможностей приложения является экспорт пользовательских файлов в PDF-формат, пример экспорта вакансии приведен на рисунке 56.

Ведущий разработчик

Отдел разработки ТПУ

Город	Томск
Улица	Советская
Номер здания	84/3
Специализация	Программирование; Интернет-технологии
Дополнительная информация	
Категории водительских прав	
Наличие транспорта	Нет
Владение языками	Английский: intermediate
Тип занятости	Полная занятость
Заработная плата	100000-150000
Валюта	руб
Расписание	Полный день
Навыки	Asp; .Net; Js; React
Сфера деятельности	IT

Рисунок 56. Экспортированная в формат PDF вакансия

3.7. Компоненты клиентского приложения

3.7.1. Компонент регистрации нового пользователя

Для регистрации новых пользователей был создан отдельный компонент, продемонстрированный на рисунке 57. Компонент позволяет выбрать пользователю роль, указать данные для авторизации и контактную информацию.

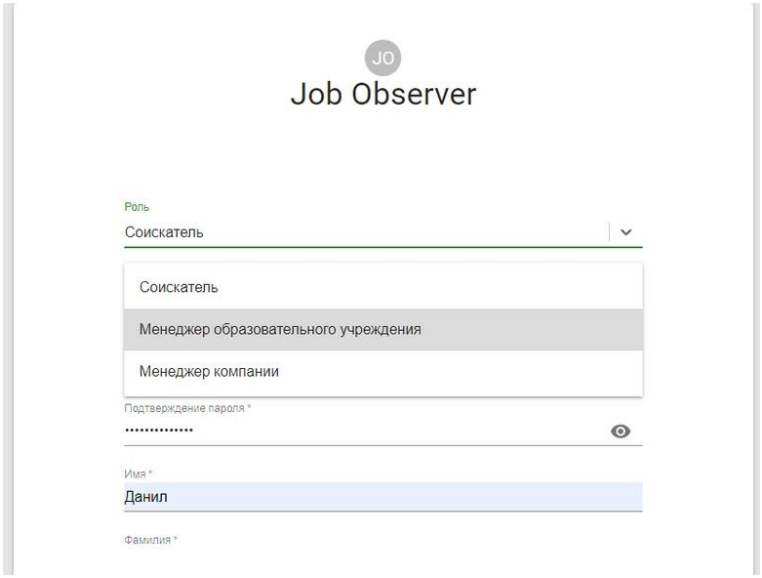

Рисунок 57. Веб-форма создания новой учетной записи

В веб-форме звездочкой помечены обязательные для заполнения поля, выбор роли осуществляется с помощью `SelectBox`.

3.7.2. Компонент авторизации

Для авторизации зарегистрированных пользователей был создан отдельный компонент, продемонстрированный на рисунке 58. Компонент представляет собой форму для ввода электронной почты и пароля.

Рисунок 58. Веб-форма авторизации пользователя

3.7.3. Компонент личного кабинета соискателя

Для организации личного кабинета соискателя был создан отдельный компонент, продемонстрированный на рисунке 59. В верхней навигационной панели есть доступ к просмотру профилю (активная вкладка на рисунке), резюме соискателя и просмотру вакансий.

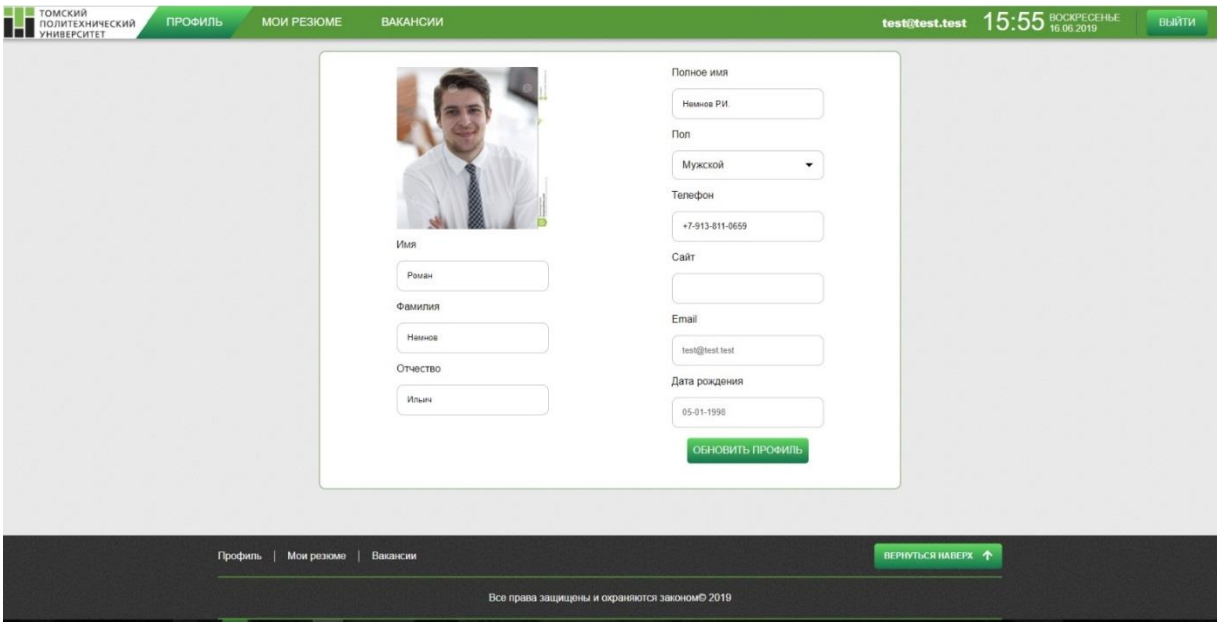

Рисунок 59. Вкладка личного кабинета соискателя

3.7.4. Компонент просмотра резюме соискателя

Для организации возможности просмотра резюме соискателя был создан отдельный компонент, продемонстрированный на рисунке 60. На широких экранах (персональные компьютеры и ноутбуки) вакансии показываются в 2 столбца. На экранах мобильных устройств разметка страницы существенно изменяется, что видно на рисунке 61.

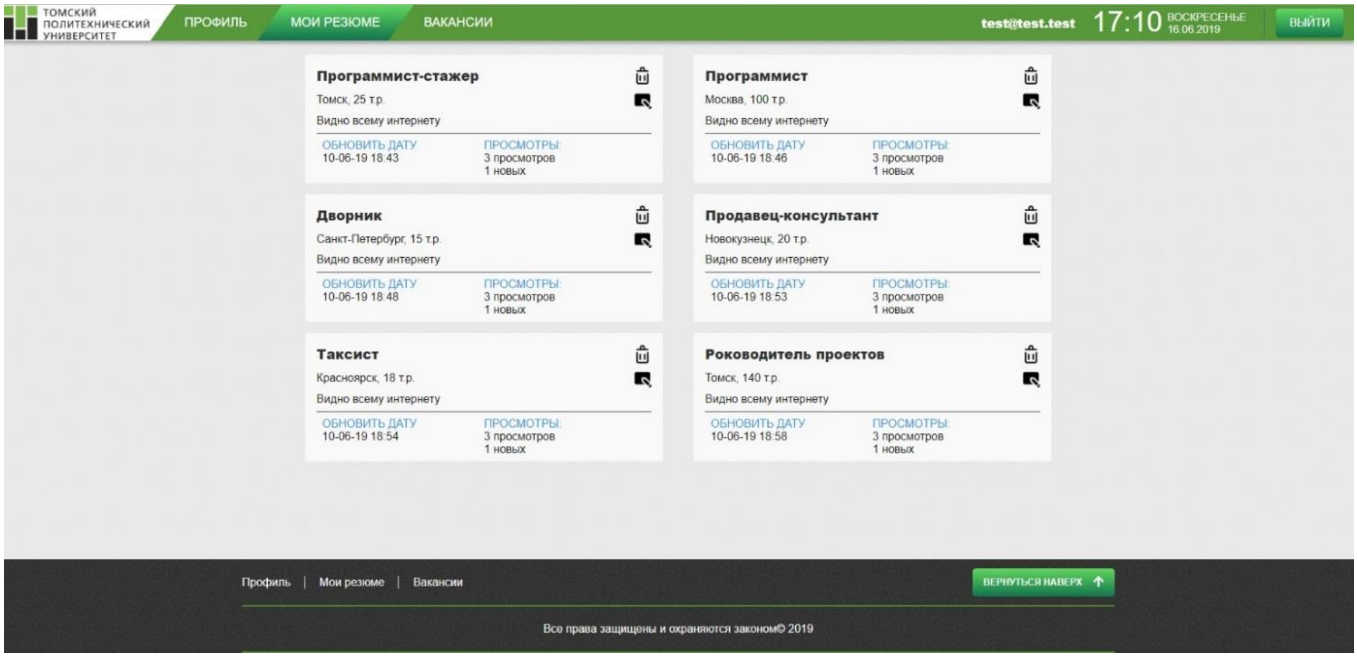

Рисунок 60. Вкладка просмотра соискателем собственных резюме

Как видно, навигационное меню дублируется в нижней части вкладки для повышения удобства использования веб-приложения. В правой верхней части приложения расположена кнопка, позволяющая прекратить сеанс работы в качестве авторизованного пользователя и вернуться на страницу авторизации.

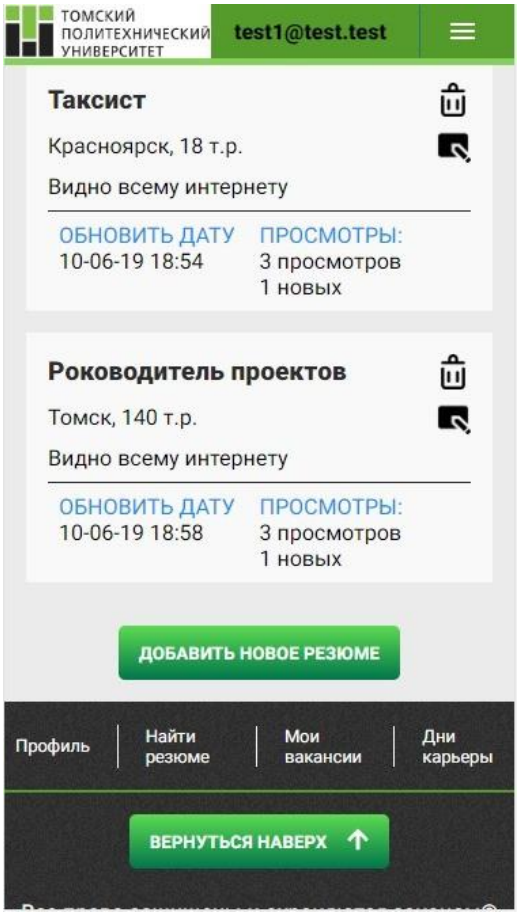

Рисунок 61. Вкладка просмотра резюме, мобильная версия

3.7.5. Компонент просмотра вакансий

Из личного кабинета соискатель также может перейти на вкладку поиска и отображения вакансий, что показано на рисунке 62.

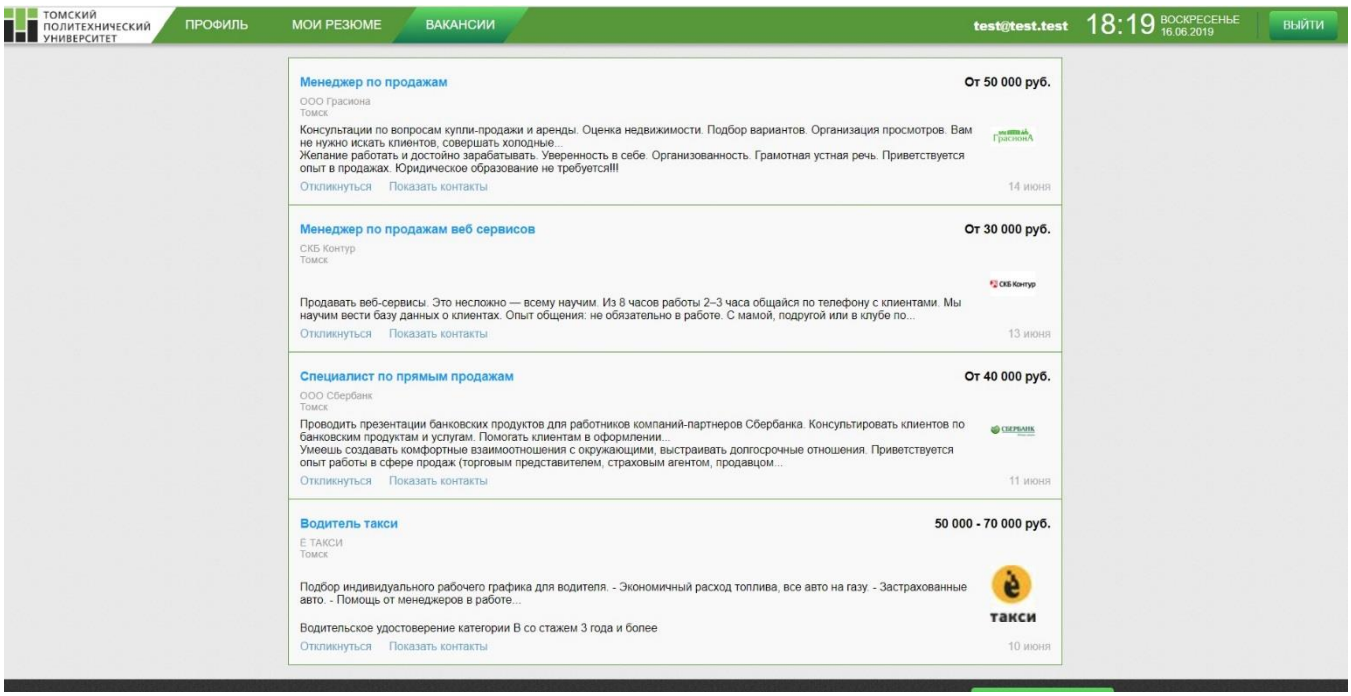

Рисунок 62. Список вакансий для соискателя

3.7.6. Компонент добавления «Дней карьеры»

Для реализации возможности добавления «Дней карьеры» менеджером учебного заведения был создан отдельный компонент, показанный на рисунке 63. В правой нижней части формы есть поле ввода представленных на мероприятии специализаций.

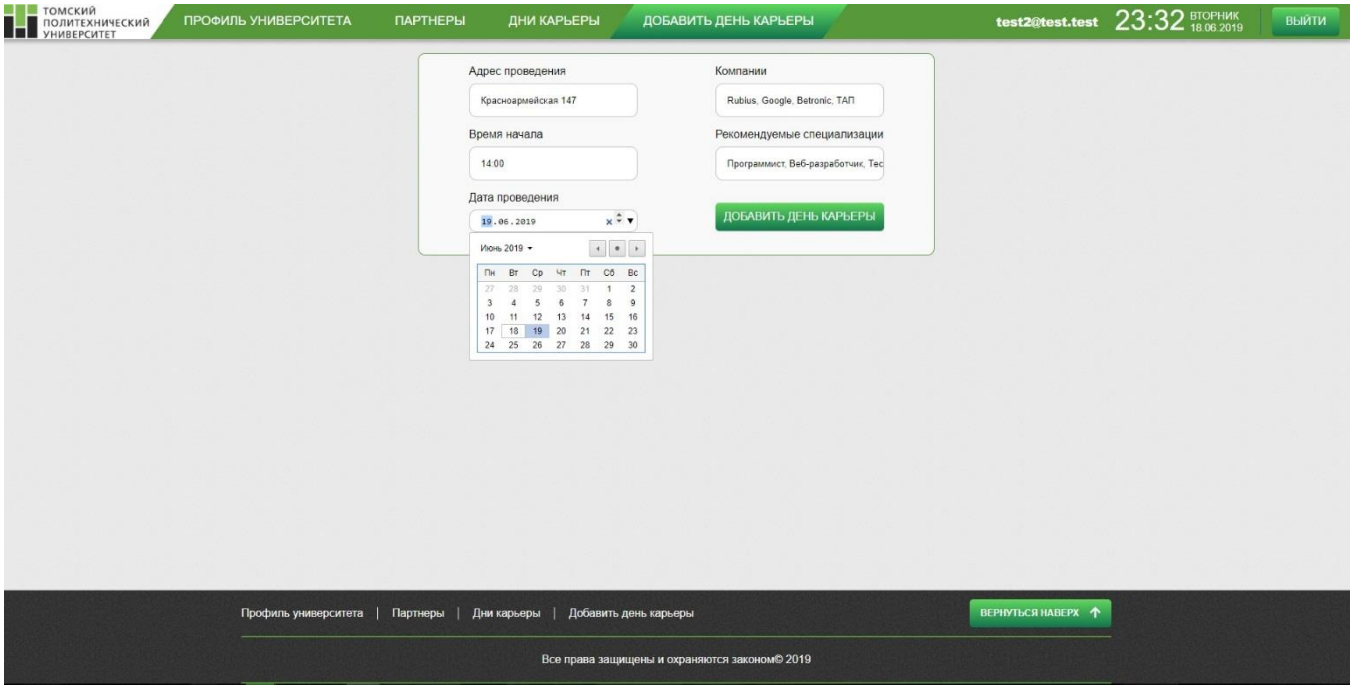

Рисунок 63. Веб-форма добавления нового «Дня карьеры»

3.7.7. Компонент добавления вакансий и просмотра вакансий компании

Компонент, позволяющий менеджеру компании просматривать все вакансии и добавлять новую, показан на рисунке 64. Визуальное представление компонента представляет 2 блока, в левый выводится список всех вакансий компании, в правом – форма добавления новой вакансии.

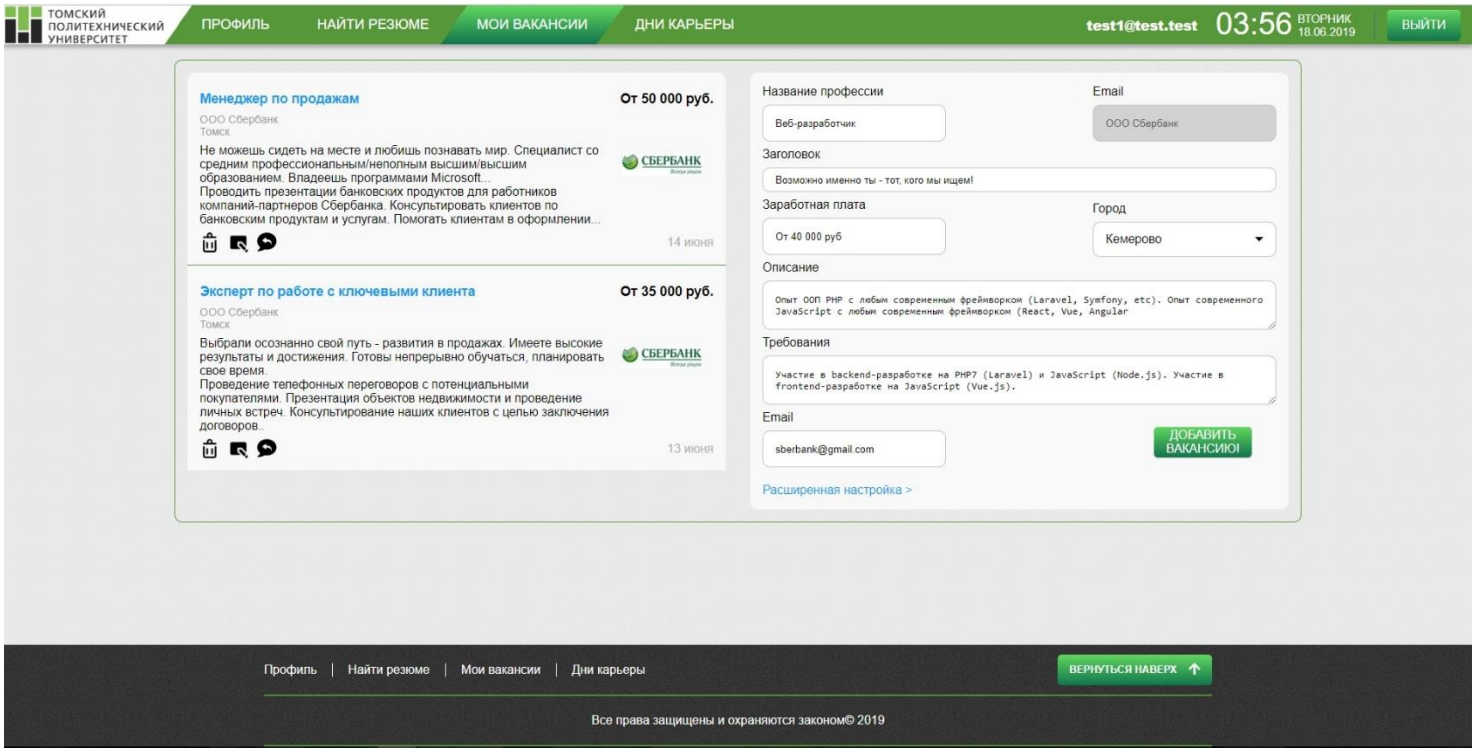

Рисунок 64. Веб-форма для просмотра и добавления вакансий компании

На рисунке 65 продемонстрирована возможность использования личного кабинете менеджера компании с мобильного устройства, а на рисунке 66 – вкладка приложения, появляющаяся после перехода в раздел «мои вакансии».

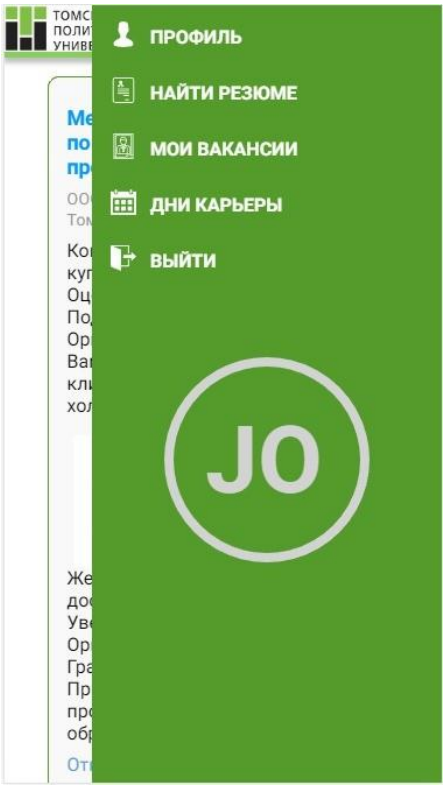

Рисунок 65. Мобильная версия личного кабинета менеджера компании

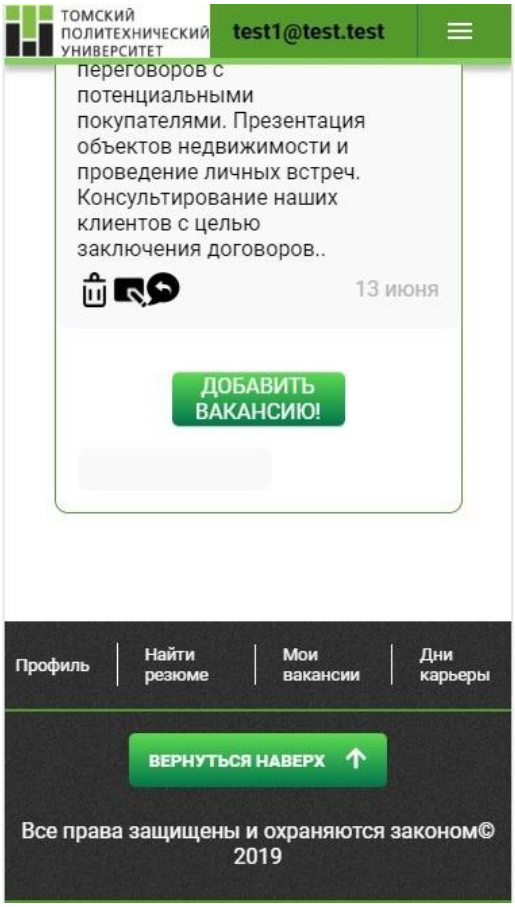

Рисунок 66. Веб-форма просмотра и добавления вакансии с помощью мобильного устройства

3.7.8. Компонент просмотра откликов на вакансию

Для реализации возможности менеджеру компании просматривать отклики на открытые вакансии, был разработан отдельный компонент, представленный на рисунке 67.

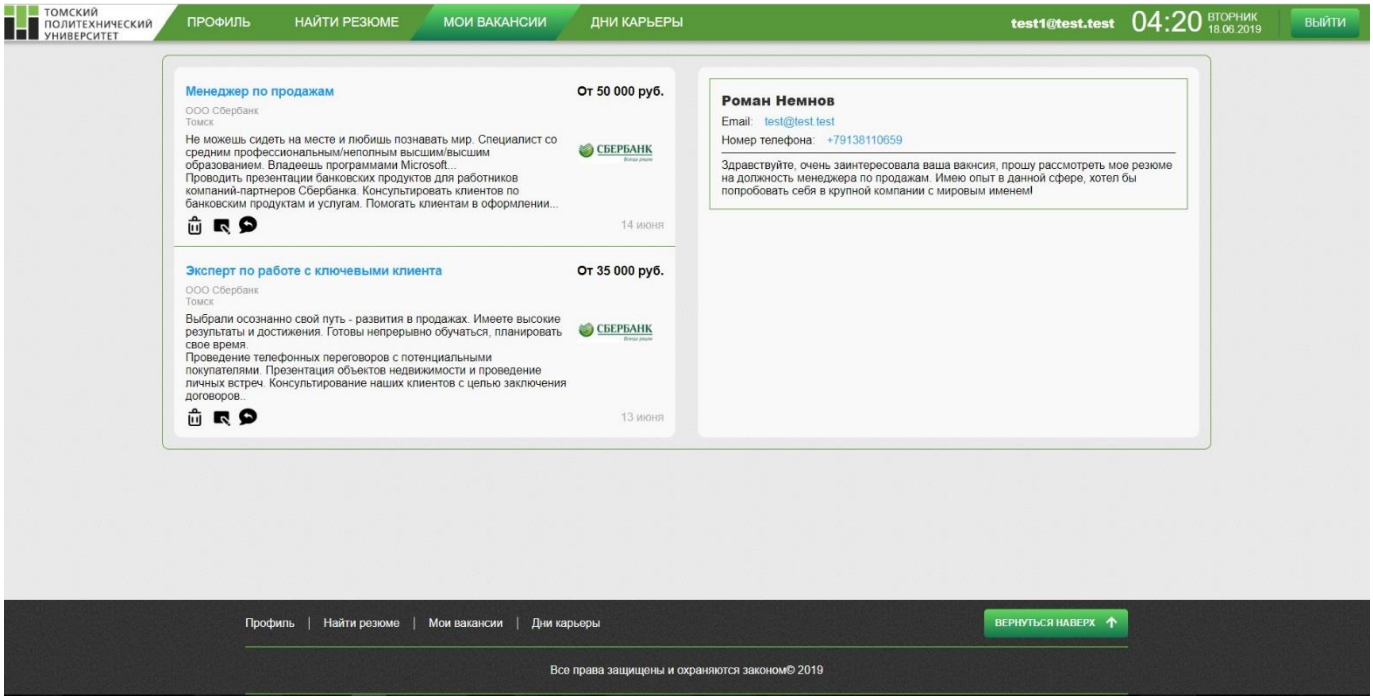

Рисунок 67. Веб-компонент для просмотра откликов на вакансию

3.8. Система оповещения

Пользователи системы могут выбрать приоритетный способ получения уведомлений – через электронную почту или через SMS, при этом также доступен вариант получения уведомления сразу через все доступные средства связи. На рисунках 68 и 69 последовательно изображены результаты работы микромодуля рассылки уведомлений – после добавления нового «Дня карьеры» тестовому соискателю пришло уведомление на указанный электронный ящик и SMS.

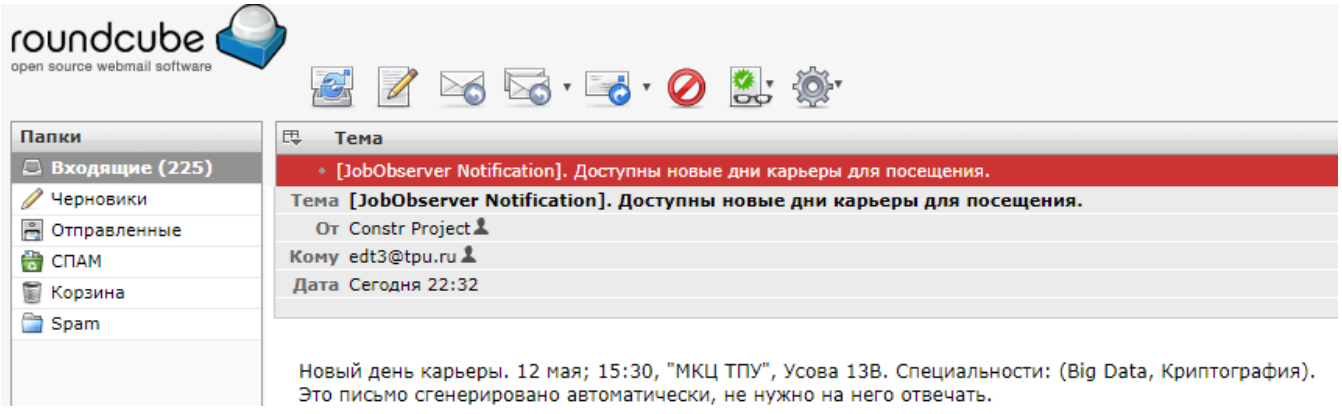

Рисунок 68. Уведомление по электронной почте

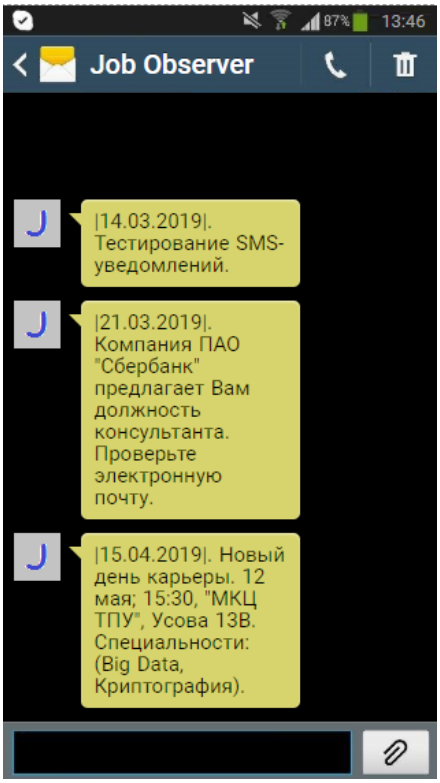

Рисунок 69. Уведомление по SMS

3.9. Создание, управление и прохождение автоматизированных тестов

На рисунке 70 показан интерфейс, который видит соискатель при прямохождении автоматизированного теста на платформе Moodle. В первой части находится навигационное меню, в левой части – краткая информация о текущем прогрессе. Основной контент расположен в центральной части.

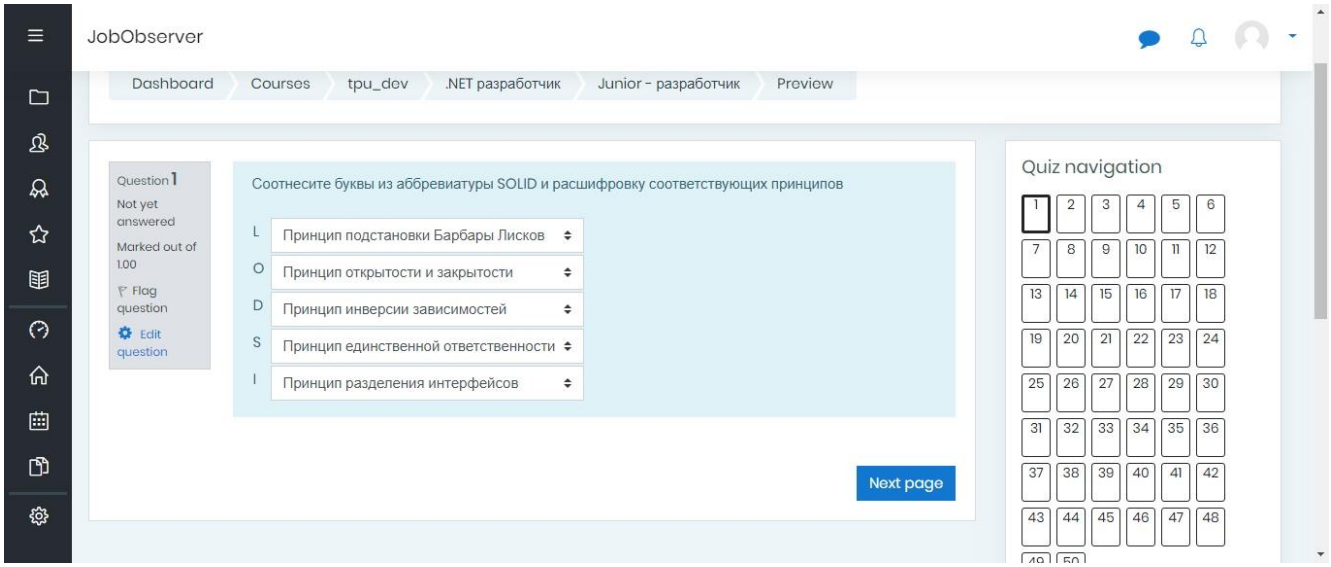

Рисунок 70. Просмотр вопроса в тесте на платформы Moodle

На рисунке 71 показана форма, появляющаяся после прохождения всех вопросов. На ней отражена статистика прохождения, включая количество набранных первичных и вторичных балов, а также итоговой результат за все попытки.

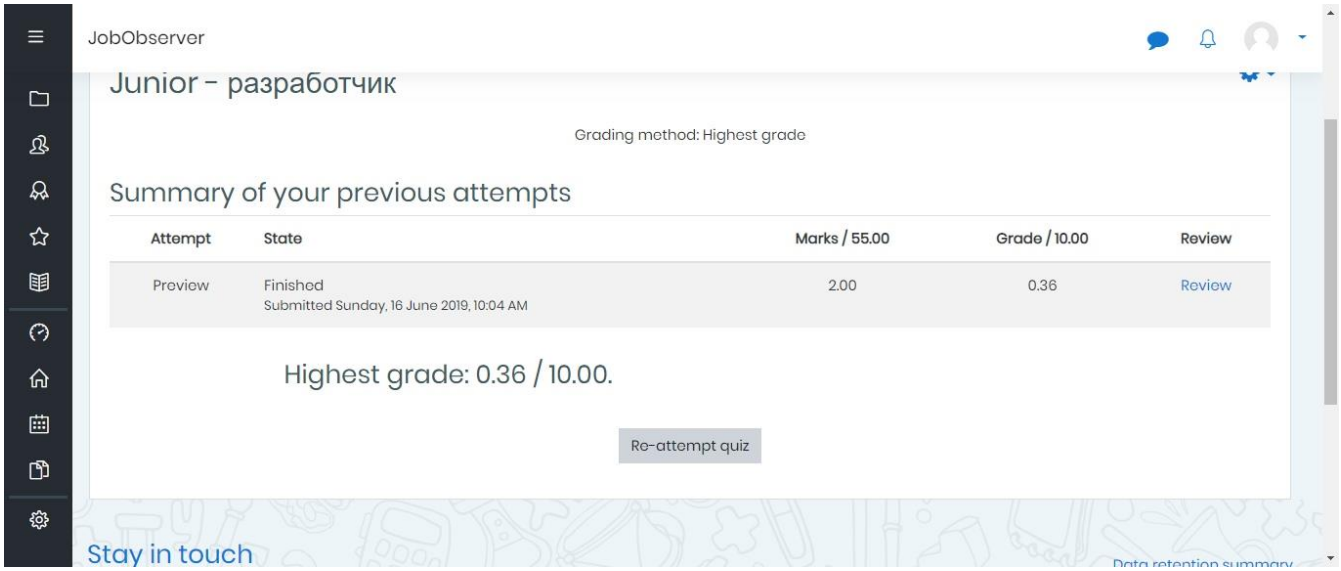

Рисунок 71. Просмотр результатов выполнения теста соискателями

Для технического персонала компании предусмотрена возможность просмотра системной информации, которая показана на рисунке 72.

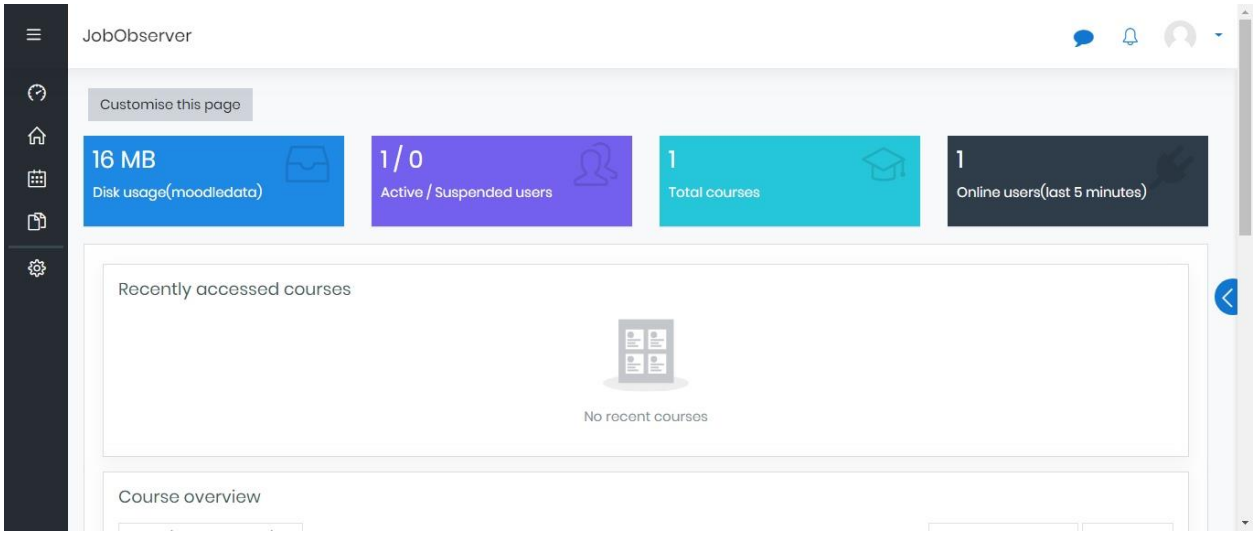

Рисунок 72. Панель управления курсами

Менеджер компании является мастером курса, то есть ему доступна возможность просмотра результатов прохождения теста всеми пользователями, что показано на рисунке 73.

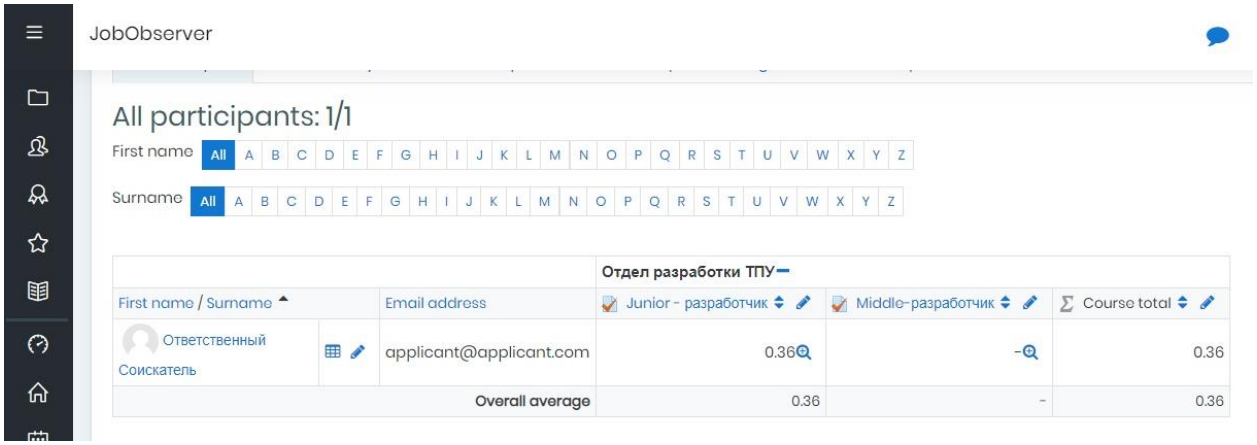

Рисунок 73. Просмотр оценок всех соискателей, прошедших тест на вакансию

3.10. Выводы по главе

В данном разделе, посвященном проведению анализа результатов разработки веб-приложения, были продемонстрированы основные разработанные компоненты клиентского интерфейса, даны пояснения по визуальному оформлению компонентов, раскрыты возможности по созданию, управлению и прохождению автоматизированных тестов. Также были приведены метрики программного кода, демонстрирующие количество затраченных усилий для разработки веб-приложения и результаты использования различных программных решений, выбранных в первой главе работы.

Глава 4. Оценка коммерческого потенциала и перспективности проведения научных исследований с позиции ресурсоэффективности и ресурсосбережения

4.1. Оценка перспективности проведения исследований

4.1.1. Потенциальные потребители результатов исследования

Проблем, вызываемых безработицей за последние 10 лет, стало меньше ввиду снижения уровня безработицы на 1,1 %. Однако ведущие аналитики считают, что в ближайшие 5 лет её уровень будет стабильно выше 4,5 % [21]. При этом следует принять во внимание тот факт, что существенная часть официально трудоустроенных граждан вынуждена работать не по той специальности, которой они обучались в учебных заведениях. Особенно остро эта проблема касается молодых специалистов, недавно окончивших обучение, или студентов старших курсов, так как эта категория лиц в большинстве случаев не имеет опыта работы, что является критичным для подавляющей части работодателей.

Работодатели же в свою очередь испытывают затруднения с вакансиями, требующими высококвалифицированного труда, особенно в области интернет-технологий, так как специфичность этой трудовой области заключатся в быстрой сменяемости применяемых программных решений. Программы обучения в учебных заведениях не могут подстраиваться под нужды работодателей быстро и своевременно, в результате чего студенты изучают устаревшие технологии и могут быть не осведомлены о современных требованиях рынка.

Программная система, автоматизирующая процессы интернет-рекрутинга, могла бы позволить соискателям и работодателям более оперативно находить друг друга и устанавливать деловой контакт. Система может взять на себя ответственность за создание электронных версий резюме и вакансий, получая пользовательские данные из форм веб-приложения. Такой подход стандартизует структуру этих документов и позволяет

пользователям быстрее получить ключевую информацию, снизить затрачиваемое время на создание вакансий или резюме, а также уменьшит риск допущения ошибки из-за невнимательности, так как приложение может оповестить пользователя о том, что какое-либо ключевое поле формируемого документа не заполнено. Система также может быть применена для улучшения системы проведения дней карьеры в учебных заведениях за счет внедрения современных способов уведомления, что способствует актуализации знаний учащихся о современных требованиях работодателей.

4.1.2. Анализ конкурентных технических решений

Процесс трудоустройства затрагивает большинство жителей всех стран мира, поэтому на момент написания работы существует множество уже реализованных и функционирующих интернет-приложений, позволяющих пользователям создавать, редактировать и фильтровать резюме и вакансии. В виду того, что разрабатываемую систему предлагается внедрить сначала только в Томский Политехнический Университет, был проведен опрос среди учащихся с целью выяснить наиболее известные для них сервисы интернет-рекрутинга. В результате опроса было выяснено, что больше всего студенты ТПУ осведомлены о следующих сервисах: Headhunter и веб-сайт Зарплата.ру.

Headhunter – крупнейший Российский сервис интернет-рекрутинга, база которого содержит более 30 млн. резюме.

Зарплата.ру – ведущий рекрутинговый сайт в Сибири и на Урале, база которого содержит более 6,5 млн. резюме.

4.1.3. Оценка конкурентоспособности проекта

Чтобы оценить конкурентоспособность проекта, необходимо составить оценочную карту, для этого выделяются наиболее важные аспекты продукта и сравниваются с аналогами на рынке.

Оценки для каждого параметра выставляются по 10-балльной шкале, при этом оценивается влияние каждой характеристики на продукт в целом. Такой метод позволяет получить оценку продукта в целом, а также сравнить различные продукты по отдельным критериям. Таким образом, можно выяснить сильные и слабые стороны разрабатываемого продукта, что отражено в таблице 8.

Таблица 8. Оценочная карта для сравнения конкурентных технических решений (разработок)

№ п/п	Продукт	Факторы конкурентоспособности							Итог
		Стоимость возможности использования полного функционала	Скорость обработки запросов	Интеграция со сторонними сервисами (Moodle)	Дополнительные возможности для молодых специалистов и учащихся	Возможность горизонтального масштабирования	Производительность клиентского интерфейса	Поддержка мобильных устройств	
1	Headhunter	5	9	8	6	10	10	10	8,275
2	Зарплата.ру	6	8	3	4	8	8	8	6,6
3	JobObserver (разрабатываемый продукт)	10	8	8	9	10	9	6	8,350
	b_i	6	7	4	6	3	6	8	40
	w_i	0,15	0,175	0,10	0,15	0,075	0,15	0,2	1,0

Из таблицы видно, что наиболее значимым критерием является поддержка мобильных устройств, это связано с тем, только на смартфоны приходится более 50 % всего создаваемого трафика, при этом конкуренты здесь оценены выше из-за наличия выделенного мобильного приложения (Android, iOS у Headhunter, только Android у Зарплаты.ру).

Положительной особенностью архитектуры разрабатываемого веб-приложения является возможность распределения сетевой нагрузки путем добавления новых вычислительных машин – горизонтальное масштабирование. Система также получает высокие оценки в категории интеграции со сторонними сервисами, что позволит работодателям создавать собственные курсы и тесты на платформе Moodle, и обеспечения дополнительных возможностей для учащихся и молодых специалистов из-за push-уведомлений о «Днях карьеры». С незначительным опережением разрабатываемый продукт превосходит главного конкурента – Headhunter. Этот отрыв может быть дополнительно увеличен при дополнительном финансировании – обновление парка серверных машин позволит повысить скорость обработки запросов – этот критерий обладает высоким удельным весом, также будет возможным создания выделенного мобильного приложения, что позволит клиентскому интерфейсу отображаться ещё плавнее из-за запуска процессов системы непосредственно в операционной системе.

Сайт Зарплата.ру значительно отстает от разрабатываемого продукта и Headhunter из-за ориентации непосредственно на работодателей и соискателей, игнорируя при этом возможности по обеспечению дополнительного функционала для молодых специалистов.

4.1.4. SWOT-анализ

Чтобы выявить достоинства и недостатки продукта с целью дальнейшего улучшения качества необходимо провести SWOT анализ.

SWOT-анализ (показан в таблице 9) –один из методов стратегического планирования, является простым и качественным инструментом оценивания конкурентоспособности. Суть метода заключается в выявлении ключевых факторов внутренней и внешней среды [22].

В названии заложены первые буквы четырех категорий, из которых складываются факторы – Strength (сильные стороны продукта), Weaknesses (слабые стороны), Opportunities (возможности) и

Threats (угрозы). Дополнительно все факторы следует проанализировать в парах, например слабые стороны и угрозы

Таблица 9. SWOT-анализ разрабатываемого продукта

		Внутренние факторы	
Внешние факторы		**Сильные стороны** 1. Отказоустойчивость системы из-за микросервисной архитектуры. 2. Балансировка нагрузки для оптимизации работы сервера. 3. Модульность клиентского интерфейса. 4. Технология надежного хранения и логирования всех событий системы. 5. Дополнительные возможности для молодых специалистов и учащихся.	**Слабые стороны** 1. Базы крупных конкурентов уже содержат миллионы резюме. 2. Использование серверов начального уровня производительности. 3. Отсутствие отдела разработки мобильных приложений для ОС Android\iOS. 4. Медленный старт из-за ограниченности рекламного бюджета.
	Возможности 1. Рост производительности вычислительной техники. 2. Попадание продукта на первые строчки выдачи поисковых систем. 3. Создание релевантных систем, интегрируемых с продуктом.	Сильные стороны и возможности показывают, что исходной производительности сервера хватит с запасом на начальный этап (до 50000 первых клиентов), модульность системы позволит предлагать пользователям самый современный функционал.	Исходя из слабых сторон и возможностей, можно предположить экспоненциальный рост числа пользователей, наблюдается сильная зависимость развития от первоначальной активности, приложение может завязнуть на начальном этапе.
	Угрозы 1. Подъем пользовательского спроса на приложения, запускаемых в ОС, а не в браузере. 2. Массированные атаки, направленные на отказ в обслуживании. 3. Появление более совершенного конкурентного продукта.	Полный отказ системы почти невозможен, но намеренный вывод из строя критичных частей может вызвать серьезное недовольство пользователей, также тенденции к использованию выделенных приложений могут снизить их активность.	Выделенные приложения конкурентов могут неожиданно обрести дополнительную популярность. В будущем существует момент, когда конкуренты обратят внимание на продукт из-за сформировавшейся базы пользователей и могут намеренно выводить части системы из строя.

Самой слабой стороной продукта является отсутствие выделенного приложения на мобильных ОС, программная система запускается в браузере с помощью веб-клиента. Для создания таких приложений в проект необходимо привлечь несколько разработчиков, однако это потребует пересмотра бюджета проекта.

Расчетный рост числа пользователей приложения является строго нелинейным, так как с увеличением их числа поисковые системы с большей вероятностью будут выдавать продукт на первых строчках результатов обработки запроса, что еще сильнее ускорит рост. Это порождает сразу 2 проблемы – этап формирования первоначальной базы пользователей может занять слишком много времени, учитывая, что в базах конкурентов десятки миллионов резюме находятся уже сейчас, при неудачном стечении обстоятельств, продукт никогда не будет способен догнать конкурентов по уровню пользовательской активности. Если популярность, а вместе с ней и сетевая нагрузка, продукта в некоторый момент времени стремительно возрастет, потребуется в срочном порядке улучшить серверный парк. На программном уровне архитектура системы позволяет это легко реализовать, но на настройку новых серверов все равно требуется значительное количество времени, которое невозможно сократить. Потребуется внимательно следить за средним количеством ежедневных активных пользователей и предсказывать всплески активности, чтобы избежать репутационных потерь из-за медленной работы серверной части.

Система ориентирована преимущественно на использование молодыми специалистами и предлагает им дополнительные возможности, что является отличительной чертой продукта. Однако всегда остается вероятность появления нового конкурентного приложения, предлагающего схожие функциональные возможности. Если их бюджет будет значительно превышать бюджет продукта, часть пользователей может уйти к конкурентам, а дальнейшие темпы прироста аудитории будут существенно снижены.

Наконец, даже выделенной группе разработчиков понадобится несколько месяцев для разработки отдельного приложения для мобильных ОС. Если же пользовательские предпочтения неожиданно сместятся в сторону таких приложений, то на период их разработки интерес к продукту может значительно снизится.

Стоит учесть, что вероятность возникновения таких событий остается низкой, и никаких серьёзных преград для реализации продукта в результате проведенного SWAT-анализа обнаружено не было.

4.2. Планирование научно-исследовательских работ.

4.2.1. Структура работ в рамках научного исследования.

В нижеприведенной таблице под словом «Все» в графе исполнители работ следует понимает следующих участников работы: Немнов Роман Ильич, Руденцов Данил Павлович и Трусов Евгений Дмитриевич.

Таблица 10. Структура работ в рамках проектирования и разработки веб-приложения

№ работы	Наименование работы	Исполнители работы
1	Выбор научного руководителя бакалаврской работы	Все
2	Составление и утверждение темы бакалаврской работы	Все, Соколова Вероника Валерьевна
3	Составление календарного плана-графика выполнения бакалаврской работы	Трусов Евгений Дмитриевич
4	Подбор и изучение литературы по теме бакалаврской работы	Руденцов Данил Павлович, Трусов Евгений Дмитриевич
5	Анализ предметной области	Все
6	Выбор программных решений для разработки клиентской части приложения	Немнов Роман Ильич
7	Выбор программных решений для разработки серверной части приложения	Руденцов Данил Павлович
8	Проработка архитектуры приложения	Руденцов Данил Павлович

№ работы	Наименование работы	Исполнители работы
9	Реализация основных микросервисов приложения	Руденцов Данил Павлович
10	Реализация дополнительных микросервисов	Трусов Евгений Дмитриевич
11	Реализация словарей данных	Руденцов Данил Павлович, Трусов Евгений Дмитриевич
12	Реализация шины событий и брокера сообщений	Трусов Евгений Дмитриевич
13	Интегрирование системы Moodle	Руденцов Данил Павлович
14	Конфигурация Docker-файлов	Руденцов Данил Павлович
16	Выработка концептуального дизайна приложения на различных устройствах	Немнов Роман Ильич
17	Реализация компонентов интерфейса пользователя	Немнов Роман Ильич
18	Настройка внутренних переходов	Немнов Роман Ильич
19	Подключение библиотеки управления состоянием к серверной части	Немнов Роман Ильич, Руденцов Данил Павлович
20	Адаптация приложения под мобильные устройства	Немнов Роман Ильич, Трусов Евгений Дмитриевич
21	Настройка политики кросс-доменных запросов	Руденцов Данил Павлович
22	Тестирование серверной части	Руденцов Данил Павлович
23	Тестирование клиентской части	Немнов Роман Ильич, Трусов Евгений Дмитриевич
24	Пользовательское тестирование приложения	Все
25	Исправление ошибок приложения	Все
26	Написание главы по проектированию приложения	Трусов Евгений Дмитриевич
27	Написание главы по реализации серверной части приложения	Руденцов Данил Павлович, Трусов Евгений Дмитриевич
28	Написание главы по реализации клиентской части приложения	Немнов Роман Ильич, Трусов Евгений Дмитриевич
29	Выполнение других частей работы (финансовый менеджмент, социальная ответственность)	Все

4.2.2. Разработка графика проведения научного исследования.

Согласно производственному календарю при расчете на 6 рабочих дней в неделю в 2019 году 365 календарных дней, 299 рабочих дней, 66 выходных или праздничных дней. Таким образом, коэффициент календарности на 2019 год вычисляется следующим образом:

$$T_{\text{кал}} = \frac{T_{\text{кал}}}{T_{\text{кал}} - T_{\text{вых}} - \text{Т}_{\text{пр}}} = 1{,}22$$

Для расчета временных показателей проведения научного исследования, необходимо для каждой задачи определить минимальную и максимальную ожидаемую трудоемкость, выраженную в человеко-днях. Зная эти показатели, ожидаемая трудоемкость может быть вычислена по следующей формуле:

$$T_{\text{ож}} = \frac{3 * \text{Т}_{min} + 2 * T_{max}}{5}$$

Для нахождения длительности этапа работы, выраженного в днях, следует найти отношение ожидаемой трудоемкости к количеству участников проектной команды, задействованной для решения этапа. Умножив полученное значение на коэффициент календарности, можно получить ожидаемую календарную длительность каждого этапа работы, выраженную в днях. Длительность работ при этом следует округлять до целых чисел согласно математическим правилам. Таблица 11 содержит рассчитанные временные показатели каждого этапа работы.

Таблица 11. Временные показатели проведения научного исследования

Наименование работы	Исполнители работы	Трудоемкость работ, чел-дни			Длительность работ, дни	
		Tmin	Tmax	Тож	Тр	Тк
Выбор научного руководителя бакалаврской работы	Все	1	2	1,4	1	1
Составление и утверждение темы бакалаврской работы	Все, Соколова Вероника	1	2	1,4	1	1

Наименование работы	Исполнители работы	Трудоемкость работ, чел-дни			Длительность работ, дни	
		Tmin	Tmax	Тож	Тр	Тк
	Валерьевна					
Составление календарного плана-графика выполнения бакалаврской работы	Трусов Евгений Дмитриевич	1	2	1,4	1	1
Подбор и изучение литературы по теме бакалаврской работы	Руденцов Данил Павлович, Трусов Евгений Дмитриевич	6	11	8,0	4	5
Анализ предметной области	Все	3	7	4,6	2	2
Выбор программных решений для разработки клиентской части приложения	Немнов Роман Ильич	1	2	1,4	1	1
Выбор программных решений для разработки серверной части приложения	Руденцов Данил Павлович	1	2	1,4	1	1
Проработка архитектуры приложения	Руденцов Данил Павлович	3	7	5,8	6	7
Реализация основных микросервисов приложения	Руденцов Данил Павлович	13	16	14,2	14	17
Реализация дополнительных микросервисов	Трусов Евгений Дмитриевич	8	13	10,0	10	12
Реализация словарей данных	Руденцов Данил Павлович, Трусов Евгений Дмитриевич	7	12	9,0	5	6
Реализация шины событий и брокера сообщений	Трусов Евгений Дмитриевич	3	6	4,2	4	5
Интегрирование системы Moodle	Руденцов Данил Павлович	4	7	5,2	5	6
Выбор цветовой палитры клиентского интерфейса	Немнов Роман Ильич	2	3	2,4	2	2
Выработка концептуального дизайна приложения на различных устройствах	Немнов Роман Ильич	5	9	6,6	7	9

Наименование работы	Исполнители работы	Трудоемкость работ, чел-дни			Длительность работ, дни	
		Tmin	Tmax	Тож	Тр	Тк
Реализация компонентов интерфейса пользователя	Немнов Роман Ильич	25	35	29,0	29	35
Настройка внутренних переходов	Немнов Роман Ильич	2	3	2,4	2	2
Подключение библиотеки управления состоянием к серверной части	Немнов Роман Ильич, Руденцов Данил Павлович	4	7	5,2	3	4
Адаптация приложения под мобильные устройства	Немнов Роман Ильич, Трусов Евгений Дмитриевич	6	9	7,2	4	5
Настройка политики кросс-доменных запросов	Руденцов Данил Павлович	1	2	1,4	1	1
Тестирование серверной части	Руденцов Данил Павлович	3	5	3,8	4	5
Тестирование клиентской части	Немнов Роман Ильич, Трусов Евгений Дмитриевич	4	7	5,2	3	4
Пользовательское тестирование приложения	Все	2	3	2,4	1	1
Исправление ошибок приложения	Все	10	20	14,0	5	6
Написание главы по проектированию приложения	Трусов Евгений Дмитриевич	5	12	7,8	8	10
Написание раздела по реализации серверной части приложения	Руденцов Данил Павлович, Трусов Евгений Дмитриевич	8	18	12,0	6	7
Написание раздела по реализации клиентской части приложения	Немнов Роман Ильич, Трусов Евгений Дмитриевич	8	18	12,0	6	7

Наименование работы	Исполнители работы	Трудоемкость работ, чел-дни			Длительность работ, дни	
		Tmin	Tmax	Тож	Тр	Тк
Выполнение других частей работы (финансовый менеджмент, социальная ответственность)	Все	15	25	21,0	7	9
Подведение итогов, оформление работы	Все	10	20	14,0	5	6

Данные из таблицы можно использовать для построения диаграммы Ганта, при этом следует учитывать, что некоторые этапы исследовательской работы могут выполняться параллельно.

Видно, что количество работ, в которые вовлечен каждый участник проектной группы, примерно одинаково, что стало возможным благодаря задействованию современных средств коммуникации и контроля исходного кода.

Вовлеченность научного руководителя в проведение работ не превышает 10 %.

Построенная диаграмма Ганта представлена на рисунке 74.

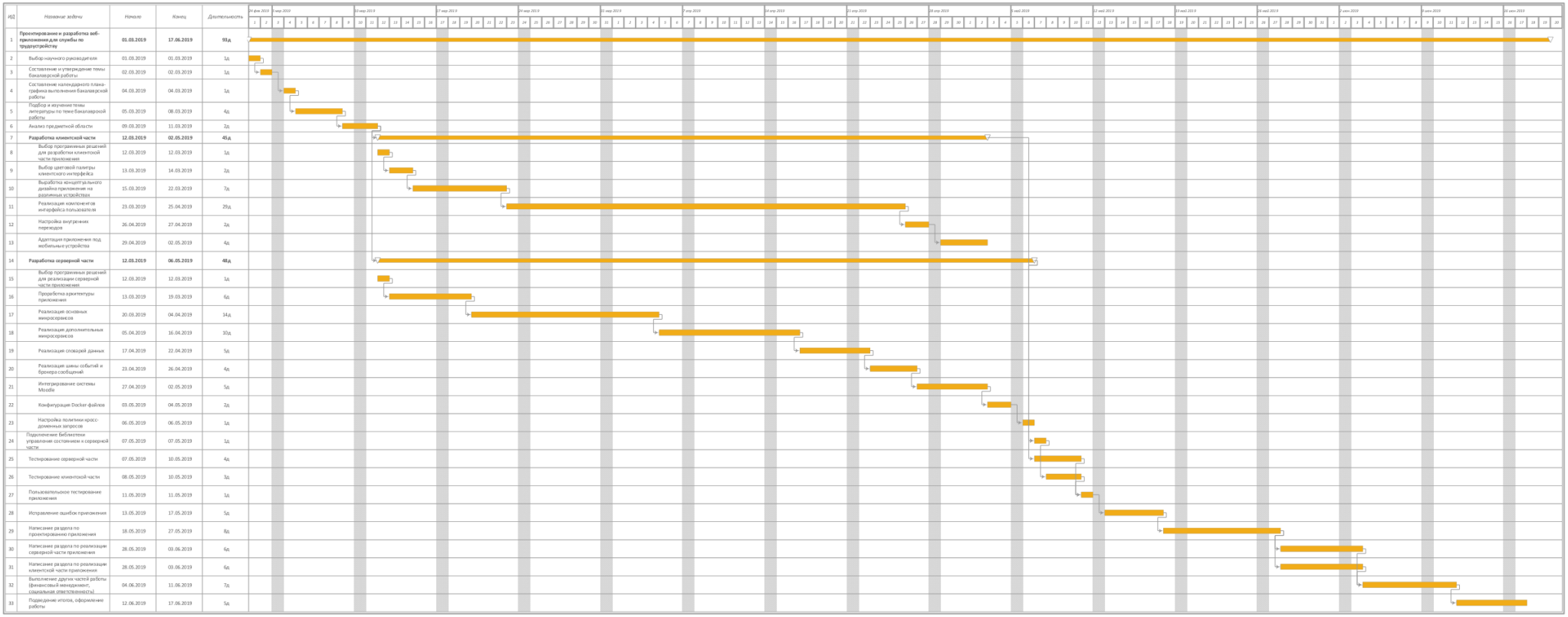

Рисунок 74. Диаграмма Ганта, охватывающая все задачи исследовательской работы.

Как видно из диаграммы, время полного выполнения исследовательской работы составило 93 календарных дня. Многие этапы выполнялись параллельно, так как затрагивали независимые компоненты веб-приложения: клиентская и серверная часть. При этом, определенные этапы требовали преждевременного завершения всех предыдущих, например, пользовательское тестирование, которое можно провести только после завершения тестирования клиентской и серверной части. Наиболее долгим этапом, занявшим 48 календарных дней (51.6 % от всего времени исследования)

оказалась разработка серверной части ввиду архитектурных сложностей и широты предметной области. На рисунках 75 и 76 показаны фрагменты полной диаграммы Ганта, соответствующие разработке клиентской и серверной частей.

1	**Проектирование и разработка веб-приложения для службы по трудоустройству**	**01.03.2019**	**17.06.2019**	**93д**
2	Выбор научного руководителя	01.03.2019	01.03.2019	1д
3	Составление и утверждение темы бакалаврской работы	02.03.2019	02.03.2019	1д
4	Составление календарного плана-графика выполнения бакалаврской работы	04.03.2019	04.03.2019	1д
5	Подбор и изучение темы литературы по теме бакалаврской работы	05.03.2019	08.03.2019	4д
6	Анализ предметной области	09.03.2019	11.03.2019	2д
7	**Разработка клиентской части**	**12.03.2019**	**02.05.2019**	**45д**
8	Выбор программных решений для разработки клиентской части приложения	12.03.2019	12.03.2019	1д
9	Выбор цветовой палитры клиентского интерфейса	13.03.2019	14.03.2019	2д
10	Выработка концептуального дизайна приложения на различных устройствах	15.03.2019	22.03.2019	7д
11	Реализация компонентов интерфейса пользователя	23.03.2019	25.04.2019	29д
12	Настройка внутренних переходов	26.04.2019	27.04.2019	2д
13	Адаптация приложения под мобильные устройства	29.04.2019	02.05.2019	4д

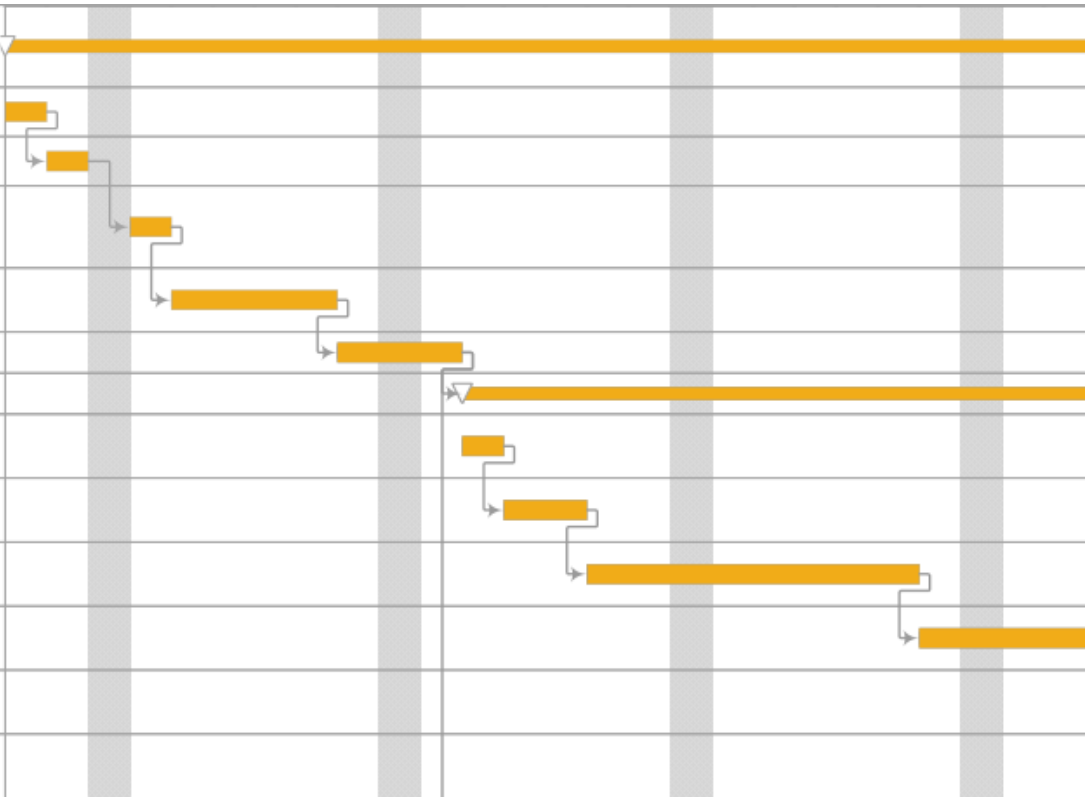

Рисунок 75. Фрагмент диаграммы Ганта – разработка клиентской части.

Реализация компонентов клиентского интерфейса заняла более 64 % времени от всего времени, затраченного на разработку клиентской части. Это связано с большим числом интерфейсных вкладок приложения и сложной внутренней маршрутизацией.

14	**Разработка серверной части**	**12.03.2019**	**06.05.2019**	**48д**
15	Выбор программных решений для реализации серверной части приложения	12.03.2019	12.03.2019	1д
16	Проработка архитектуры приложения	13.03.2019	19.03.2019	6д
17	Реализация основных микросервисов	20.03.2019	04.04.2019	14д
18	Реализация дополнительных микросервисов	05.04.2019	16.04.2019	10д
19	Реализация словарей данных	17.04.2019	22.04.2019	5д
20	Реализация шины событий и брокера сообщений	23.04.2019	26.04.2019	4д
21	Интегрирование системы Moodle	27.04.2019	02.05.2019	5д
22	Конфигурация Docker-файлов	03.05.2019	04.05.2019	2д
23	Настройка политики кросс-доменных запросов	06.05.2019	06.05.2019	1д
24	Подключение библиотеки управления состоянием к серверной части	07.05.2019	07.05.2019	1д
25	Тестирование серверной части	07.05.2019	10.05.2019	4д
26	Тестирование клиентокой части	08.05.2019	10.05.2019	3д
27	Пользовательское тестирование приложения	11.05.2019	11.05.2019	1д
28	Исправление ошибок приложения	13.05.2019	17.05.2019	5д
29	Написание раздела по проектированию приложения	18.05.2019	27.05.2019	8д
30	Написание раздела по реализации серверной части приложения	28.05.2019	03.06.2019	6д
31	Написание раздела по реализации клиентской части приложения	28.05.2019	03.06.2019	6д
32	Выполнение других частей работы (финансовый менеджмент, социальная ответственность)	04.06.2019	11.06.2019	7д
33	Подведение итогов, оформление работы	12.06.2019	17.06.2019	5д

Рисунок 76. Фрагмент диаграммы Ганта – разработка серверной части и написание отчетности по исследовательской работе

4.3. Бюджет научно-технического исследования

4.3.1. Расчет материальных затрат научно-технического исследования

На момент начала исследовательской работы группа студентов обладала всеми необходимыми аппаратно-техническим средствами, в связи с

чем потребности в приобретении дополнительных средств не возникало. Однако для расчета материальных затрат на исследование необходимо рассчитать амортизацию основных средств и нематериальных активов.

Для написания исследовательской работы использовался персональный компьютер повышенного класса мощности с периферийными устройствами на сумму 106000 рублей и 2 ноутбука среднего класса мощности стоимостью 45000 рублей каждый. Общая стоимость технического оборудования составила 196000 рублей. Срок полезного использования офисных машин (код 330.28.23.23) составляет от 2 до 3 лет, в расчетах его можно принять его за 3 года. Время, требуемое для написания ВКР – около 5 месяцев.

Норма амортизации вычисляется по следующей формуле:

$$\mathrm{А_н} = \frac{100\%}{3} = 33{,}33\ \%$$

Годовые амортизационные отчисления:

$$\mathrm{А_г} = S * \frac{\mathrm{А_н}}{100}\% = 196000 * 0.33 = 65333 \text{ рубля}$$

Ежемесячные амортизационные отчисления при этом составят:

$$\mathrm{А_м} = \frac{\mathrm{А_г}}{12} = 5444 \text{ рубля}$$

Итоговая сумма амортизации основных средств за время написания исследовательской работы составит: А = 5444*5 = 27222 рубля.

Для разработки программного продукта использовалось программное обеспечение со свободной моделью распространения или ПО бесплатной версии, уменьшенного функционала которого по сравнению с полной версией, все равно было бы достаточно для проведения исследовательской работы. Из этого следует, что амортизация ПО в рамках проведения исследовательской работы была бесплатной.

4.3.2. Основная заработная плата исполнителей исследовательской работы

Ниже представлена формула для расчета затрат на заработную платы

$$З_п = З_{осн} + З_{доп}, \text{ где}$$

$З_{осн}$ – выраженная в рублях основная заработная плата;

$З_{доп}$ – выраженная в рублях дополнительная заработная плата.

Основную заработную плату можно вычислить по формуле:

$$З_{осн} = З_{дн} * Т_р * (1 + К_{пр} + К_д) * К_{р,} \text{ где}$$

$З_{дн}$ – среднедневная заработная плата, руб.;

$Т_р$ – продолжительность работ, выполняемых работником (рабочие дни);

$К_{пр}$ – премиальный коэффициент (0,3);

$К_д$ – коэффициент доплат и надбавок (0,3-0,5);

$К_р$ – районный коэффициент, для Томска составляет 1,3.

Среднедневная заработная плата вычисляется по формуле:

$$Здн = \frac{Зм * М}{Fд}$$

Зм – месячный оклад работника, руб. М – количество месяцев работы без отпуска в течение года (для 6-дневной рабочей недели М=10,4). Fд – действительный годовой фонд рабочего времени персонала, выражен в рабочих днях. Расчет баланса приведен в таблице 12.

Таблица 12. Баланс рабочего времени (для 6-дневной недели)

Показатели рабочего времени	Дни
Календарные дни	365
Нерабочие дни (праздники/выходные)	66
Потери рабочего времени (отпуск/невыходы по болезни)	56
Действительный годовой фонд рабочего времени	243

Таким образом, в 2019 году действительный годовой фонд рабочего времени составляет 243 дня. Найденных показателей достаточно, чтобы составить таблицу расчета основной заработной платы, при этом зарплата студента в месяц равняется 21760, а руководителя – 33664 рублей.

Таблица 13. Расчет основной заработной платы

Исполнители	$З_{дн}$, руб	$К_{пр}$	$К_{д}$	$К_{р}$	$Т_{р}$	$З_{осн}$
Немнов Роман Ильич	931,29	0,3	0,2	1,3	85	154361,32
Руденцов Данил Павлович	931,29	0,3	0,2	1,3	85	154361,32
Трусов Евгений Дмитриевич	931,29	0,3	0,2	1,3	85	154361,32
Соколова Вероника Валерьевна	1440,76	0,3	0,5	1,3	4	13485,51
Итого						**476569,47**

4.3.3. Расчет дополнительной заработной платы

Дополнительная заработная плата может быть вычислена путем умножения основной заработной платы на соответствующий коэффициент. Для расчетов значение коэффициента было установлено в 0,15.

Таблица 14. Расчет дополнительной заработной платы

Исполнитель	Основная заработная плата, руб.	Коэффициент дополнительной заработной платы	Дополнительная заработная плата, руб
Немнов Р.И.	154361,32	0,15	23154,2
Руденцов Д.П.	154361,32		23154,2
Трусов Е.Д.	154361,32		23154,2
Соколова В.В.	13485,51		2022,82
Итого:			**71485,43**

4.3.4. Отчисления во внебюджетные фонды

Подсчитав основную и дополнительную заработную плату, можно рассчитать сумму отчислений во внебюджетные фонды через соответствующей коэффициент, что отражено в таблице 15.

Таблица 15. Расчет отчислений во внебюджетные фонды

Исполнитель	Основная заработная плата + дополнительная, руб.	Коэффициент отчислений во внебюджетные фонды	Сумма отчислений
Немнов Р.И.	177515,52	0,28	49704,35
Руденцов Д.П.	177515,52		49704,35
Трусов Е.Д.	177515,52		49704,35
Соколова В.В.	15508,33		4342,33
Итого:			**153455,38**

4.3.5. Накладные расходы

Помимо затрат на амортизацию и выплату заработной платы, компания также несет издержки за оплату электроэнергии, услуг связи, ксерокопии материалов и пр., эти издержки объединяются в накладные расходы, которые могут быть вычислены по следующей формуле:

$$З_{накл} = \sum_{i=1}^{4} \text{Сумма } i - \text{той статьи} * К_{нр}, \text{ где}$$

$К_{нр}$ – коэффициент, учитывающий накладные расходы (16 %)

4.3.6. Формирование бюджета затрат научно-исследовательского проекта

Зная затраты на амортизацию, выплату основной и дополнительной заработной платы, отчисления во внебюджетные фонды и накладные расходы, можно вычислить общий бюджет научно-исследовательского проекта, что показано в таблице 16.

Таблица 16. Бюджет затрат научно-исследовательского проекта

Наименование статьи	Сумма (руб.)	Примечание
Амортизационные затраты на спецоборудование	27222,00	См. раздел 4.3.1.
Затраты на основную заработную плату	476569,47	См. графу «итого» в таблице 13
Затраты на дополнительную заработную плату	71485,43	См. графу «итого» в таблице 14
Затраты на отчисление во внебюджетные фонды	153455,38	См. графу «итого» в таблице 15
Накладные расходы	102022,34	16 % от суммы статей 1-4
Бюджет затрат НТИ	830753,72	Сумма статей 1-5

Таким образом, было установлено, что бюджет затрат научно-исследовательского проекта составляет 831 000 рублей.

4.4. Определение ресурсной (ресурсосберегающей), финансовой, бюджетной, социальной и экономической эффективности исследования

Эффективность проекта можно рассчитать с помощью интегрального показателя эффективности научного исследования. Интегральный показатель финансовой эффективности получают в ходе оценки бюджета затрат нескольких вариантов выполнения исследования по следующей формуле:

$$I_{\text{фин р}}^{\text{исп } i} = \frac{\Phi_{pi}}{\Phi_{max}}, \text{ где}$$

$I_{\text{фин } p}^{\text{исп } i}$ – интегральный финансовый показатель разработки;

Φ_{pi} – стоимость i -го варианта исполнения;

Φ_{max} – максимальная стоимость исполнения научно-исследовательского проекта.

Интегральный показатель ресурсоэффективности может быть вычислен по следующей формуле:

$$I_{pi} = \sum a_i * b_i, \text{ где}$$

I_{pi} – интегральный показатель ресурсоэффективности для i-го варианта исполнения разработки;

a_i – весовой коэффициент i-го варианта исполнения разработки;

b_i – оценка i-го варианта исполнения разработки, выраженная в баллах, устанавливается экспертным путем по выбранной шкале оценивания;

n – число параметров сравнения.

Таблица 17. Сравнительная оценка характеристик вариантов исполнения проекта

Объект исследования/ критерий	Весовой коэффициент параметра	Исп. 1	Исп. 2	Исп. 3
Способствует росту производительности труда пользователя	0,15	3	5	3
Удобство в эксплуатации (соответствует требованиям потребителей)	0,25	3	4	5
Помехоустойчивость	0,10	4	5	3
Энергосбережение	0,15	5	2	4
Надежность	0,25	5	3	4
Материалоемкость	0,10	4	5	4

$I_{р\text{-}исп1}$ = 3 * 0,15 + 3 * 0,25 + 4 * 0,1 + 4 * 0,15 + 5 * 0,25 + 4 * 0,1 = 3,85

$I_{р\text{-}исп2}$ = 5 * 0,15 + 4 * 0,25 + 5 * 0,1 + 2 * 0,15 + 3 * 0,25 + 5 * 0,1 = 3,8

$I_{р\text{-}исп3}$= 4 * 0,15 + 5 * 0,25 + 3 * 0,1 + 4 * 0,15 + 4 * 0,25 + 4 * 0,1 = 4,0

Интегральный показатель эффективности вариантом использования определяется на основании интегрального показателя ресурсоэффективности и интегрального финансового показателя по формуле [23]:

$$\mathrm{I}_{\text{испi}} = \frac{\mathrm{I}_{\text{р}-\text{испi}}}{I^{\text{исп.}i}_{\text{финр}}}$$

Сравнительная эффективность проекта:

$$Э_{ср} = \frac{I_{исп1}}{I_{исп2}}$$

Используя эти формулы, можно найти наиболее эффективный вариант исполнения. Расчеты приведены в таблице 18.

Таблица 18. Сравнительная эффективность разработки

№	Показатели	Исп. 1	Исп. 2	Исп. 3
1	Интегральный финансовый показатель разработки	0,90	0,85	0,95
2	Интегральный показатель ресурсоэффективности разработки	3,85	3,8	4,0
3	Интегральный показатель эффективности	4,28	4,47	4,21
4	Сравнительная эффективность вариантов исполнения	0,96	1	0,94

Выводы по главе

В разделе исследовательской работы, посвященной оценке коммерческого потенциала и перспективности, были определены финансовый показатель разработки, показатель ресурсоэффективности, с помощью которых был вычислен интегральный показатель эффективности для каждого возможного варианта исполнения работы. Сравнение интегрального показателя эффективности трех вариантов исполнения показал, что наиболее эффективным решением является 2 вариант исполнения.

Глава 5. Социальная ответственность

5.1. Введение

Исследовательская работа заключалась в проектировании и разработке веб-приложения, автоматизирующего процессы, связанные с интернет-рекрутингом. Используя приложение, соискатели и работодатели будут способны устанавливать деловой контакт значительно быстрее, что в перспективе снизит уровень безработицы в России. Поиск вакансий с устанавливаемыми параметрами фильтрации позволит соискателям быстро и точно находить все интересующие их вакансии, оперативно узнавать о новых тенденциях на рынке трудоустройства, корректируя при этом программу личного роста. Особое внимание при разработке приложения было уделено определенной категории соискателей – учащимся старших курсов и молодым специалистам, так как именно они испытают наибольшие затруднения при трудоустройстве ввиду отсутствия опыта. К тому же, именно эта категория лиц является наиболее активной в смысле использования современных информационных систем и технологий.

Согласно требованиям, предъявленным к разрабатываемому веб-приложению, клиентами могут являться пользователи, использующие как стационарные персональные компьютеры и ноутбуки, так и мобильные устройства. По этой причине интерфейс веб-приложения необходимо было адаптировать под различные пользовательские устройства, проверка корректности разметки содержимого осуществлялась лично участниками исследовательской работы. Сама разработка при этом велась с использованием ноутбуков и персонального компьютера, в связи с чем участники проектной группы подвергались таким вредным факторам, как:

1. Некачественные TN-матрицы ноутбуков под острым углом сильно искажают цвета, вплоть до полного инвертирования при угле, близком к 180 градусов. Неправильная цветопередача может привести к легкой дезориентации и головным болям;

2. Частота вертикальной развертки мониторов ограничена 60 Гц, мерцание, возникающее при этом из-за особенностей работы ШИМ-контроллера, быстро утомляет глаза;
3. В непосредственной близости от электронно-вычислительных устройств существует область повышенного электромагнитного излучения, длительное нахождение в которой вредно для внутренних органов;
4. Специфичность трудовой деятельности, связанной с разработкой веб-приложений, состоит в необходимости выполнения работ преимущественно в сидячем положении, что может привести к нарушению кровообращения и образованию зловредных новообразований;
5. Сбои в электропитании могут нарушить равномерность подсветки матриц жидкокристаллических мониторов, в результате утомляемость глаз может дополнительно усилиться;
6. В вечернее и ночное время разработчики вынуждены использовать искусственное освещение.

5.2. Правовые и организационные вопросы обеспечения безопасности

5.2.1. Организационные мероприятия обеспечения безопасности

Разработка веб-приложения велась в помещении общей площадью 26 квадратных метров, в качестве искусственного источника освещения использовались 3 лампы накаливания, общей мощностью 85 Вт. Помещение оборудовано одним крупным столом, позволяющим работать одновременно до двух человек, и одним компьютерным столом стандартного размера с выдвижной клавиатурой. Участники научно-исследовательской работы взаимодействовали с электронно-вычислительными устройствами, находясь в операторских креслах, выполненных в виде компьютерного кресла с регулируемыми подлокотниками и углом наклона спинки. Перемещение кресла внутри помещения обеспечивают 5 пластиковых колес диаметром 50 мм. Каждый участник исследования постоянно имел доступ к свободному операторскому креслу.

С целью минимизации воздействия вредных факторов на группу разработчиков при выполнении научно-исследовательской работы, рабочее место должно быть организовано с учетом требований ГОСТ 12.2.032-78 «ССБТ. Рабочее место при выполнении работ сидя. Общие эргономические требования» и СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы».

Ниже приведены наиболее важные для соблюдения фрагменты стандарта (используется оригинальная нумерация пунктов соответствие с ГОСТ 12.2.032-78 «ССБТ. Рабочее место при выполнении работ сидя. Общие эргономические требования») [24]:

1. Подвижность кресла относительно пола или другой поверхности, на которой оно установлено, может не ограничиваться. В случае необходимости обеспечения строго определенного положения человека-оператора по отношению к средствам отображения информации и органам управления, а также в случае, если трудовая деятельность человека-оператора сопряжена с силовыми и резкими движениями, кресло должно быть фиксировано. При этом, в зависимости от характера трудовой деятельности оператора, должна быть обеспечена возможность изменения положения кресла или сиденья в горизонтальной плоскости с фиксацией его в нужном положении. При необходимости подвижность кресла должна задаваться также вращением кресла на 180-360° вокруг вертикальной оси опорной конструкции кресла с фиксацией в нужном положении.
2. Подставка для ног должна быть регулируемой по высоте. Ширина должна быть не менее 300 мм, длина - не менее 400 мм. Поверхность подставки должна быть рифленой. По переднему краю следует предусматривать бортик высотой 10 мм.
3. При работе двумя руками органы управления размещают с таким расчетом, чтобы не было перекрещивания рук.

4. Помещения, где размещаются рабочие места с ПЭВМ, должны быть оборудованы защитным заземлением (занулением) в соответствии с техническими требованиями по эксплуатации.

5. Расстояние между внутренними гранями подлокотников определяется межлоктевым диаметром, измеренным в положении сидя, для 95 перцентиля данного антропометрического признака, с поправкой на специальную одежду и снаряжение, а при наличии регулировки параметра - 50-95 перцентилями данного антропометрического признака.

6. В помещениях, оборудованных ПЭВМ, проводится ежедневная влажная уборка и систематическое проветривание после каждого часа работы на ПЭВМ.

7. Рабочие столы следует размещать таким образом, чтобы видеодисплейные терминалы были ориентированы боковой стороной к световым проемам, чтобы естественный свет падал преимущественно слева.

8. В качестве источников света при искусственном освещении следует применять преимущественно люминесцентные лампы типа ЛБ и компактные люминесцентные лампы (КЛЛ). При устройстве отраженного освещения в производственных и административно-общественных помещениях допускается применение металлогалогенных ламп. В светильниках местного освещения допускается применение ламп накаливания, в том числе галогенные.

9. Рабочие места с ПЭВМ при выполнении творческой работы, требующей значительного умственного напряжения или высокой концентрации внимания, рекомендуется изолировать друг от друга перегородками высотой 1,5 - 2,0 м.

10. Экран видеомонитора должен находиться от глаз пользователя на расстоянии 600-700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов.

Проектной группой научно-исследовательской работы были соблюдены в допустимой мере все требования, предусматриваемые государственным стандартом 12.2.032-78. Во время выполнения выпускной квалификационной работы не происходило случаев, несущих в себе угрозы для здоровья и жизни сотрудников и участников проектной группы, а также представляли опасности для окружающей среды.

5.2.2. Особенности законодательного регулирования проектных решений

Основными документами предметной области, связанной с трудоустройством, являются вакансии работодателей и резюме соискателей. В документах в большом объеме содержатся личные данные пользователей – ФИО, контакты, прошлый опыт работы, личные достижения и предпочтения и пр. Поэтому было принято решение обратиться к федеральному закону Российской Федерации «О персональных данных», глава 2, статья 5 [25]:

1. Обработка персональных данных должна осуществляться на законной и справедливой основе.
2. Обработка персональных данных должна ограничиваться достижением конкретных, заранее определенных и законных целей. Не допускается обработка персональных данных, несовместимая с целями сбора персональных данных.
3. Не допускается объединение баз данных, содержащих персональные данные, обработка которых осуществляется в целях, несовместимых между собой.
4. Обработке подлежат только персональные данные, которые отвечают целям их обработки.
5. Содержание и объем обрабатываемых персональных данных должны соответствовать заявленным целям обработки. Обрабатываемые персональные данные не должны быть избыточными по отношению к заявленным целям их обработки.

6. При обработке персональных данных должны быть обеспечены точность персональных данных, их достаточность, а в необходимых случаях и актуальность по отношению к целям обработки персональных данных. Оператор должен принимать необходимые меры либо обеспечивать их принятие по удалению или уточнению неполных, или неточных данных.
7. Хранение персональных данных должно осуществляться в форме, позволяющей определить субъекта персональных данных, не дольше, чем этого требуют цели обработки персональных данных, если срок хранения персональных данных не установлен федеральным законом, договором, стороной которого, выгодоприобретателем или поручителем, по которому является субъект персональных данных. Обрабатываемые персональные данные подлежат уничтожению либо обезличиванию по достижении целей обработки или в случае утраты необходимости в достижении этих целей, если иное не предусмотрено федеральным законом.

Исходя из положения федерального закона, участниками проектной работы было принято решение отказаться от использования NoSQL СУБД в микросервисах приложения, связанных с хранением документов, хранящих личные данные пользователей из-за отсутствия в них эффективного механизма транзакций и меньших уровней защиты. В клиентские компоненты интерфейса, связанные с созданием и редактированием резюме и вакансий, был добавлен графический элемент, сообщающий пользователям о согласии на обработку персональной информации.

5.3. Производственная безопасность

В данном пункте производится анализ вредных и опасных факторов, которые могут возникнуть на одном из этапов выполнения работы.

Химические факторы не рассмотрены в связи с использованием высококачественных материалов и отсутствием необходимости взаимодействия с химикатами в процессе выполнения научно-исследовательской работы. Помещение, где выполнялись работы, располагалось в дали от крупных городских

улиц на 6 этаже, в связи с чем шумовые факторы были неспособны оказать существенное влияние на ход проведения работ.

Рассмотрены следующие факторы: отклонение показателей микроклимата и естественного, недостаточная освещенность и повышенный уровень электромагнитного излучения. Таблица 19 содержит информацию о воздействии факторов в зависимости от этапов научного исследования.

Таблица 19. Возможные опасные и вредные факторы

Факторы (по ГОСТ 12.0.003-74)	Этапы работ			Нормативные документы
	Проектирование	Разработка	Формирование отчетности	
Отклонение показателей микроклимата	+	+	+	СанПиН 2.2.2.548-96
Отсутствие или недостаток естественного света	+	+	+	СНиП 23-05-95*
Недостаточная освещенность рабочей зоны	+	+	+	СанПиН 2.2.1/2.1.1.1278–03
Повышенный уровень электромагнитных излучений	+	+	+	СанПиН 2.2.4/2.1.8.055-96.

Как видно из таблицы 19, опасные и вредные факторы воздействовали на участников разработки на всем протяжении работ, так как все этапы научно-исследовательской работы были произведены в одном и том же помещении.

5.3.1. Анализ опасных и вредных производственных факторов

Таблица 20. Влияние опасных и вредных факторов

Фактор	Источник	Воздействие	Допустимые нормы
Отклонение показателей	Отсутствие кондиционеров	Вялость, усталость,	Таблица 21

Фактор	Источник	Воздействие	Допустимые нормы
микроклимата		сниженная концентрация	
Отсутствие или недостаток естественного света	Периодическая необходимость работы за ЭВМ в ночное время	Ухудшение зрения, усталость глаз	КЕО не ниже 1,2 %-1,5 %
Недостаточная освещенность рабочей зоны	Недостаточная мощность осветительных приборов	Ухудшение зрения, усталость глаз	Освещенность на рабочей поверхности от системы общего искусственного освещения 200-300 лк.
Повышенный уровень электромагнитных излучений	Компоненты персональных компьютеров и ноутбуков	Возможно возникновение рака	Напряженность электростатического поля не более 20 кВ/м

Норма микроклимата является плавающим параметров и зависит от температуры помещения, поверхностей, влажности и скорости воздуха [26]. Допустимые величины показателей микроклимата продемонстрированы в таблице 21.

Таблица 21. Допустимые величины показателей микроклимата

Период года	Температура воздуха, °С	Температура поверхностей, °С	Относительная влажность воздуха, %	Скорость движения воздуха, м/с
Холодный	20,0-21,9	19,0-26,0	15-75	0,1
Теплый	21,0-22,9	20,0-29,0		

5.3.2. Обоснование мероприятий по снижению уровней воздействия опасных и вредных факторов

5.3.2.1. Мероприятия по снижению воздействия недопустимого микроклимата

Для восстановления и поддержания допустимого микроклимата необходимо придерживаться следующих правил:

- Оборудование помещения системами обогрева, вентилирования и увлажнения.
- Оборудование помещения современными пластиковыми окнами, поддерживающими возможность микровентилирования.
- Защита фасада здания от солнца: шторы, жалюзи, навесы и т.д.
- Рационально размещать рабочие места.
- Ежедневная влажная уборка рабочего помещения.

5.3.2.2. Мероприятия по снижению воздействия недопустимого

Для решения проблемы отсутствия или недостатка естественного света и плохой освещенности рабочего места подходят следующие пункты:

- Сокращение времени работы.
- Своевременная чистка стекол в светопроемах.
- Снос деревьев, препятствующих проникновению света в помещение.
- Ремонт помещения в светлых тонах.
- Установка более мощных ламп или в большем количестве.
- Установка ламп в правильном положении.

5.3.2.3. Мероприятия по снижению воздействия недопустимого уровня электромагнитного излучения

Повышенный уровень электромагнитных излучений можно избежать, если следовать следующим пунктам:

- Прекратить использование мониторов с электронно-лучевой трубкой.
- Использовать высокоэффективные блоки питания и прочие преобразователи напряжения.
- Располагать монитор в углу помещения для того, чтобы стены поглощали излучение.

- Выключать компьютер при его неиспользовании.
- Сокращать время, проводимое за компьютером.

5.4. Экологическая безопасность

Экологической безопасностью называется комплекс мероприятий по снижению негативных влияний производственной деятельности человека на окружающую среду и защиту человека от последствий этого влияния [27].

Для выполнения научно-исследовательской работы использовались компьютеры, ноутбуки и смартфоны средней мощности. Современные электронно-вычислительные устройства не выбрасывают в окружающую среду каких-либо вредных веществ, однако используют для работы электроэнергию и создают электромагнитные поля. Рост потребления электроэнергии может привести к строительству дополнительных атомных электростанций, что повышает вероятность ядерной катастрофы. Несанкционированное использование смартфонов может вывести из строя технику, не предназначенную для длительного нахождения в электромагнитных полях. Вывод из строя бортовых систем управления самолетов может привести к авиакатастрофам. Производство и утилизация современных вычислительных устройств составляют серьезную проблему: текстолит, используемый при производстве микросхем, имеющих срок разложения более тысячи лет.

Также при производстве смартфонов и персональных компьютеров используются тяжелые, щелочноземельные металлы, ртуть, пластик и стекло, что без должной утилизации по окончании службы попадает в природу и остается в не переработанном виде.

Мероприятия, позволяющие сохранять экологическую безопасность находясь на своем рабочем месте:

- Правильная утилизация персональных компьютеров и смартфонов, а также их комплектующих;
- Использование энергосберегающих ламп;
- Использование аккумуляторов вместо солевых батареек.

5.5. Безопасность в чрезвычайных ситуациях

Чрезвычайная ситуация (ЧС) – обстановка на определенной территории, сложившаяся в результате аварии, опасного природного явления, катастрофы, стихийного или иного бедствия, которая может повлечь за собой человеческие жертвы, а также ущерб здоровью человека или окружающей среде, значительные материальные потери и нарушение условий жизнедеятельности [28].

5.5.1. Пожар

Научно-исследовательская работа проходила в помещении, подходящем под определение офис. Одной из наиболее возможной чрезвычайной ситуацией, которая может возникнуть при работе в помещениях такого типа – пожар. К пожару могут привести неисправности в технических средствах, оргтехнике, а также действия самих сотрудников [29]. Главное во время пожара – не поддаваться панике и действовать согласно правилам поведения при пожаре. Для сотрудника существует порядок действий и правила поведения в подобной чрезвычайной ситуации:

1. Заметив пожар или загорание, необходимо немедленно организовать оповещение об этом всех находящихся в здании людей, независимо от размеров и места пожара или загорания, равно как и при обнаружении хотя бы малейших признаков горения (дыма, запаха гари) и немедленно вызвать пожарную охрану по телефону «01». Очевидно, что быстрота прибытия пожарной помощи, позволит успешнее ликвидировать пожар и быстрее помочь людям, находящимся в опасности.
2. Сообщения о пожаре, как правило, передаются по телефону. Поэтому каждый человек должен хорошо знать места расположения телефонных аппаратов, особенно тех, которые доступны в любое время суток. Следует помнить, что с помощью сотового телефона можно вызвать помощь даже при отсутствии денег на счете или SIM-карты по номеру «112».

3. Каждый работник образовательного учреждения, обнаруживший пожар или его признаки (задымление, запах горения или тления различных материалов, повышение температуры и т.п.) обязан:
4. немедленно сообщить об этом по телефону в пожарную часть (при этом необходимо четко назвать адрес учреждения, место возникновения пожара, а также сообщить свою должность, фамилию и номер своего телефона);
5. задействовать систему оповещения людей о пожаре, приступить самому и привлечь других лиц к эвакуации детей из здания в безопасное место согласно плану эвакуации;
6. известить о пожаре руководителя образовательного учреждения или заменяющего его работника;
7. организовать встречу пожарных подразделений, принять меры по тушению пожара имеющимися в учреждении средствами пожаротушения.

5.5.2. Землетрясение

Под землетрясением понимают подземные толчки и колебания земной поверхности. Землетрясения отражают процесс геологического преобразования планеты, первопричиной землетрясений являются глобальные геологические и тектонические силы

Томская область располагается на значительном отдалении от зон сейсмической активности, однако, за последние 10 лет в пределах городской зоны произошло несколько землетрясений, максимальная зафиксированная амплитуда толчков составила 5,9. В связи с тем, что научно-исследовательская работа происходила в помещении на 6 этаже, что создает дополнительные факторы риска при земных толчках, участникам работ следует знать инструкции по поведению во время землетрясения [30]:

1. При возможности захватить с собой документы, деньги, предметы первой необходимости, фонарик.
2. Остерегаться падающих предметов, оборванных проводов и других источников опасности.

3. Сохранять спокойствие и не допускать паники.
4. При нахождении на верхних этажах многоэтажного здания — оставаться в здании, предварительно открыть входную дверь, которая в дальнейшем может оказаться перекошенной и заклиненной.
5. Быстро занять наиболее безопасное место в помещении: в дверных проемах капитальных стен, у ближайшей к центру здания капитальной стены, опорной колонны, в углу комнаты, всегда подальше от окон, тяжелых предметов и мебели, которые могут опрокинуться.
6. В случае разрушения здания, сопровождающегося падением отдельных элементов перекрытия или частей капитальных стен, необходимо немедленно покинуть здание.
7. Покидая здание, не выпрыгивать из окон, расположенных выше первого этажа, стекла выбивать подручными средствами (стулом, табуреткой), в крайнем случае, рукой, обмотанной тряпкой.

Выводы по главе

В результате изучения и анализа стандартов и правил, касающихся работе в помещениях с электронно-вычислительными устройствами, можно сделать вывод, что выполнение проекта соответствовало всем заявленным нормам безопасности жизнедеятельности. Участники проектной группы не подвергались серьезному воздействию опасных факторов.

Рабочие места и помещение в целом во время проведения исследовательской работы соответствовало региональным стандартам, а также санитарно-эпидемиологическим правилам и нормам. Федеральный закон «О персональных данных» внес изменения в архитектуру серверной части и визуальному изменению некоторых компонентов приложения, члены проектной группы готовы к поведению в условиях чрезвычайных ситуаций.

Заключение

Для реализации веб-приложения для системы трудоустройства командой разработчиков было написано более 171000 строчек кода, обеспечивающих развертывание и функционирование 11 микросервисов и 16 различных вкладок клиентской части веб-приложения, включающих в себя более 30 пользовательских реактивных компонентов. Благодаря выбору микросервисной архитектуры и использованию проверенных паттернов проектирования, удалось достичь индекса поддерживаемости кода в 91,9 %, что означает высокую степень готовности веб-приложения к дальнейшим изменениям и масштабированию.

Использование средства контейнеризации Docker позволило веб-приложению не зависеть от программного оснащения конкретной серверной машины или кластера машин, что позволяет более оперативно развёртывать веб-приложение на удаленном хостинге.

Проанализировав предметную область и выявив список наиболее опасных конкурирующих программных продуктов в сфере автоматизации процессов интернет-рекрутинга, были скорректированы функциональные требования к веб-приложению. Для молодых специалистов и учащихся старших курсов была добавлена возможность выбора специализаций «Дней карьеры» и дальнейшего получения SMS- или E-mail уведомлений. По результатам опроса, проведенному среди студентов 4 курса ТПУ, было выяснено, что около 10-15 % учащихся не посещали «Дни карьеры» из-за отсутствия напоминания. Внедрение данного веб-приложения в информационные системы образовательных учреждений может повысить явку студентов на «Дни карьеры» и снизить уровень безработицы среди молодых специалистов в целом.

Внедрение возможности создания пользовательских тестов на платформе Moodle для менеджеров компаний позволит проводить отбор квалифицированных соискателей быстрее и эффективнее, а самим соискателям – получать актуальную информацию о требованиях работодателей.

Список использованных источников

1. Анализ состояния безработицы в экономике России [Электронный ресурс]. – Режим доступа: https://novainfo.ru/article/12041 (дата обращения: 04.05.2019).
2. Экономически активное население [Электронный ресурс]. – Режим доступа: https://www.e-reading.club/chapter.php/127765/49/Maksakovskiii_-_Geograficheskaya_kartina_mira_Posobie_dlya_vuzov_Kn._I__Obshchaya_harakteristika_mira._Global%27nye_p--chestva.html / (дата обращения: 02.06.2019).
3. Социальная сеть для поиска и установления деловых контактов [Электронный ресурс]. – Режим доступа: https://ru.linkedin.com/ (дата обращения: 12.05.2019).
4. Крупнейшая Российская компания интернет-рекрутинга [Электронный ресурс]. – Режим доступа: https://hh.ru (дата обращения: 13.05.2019).
5. Российская компания интернет-рекрутмента [Электронный ресурс]. – Режим доступа: https://zarplata.ru (дата обращения: 13.05.2019).
6. Использование криптографического алгоритма RSA [Электронный ресурс]. – Режим доступа: http://www.e-nigma.ru/stat/rsa/ (дата обращения: 02.06.2019).
7. Просто о микросервисах [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/raiffeisenbank/blog/346380/ (дата обращения: 27.05.2019).
8. Успешные истории внедрения протокола AMQP [Электронный ресурс]. – Режим доступа: https://www.amqp.org/about/examples (дата обращения: 22.05.2019).
9. Исследование производительности RabbitMQ [Электронный ресурс]. – Режим доступа: https://www.cloudamqp.com/blog/2018-01-08-part2-rabbitmq-best-practice-for-high-performance.html (дата обращения: 27.05.2019).

10. Сравнение производительности NoSQL СУБД [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/mailru/blog/352760/ (дата обращения: 27.05.2019).
11. Как работает виртуальный DOM [Электронный ресурс]. – Режим доступа: https://learn-reactjs.ru/faq/virtual-dom-and-internals (дата обращения: 28.05.2019).
12. Trello – программная для управления проектами небольших групп [Электронный ресурс]. – Режим доступа: https://trello.com/ (дата обращения: 03.04.2019).
13. Docker: основные термины и принципы [Электронный ресурс]. – Режим доступа: https://aws.amazon.com/ru/docker/ (дата обращения: 12.04.2019).
14. БЭМ – технологии Яндекса [Электронный ресурс]. – Режим доступа: https://yandex.ru/dev/bem/ (дата обращения: 20.04.2019).
15. Redux и Thunk вместе с Redux-thunk, пособие для начинающих [Электронный ресурс]. – Режим доступа: https://tuhub.ru/posts/redux-i-thunk-vmeste-react-rukovodstvo-dlya-chajnikov (дата обращения: 29.04.2019).
16. Препроцессор SASS/SCSS: документация на русском для начинающих [Электронный ресурс]. – Режим доступа: https://zaurmag.ru/html5-css3/preprotsessor-sass.html (дата обращения: 04.05.2019).
17. Чистые функции – Введение в язык программирования Haskell [Электронный ресурс]. – Режим доступа: https://ru.hexlet.io/courses/introduction_to_programming/lessons/pure/theory_unit (дата обращения: 25.05.2019).
18. Приемы проектирования – применение паттерна «декоратор» [Электронный ресурс]. – Режим доступа: http://cpp-reference.ru/patterns/structural-patterns/decorator/ (дата обращения: 01.06.2019).
19. JSReport – платформа с открытым исходным кодом, использующая движки шаблонов [Электронный ресурс]. – Режим доступа: https://jsreport.net/ (дата обращения: 09.06.2019).

20. Seq – система централизованных спроектированных логов для Java и .NET [Электронный ресурс]. – Режим доступа: https://datalust.co/seq (дата обращения: 01.04.2019).
21. Уровень безработицы в России [Электронный ресурс]. – Режим доступа: https://nangs.org/analytics/rosstat-zanyatost-i-bezrabotitsa-v-rossijskoj-federatsii (дата обращения: 04.06.2019).
22. Применение метода SWOT-анализа в стратегическом управлении [Электронный ресурс]. – Режим доступа: http://powerbranding.ru/biznes-analiz/swot/ (дата обращения: 29.05.2019).
23. Интегральный финансовый анализ [Электронный ресурс]. – Режим доступа: http://afdanalyse.ru/publ/finansovyj_analiz/integralnyj_finansovyj_analiz/9-1-0-81 (дата обращения: 01.06.2019).
24. ГОСТ 12.2.0.32-78. Система стандартов безопасности труда. Рабочее место при выполнении работ сидя [Электронный ресурс]. – Режим доступа: https://internet-law.ru/gosts/gost/31970/ (дата обращения: 15.05.2019).
25. Федеральный закон, глава 2, статья 5 «О защите персональной информации.» [Электронный ресурс]. – Режим доступа: http://www.consultant.ru/document/cons_doc_LAW_61801/ (дата обращения: 16.05.2019).
26. Оптимальные и допустимые параметры воздуха [Электронный ресурс]. – Режим доступа: http://www.technoholod.ru/site.aspx?SECTIONID=1940097 (дата обращения: 16.05.2019).
27. Состояние окружающей среды. Экологическая безопасность [Электронный ресурс]. – Режим доступа: http://www.infoeco.ru/index.php?id=58 (дата обращения: 15.05.2019).
28. МЧС России. Чрезвычайные ситуации [Электронный ресурс]. – Режим доступа: https://www.mchs.gov.ru/dop/terms/item/86578/ (дата обращения: 15.05.2019).

29. Должностные инструкции при пожаре [Электронный ресурс]. – Режим доступа: https://hr-portal.ru/doki/dolzhnostnaya-instrukciya-specialista-po-pozharnoy-bezopasnosti (дата обращения: 20.05.2019).
30. Должностные инструкции при землетрясении [Электронный ресурс]. – Режим доступа: http://my.sfu-kras.ru/safety/earthquake (дата обращения: 20.05.2019).

Приложение А. Программный код абстрактного класса проверки доступа к ресурсам

```
using System;
using System.Collections.Generic;
using System.Collections.Immutable;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BuildingBlocks.Security.Abstract
{
    public abstract class AbstractAccessor
    {
        /// <summary>
        /// Метод хэндлера для проверки прав.
        /// </summary>
        protected const string HANDLER_METHOD_NAME = "HasPermissionAsync";

        /// <summary>
        /// Контейнер зависимостей.
        /// </summary>
        protected readonly IServiceProvider _serviceProvider;

        /// <summary>
        /// Словарь "тип ресурса - тип обработчика".
        /// </summary>
        protected readonly IImmutableDictionary<Type, Type> _eventHandlersDictionary;

        /// <summary>
        /// Конструктор.
        /// </summary>
        /// <param name="serviceProvider"> Контейнер зависимостей. </param>
        /// <param name="eventHandlersDictionary"> Словарь "тип ресурса - тип обработчика". </param>
        public AbstractAccessor(
            IServiceProvider serviceProvider,
            IImmutableDictionary<Type, Type> eventHandlersDictionary)
        {
            _serviceProvider = serviceProvider;
            _eventHandlersDictionary = eventHandlersDictionary;
        }

        /// <summary>
        /// Проверка прав доступа к ресурсу.
        /// </summary>
        /// <typeparam name="TEntity"> Тип ресурса. </typeparam>
        /// <param name="event"> ДТО для передачи данных о ресурсе для проверки доступа. </param>
        /// <param name="operation"> Проверяемая операция. </param>
        /// <returns> Доступность ресурса. </returns>
        public virtual async Task<bool> HasPermissionAsync<TEntity>(
            AccessEvent<TEntity> @event,
            AccessOperation operation)
            where TEntity : class
        {
            var eventType = @event.GetType();
            var hasHandler = _eventHandlersDictionary.TryGetValue(eventType, out var handlerType);

            if (!hasHandler)
                await OnHandlerNotFound(@event, operation);

            var handler = Activator.CreateInstance(handlerType);

            if (handler == null)
                return await OnHandlerNotFound(@event, operation);
```

```
                var concreteType = typeof(IAccessHandler<,>).MakeGenericType(eventType,
typeof(TEntity));
                var methodParams = new object[] { @event, operation };
                var                                result                                =
await(Task<bool>)concreteType.GetMethod(HANDLER_METHOD_NAME).Invoke(handler, methodParams);

                return result;
            }

            /// <summary>
            /// Коллбэк, вызываемый, если не найден обработчик.
            /// </summary>
            /// <typeparam name="TEntity"> Тип ресурса. </typeparam>
            /// <param name="event"> ДТО для передачи данных о ресурсе для проверки
доступа. </param>
            /// <param name="operation"> Проверяемая операция. </param>
            /// <returns> Доступность ресурса. </returns>
            protected virtual Task<bool> OnHandlerNotFound<TEntity>(
                AccessEvent<TEntity> @event,
                AccessOperation operation)
                where TEntity: class
            {
                throw new NullReferenceException($"Не зарегистрирован обработчик для " +
                        $"для сущности {typeof(TEntity).Name}");
            }
        }
    }
```

Приложение Б. Программный код менеджера обработки прав доступа

```
using BuildingBlocks.Security;
using BuildingBlocks.Security.Abstract;
using Microsoft.AspNetCore.Http;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace BuildingBlocks.Security
{
    public class SecurityManager : ISecurityManager
    {
        public HttpContext HttpContext { get; set; }

        public IAccessorFactory AccessorFactory { get; }

        public IAccessEventFactory EventFactory { get; }

        public SecurityManager(
            IAccessorFactory accessorFactory,
            IAccessEventFactory eventFactory)
        {
            AccessorFactory = accessorFactory;
            EventFactory = eventFactory;
        }

        public AbstractAccessor CreateAccessor()
        {
            return AccessorFactory.Create(HttpContext.User);
        }

        public AccessEvent<TEntity> CreateEvent<TEntity>(AccessOperation operation)
            where TEntity : class
        {
            return EventFactory.Create<TEntity>(HttpContext.User, operation);
        }

        public AccessEvent<TEntity> CreateEvent<TEntity>(TEntity entity,
AccessOperation operation)
            where TEntity : class
        {
            var @event = CreateEvent<TEntity>(operation);
            @event.Entity = entity;
            return @event;
        }

        public AccessEvent<TEntity> CreateEvent<TEntity>(IEnumerable<TEntity>
entities, AccessOperation operation)
            where TEntity : class
        {
            var @event = CreateEvent<TEntity>(operation);
            @event.EnumerableEntities = entities;
            return @event;
        }

        public AccessEvent<TEntity> CreateEvent<TEntity>(IQueryable<TEntity> entities,
AccessOperation operation)
            where TEntity : class
        {
            var @event = CreateEvent<TEntity>(operation);
            @event.QueriableEntities = entities;
            return @event;
        }

        public async Task<bool> HasPermissionAsync<TEntity>(TEntity entity,
AccessOperation operation)
            where TEntity : class
        {
```

```
                var @event = CreateEvent(entity, operation);
                var accessor = CreateAccessor();
                var allowed = await accessor.HasPermissionAsync(@event, operation);

                return allowed;
            }

            public async Task<bool> HasPermissionAsync<TEntity>(IEnumerable<TEntity>
entities, AccessOperation operation)
                where TEntity : class
            {
                var @event = CreateEvent(entities, operation);
                var accessor = CreateAccessor();
                var allowed = await accessor.HasPermissionAsync(@event, operation);

                return allowed;
            }

            public async Task<bool> HasPermissionAsync<TEntity>(IQueryable<TEntity>
entities, AccessOperation operation)
                where TEntity : class
            {
                var @event = CreateEvent(entities, operation);
                var accessor = CreateAccessor();
                var allowed = await accessor.HasPermissionAsync(@event, operation);

                return allowed;
            }
        }
    }
```

Приложение В. Программный код конфигурационного файла для развертывания веб-приложения

```
      version: '3.4'

      # The default docker-compose.override file can use the "localhost" as the external
name for testing web apps within the same dev machine.
      # The ESHOP_EXTERNAL_DNS_NAME_OR_IP environment variable is taken by default from the
".env" file defined like:
      #     ESHOP_EXTERNAL_DNS_NAME_OR_IP=localhost
      # but values present in the environment vars at runtime will always override those
defined inside the .env file
      # An external IP or DNS name has to be used (instead localhost and the 10.0.75.1 IP)
when testing the Web apps and the Xamarin apps from remote machines/devices using the same
WiFi for instance.

      services:
        sql.data:
          environment:
            - SA_PASSWORD=Pass@word
            - ACCEPT_EULA=Y
          ports:
            - "5433:1433"    # Important: In a production environment your should remove the
external port
          volumes:
            - ./mssql:/var/opt/mssql

        rabbitmq:
          ports:
            - "15672:15672"  # Important: In a production environment your should remove the
external port
            - "5672:5672"    # Important: In a production environment your should remove the
external port

        mariadb:
          environment:
            - MARIADB_USER=bn_moodle
            - MARIADB_DATABASE=bitnami_moodle
            - ALLOW_EMPTY_PASSWORD=yes
          ports:
           - '3306:3306'
          volumes:
            - ./moodle/data:/bitnami

        moodle:
          environment:
            - MARIADB_HOST=mariadb
            - MARIADB_PORT_NUMBER=3306
            - MOODLE_DATABASE_USER=bn_moodle
            - MOODLE_DATABASE_NAME=bitnami_moodle
            - ALLOW_EMPTY_PASSWORD=yes
          ports:
            - '5150:80'
            - '5151:443'
          volumes:
            - ./moodle/bitnami:/bitnami

        js_report:
          ports:
            - '5488:5488'

        # elk:
        #   environment:
        #     - MAX_MAP_COUNT=262144
        #   ports:
        #     - "5601:5601"
        #     - "9200:9200"
        #     - "5044:5044"
```

```
        #   volumes:
        #     - ./database/elk-data:/var/lib/elasticsearch

        seq:
          environment:
            - ACCEPT_EULA=Y
          ports:
            - "5340:80"

        proxy:
          environment:
            - ENABLE_SSL=true
          ports:
            - 80:80
            # - 443:443
          volumes:
            - ./nginx/nginx.conf:/etc/nginx/nginx.conf
            - ./ssl:/etc/ssl

        brandedtemplates.api:
          environment:
            - ASPNETCORE_URLS=https://0.0.0.0:80
            - ASPNETCORE_HTTPS_PORT=5101
            - Kestrel__Certificates__Default__Path=${SSL_PATH}/${SSL_CERT_NAME}
            - Kestrel__Certificates__Default__Password=${SSL_CERT_PASSWORD}
            -               ConnectionString=Server=sql.data;Database=BRANDED_TEMPLATES;User
Id=sa;Password=Pass@word
            - CorsPolicy=${CORS_POLICY}
            - IdentityUrl=https://${EXTERNAL_DNS_NAME_OR_IP}:5105
            - ElasticSearch=elk:9200
            - Seq=http://seq
          ports:
            - "5101:80"
          volumes:
            - ./ssl:${SSL_PATH}

        careerdays.api:
          environment:
            - ASPNETCORE_URLS=https://0.0.0.0:80
            - ASPNETCORE_HTTPS_PORT=5102
            - Kestrel__Certificates__Default__Path=${SSL_PATH}/${SSL_CERT_NAME}
            - Kestrel__Certificates__Default__Password=${SSL_CERT_PASSWORD}
            -                       ConnectionString=Server=sql.data;Database=CAREER_DAYS;User
Id=sa;Password=Pass@word
            - CorsPolicy=${CORS_POLICY}
            - IdentityUrl=https://${EXTERNAL_DNS_NAME_OR_IP}:5105
            - EventBusRetryCount=${EVENT_BUS_RETRY_COUNT}
            - EventBusConnection=rabbitmq
            - EventBusUserName=${EVENT_BUS_USER_NAME}
            - EventBusPassword=${EVENT_BUS_PASSWORD}
            - ElasticSearch=elk:9200
            - Seq=http://seq
          ports:
            - "5102:80"
          volumes:
            - ./ssl:${SSL_PATH}

        educationalinstitutions.api:
          environment:
            - ASPNETCORE_URLS=https://0.0.0.0:80
            - ASPNETCORE_HTTPS_PORT=5103
            - Kestrel__Certificates__Default__Path=${SSL_PATH}/${SSL_CERT_NAME}
            - Kestrel__Certificates__Default__Password=${SSL_CERT_PASSWORD}
            -         ConnectionString=Server=sql.data;Database=EDUCATIONAL_INSTITUTIONS;User
Id=sa;Password=Pass@word
            - CorsPolicy=${CORS_POLICY}
            - IdentityUrl=https://${EXTERNAL_DNS_NAME_OR_IP}:5105
            - EventBusRetryCount=${EVENT_BUS_RETRY_COUNT}
            - EventBusConnection=rabbitmq
            - EventBusUserName=${EVENT_BUS_USER_NAME}
```

```
            - EventBusPassword=${EVENT_BUS_PASSWORD}
            - ElasticSearch=elk:9200
            - Seq=http://seq
          ports:
            - "5103:80"
          volumes:
            - ./ssl:${SSL_PATH}

        employers.api:
          environment:
            - ASPNETCORE_URLS=https://0.0.0.0:80
            - ASPNETCORE_HTTPS_PORT=5104
            - Kestrel__Certificates__Default__Path=${SSL_PATH}/${SSL_CERT_NAME}
            - Kestrel__Certificates__Default__Password=${SSL_CERT_PASSWORD}
            -                       ConnectionString=Server=sql.data;Database=EMPLOYERS;User
Id=sa;Password=Pass@word
            - CorsPolicy=${CORS_POLICY}
            - IdentityUrl=https://${EXTERNAL_DNS_NAME_OR_IP}:5105
            - EventBusRetryCount=${EVENT_BUS_RETRY_COUNT}
            - EventBusConnection=rabbitmq
            - EventBusUserName=${EVENT_BUS_USER_NAME}
            - EventBusPassword=${EVENT_BUS_PASSWORD}
            - ElasticSearch=elk:9200
            - Seq=http://seq
          ports:
            - "5104:80"
          volumes:
            - ./ssl:${SSL_PATH}

        login.api:
          environment:
            - ASPNETCORE_URLS=https://0.0.0.0:80
            - ASPNETCORE_HTTPS_PORT=5105
            - Kestrel__Certificates__Default__Path=${SSL_PATH}/${SSL_CERT_NAME}
            - Kestrel__Certificates__Default__Password=${SSL_CERT_PASSWORD}
            - ConnectionString=Server=sql.data;Database=LOGIN;User Id=sa;Password=Pass@word
            - CorsPolicy=${CORS_POLICY}
            - EventBusRetryCount=${EVENT_BUS_RETRY_COUNT}
            - EventBusConnection=rabbitmq
            - EventBusUserName=${EVENT_BUS_USER_NAME}
            - EventBusPassword=${EVENT_BUS_PASSWORD}
            - ElasticSearch=elk:9200
            - Seq=http://seq
            - SymmetricKey=kAkoYto_yeb!chesk!y_k1uch
            - IdentityServiceUrl=https://${EXTERNAL_DNS_NAME_OR_IP}:5105/
            - WebAppClient=http://${EXTERNAL_DNS_NAME_OR_IP}:${FRONTEND_PORT}/
          ports:
            - "5105:80"
          volumes:
            # - ./src/Services/Domain/Login/Login.API/Certificates:/app/Certificates
            - ./ssl:${SSL_PATH}

        paidservices.api:
          environment:
            - ASPNETCORE_ENVIRONMENT=Development
            - ASPNETCORE_URLS=https://0.0.0.0:80
            - ASPNETCORE_HTTPS_PORT=5106
            - Kestrel__Certificates__Default__Path=${SSL_PATH}/${SSL_CERT_NAME}
            - Kestrel__Certificates__Default__Password=${SSL_CERT_PASSWORD}
            -                   ConnectionString=Server=sql.data;Database=PAID_SERVICES;User
Id=sa;Password=Pass@word
            - CorsPolicy=${CORS_POLICY}
            - IdentityUrl=https://${EXTERNAL_DNS_NAME_OR_IP}:5105
            - ElasticSearch=elk:9200
            - Seq=http://seq
          ports:
            - "5106:80"
          volumes:
            - ./ssl:${SSL_PATH}
```

```
        resumes.api:
          environment:
            - ASPNETCORE_ENVIRONMENT=Development
            - ASPNETCORE_URLS=https://0.0.0.0:80
            - ASPNETCORE_HTTPS_PORT=5107
            - Kestrel__Certificates__Default__Path=${SSL_PATH}/${SSL_CERT_NAME}
            - Kestrel__Certificates__Default__Password=${SSL_CERT_PASSWORD}
            -                         ConnectionString=Server=sql.data;Database=RESUMES;User
Id=sa;Password=Pass@word
            - CorsPolicy=${CORS_POLICY}
            - IdentityUrl=https://${EXTERNAL_DNS_NAME_OR_IP}:5105
            - EventBusRetryCount=${EVENT_BUS_RETRY_COUNT}
            - EventBusConnection=rabbitmq
            - EventBusUserName=${EVENT_BUS_USER_NAME}
            - EventBusPassword=${EVENT_BUS_PASSWORD}
            - ElasticSearch=elk:9200
            - Seq=http://seq
            - JsReport=http://js_report:5488
            - ExportTemplate=Resume.cshtml
            - ExportTemplatesPath=Templates
          ports:
            - "5107:80"
          volumes:
            - ./ssl:${SSL_PATH}

        vacancies.api:
          environment:
            - ASPNETCORE_ENVIRONMENT=Development
            - ASPNETCORE_URLS=https://0.0.0.0:80
            - ASPNETCORE_HTTPS_PORT=5108
            - Kestrel__Certificates__Default__Path=${SSL_PATH}/${SSL_CERT_NAME}
            - Kestrel__Certificates__Default__Password=${SSL_CERT_PASSWORD}
            -                       ConnectionString=Server=sql.data;Database=VACANCIES;User
Id=sa;Password=Pass@word
            - CorsPolicy=${CORS_POLICY}
            - IdentityUrl=https://${EXTERNAL_DNS_NAME_OR_IP}:5105
            - EventBusRetryCount=${EVENT_BUS_RETRY_COUNT}
            - EventBusConnection=rabbitmq
            - EventBusUserName=${EVENT_BUS_USER_NAME}
            - EventBusPassword=${EVENT_BUS_PASSWORD}
            - ElasticSearch=elk:9200
            - Seq=http://seq
            - JsReport=http://js_report:5488
            - ExportTemplate=Vacancy.cshtml
            - ExportTemplatesPath=Templates
          ports:
            - "5108:80"
          volumes:
            - ./ssl:${SSL_PATH}

        dictionaries.api:
          environment:
            - ASPNETCORE_ENVIRONMENT=Development
            - ASPNETCORE_URLS=https://0.0.0.0:80
            - ASPNETCORE_HTTPS_PORT=5109
            - Kestrel__Certificates__Default__Path=${SSL_PATH}/${SSL_CERT_NAME}
            - Kestrel__Certificates__Default__Password=${SSL_CERT_PASSWORD}
            -                     ConnectionString=Server=sql.data;Database=DICTIONARIES;User
Id=sa;Password=Pass@word
            - CorsPolicy=${CORS_POLICY}
            - IdentityUrl=https://${EXTERNAL_DNS_NAME_OR_IP}:5105
            - EventBusRetryCount=${EVENT_BUS_RETRY_COUNT}
            - EventBusConnection=rabbitmq
            - EventBusUserName=${EVENT_BUS_USER_NAME}
            - EventBusPassword=${EVENT_BUS_PASSWORD}
            - ElasticSearch=elk:9200
            - Seq=http://seq
          ports:
            - "5109:80"
          volumes:
```

```
    - ./src/Services/Integration/Dictionaries/Dictionaries.API/Sources:/app/Sources
    - ./ssl:${SSL_PATH}

moodle.api:
  environment:
    - ASPNETCORE_ENVIRONMENT=Development
    - ASPNETCORE_URLS=https://0.0.0.0:80
    - ASPNETCORE_HTTPS_PORT=5111
    - Kestrel__Certificates__Default__Path=${SSL_PATH}/${SSL_CERT_NAME}
    - Kestrel__Certificates__Default__Password=${SSL_CERT_PASSWORD}
    - CorsPolicy=${CORS_POLICY}
    - IdentityUrl=https://${EXTERNAL_DNS_NAME_OR_IP}:5105
    - EventBusRetryCount=${EVENT_BUS_RETRY_COUNT}
    - EventBusConnection=rabbitmq
    - EventBusUserName=${EVENT_BUS_USER_NAME}
    - EventBusPassword=${EVENT_BUS_PASSWORD}
    - ElasticSearch=elk:9200
    - MoodleRestUrl=moodle/webservice/rest/server.php
    - MoodleToken=f3f5be086fd0b9cf4ed3d09a3fdf1da0
    - MoodleRestFormat=json
    - Seq=http://seq
  ports:
    - "5111:80"
  volumes:
    - ./ssl:${SSL_PATH}

emailnotifications.api:
  environment:
    - ASPNETCORE_ENVIRONMENT=Development
    - ASPNETCORE_URLS=https://0.0.0.0:80
    - ASPNETCORE_HTTPS_PORT=5110
    - Kestrel__Certificates__Default__Path=${SSL_PATH}/${SSL_CERT_NAME}
    - Kestrel__Certificates__Default__Password=${SSL_CERT_PASSWORD}
    - CorsPolicy=${CORS_POLICY}
    - EventBusRetryCount=${EVENT_BUS_RETRY_COUNT}
    - EventBusConnection=rabbitmq
    - EventBusUserName=${EVENT_BUS_USER_NAME}
    - EventBusPassword=${EVENT_BUS_PASSWORD}
    - ElasticSearch=elk:9200
    - Seq=http://seq
  ports:
    - "5110:80"
  volumes:
    - ./ssl:${SSL_PATH}

apigw:
  environment:
    - ASPNETCORE_ENVIRONMENT=Development
    - ASPNETCORE_URLS=https://0.0.0.0:80
    - ASPNETCORE_HTTPS_PORT=5200
    - Kestrel__Certificates__Default__Path=${SSL_PATH}/${SSL_CERT_NAME}
    - Kestrel__Certificates__Default__Password=${SSL_CERT_PASSWORD}
    - CorsPolicy=${CORS_POLICY}
    - IdentityUrl=https://${EXTERNAL_DNS_NAME_OR_IP}:5105
    - ElasticSearch=elk:9200
    - Seq=http://seq
  ports:
    - "5200:80"
  volumes:
    - ./src/ApiGateways/WebApp.Bff/apigw:${OCELOT_VOLUME_SPEC:-/app/configuration}
    - ./ssl:${SSL_PATH}
```

Приложение Г. Программный код Razor Page для экспорта пользовательских данных в PDF-формат

```
@using System.Collections.Generic
@model IDictionary<string, string>

@{
    const string TITLE_KEY = "Должность";
    const string FIO_KEY = "ФИО";
    const string DATE_KEY = "Дата рождения";
    const string EMAIL_KEY = "Email";
    var excludedFields = new[] { TITLE_KEY, FIO_KEY, DATE_KEY, EMAIL_KEY };
}

<style>
    .content, .table {
        display: flex;
        width: 100%;
        flex-direction: column;
        align-content: center;
    }

    .body {
        display: flex;
        width: 100%;
        flex-direction: column;
        align-content: center;
    }

    .header, .applicant_info {
        display: flex;
        width: 100%;
        flex-direction: column;
        align-content: center;
    }

    .applicant_info > span, .header {
        padding-bottom: 15px
    }

    .row {
        display: flex;
        width: 100%;
        flex-direction: row;
        justify-content: space-between
    }

    .row:nth-child(2n+1) {
        background-color: lightgray;
    }

    .key {
        display: flex;
        width: 40%;
        justify-content: center;
        align-content: center;
    }

    .value {
        display: flex;
        width: 60%;
        border-left: 1px solid black;
        justify-content: center;
        align-content: center;
    }

    .key > span, .value > span {
        padding: 5px;
```

```
    }

</style>

    <div class="content">
        <div class="header">
            <h1>
                @Model[TITLE_KEY]
            </h1>
        </div>
        <div class="body">
            <div class="applicant_info">
                <span>@Model[FIO_KEY]</span>
            </div>
            <div class="applicant_info">
                <span>@Model[DATE_KEY]</span>
            </div>
            <div class="applicant_info">
                <span>@Model[EMAIL_KEY]</span>
            </div>
            <div class="table">
                @foreach (var pair in Model)
                {
                    @if (excludedFields.Contains(pair.Key))
                    {
                        continue;
                    }
                    <div class="row">
                        <div class="key">
                            <span>@pair.Key</span>
                        </div>
                        <div class="value">
                            <span>@pair.Value</span>
                        </div>
                    </div>
                }
            </div>
        </div>
    </div>
```