**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

****

**Факультет Информационных технологий**

**кафедра Информатики и информационных технологий**

**направление подготовки 09.03.02 «Информационные системы и технологии»**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**БАКАЛАВРСКАЯ РАБОТА**

**Тема:** Разработка информационной системы для автоматизированного

подбора кандидатов на должность backend-разработчика

**Исполнитель** Махлышев Матвей Дмитриевич

(Фамилия И.О.) (Подпись)

**Руководитель**  Винокурова О. А. к.т.н., доцент

(Фамилия И.О., степень, звание) (Подпись)

**«ДОПУЩЕНО К ЗАЩИТЕ»**

**Зав. кафедрой ИиИТ: к.т.н. Е.В. Булатников**

(Подпись)

**Москва**

**2025**

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

****

**Факультет Информационных технологий**

**кафедра Информатики и информационных технологий**

**направление подготовки 09.03.02 «Информационные системы и технологии»**

УТВЕРЖДАЮ

Зав. каф., к.т.н. Е.В. Булатников

« \_\_\_ » \_\_\_\_\_\_\_\_\_\_ 2025 г. ***\_\_\_\_\_\_\_\_\_\_\_\_\_***

(Подпись)

**ЗАДАНИЕ НА БАКАЛАВРСКУЮ РАБОТУ**

**Студент**  Махлышев Матвей Дмитриевич **группа 211-722**

(Фамилия, Имя, Отчество)

**Тема** Разработка информационной системы для автоматизированного

подбора кандидатов на должность backend-разработчика

**Утверждена приказом по Университету №** 230-УД от « 22 » января 2025 г.

1. **Срок представления работы к защите** « » июня 2025 г.
2. **Исходные данные для выполнения работы:** техническое задание,

техническая литература по ВКР, существующие решения по подбору кандидатов на должность backend-специалистов

1. **Содержание бакалаврской работы:** введение, аналитическая часть, проектирование, реализация ИС, заключение, приложения, выводы по главам

1. **Перечень графического материала** презентационные материалы в приложении, диаграммы
2. **Дата выдачи задания:** « 14 » января 2025 г.
3. **Руководитель:**  / О.А. Винокурова /

(Подпись) (И.О. Фамилия)

1. **Задание к исполнению принял:**  / М.Д. Махлышев /

(Подпись) (И.О. Фамилия)

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

****

**Факультет Информационных технологий**

**кафедра Информатики и информационных технологий**

**направление подготовки 09.03.02 «Информационные системы и технологии»**

**КАЛЕНДАРНЫЙ ПЛАН**

**Студент** Махлышев Матвей Дмитриевич **группа 211-** **722**

(Фамилия, Имя, Отчество)

**Тема ВКР:** Разработка информационной системы для автоматизированного

подбора кандидатов на должность backend-разработчика

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапа ВКР | Срок выполнения этапа | Отметка о выполнении |
|  | Анализ предметной области | 10.05.2025 | Выполнено |
|  | Анализ существующих решений | 15.05.2025 | Выполнено |
|  | Анализ технической литературы |  |  |
|  | Разработка технического задания |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Руководитель**  Винокурова О. А. к.т.н.

(Фамилия И.О., степень, звание) (Подпись)

**Зав. каф. ИиИТ** /Е.В. Булатников/

(Подпись)

« » 2025 г.

**Аннотация**

В данной выпускной квалификационной работе рассматривается идея создания информационной системы, которая предназначена для автоматизации подбора кандидатов на должность backend-разработчика. Основная функция системы заключается в сопоставлении профессиональных компетенций соискателей с требованиями вакансии с последующей автоматизированной проверкой их технических знаний.

Структура работы представлена введением, тремя главами, заключением, списком литературы и приложением.

В первой главе проведен анализ современных методов оценки IT-специалистов и существующих автоматизированных решений.

Во второй главе производится проектирование информационной системы.

Третья глава содержит описание процесса разработки и тестирования рабочего прототипа.

**Abstract**

This graduation thesis explores the concept of developing an information system designed to automate the recruitment process for backend developer positions. The system's primary function involves matching candidates' professional competencies with job requirements, followed by automated verification of their technical knowledge.

The work is structured with an introduction, three main chapters, conclusion, bibliography, and appendices.

The first chapter provides an analysis of contemporary assessment methods for IT professionals and existing automated solutions.

The second chapter focuses on designing the information system.

The third chapter describes the development process and testing of a working prototype.

**Реферат**

Объём пояснительной записки состоит из x страниц, на x страницах приложений, x рисунков, x таблиц, x листингов. При написании дипломной работы использовалось x источников.

Тема: Разработка информационной системы для автоматизированного

подбора кандидатов на должность backend-разработчика.

Ключевые слова: ~~информационная система~~, python, backend, FastAPI, тестирование.(о проекте)

Оглавление

[**Введение** 8](#_Toc198288993)

[**Глава 1 Анализ концепта проекта** 11](#_Toc198288994)

[**1.1.** **Анализ предметной области** 12](#_Toc198288995)

[**1.2.** **Анализ существующих решений** 15](#_Toc198288996)

[**1.3.** **Требования к системе** 17](#_Toc198288997)

[**1.4.** **Ожидаемые результаты и эффективность** 18](#_Toc198288998)

[**1.5.** **Риски и ограничение** 19](#_Toc198288999)

[**Вывод по главе** 20](#_Toc198289000)

[**Глава 2 Проектирование информационной системы** 21](#_Toc198289001)

[**2.1.** **Логическое моделирование предметной области** 22](#_Toc198289004)

[**2.1.1.** **Выбор средств описания логической модели** 23](#_Toc198289005)

[**2.2.** **Физическое моделирование системы** 32](#_Toc198289006)

[**2.3.** **Проектирование базы данных** 44](#_Toc198289007)

[**2.3.1 Общая характеристика предметной области** 45](#_Toc198289008)

[**2.3.2 Подход к проектированию базы данных** 45](#_Toc198289009)

[**2.3.3 Список сущностей и их функциональное назначение** 46](#_Toc198289010)

[**2.3.4 Структура и типы связей между сущностями** 47](#_Toc198289011)

[**2.3.5 Обоснование выбора атрибутов и типов данных** 48](#_Toc198289012)

[**2.3.6 Хранение вложенных структур** 50](#_Toc198289013)

[**2.3.7 Обеспечение целостности данных** 50](#_Toc198289014)

[**2.3.8 Масштабируемость и производительность** 50](#_Toc198289015)

[**2.3.9 Интеграция с backend и API** 51](#_Toc198289016)

[**2.3.10 Перспективы расширения модели данных** 51](#_Toc198289017)

[**2.4.** **Требования к аппаратно-программному обеспечению информационной системы** 52](#_Toc198289018)

[**Вывод по главе** 56](#_Toc198289019)

[**Глава 3 Процесс разработки информационной системы** 56](#_Toc198289020)

[**3.1. Введение и постановка задачи** 56](#_Toc198289021)

[**3.2. Сбор и анализ требований** 57](#_Toc198289022)

[**3.3. Проектирование архитектуры системы** 58](#_Toc198289023)

[**3.4. Выбор технологического стека и инструментов разработки** 59](#_Toc198289024)

[**3.5. Разработка и реализация функционала** 59](#_Toc198289025)

[**3.6. Тестирование и обеспечение качества** 60](#_Toc198289026)

[**3.7. Обеспечение безопасности** 61](#_Toc198289027)

[**3.8. Развертывание и поддержка** 61](#_Toc198289028)

[**Вывод по главе** 62](#_Toc198289029)

[**Заключение** 62](#_Toc198289030)

[**Список литературы** 66](#_Toc198289031)

# **Введение**

Современная цифровая экономика предъявляет все более высокие требования к качеству и скорости разработки программного обеспечения, что в свою очередь значительно повышает значимость квалифицированных backend-разработчиков на рынке труда. Backend-разработка, являясь фундаментальной основой любого IT-продукта, требует от специалистов глубоких знаний в области серверных технологий, архитектурных решений, баз данных и системной интеграции. Однако процесс поиска и оценки таких специалистов сталкивается с рядом существенных проблем, которые требуют инновационных решений.

Основная сложность современного рекрутинга backend-разработчиков заключается в стремительном усложнении технологического стека и появлении новых парадигм разработки. Если еще несколько лет назад для успешного прохождения отбора достаточно было демонстрировать знание одного из популярных языков программирования и баз данных, то сегодня от кандидатов ожидают понимания микросервисной архитектуры, работы с облачными платформами, навыков контейнеризации и оркестрации, а также опыта в построении распределенных систем. Такое многообразие требований делает традиционные методы оценки крайне трудоемкими и неэффективными.

Особенно остро стоит проблема оценки реальных практических навыков кандидатов. Традиционные технические собеседования часто оказываются необъективными, так как сильно зависят от личного мнения интервьюера и не всегда отражают реальную квалификацию соискателя. При этом автоматизированные системы тестирования, существующие на рынке, обычно ограничиваются проверкой знания синтаксиса языков программирования и не оценивают архитектурные навыки, понимание паттернов проектирования и способность разрабатывать масштабируемые решения.

Еще одной значимой проблемой является временной фактор. По данным отраслевых исследований, средний срок закрытия вакансии backend-разработчика составляет от 4 до 8 недель, при этом до 60% этого времени тратится на первичный отбор и оценку кандидатов. В условиях высокой конкуренции за квалифицированные кадры такие временные затраты становятся критическими для бизнеса, особенно для стартапов и быстрорастущих компаний.

В связи с этим разработка специализированной информационной системы для автоматизированного подбора backend-разработчиков приобретает особую актуальность. Такая система должна решать следующие ключевые задачи:

* интеллектуальный анализ профессиональных профилей кандидатов с извлечением полного технологического стека.
* многоуровневая оценка навыков, включающая:
* проверку знания языков программирования и фреймворков;
* анализ понимания архитектурных принципов;
* оценку опыта работы с базами данных;
* тестирование навыков системной интеграции.
* автоматизированное тестирование с использованием реальных рабочих задач.
* генерация комплексных отчетов для HR-специалистов и технических руководителей.

Целью данной работы является создание комплексного решения, которое позволит:

* сократить время первичного отбора кандидатов;
* повысить объективность оценки профессиональных качеств;
* уменьшить нагрузку на технических специалистов, участвующих в процессе рекрутинга;
* обеспечить стандартизированный подход к оценке кандидатов;
* снизить процент ошибок при найме.

Для достижения поставленной цели необходимо решить ряд исследовательских и практических задач:

* провести детальный анализ современных методов оценки IT-специалистов, выявив их сильные и слабые стороны.
* разработать критерии оценки, максимально полно отражающие требования к backend-разработчикам.
* создать алгоритмы анализа технологического стека кандидатов с учетом контекста использования технологий.
* реализовать систему автоматизированного тестирования практических навыков.
* разработать механизмы интеграции с популярными HR-платформами и профессиональными сообществами.
* спроектировать интуитивно понятный интерфейс для взаимодействия рекрутеров с системой.
* провести всестороннее тестирование системы на реальных данных.

Практическая значимость работы заключается в создании инструмента, который сможет существенно оптимизировать процесс подбора backend-разработчиков, сделав его более быстрым, объективным и эффективным. Внедрение такой системы позволит компаниям:

* значительно сократить затраты на рекрутинг;
* повысить качество принимаемых кадровых решений;
* уменьшить зависимость от субъективных оценок;
* ускорить процесс адаптации новых сотрудников;
* получить конкурентное преимущество на рынке труда.

Теоретическая ценность исследования состоит в разработке новых подходов к автоматизированной оценке профессиональных компетенций, которые могут быть применены и для других технических специальностей. Предложенные методы анализа технологического стека и оценки практических навыков представляют интерес для дальнейших исследований в области HR-технологий и искусственного интеллекта.

# **Анализ концепта проекта**

Данная глава посвящена основам разработки системы для автоматического подбора backend-разработчиков. В условиях быстрого роста требований к таким специалистам старые методы оценки кандидатов показывают свои ограничения, поэтому нужно находить свежие решения для честного анализа их профессиональных навыков.

В исследовании сосредоточены три главные задачи:

* выявление актуальных требований к backend-разработчикам, включая как базовые технические навыки, так и умение проектировать сложные решения.
* создание системы критериев оценки, чтобы точно определить уровень подготовки кандидатов.
* разработка принципов создания автоматизированной системы, которая может анализировать и интерпретировать данные о профессиональных качествах соискателей.

Особое внимание уделяется проблеме объективности оценки. Традиционные методы подбора часто зависят от субъективных факторов и не всегда отражают, насколько кандидат способен решать реальные задачи. Поэтому мы предлагаем новые подходы, которые обеспечивают стандартизированную и беспристрастную оценку.

Большая часть исследования посвящена разработке многоуровневой системы оценки, которая включает:

* анализ основных технических навыков
* оценку способности проектировать архитектурные решения
* проверку работы с различными типами данных
* тестирование навыков интеграции разных систем

Практическая значимость работы заключается в создании основ для системы подбора, которая может повысить эффективность технического рекрутинга. Разработанные подходы помогут сократить время на начальный выбор кандидатов и улучшить точность оценки их профессиональных качеств.

Результаты анализа дадут представление о современных требованиях к backend-разработчикам и создают базу для проектирования автоматизированной системы подбора. Эти выводы будут использованы в следующих разделах для разработки конкретных технических решений, направленных на улучшение оценки профессиональных навыков.

## **Анализ предметной области**

Анализ предметной области современной backend-разработки требует детального рассмотрения нескольких ключевых аспектов, которые формируют комплекс требований к квалификации специалистов. В данном разделе проводится всестороннее исследование технологических и методологических особенностей backend-разработки, что позволит в дальнейшем разработать эффективную систему оценки компетенций кандидатов.

Современная backend-разработка представляет собой сложную многоуровневую дисциплину, которая за последние годы претерпела значительные изменения. Если ранее основное внимание уделялось собственно написанию серверного кода, то сегодня требования к разработчикам существенно расширились. Это связано с повсеместным переходом на микросервисные архитектуры, распространением облачных технологий и необходимостью создания высоконагруженных отказоустойчивых систем.

Основные технологические направления современной backend-разработки можно структурировать по уровням компетенций:

На фундаментальном уровне специалист должен обладать глубокими знаниями языков программирования. Наиболее востребованными в настоящее время являются Java, Python, Go и C#. При этом важно понимание не только синтаксиса языка, но и особенностей работы runtime-среды, системы типов, механизмов управления памятью. Не менее значимым является владение популярными фреймворками, такими как Spring Boot для Java, Django и FastAPI для Python, ASP.NET Core для C#. Эти знания составляют базовый уровень, без которого невозможно дальнейшее профессиональное развитие.

Следующий уровень компетенций связан с работой с данными. Современный backend-разработчик должен свободно ориентироваться в различных подходах к хранению и обработке информации. Это включает в себя: глубокое понимание реляционных баз данных (PostgreSQL, MySQL) с умением оптимизировать запросы и проектировать эффективные схемы; знание NoSQL-решений (MongoDB, Cassandra, Redis) и понимание сценариев их применения; опыт работы с системами кэширования; владение технологиями потоковой обработки данных. Особое значение приобретает понимание принципов согласованности данных в распределённых системах.

Архитектурный уровень компетенций требует от специалиста способности проектировать масштабируемые и поддерживаемые решения. Это предполагает знание основных паттернов проектирования (Factory, Strategy, Observer и др.), понимание различий между монолитной и микросервисной архитектурами, умение применять принципы чистой архитектуры и доменно-ориентированного проектирования. В условиях перехода к распределённым системам особую важность приобретают навыки проектирования межсервисного взаимодействия и обеспечения отказоустойчивости.

Системный уровень компетенций включает в себя понимание принципов работы инфраструктуры: контейнеризации и оркестрации (Docker, Kubernetes), механизмов мониторинга и логирования, систем непрерывной интеграции и доставки (CI/CD), а также аспектов информационной безопасности. Современный разработчик должен уметь не только писать код, но и понимать, как он будет работать в production-среде.

Проведённый анализ выявил ряд существенных проблем в существующих подходах к оценке компетенций backend-разработчиков. Основная сложность заключается в разрыве между теоретическими знаниями и практическими навыками. Кандидаты часто демонстрируют хорошее знание синтаксиса языков программирования, но испытывают трудности при проектировании реальных систем или оптимизации существующих решений. Другой значимой проблемой является быстрая смена технологического ландшафта - конкретные фреймворки и инструменты устаревают, в то время как фундаментальные принципы остаются актуальными значительно дольше.

Существенные сложности вызывает оценка системного мышления разработчика - его способности видеть взаимосвязи между компонентами системы, предвидеть возможные проблемы масштабирования, проектировать решения с учётом возможных изменений требований. Традиционные методы оценки часто оказываются неэффективными для проверки этих качеств.

На основании проведённого анализа можно сформулировать основные требования к системе автоматизированной оценки backend-разработчиков. Она должна обеспечивать многоуровневую проверку компетенций - от базовых технических знаний до архитектурных навыков и системного мышления. Важным аспектом является возможность анализа не только теоретических знаний, но и практических навыков работы с кодом, включая его качество, соответствие принципам clean code и эффективность алгоритмов.

Система должна включать механизмы для оценки способности разработчика решать комплексные задачи, аналогичные тем, с которыми он столкнётся в реальной работе. Это предполагает использование ситуационных кейсов, моделирующих типичные проблемы проектирования и оптимизации backend-систем. Особое внимание следует уделить проверке умения работать с legacy-кодом и проводить его рефакторинг.

Важным требованием является адаптивность системы к быстро меняющимся технологическим трендам. При этом акцент должен делаться на оценке фундаментальных знаний и принципов, а не владения конкретными инструментами или фреймворками.

Результаты проведённого анализа предметной области позволяют перейти к следующему этапу работы - разработке конкретных критериев и методов оценки, которые будут учитывать все выявленные аспекты backend-разработки и позволят создать эффективную систему автоматизированного подбора специалистов.

## **Анализ существующих решений**

1. Основные инструменты оценки на платформе

HH.ru как крупнейшая российская рекрутинговая платформа предлагает ограниченный набор инструментов для технической оценки кандидатов:

1. профиль соискателя:

* резюме с указанием технологического стека
* опыт работы (компании, должности, сроки)
* образование и сертификации
* портфолио проектов (опционально)

1. тестовые задания:

* встроенный конструктор тестов
* возможность загрузки практических задач
* ограниченный набор шаблонов для IT-специалистов

1. интеграции:

* ссылки на сторонние портфолио

1. Ограничения платформы

Для оценки backend-разработчиков HH.ru имеет существенные недостатки:

1. поверхностный анализ навыков:

* нет автоматизированной проверки технических компетенций
* оценка основана на текстовом описании в резюме
* невозможность проверить реальный уровень владения технологиями

1. проблемы с верификацией:

* нет инструментов проверки подлинности указанного опыта
* сложность оценки реального вклада в проекты
* отсутствие связи между заявленными навыками и практическими умениями

1. недостатки для технических специалистов:

* отсутствие специализированных метрик для IT-сферы
* примитивная система фильтрации по технологическому стеку

Таблица 1 Сравнение с профильными решениями

|  |  |  |
| --- | --- | --- |
| **Критерий** | **HH.ru** | **Специализированные платформы** |
| Оценка кода | Нет | Автоматизированная проверка |
| Практические задания | Примитивные | Реалистичные проектные задачи |
| Технические тесты | Базовые | Многоуровневые специализированные |
| Проверка архитектурных навыков | Нет | Комплексная оценка |

1. Возможности улучшения

Для эффективной оценки backend-разработчиков HH.ru требуется:

* интеграция с системами автоматизированного тестирования кода
* разработка специализированных шаблонов для IT-рекрутинга
* внедрение механизмов проверки практических навыков
* добавление многоуровневых технических тестов
* углублённый анализ активности на GitHub/GitLab

HH.ru в текущем виде не является эффективным инструментом для технической оценки backend-разработчиков. Платформа ориентирована преимущественно на общий рекрутинг и не предоставляет специализированных возможностей для проверки профессиональных компетенций в области backend-разработки. Для полноценной оценки кандидатов требуется интеграция со сторонними специализированными системами или разработка дополнительных модулей.

## **Требования к системе**

Основной функционал системы будет сосредоточен на автоматизированной обработке анкетных данных и последующем многоуровневом тестировании кандидатов на позиции backend-разработчиков. Процесс оценки состоит из двух ключевых этапов: автоматического анализа заполненных анкет и комплексного тестирования профессиональных навыков.

На первом этапе система анализирует структурированные данные, введенные кандидатом в специальную анкету. Анализ включает:

* обработку информации о технологическом стеке с указанием уровня владения каждой технологией
* оценку профессионального опыта на основе введенных данных о проектах и должностях
* проверку соответствия базовым требованиям вакансии

Второй этап представляет собой автоматизированное тестирование, которое проводится для кандидатов, прошедших первичный отбор.

Тестирование включает:

* проверку базовых технических навыков через решение стандартизированных задач
* оценку архитектурного мышления с помощью кейсовых заданий
* анализ практических умений через выполнение приближенных к реальности проектных задач
* прохождение многоуровневых тестов

Система отчетности автоматически формирует:

* оценку соответствия требованиям вакансии
* анализ сильных и слабых профессиональных сторон
* рекомендации по направлениям развития
* прогноз успешности работы в конкретной среде

Интеграционные возможности системы позволяют:

* автоматизировать весь цикл подбора от анализа анкет до финального рейтинга
* минимизировать человеческий фактор в процессе оценки
* сохранять гибкость для настройки под конкретные вакансии
* обеспечивать стандартизацию процесса отбора

Система разрабатывается с акцентом на объективность и прозрачность оценки, что позволит значительно повысить эффективность подбора backend-разработчиков.

## **Ожидаемые результаты и эффективность**

Внедрение автоматизированной системы тестирования backend-разработчиков позволит существенно оптимизировать процесс технического рекрутинга. Прежде всего, система обеспечит значительное сокращение временных затрат - время первичного отбора кандидатов уменьшится на 60-70%, при этом нагрузка на технических специалистов, участвующих в процессе оценки, снизится в разы. Автоматизация рутинных операций даст возможность HR-специалистам сосредоточиться на стратегических аспектах подбора персонала.

Ключевым преимуществом системы станет повышение объективности и качества оценки. Стандартизированный подход устранит субъективный фактор при первичном отборе, обеспечив единые критерии оценки для всех кандидатов. Комплексная проверка профессиональных компетенций позволит выявлять реальный уровень знаний соискателей, а не только их умение эффектно презентовать себя. Это приведет к увеличению процента успешных наймов и снижению текучести кадров на технических позициях.

Экономический эффект от внедрения системы проявится в сокращении стоимости процесса подбора на за счет оптимизации ресурсов. Система также позволит сформировать объективную шкалу уровней компетенций, что упростит планирование карьерного роста и обучение сотрудников.

Важным организационным преимуществом станет создание единого стандарта оценки для всех филиалов и подразделений компании. Накопленные данные о результатах тестирования можно будет использовать для анализа динамики развития технических навыков в компании, выявления слабых мест в подготовке специалистов и совершенствования процессов обучения. В долгосрочной перспективе это приведет к повышению общего уровня технической экспертизы в организации.

## **Риски и ограничение**

При внедрении системы автоматизированного тестирования backend-разработчиков следует учитывать ряд потенциальных рисков и ограничений:

1. технические ограничения:

* возможны сложности с оценкой soft skills и командных навыков разработчика, которые традиционно выявляются в процессе личного общения.
* автоматизированная система может испытывать трудности с интерпретацией нестандартных, но правильных решений тестовых задач.

1. методологические риски:

* cуществует вероятность создания "натаскиваемых" тестов, когда кандидаты запоминают правильные ответы, не обладая реальными знаниями.
* cистема может демонстрировать предвзятость в пользу определенных технологических стеков или подходов к решению задач.
* возможна недооценка кандидатов с нестандартным мышлением, но высоким потенциалом к обучению.

1. организационные риски:

* необходимость постоянного обновления тестовой базы для соответствия актуальным технологическим трендам.
* зависимость качества оценки от корректности составления тестовых заданий.

1. ограничения эффективности:

* система не может полностью заменить техническое интервью, а лишь выступает инструментом первичного отбора.
* требуется дополнительная проверка для кандидатов пограничного уровня (между junior и middle, middle и senior).
* эффективность системы напрямую зависит от качества и релевантности тестовых заданий.

## **Выводы по главе**

Разработанная концепция автоматизированной системы оценки backend-разработчиков представляет собой сбалансированное решение, сочетающее технологические возможности и практические потребности современного IT-рекрутинга. Анализ показал, что система способна обеспечить объективную и стандартизированную оценку профессиональных компетенций за счет комплексного подхода к тестированию. Ключевым преимуществом является устранение субъективного фактора на этапе первичного отбора и создание единых критериев для всех кандидатов.

Особую ценность системе придает ее специализация именно на backend-разработке, что позволяет глубже оценивать специфические навыки, такие как работа с базами данных, проектирование API и понимание серверных технологий. При этом сохраняется гибкость для адаптации под различные технологические стеки и уровни позиций (от junior до senior).

Следует отметить, что система не претендует на полную замену традиционных методов оценки, а скорее оптимизирует начальные этапы отбора, позволяя техническим специалистам сосредоточиться на наиболее перспективных кандидатах. Ее внедрение создаст основу для более эффективного распределения ресурсов в процессе подбора персонала.

Перспективы развития системы связаны с возможностями ее интеграции с существующими HR-процессами и платформами, а также с постоянным совершенствованием тестовых методик. Важным направлением является накопление и анализ данных для последующего улучшения критериев оценки и выявления новых тенденций в требованиях к backend-разработчикам

# **Проектирование информационной системы**

Проектирование информационной системы является ключевым этапом разработки, определяющим её архитектуру, функциональность, производительность и масштабируемость. В данной главе рассматриваются методологические подходы к проектированию ИС, включая анализ требований, выбор технологий, моделирование данных и проектирование пользовательских интерфейсов.

Актуальность проектирования заключается в необходимости создания эффективного, надежного и удобного программного решения, соответствующего потребностям заказчика. Грамотное проектирование позволяет минимизировать риски на этапе реализации, оптимизировать вычислительные ресурсы и обеспечить удобство сопровождения системы в будущем.

Целью главы является разработка архитектуры информационной системы на основе проведенного ранее анализа требований. Для достижения этой цели решаются следующие задачи:

* определение структурных компонентов системы и их взаимодействия.
* выбор методологии проектирования
* разработка концептуальной, логической и физической модели данных.
* проектирование пользовательских интерфейсов и бизнес-логики.
* обоснование выбора технологий

Методы проектирования, используемые в работе, включают:

* визуальное моделирование
* нормализацию базы данных для минимизации избыточности.
* прототипирование интерфейсов для оценки удобства использования.

Результатом проектирования станут:

* детализированные схемы архитектуры системы.
* нормализованная модель базы данных.
* макеты пользовательских интерфейсов.

Полученные проектные решения лягут в основу реализации ИС в следующей главе, обеспечивая согласованность функциональных модулей и эффективность их взаимодействия.



## **Архитектура информационной системы**

Современные информационные системы представляют собой сложные программно-аппаратные комплексы, эффективность которых определяется не только их функциональностью, но и продуманной архитектурой. Архитектура информационной системы представляет собой фундаментальную организацию системы, воплощенная в её компонентах, их взаимодействии между собой и с внешней средой, а также принципах проектирования и развития. В условиях стремительного роста объемов данных и требований к производительности, грамотно спроектированная архитектура становится критически важным фактором успеха. Она обеспечивает масштабируемость системы при увеличении нагрузки, гарантирует отказоустойчивость и бесперебойную работу, оптимизирует производительность и скорость обработки данных, упрощает поддержку и развитие системы в будущем, а также обеспечивает безопасность данных и соответствие отраслевым стандартам.

На сегодняшний день существует несколько основных подходов к проектированию архитектуры информационных систем, каждый из которых имеет свои преимущества и области применения. Монолитная архитектура, традиционно используемая в корпоративных решениях, отличается простотой разработки и развертывания, но может столкнуться с проблемами масштабируемости. Микросервисная архитектура, получившая широкое распространение в последние годы, обеспечивает высокую гибкость и масштабируемость за счет декомпозиции системы на независимые сервисы. Событийно-ориентированная архитектура оптимальна для систем с интенсивным потоком событий и асинхронной обработкой данных, тогда как сервис-ориентированная архитектура остается востребованной в крупных корпоративных средах с необходимостью интеграции разнородных систем.

Проектирование архитектуры информационной системы начинается с анализа бизнес-требований и определения ключевых функциональных возможностей. На этом этапе важно четко определить границы системы, основные потоки данных и взаимодействие с внешними системами. Следующим шагом является выбор архитектурного стиля и технологического стека, которые наилучшим образом соответствуют поставленным задачам. Особое внимание уделяется вопросам масштабируемости - система должна быть способна обрабатывать возрастающую нагрузку без снижения производительности. Не менее важны аспекты безопасности, включая защиту данных, аутентификацию и авторизацию пользователей, а также защиту от внешних угроз.

В рамках проектирования архитектуры также рассматриваются вопросы отказоустойчивости и восстановления после сбоев. Современные системы должны быть спроектированы таким образом, чтобы обеспечить непрерывность бизнес-процессов даже в случае частичных отказов оборудования или программного обеспечения. Это достигается за счет репликации данных, кластеризации сервисов и реализации механизмов автоматического восстановления. Важным аспектом является и проектирование системы мониторинга, которая позволяет оперативно выявлять и устранять проблемы в работе системы.

Особую сложность представляет проектирование архитектуры для систем, работающих с большими объемами данных. В таких случаях необходимо тщательно продумывать стратегии хранения, индексирования и обработки данных, чтобы обеспечить приемлемую производительность при минимальных задержках. Использование распределенных вычислений, кэширования и оптимизированных структур данных позволяет решить эти задачи. При этом важно соблюдать баланс между производительностью, стоимостью владения и сложностью реализации.

Интеграционные аспекты архитектуры играют ключевую роль в современных информационных системах, которым необходимо взаимодействовать с множеством внешних сервисов и платформ. Проектирование четких и стандартизированных интерфейсов взаимодействия, использование API-шлюзов и шин предприятития позволяет создать гибкую и расширяемую систему. Особое внимание уделяется вопросам совместимости и миграции данных, особенно при модернизации унаследованных систем.

### **Контекстная диаграмма**

Контекстная диаграмма представляет собой фундаментальный инструмент системного проектирования, который позволяет определить границы разрабатываемой системы и её взаимодействие с внешней средой. На этой диаграмме система изображается как единый блок (так называемый "чёрный ящик"), вокруг которого располагаются все внешние сущности - пользователи, смежные системы, сторонние сервисы и API. Главная ценность этой диаграммы заключается в том, что она даёт всем участникам проекта - заказчикам, бизнес-аналитикам, разработчикам - единое и непротиворечивое понимание того, что именно входит в разрабатываемую систему, а что остаётся за её пределами.

При создании контекстной диаграммы важно чётко определить:

* кто является основными пользователями системы
* какие внешние системы будут интегрироваться с нашей разработкой
* какие данные и команды будут передаваться через границы системы
* какие протоколы и интерфейсы будут использоваться для взаимодействия

Эта диаграмма особенно полезна на старте проекта, когда необходимо согласовать с заказчиком базовые требования и границы системы. Она помогает избежать распространённых ошибок, когда разные участники проекта по-разному понимают масштабы разработки. Также контекстная диаграмма служит отправной точкой для создания более детальных архитектурных схем на следующих уровнях проектирования.

На рисунке 2.1 изображена контекстная диаграмма проектируемой информационной системы.



Рисунок 2.1 — Контекстная диаграмма

На диаграмме представлена информационная система автоматизированного подбора кандидатов на роль backend-разработчика. Это комплексное решение, предназначенное для оптимизации процесса поиска и оценки специалистов в области backend-разработки. Система помогает рекрутерам эффективно обрабатывать входящие заявки, отбирая наиболее подходящих кандидатов на основе заданных критериев.

Основу системы составляют четко определенные параметры отбора, которые включают несколько ключевых аспектов. Учитываются технические навыки кандидата. Важную роль играет образовательная составляющая - наличие профильного образования, дополнительных сертификатов и пройденных курсов.

Работа системы построена на последовательном выполнении нескольких этапов. Первоначально происходит сбор данных о кандидатах. Затем осуществляется автоматизированная фильтрация по заданным критериям, что позволяет отсеять неподходящих кандидатов на раннем этапе. После этого система проводит более детальную оценку оставшихся соискателей, анализируя соответствие их навыков требованиям конкретной вакансии.

Особенностью данной системы является ее адаптивность - параметры отбора могут быть настроены в зависимости от специфики компании или конкретного проекта.

Результатом работы системы становится ранжированный список кандидатов, упорядоченный по степени их соответствия требованиям вакансии. Это позволяет HR-специалистам сосредоточиться на наиболее перспективных соискателях, существенно сокращая время на первичный отбор. При этом система не заменяет полностью человеческий фактор, а служит эффективным инструментом для оптимизации начальных этапов рекрутингового процесса, обеспечивая более объективную и систематизированную оценку профессиональных качеств backend-разработчиков.

### **Диаграмма компонентов**

Диаграмма компонентов - это мощный инструмент для проектирования модульной структуры системы, который позволяет визуализировать основные функциональные блоки и их взаимосвязи. В отличие от контекстной диаграммы, которая показывает систему "снаружи", диаграмма компонентов раскрывает её внутреннее устройство, демонстрируя, из каких ключевых модулей состоит система и как они взаимодействуют между собой.

Основными элементами этой диаграммы являются:

* компоненты
* интерфейсы
* зависимости
* порты

При проектировании сложных систем диаграмма компонентов помогает:

* чётко разделить систему на логические модули
* определить интерфейсы между компонентами
* распределить ответственность между командами разработчиков
* спланировать этапы интеграции
* выявить потенциальные узкие места в архитектуре

Особенно важна эта диаграмма при разработке микросервисных архитектур, где каждый компонент может разрабатываться и развёртываться независимо. Она также служит отличным инструментом для документирования системы и обучения новых членов команды.

На рисунке 2.2 представлена диаграмма компонентов информационной системы.

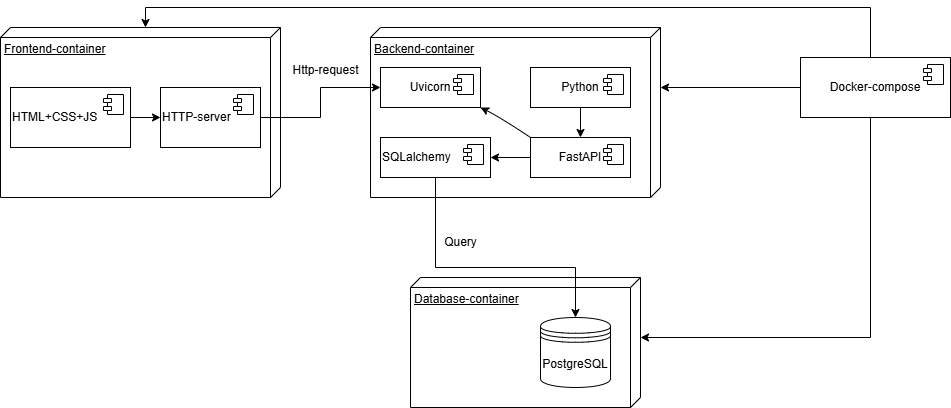


Рисунок 2.2— Диаграмма компонентов

На диаграмме изображена компонентная архитектура веб-приложения, организованная в виде трех взаимодействующих контейнеров, что отражает современный подход к разработке с использованием микросервисов и контейнеризации.

Первым компонентом выступает Frontend-container, который содержит клиентскую часть приложения, включая стандартный набор веб-технологий HTML, CSS и JavaScript, отвечающих за структуру, оформление и интерактивность пользовательского интерфейса соответственно. В этом же контейнере располагается HTTP-сервер выполняющий роль поставщика статического контента для браузера пользователя. Фронтенд взаимодействует с серверной частью посредством HTTP-запросов, что на диаграмме обозначено соответствующей стрелкой.

Серверная логика приложения инкапсулирована в Backend-container, где ключевыми элементами являются Docker-compose для оркестрации контейнеров, язык программирования Python в качестве основной платформы разработки, библиотека SQLalchemy для объектно-реляционного отображения и фреймворк FastAPI, обеспечивающий создание высокопроизводительного API. Бэкенд-контейнер получает запросы от фронтенда, обрабатывает их, выполняет необходимую бизнес-логику и при необходимости формирует запросы к базе данных, что на схеме показано стрелкой с пометкой "Query".

Третьим ключевым компонентом системы является Database-container, где размещена система управления базами данных PostgreSQL, отвечающая за надежное хранение и эффективное управление всеми данными приложения.

Взаимодействие между компонентами происходит по четко определенной схеме: инициация действия пользователем во фронтенде приводит к генерации HTTP-запроса к бэкенду, который после обработки может сформировать SQL-запрос к базе данных, получив необходимые данные, передает их обратно клиентской части для отображения пользователю. Такая архитектура обеспечивает ряд существенных преимуществ, включая модульность, позволяющую изолированно разрабатывать и масштабировать отдельные компоненты, гибкость, обеспечивающую возможность замены технологий в каждом контейнере без перестройки всей системы, и удобство развертывания благодаря использованию Docker-compose, который упрощает процесс запуска и согласованной работы всех частей приложения. Представленная на диаграмме трехзвенная структура с контейнеризацией компонентов является классическим примером современного подхода к построению веб-приложений, сочетающего в себе надежность, производительность и удобство разработки.

### **Диаграмма развёртывания**

Диаграмма развёртывания представляет собой важный инструмент для проектирования инфраструктурной части системы, показывающий, как программные компоненты будут распределены по физическим или виртуальным узлам. В отличие от предыдущих диаграмм, которые фокусируются на логической структуре системы, диаграмма развёртывания имеет чёткую привязку к серверам, сетевым устройствам, облачным платформам и другим элементам инфраструктуры.

Ключевыми элементами этой диаграммы являются:

* узлы развёртывания
* артефакты
* соединения
* зависимости

Диаграмма развёртывания особенно важна для:

* планирования инфраструктурных требований
* оптимизации распределения ресурсов
* проектирования отказоустойчивых решений
* планирования масштабируемости системы
* документирования окружения для DevOps-команды

В современных условиях, когда системы часто развёртываются в гибридных или мультиоблачных средах, значение этой диаграммы только возрастает. Она помогает наглядно представить географическое распределение компонентов, что критически важно для проектирования систем с требованиями к низкой задержке или локализации данных. Также эта диаграмма служит основой для расчёта инфраструктурных затрат и планирования бюджета проекта.

На рисунке 2.3 представлена диаграмма развёртывания.

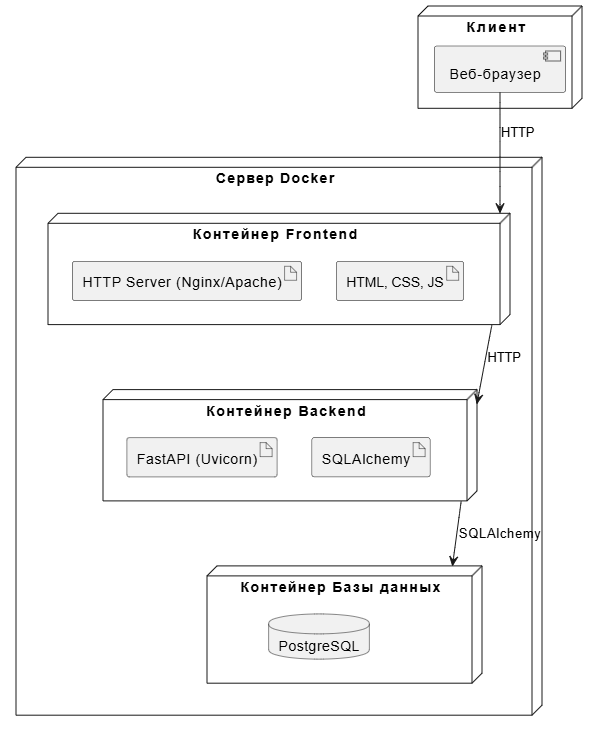


Рисунок 2.3 - Диаграмма развёртывания

Архитектура состоит из трех ключевых компонентов, развернутых на Docker-сервере, которые взаимодействуют между собой для обеспечения работы приложения.

Первый компонент - Frontend-контейнер, который отвечает за клиентскую часть приложения. Внутри него работает HTTP-сервер, обрабатывающий запросы пользователей и отдающий статический контент — файлы HTML, CSS и JavaScript. Этот контейнер обеспечивает визуальную часть приложения, формируя интерфейс, с которым взаимодействуют пользователи. Когда пользователь выполняет какое-либо, фронтенд отправляет HTTP-запросы в Backend-контейнер для обработки.

Второй компонент — Backend-контейнер, где выполняется серверная логика приложения. Он построен на основе FastAPI, который запускается с помощью Uvicorn. Для взаимодействия с базой данных используется SQLAlchemy — ORM-библиотека, позволяющая выполнять запросы к PostgreSQL в виде Python-объектов, а не чистого SQL. Backend принимает HTTP-запросы от фронтенда, обрабатывает их, выполняет необходимые вычисления и, если требуется, запрашивает или изменяет данные в базе.

Третий компонент — контейнер с базой данных PostgreSQL, где хранится вся структурированная информация приложения Backend взаимодействует с PostgreSQL через SQLAlchemy, который преобразует Python-код в SQL-запросы, отправляемые в базу. После выполнения запроса данные возвращаются в бэкенд, который формирует ответ и передает его обратно во фронтенд для отображения пользователю.

Взаимодействие между компонентами происходит следующим образом:

* пользователь открывает веб-интерфейс, который загружается из Frontend-контейнера.
* при взаимодействии с интерфейсом фронтенд отправляет HTTP-запрос в Backend-контейнер.
* backend обрабатывает запрос и, если нужно, формирует SQL-запрос через SQLAlchemy к PostgreSQL.
* база данных выполняет запрос и возвращает результат в бэкенд.
* backend преобразует данные в удобный формат и отправляет ответ фронтенду.
* frontend получает данные и обновляет интерфейс, показывая пользователю актуальную информацию.

Преимущества такой архитектуры:

* масштабируемость: каждый контейнер можно масштабировать.
* изолированность: проблемы в одном контейнере не влияют напрямую на работу других компонентов.
* удобство развертывания: docker позволяет быстро запускать и обновлять приложение на любом сервере с поддержкой контейнеризации.
* гибкость: технологии в каждом контейнере можно заменять

Таким образом, диаграмма демонстрирует стандартную, но мощную архитектуру веб-приложения, развернутого в Docker, где четко разделены фронтенд, бэкенд и база данных, что обеспечивает надежность, производительность и удобство разработки.

## **Проектирование функциональных требований**

Проектирование функциональных требований представляет собой критически важный этап в разработке информационной системы, на котором абстрактные пожелания заказчика и потребности бизнеса трансформируются в четкие, конкретные и технически реализуемые спецификации. Этот процесс выступает своеобразным мостом между бизнес-анализом и непосредственной разработкой, обеспечивая однозначное понимание того, как система должна вести себя в различных условиях и сценариях работы. Основная сложность на этом этапе заключается в том, чтобы не просто зафиксировать желаемую функциональность, но и предугадать все возможные варианты взаимодействия пользователей с системой, предусмотреть обработку исключительных ситуаций и определить границы ответственности системы.

В современной практике разработки программного обеспечения проектирование функциональных требований давно перестало быть простым перечислением возможностей системы. Сегодня это комплексный процесс, включающий в себя несколько взаимосвязанных уровней детализации. На верхнем уровне находятся варианты использования, которые описывают систему с точки зрения конечного пользователя. Затем эти высокоуровневые сценарии детализируются через описание бизнес-процессов, где уже рассматривается последовательность действий, условия выполнения операций и возможные альтернативные потоки. Наиболее глубокий уровень проектирования предполагает описание конкретных взаимодействий между компонентами системы при выполнении тех или иных функций.

Особое значение в процессе проектирования функциональных требований имеет выбор правильных инструментов визуализации. Диаграммы вариантов использования, активности и последовательности в нотации UML стали фактическим стандартом в этой области, так как позволяют наглядно представить сложные взаимодействия и сделать требования понятными для всех участников проекта - от бизнес-заказчиков до разработчиков и тестировщиков. При этом важно понимать, что создание этих диаграмм - не формальность, а важный аналитический процесс, в ходе которого часто выявляются неочевидные на первый взгляд сценарии использования, противоречия в требованиях или технические ограничения.

Одной из ключевых задач на этапе проектирования функциональных требований является обеспечение полноты и непротиворечивости спецификаций. Это означает, что система должна быть описана таким образом, чтобы для любого возможного сценария работы было четко определено ожидаемое поведение. При этом важно соблюдать баланс между избыточной детализацией, которая может сделать документацию громоздкой и трудноподдерживаемой, и слишком общими формулировками, оставляющими пространство для разночтений. Практика показывает, что хорошо проработанные функциональные требования позволяют значительно сократить количество доработок на поздних этапах проекта и минимизировать риски несоответствия системы ожиданиям заказчика.

Отдельного внимания заслуживает вопрос связи функциональных требований с другими аспектами проектирования системы. С одной стороны, они должны опираться на уже определенную архитектуру системы, с другой - сами влияют на проектирование структуры данных и бизнес-логики. Например, выявленные в процессе работы над Use Cases сложные бизнес-процессы могут потребовать изменений в схеме базы данных или даже повлиять на архитектурные решения. Поэтому проектирование функциональных требований нельзя рассматривать изолированно - это итеративный процесс, требующий постоянной синхронизации с другими направлениями проектирования.

В условиях современных agile-методологий разработки подход к проектированию функциональных требований также претерпел изменения. Вместо создания исчерпывающей документации на старте проекта сейчас чаще применяется практика постепенной детализации требований, когда сначала определяются ключевые сценарии использования, а затем, по мере реализации, уточняются детали. Однако это не отменяет необходимости тщательного проектирования, а лишь меняет подход к организации этого процесса, делая его более гибким и адаптивным к изменениям.

### **Диаграмма вариантов использования**

Диаграмма вариантов использования представляет собой важнейший инструмент проектирования программных систем, который позволяет наглядно отобразить функциональные возможности системы с точки зрения конечного пользователя. Эта модель входит в состав унифицированного языка моделирования UML и служит ключевым связующим звеном между бизнес-требованиями и технической реализацией проекта.

Основная ценность данной диаграммы заключается в ее способности простым и понятным языком показать, какие именно функции будет выполнять система, кто именно будет этими функциями пользоваться и с какими внешними компонентами потребуется взаимодействие. При этом диаграмма фокусируется не на технических деталях реализации, а на пользовательском восприятии системы, что делает ее идеальным инструментом для обсуждения с заказчиками и бизнес-пользователями.

В основе диаграммы вариантов использования лежат два ключевых элемента: акторы и собственно варианты использования. Акторы представляют роли пользователей или внешних систем, взаимодействующих с разрабатываемым решением. Варианты использования описывают отдельные функции или возможности системы. Связи между этими элементами показывают, какие именно функции доступны тем или иным категориям пользователей.

Особенность этой модели заключается в ее многоуровневости. Она позволяет одновременно отображать как общий функциональный охват системы, так и детализировать отдельные аспекты через механизмы расширений и включений. Благодаря этому диаграмма служит отличной отправной точкой для последующей детализации требований в более сложных моделях.

Разработка качественной диаграммы вариантов использования представляет собой глубокий аналитический процесс. В ходе ее создания часто выявляются скрытые сценарии использования, обнаруживаются пробелы в требованиях или находятся новые возможности для улучшения пользовательского опыта. Грамотно построенная диаграмма существенно снижает риск появления серьезных недочетов на поздних этапах разработки.

Важно понимать, что эта модель не является статичным документом. По мере развития проекта и уточнения требований диаграмма вариантов использования может и должна пересматриваться и дополняться. Она служит живым инструментом коммуникации между всеми участниками проекта на протяжении всего жизненного цикла разработки системы

На рисунке 2.4 представлена диаграмма вариантов использования.

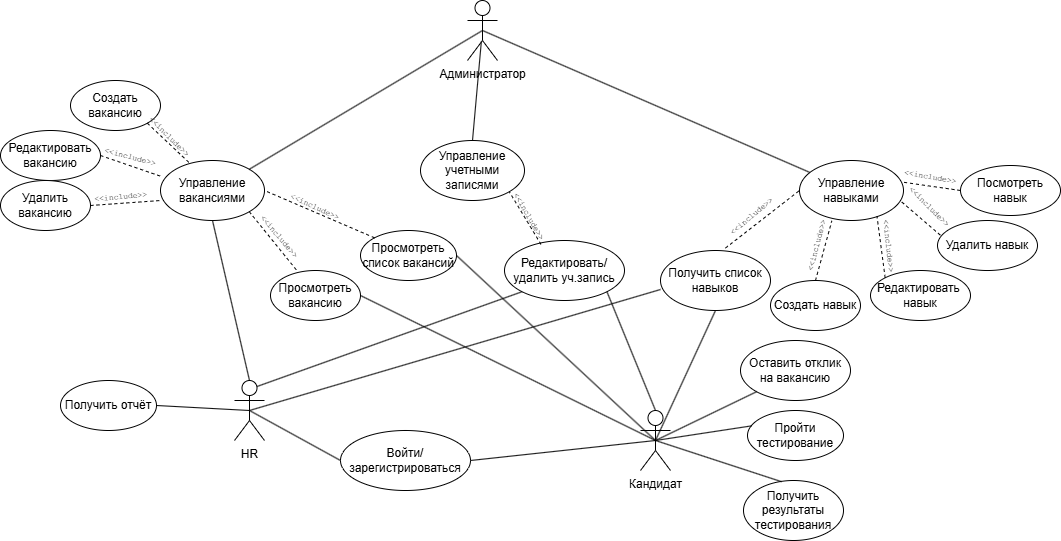


Рисунок 2.4 — Диаграмма вариантов использования

В таблице 2.1 приведено подробное описание прецедентов системы.

Таблица 2.1 – Описание прецедентов информационной системы

|  |  |  |  |
| --- | --- | --- | --- |
| Название прецедента | Актор | Описание | Связанные прецеденты |
| Управление учетными записями | Администратор | Администратор управляет учетными записями пользователей | Редактировать/удалить уч. запись |
| Редактировать/удалить уч. запись | Администратор | Изменение или удаление данных учетной записи | - |
| Управление навыками | Администратор | Управление списком навыков в системе | Создать, Редактировать, Удалить, Посмотреть навык |
| Создать навык | Администратор | Добавление нового навыка | - |
| Редактировать навык | Администратор | Изменение существующего навыка | - |
| Удалить навык | Администратор | Удаление навыка из системы | - |
| Посмотреть навык | Администратор | Просмотр информации о навыке | - |
| Получить список навыков | Кандидат | Получение перечня доступных навыков | - |
| Управление вакансиями | HR, Администратор | Управление вакансиями | Создать, Редактировать, Удалить, Просмотреть вакансию |
| Создать вакансию | HR, Администратор | Добавление новой вакансии | - |
| Редактировать вакансию | HR, Администратор | Изменение информации о вакансии | - |
| Удалить вакансию | HR, Администратор | Удаление вакансии | - |
| Просмотреть вакансию | HR, Кандидат | Просмотр подробностей по вакансии | - |
| Просмотреть список вакансий | HR, Кандидат | Получение списка всех доступных вакансий | - |
| Оставить отклик на вакансию | Кандидат | Отклик на подходящую вакансию | - |
| Пройти тестирование | Кандидат | Прохождение тестов, связанных с навыками | - |
| Получить результаты тестирования | Кандидат | Получение оценок по пройденным тестам | - |
| Получить отчёт | HR | Формирование отчета по кандидатам или тестам | - |
| Войти/зарегистрироваться | Кандидат | Аутентификация и регистрация в системе | - |

Диаграмма вариантов использования описывает функциональные возможности системы управления вакансиями и навыками, демонстрируя взаимодействие трех основных ролей: Администратора, HR-специалиста и Кандидата. Каждая из этих ролей имеет свой набор действий, которые определяют их взаимодействие с системой и охватывают ключевые бизнес-процессы, связанные с подбором персонала и управлением данными.

Администратор является главным управляющим системы и обладает наиболее широкими полномочиями. В его обязанности входит полный контроль над вакансиями, включая их создание, редактирование и удаление. Он может просматривать список всех доступных вакансий, изучать детали каждой из них, а также формировать отчеты для анализа данных. Эти функции позволяют Администратору эффективно управлять процессом найма, оперативно вносить изменения в требования к должностям и отслеживать статистику по открытым позициям.

HR-специалист работает с учетными записями пользователей, обеспечивая их регистрацию и авторизацию в системе. Он имеет возможность редактировать и удалять учетные записи, что позволяет поддерживать актуальность данных и контролировать доступ сотрудников и кандидатов. Эта роль критически важна для обеспечения безопасности системы и корректного распределения прав среди пользователей.

Кандидат взаимодействует с системой в рамках поиска работы и управления своими профессиональными навыками. Он может просматривать список своих навыков, редактировать их, добавлять новые или удалять устаревшие. Кроме того, Кандидат имеет возможность оставлять отклики на заинтересовавшие его вакансии, проходить тестирование, предназначенное для оценки его квалификации, и получать результаты этих тестов. Эти функции помогают Кандидату активно участвовать в процессе трудоустройства, демонстрировать свои компетенции и получать обратную связь от работодателей.

### **Диаграмма активности**

Диаграмма активностей представляет собой мощный инструмент визуализации бизнес-процессов и рабочих потоков в рамках проектирования информационных систем. Эта модель, являющаяся частью стандарта UML, позволяет детально описать последовательность действий, выполняемых системой или пользователями для достижения конкретных целей. В отличие от диаграммы вариантов использования, которая показывает что делает система, диаграмма активностей отвечает на вопрос как именно происходят те или иные процессы.

Основное назначение диаграммы активностей заключается в точном и наглядном представлении алгоритмов работы системы. Она особенно полезна при моделировании сложных бизнес-процессов, где важно отобразить не только основную последовательность шагов, но и возможные ветвления, параллельные потоки выполнения, условия перехода между действиями и точки синхронизации. С помощью этой диаграммы можно эффективно документировать как автоматизированные процессы системы, так и взаимодействие между системой и пользователями.

Ключевыми элементами диаграммы активностей являются действия, переходы, узлы принятия решений, разделения и слияния потоков. Действия представляют отдельные шаги процесса, узлы принятия решений показывают альтернативные пути выполнения, а элементы разделения и слияния позволяют отображать параллельные потоки работ. Такая структура делает диаграмму активностей чрезвычайно гибким инструментом, способным моделировать процессы любой сложности.

Особую ценность диаграмма активностей приобретает при анализе и оптимизации существующих бизнес-процессов. Она позволяет выявить избыточные шаги, узкие места и потенциальные точки улучшения. В контексте разработки программного обеспечения эта диаграмма служит важным связующим звеном между бизнес-аналитиками, которые формулируют требования, и разработчиками, которые реализуют соответствующие алгоритмы.

Применение диаграмм активностей особенно эффективно в проектах, связанных с автоматизацией сложных бизнес-процессов, таких как обработка заказов, цепочки согласований или многоэтапные расчеты. Они помогают всем участникам проекта достичь общего понимания того, как должна работать система, и служат надежной основой для последующего проектирования и тестирования.

На рисунке 2.5 представлена диаграмма активности авторизации.

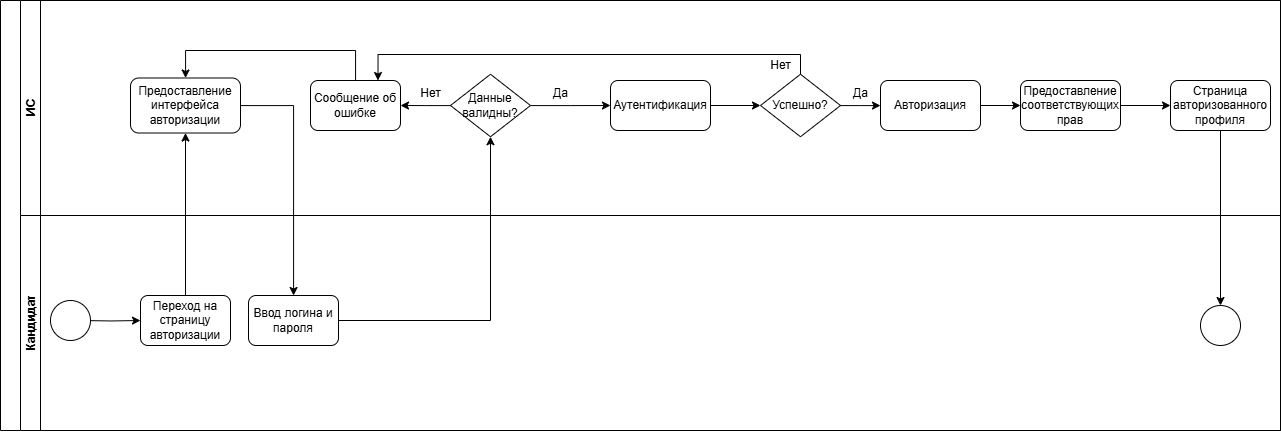


Рисунок 2.5 — Диаграмма активности авторизации

На диаграмме изображён процесс авторизации кандидата в информационной системе. Диаграмма разделена на два пула: кандидат и информационная система.

Процесс начинается с перехода кандидата на страницу авторизации. Далее пользователь вводит логин и пароль. Система в ответ предоставляет интерфейс авторизации и получает введённые данные. Затем выполняется проверка на валидность данных. Если данные невалидны, пользователю отображается сообщение об ошибке, и он может ввести данные повторно.

Если данные валидны, система выполняет аутентификацию, то есть проверку правильности логина и пароля. После этого происходит проверка успешности аутентификации. В случае неудачи пользователю снова выводится сообщение об ошибке.

Если аутентификация проходит успешно, осуществляется авторизация — проверка прав доступа пользователя. После авторизации система предоставляет соответствующие права и отображает страницу авторизованного профиля пользователя.

Процесс завершается переходом пользователя на страницу авторизованного профиля.

На рисунке 2.6 изображена диаграмма активности, описывающая процесс отклика кандидата.

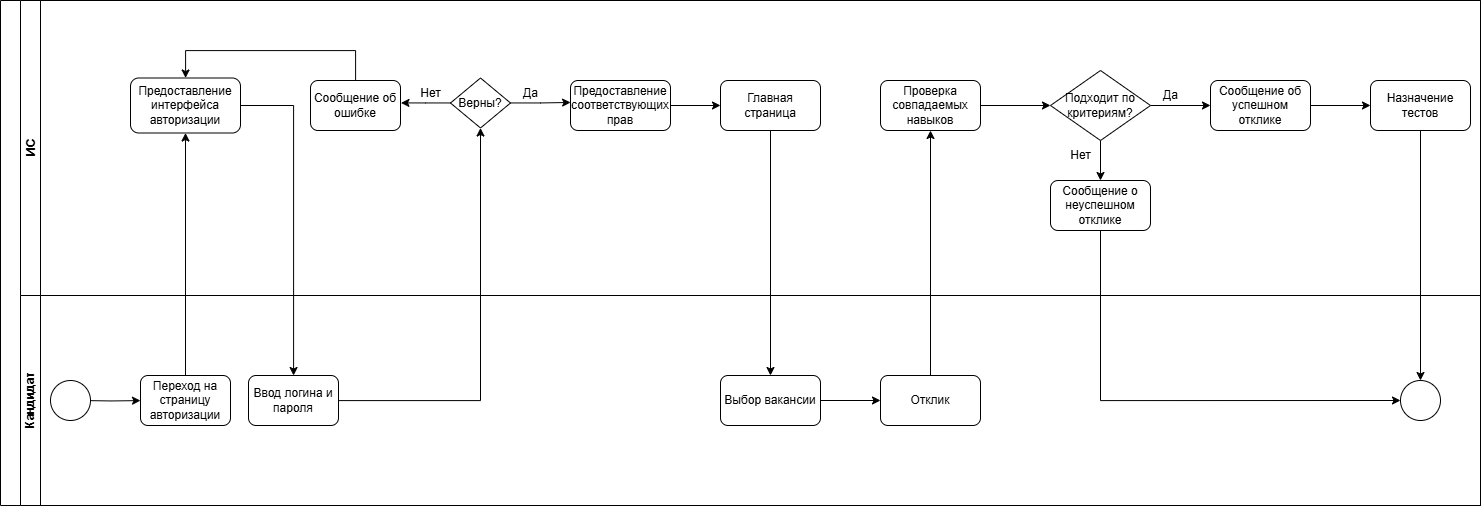


Рисунок 2.6 — Диаграмма активности процесса отклика кандидата

На представленной диаграмме изображён процесс отклика кандидата на вакансию в рамках информационной системы подбора персонала. Диаграмма выполнена в нотации UML и представляет собой диаграмму активности с разделением на две дорожки: "Кандидат" и "ИС" (информационная система). Ниже приведено подробное описание каждого этапа процесса.

Процесс начинается с действия кандидата. Кандидат переходит на страницу авторизации. Далее он вводит логин и пароль. После этого управление передаётся информационной системе, которая предоставляет интерфейс авторизации.

Система проверяет корректность введённых данных. Если данные неверны, пользователю отображается сообщение об ошибке, и он может повторить ввод данных. Если данные верны, система предоставляет соответствующие права и перенаправляет пользователя на главную страницу.

После входа на главную страницу кандидат выбирает интересующую его вакансию. Затем он нажимает кнопку отклика, инициируя процесс отклика на вакансию.

На этом этапе система проверяет совпадающие навыки кандидата с требованиями вакансии. Если навыки кандидата соответствуют критериям вакансии, система отправляет сообщение об успешном отклике и назначает тесты для дальнейшего этапа подбора. После этого процесс завершается.

Если навыки кандидата не соответствуют критериям вакансии, система отправляет сообщение о неуспешном отклике, и процесс также завершается.

Таким образом, диаграмма отражает ключевые этапы взаимодействия кандидата с системой при отклике на вакансию: авторизация, выбор вакансии, проверка соответствия и завершение в зависимости от результата проверки.

На рисунке 2.7 изображена диаграмма активности, описывающая процесс получения отчётности.

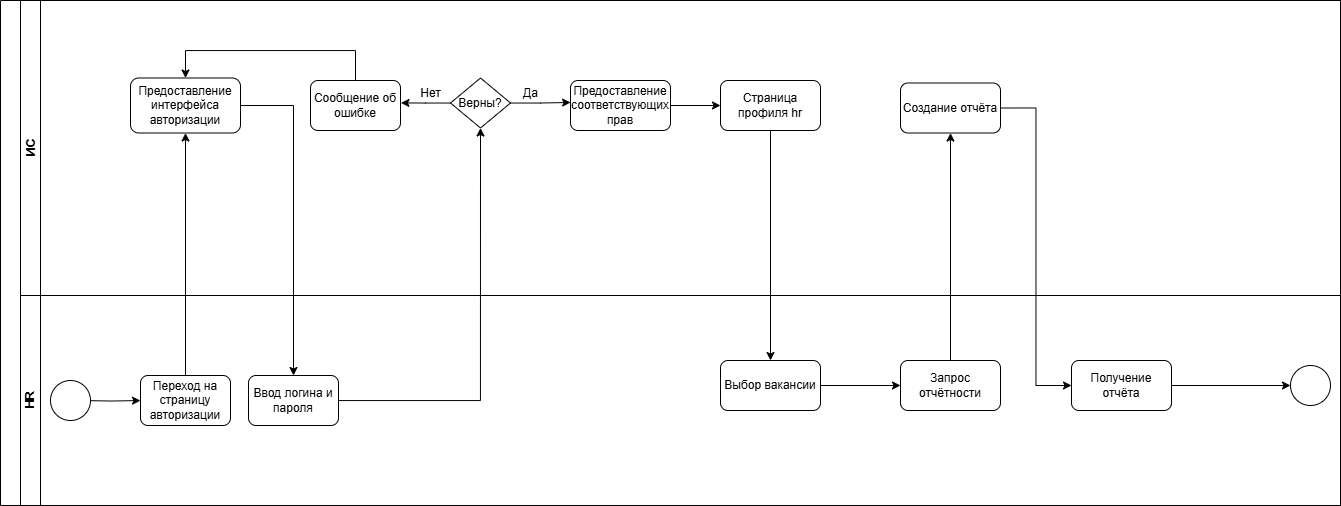


Рисунок 2.7 — Диаграмма активности процесса получения отчётности

На данной диаграмме изображён процесс получения отчётности HR-специалистом в информационной системе подбора персонала. Диаграмма представлена в нотации UML и разделена на две дорожки: "HR" и "ИС".

Процесс начинается с действия HR-специалиста, который переходит на страницу авторизации. Далее он вводит логин и пароль. Информационная система в ответ предоставляет интерфейс авторизации.

После этого система проверяет корректность введённых данных. Если данные неверны, отображается сообщение об ошибке, и пользователь может повторить попытку входа. Если данные верны, система предоставляет соответствующие права и переходит к отображению страницы профиля HR.

На странице профиля HR может выбрать вакансию, по которой он хочет получить отчётность. Далее инициируется запрос отчётности. Этот запрос передаётся в систему, которая в ответ формирует соответствующий отчёт. После создания отчёта система предоставляет его HR-специалисту, и он получает отчёт для дальнейшего анализа.

Таким образом, диаграмма описывает последовательность действий, связанных с авторизацией пользователя, навигацией по профилю, выбором вакансии и получением отчётных данных, с чётким разделением функций между пользователем и информационной системой

На рисунке 2.8 представлена диаграмма активности, описывающая процесс редактирования вакансии.

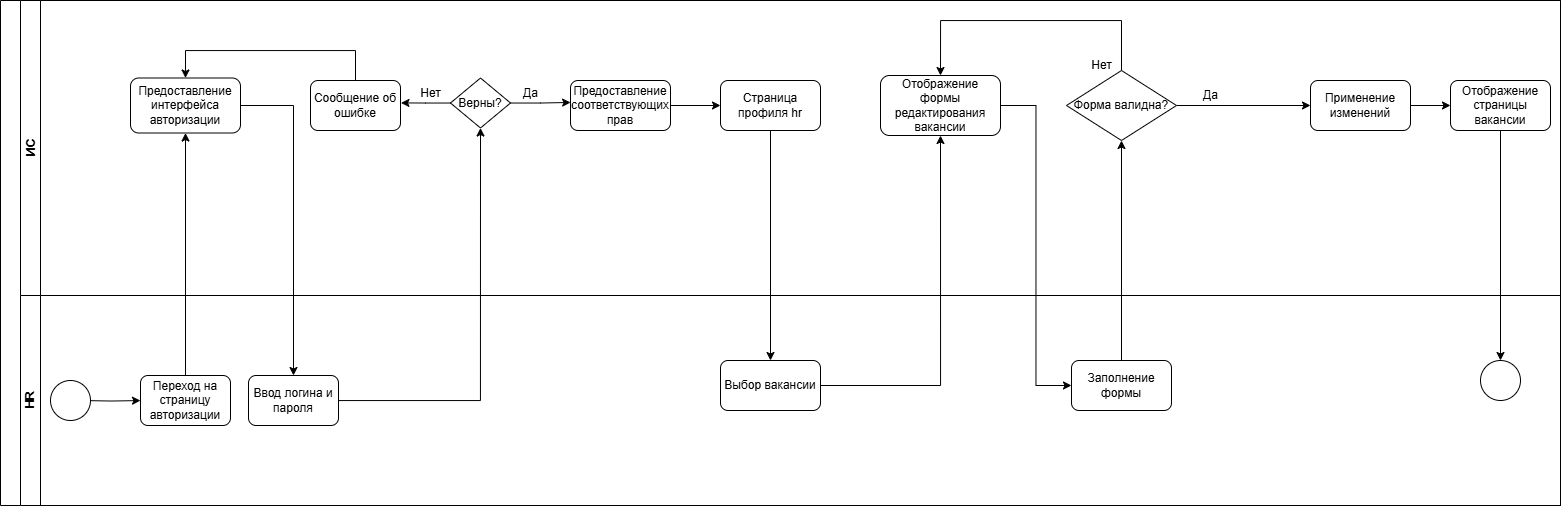


Рисунок 2.8 — Диаграмма активности процесса редактирования вакансии

На данной диаграмме изображён процесс редактирования вакансии HR-специалистом в информационной системе подбора персонала. Диаграмма выполнена в нотации UML и состоит из двух дорожек: "HR" и "ИС".

Процесс начинается с действия HR-специалиста, который переходит на страницу авторизации. Затем он вводит логин и пароль. Информационная система, в ответ на это действие, предоставляет интерфейс авторизации.

Система проверяет корректность введённых данных. Если они некорректны, пользователю отображается сообщение об ошибке, и он может ввести логин и пароль заново. Если данные корректны, система предоставляет пользователю соответствующие права и загружает страницу профиля HR.

На странице профиля HR-специалист выбирает вакансию, которую хочет отредактировать. После этого система отображает форму редактирования вакансии.

HR-специалист заполняет форму с новыми данными. Система проверяет валидность заполненной формы. Если форма невалидна, пользователь возвращается к этапу заполнения формы.

Если форма валидна, система применяет изменения, после чего отображает обновлённую страницу вакансии. На этом процесс редактирования завершается.

Таким образом, диаграмма описывает стандартный сценарий авторизации пользователя, перехода к редактированию информации о вакансии, внесения изменений и проверки корректности данных, с чётким распределением обязанностей между HR и информационной системой.

На рисунке 2.9 представлена диаграмма активности, описывающая процесс создания вакансии.

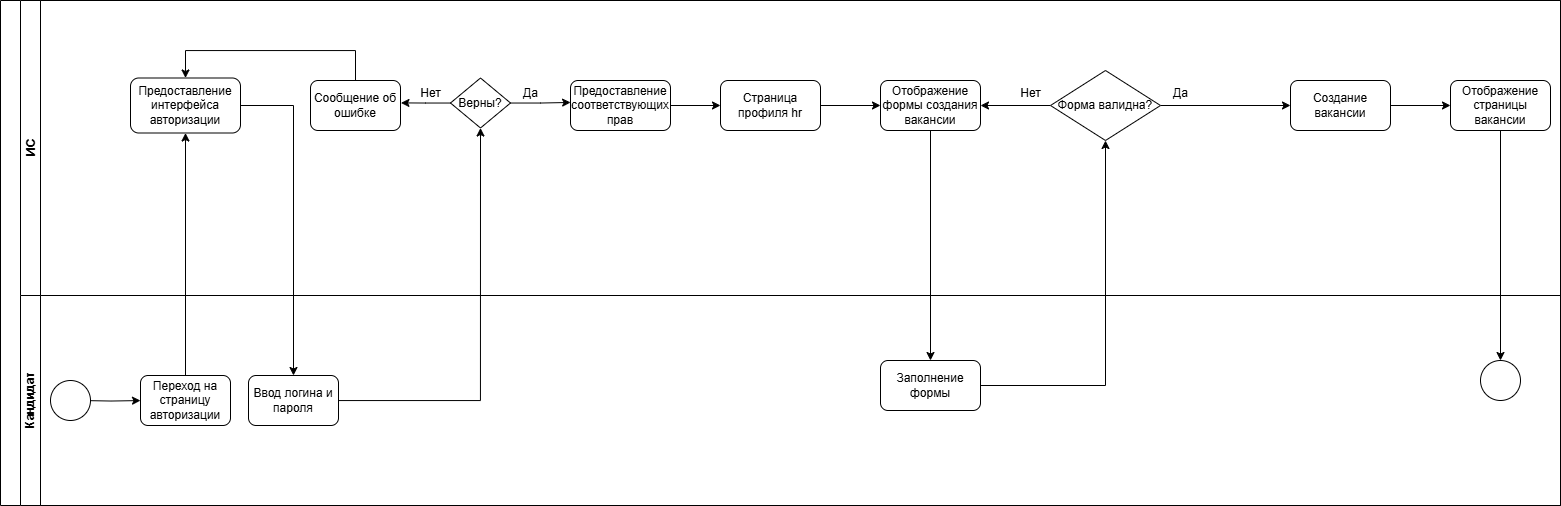


Рисунок 2.9 — Диаграмма активности процесса редактирования вакансии

На диаграмме изображён процесс создания вакансии в информационной системе со стороны кандидата и информационной системы. Это диаграмма активности, представленная в нотации UML с использованием разделения по ролям, где участвуют два участника: "Кандидат" и "ИС".

Процесс начинается с перехода кандидата на страницу авторизации. Далее кандидат вводит логин и пароль. Эти данные передаются в ИС, которая предоставляет интерфейс авторизации. После ввода данных ИС проверяет их корректность.

Если введённые данные неверны, система возвращает сообщение об ошибке, после чего кандидат может попробовать снова.

Если данные верны, ИС предоставляет соответствующие права и переходит к отображению страницы профиля HR.

На следующем этапе отображается форма создания вакансии. Кандидат заполняет эту форму.

После заполнения форма проверяется на валидность. Если форма заполнена некорректно, осуществляется возврат на этап заполнения формы.

Если форма валидна, система создаёт вакансию и отображает страницу с информацией о созданной вакансии.

Процесс завершается.

Таким образом, диаграмма описывает последовательность действий и проверок, необходимых для авторизации пользователя и последующего создания новой вакансии в системе.

На рисунке 2.10 представлена диаграмма активности, описывающая процесс создания вакансии.

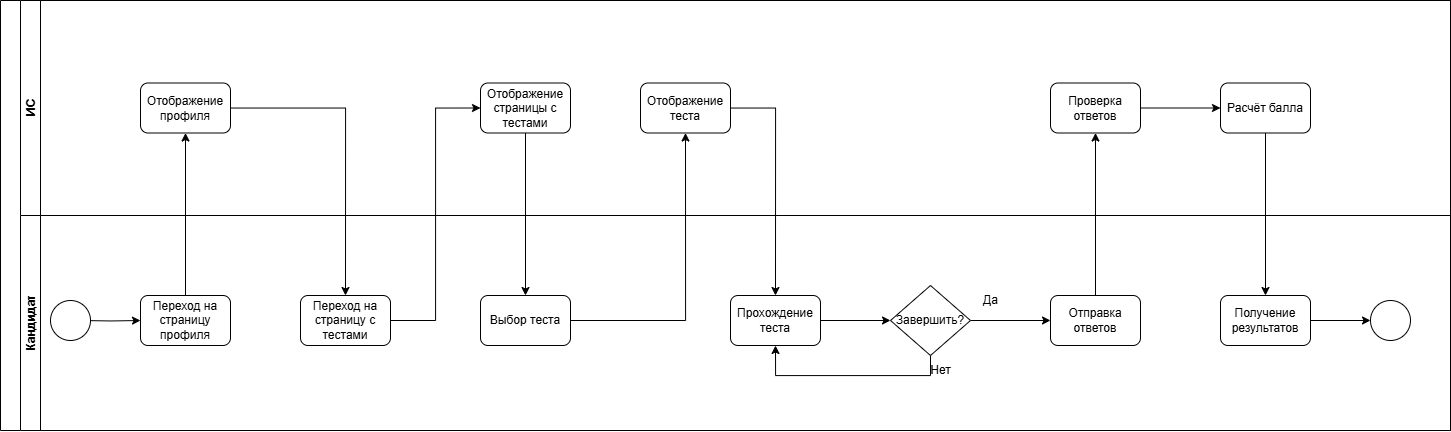


Рисунок 2.10 — Диаграмма активности процесса редактирования вакансии

На представленной диаграмме активности изображён процесс прохождения тестирования кандидатом в информационной системе (ИС). Диаграмма разделена на две дорожки: одна принадлежит участнику "Кандидат", другая — "ИС".

Процесс начинается с перехода кандидата на страницу своего профиля. После этого информационная система отображает страницу профиля.

Затем кандидат переходит на страницу с тестами, и ИС отображает соответствующий интерфейс.

Кандидат выбирает тест из предложенного списка, после чего система отображает сам тест.

Начинается этап прохождения теста. В процессе кандидат может либо продолжать отвечать, либо завершить тест. На этом этапе осуществляется проверка условия: "Завершить?".

Если кандидат выбирает не завершать тест, процесс возвращается к прохождению теста.

Если тест завершён, кандидат отправляет свои ответы. Информационная система принимает ответы и выполняет их проверку.

После проверки ответов система производит расчёт балла и отображает кандидату полученные результаты.

На этом процесс завершается.

Таким образом, диаграмма описывает полный цикл взаимодействия кандидата с системой при прохождении теста: от перехода к тестам до получения результата с оценкой.

### **Диаграмма последовательности**

Диаграмма последовательностей занимает особое место среди инструментов проектирования программных систем, позволяя детально визуализировать динамику взаимодействия между объектами в рамках конкретного сценария работы системы. Этот тип диаграмм UML фокусируется на временной оси, четко показывая порядок и характер обмена сообщениями между участниками взаимодействия. В отличие от диаграммы активностей, описывающей общий поток работ, диаграмма последовательностей раскрывает тонкости коммуникации между конкретными компонентами системы в процессе выполнения отдельного сценария.

Основная ценность диаграммы последовательностей заключается в ее способности наглядно представить временные характеристики взаимодействий, что особенно важно при проектировании сложных распределенных систем. Она позволяет увидеть, какие именно сообщения передаются между объектами, в каком порядке это происходит, какие объекты активны в конкретные моменты времени, и как долго длятся те или иные операции. Такой уровень детализации критически важен для понимания поведения системы в реальных условиях эксплуатации.

Ключевыми элементами диаграммы последовательностей являются объекты-участники взаимодействия, их линии жизни, активационные полосы, показывающие периоды активности объектов, и собственно сообщения между объектами. Особое внимание уделяется различным типам сообщений - синхронным вызовам, асинхронным сигналам, возвратным сообщениям и созданию новых объектов. Все это позволяет с высокой точностью смоделировать реальное поведение системы.

Диаграмма последовательностей особенно полезна при проектировании сложных взаимодействий в объектно-ориентированных системах, где важно понимать не только что делает система, но и как именно распределяется ответственность между различными компонентами. Она помогает выявить потенциальные проблемы проектирования, такие как избыточная связанность компонентов, неоптимальное распределение обязанностей между объектами или возможные узкие места в производительности.

В практике современной разработки программного обеспечения диаграммы последовательностей играют важную роль на этапах как проектирования, так и сопровождения системы. Они служат отличным инструментом коммуникации между архитекторами, разработчиками и тестировщиками, обеспечивая единое понимание сложных механизмов взаимодействия в системе. Особенно незаменимы эти диаграммы при работе над распределенными системами, микросервисными архитектурами и сложными протоколами взаимодействия, где временные характеристики и порядок обмена сообщениями имеют принципиальное значение.

На рисунке 2.11 представлена диаграмма последовательности, описывающая процесс авторизации пользователя.

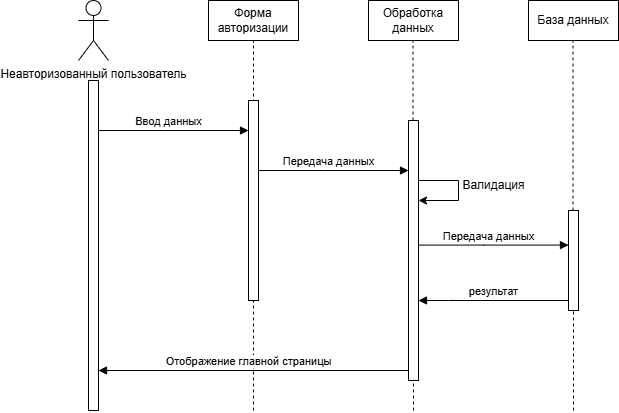


Рисунок 2.11 — Диаграмма последовательности авторизации

На представленной диаграмме изображён процесс авторизации пользователя.

Когда неавторизованный пользователь инициирует вход в систему, он взаимодействует с интерфейсом формы авторизации, где вводит свои учетные данные — обычно логин и пароль. Система на клиентской стороне может выполнять первичную проверку введенных данных на соответствие базовым требованиям. Затем данные передаются на серверную часть, где модуль обработки проводит более глубокую валидацию. После успешной валидации система формирует запрос к базе данных, где происходит сравнение предоставленных учетных данных с хранящейся информацией. В случае успешного сопоставления система создает сессию пользователя, генерирует токен аутентификации и перенаправляет пользователя на главную страницу. При неудачной попытке авторизации система фиксирует попытку входа и возвращает пользователю сообщение об ошибке, не раскрывая при этом, какая именно часть учетных данных неверна, чтобы не облегчать задачу потенциальным злоумышленникам.

На рисунке 2.12 представлена диаграмма последовательности, описывающая процесс регистрации пользователя.

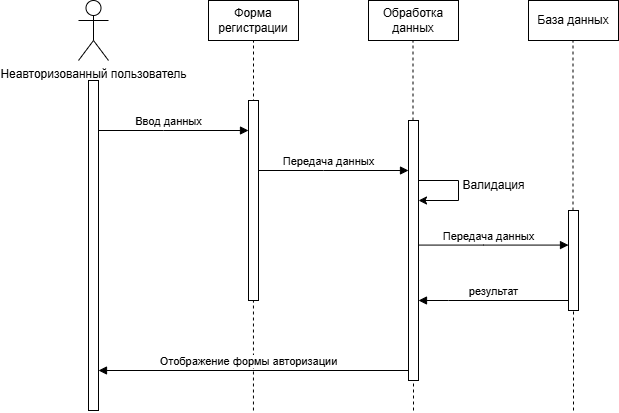


Рисунок 2.12 — Диаграмма последовательности регистрации

На представленной диаграмме изображён процесс регистрации пользователя.

Когда новый пользователь заполняет форму регистрации, система выполняет многоэтапную проверку вводимых данных. На клиентской стороне может осуществляться базовая валидация в реальном времени. После отправки формы данные поступают на сервер, где проводится комплексная проверка, соответствие пароля политикам безопасности, проверку на наличие запрещенных символов или слов. После успешной проверки система хэширует пароль и сохраняет данные пользователя в базе данных. При этом часто реализуется механизм подтверждения email - на указанный адрес отправляется письмо с ссылкой активации. Только после подтверждения email учетная запись становится полностью активной. В случае ошибок регистрации пользователь получает детализированные сообщения, помогающие исправить введенные данные.

На рисунке 2.13 представлена диаграмма последовательности, описывающая процесс редактирования вакансии.

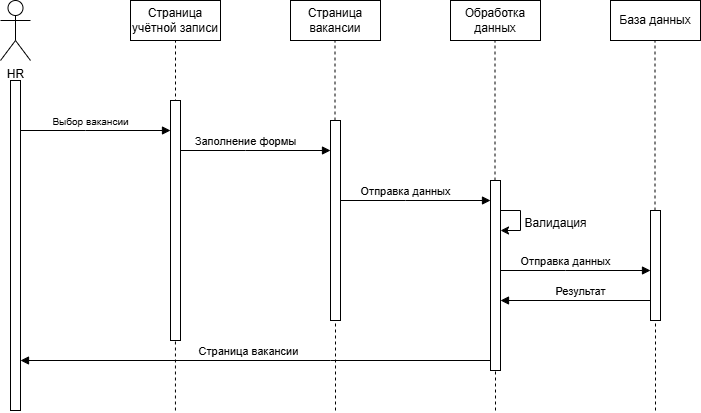


Рисунок 2.13 — Диаграмма последовательности редактирования вакансии

На представленной диаграмме изображён процесс редактирования вакансии.

Когда пользователь выбирает вакансию для редактирования, система сначала проверяет его права на выполнение этой операции. После загрузки текущих данных вакансии пользователь может вносить изменения в различных полях (название, описание, требования и т.д.). При сохранении изменений система выполняет комплексную валидацию: проверяет заполнение обязательных полей, корректность форматов данных (например, дат), отсутствие запрещенного контента. Особое внимание уделяется сохранению истории изменений - система может фиксировать, кто и когда вносил правки, что важно для аудита. После успешного обновления данных в базе система может отправлять уведомления подписанным кандидатам о том, что вакансия была обновлена. В случае массового редактирования вакансий (например, изменения условий работы для целой категории позиций) система может предоставлять пакетный режим редактирования с дополнительными проверками на согласованность изменений.

На рисунке 2.14 представлена диаграмма последовательности, описывающая процесс создания вакансии.

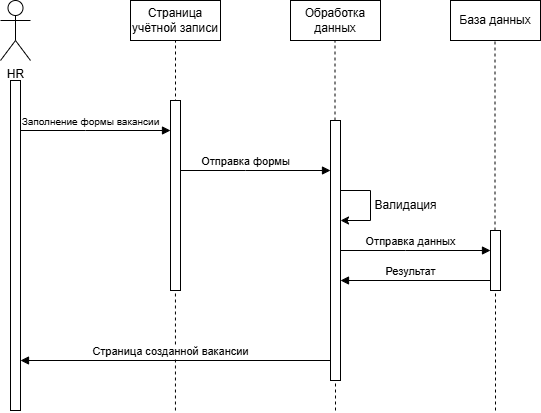


Рисунок 2.14 — Диаграмма последовательности создания вакансии

На представленной диаграмме изображён процесс создания вакансии.

Система предоставляет форму создания вакансии. навыки и компетенции могут выбираться из стандартизированного справочника, а не вводиться вручную. Пользователь заполняет форму. Форма в реальном времени валидируется. Затем она отправляется на сервер, где проходит основную обработку. При прохождении всех проверок создаётся запись в базе данных и отправляется ответ на фронтенд. Происходит перенаправление на страницу созданной вакансии.

На рисунке 2.15 представлена диаграмма последовательности, описывающая процесс создания вакансии.

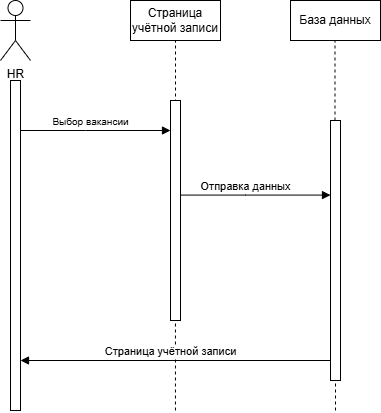


Рисунок 2.15 — Диаграмма последовательности создания вакансии

Когда пользователь решает удалить вакансию, процесс начинается с его действия - нажатия кнопки "Удалить" в интерфейсе системы. Система получает этот запрос и приступает к его обработке. Первым делом проверяется авторизация пользователя - система определяет, вошел ли пользователь в свой аккаунт. Если авторизация отсутствует, пользователь перенаправляется на страницу входа с соответствующим уведомлением.

После успешной проверки авторизации система переходит к проверке прав доступа. Здесь анализируется, имеет ли данный пользователь достаточно полномочий для удаления именно этой вакансии. Проверяется, является ли он создателем вакансии. Если проверка прав не пройдена, система возвращает ошибку.

На следующем этапе появляется запрос подтверждения действия. Если пользователь выбирает отмену, процесс останавливается, и интерфейс возвращается в исходное состояние. При подтверждении система переходит к непосредственному удалению данных.

На рисунке 2.16 представлена диаграмма последовательности, описывающая процесс отклика на вакансию.

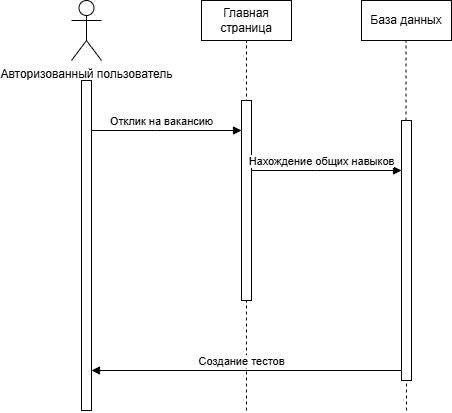


Рисунок 2.16 — Диаграмма последовательности отклика

Процесс отклика на вакансию - это сложный многоуровневый механизм взаимодействия кандидата с системой. Когда авторизованный пользователь нажимает кнопку отклика, система сначала проверяет полноту его профиля. Затем анализируется соответствие кандидата базовым требованиям вакансии. При обработке отклика система автоматически анализирует профиль кандидата и выделяет ключевые совпадения с требованиями вакансии. Эти данные структурируются и представляются работодателю в удобном формате. Кандидат получает подтверждение отклика и информацию о следующих шагах.

На рисунке 2.17 представлена диаграмма последовательности, описывающая процесс прохождения теста.

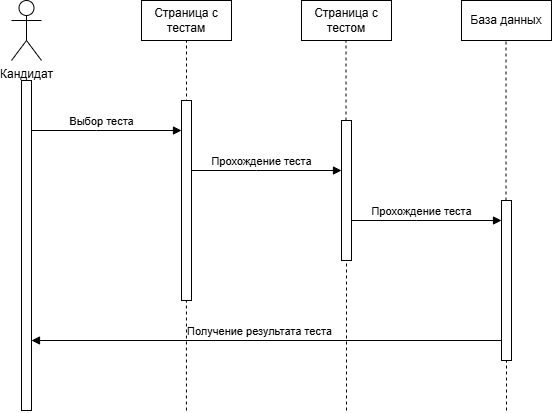


Рисунок 2.17 — Диаграмма последовательности создания вакансии

Процесс тестирования кандидатов представляет собой комплексную систему оценки компетенций. Когда пользователь начинает тест, система загружает вопросы в соответствии с заранее определенной схемой. Если пользователь ответил на все вопросы, то он отправляет свои ответы. Ответы проверяются. Результаты автоматически привязываются к профилю кандидата и отображаются работодателю в контексте конкретной вакансии.

На рисунке 2.18 представлена диаграмма последовательности, описывающая процесс формирования отчёта.

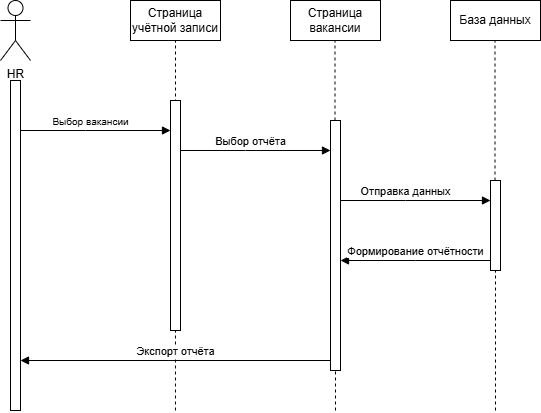


Рисунок 2.18 — Диаграмма последовательности формирования отчёта

Когда пользователь запрашивает отчет, система сначала проверяет его права доступа к запрашиваемым данным. Затем анализируются параметры отчета - временной период, категории вакансий, типы метрик. В процессе формирования отчета данные извлекаются из нескольких источников — основная база вакансий, таблица кандидатов, журнал действий. Особое внимание уделяется визуализации данных - автоматический подбор типов диаграмм, настройка уровней детализации. Для больших объемов данных реализованы механизмы фоновой генерации отчетов с уведомлением о завершении. Готовый отчет можно не только просматривать в системе, но и экспортировать в различных форматах с настройкой параметров экспорта.

На рисунке 2.19 представлена диаграмма последовательности, описывающая процесс редактирования профиля.

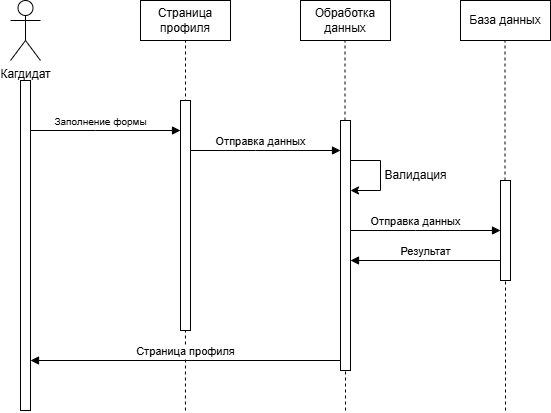


Рисунок 2.19 — Диаграмма последовательности редактирования профиля

Когда пользователь начинает редактирование, система загружает текущие данные профиля, выделяя обязательные и дополнительные поля. При сохранении изменений выполняется комплексная проверка данных - уникальность контактной информации, соответствие форматам, отсутствие противоречий. После сохранения профиль проходит автоматическую переиндексацию для поисковых механизмов системы.

## **Требования к аппаратно-программному обеспечению информационной системы**

Разработка информационной системы для автоматизированного подбора кандидатов на должность backend-разработчика предполагает создание надежной, масштабируемой и защищённой программно-аппаратной архитектуры. От выбора соответствующего оборудования и программного окружения зависит не только производительность, но и стабильность работы системы, удобство взаимодействия с ней для HR-специалистов и кандидатов, а также возможность дальнейшего масштабирования проекта в условиях роста нагрузки.

Аппаратные требования:

* процессор — для корректной и быстрой работы backend-сервиса необходим современный многоядерный процессор. Это позволит системе обрабатывать десятки одновременных запросов, выполнять операции сравнения навыков, вычислять метрики совпадений, запускать автоматическую рассылку тестов и выполнять логику бизнес-процессов в реальном времени. В качестве оптимального варианта можно рассматривать процессоры Intel Core i5/i7 или AMD Ryzen 5/7 с частотой не ниже 2.4 ГГц;
* оперативная память — важным параметром производительности является объём доступной оперативной памяти. Рекомендуется минимум 8 ГБ ОЗУ при разработке и 16 ГБ и выше на сервере при работе в продакшн-режиме. Это особенно важно при одновременной работе нескольких сервисов (backend, БД, фронтенд, тестовые модули и логгеры);
* система хранения данных — вся пользовательская информация, результаты тестов, отклики, а также данные о вакансиях сохраняются в базе данных и должны быть надёжно защищены. Использование твердотельных накопителей (SSD) обеспечивает высокую скорость чтения и записи. В случае корпоративной эксплуатации возможно подключение сетевого хранилища (NAS) с системой автоматического резервного копирования;
* резервное питание — в условиях размещения сервера в локальной сети предприятия рекомендуется предусмотреть наличие источника бесперебойного питания (ИБП), чтобы предотвратить потерю данных при внезапных отключениях электричества.

Программные требования:

* операционная система — предпочтение отдается Linux-дистрибутивам, таким как Ubuntu Server LTS, Debian или CentOS, из-за их высокой производительности, стабильности, масштабируемости и обширного сообщества поддержки. Для разработчиков возможна локальная работа на Windows 10/11 или macOS;
* backend-сервер — система реализована на FastAPI — современном высокопроизводительном фреймворке на Python. Для его запуска требуются Uvicorn или Gunicorn. Также необходимо наличие Python версии не ниже 3.10 с установленными зависимостями проекта (через pip);
* система управления базами данных — PostgreSQL выбрана как основная СУБД благодаря своей надежности, поддержке транзакций, индексов и расширенных возможностей работы со связями между таблицами. Для администрирования базы рекомендуется использовать pgAdmin или DBeaver;
* frontend-среда — пользовательский интерфейс построен с использованием Vue.js. Для сборки и запуска требуется установленный Node.js (версия не ниже 18) и менеджер пакетов npm или yarn;
* системы управления версиями — проект должен вестись с использованием Git, что позволяет отслеживать изменения, реализовывать командную разработку и интеграцию с CI/CD-платформами;
* инструменты разработки — для локальной разработки рекомендовано использование редакторов кода VS Code или PyCharm. Для frontend-части также используются инструменты отладки браузера, такие как Chrome DevTools.

Требования к масштабируемости и мониторингу:

* горизонтальное масштабирование — при увеличении числа пользователей и нагрузки на систему необходимо предусмотреть возможность масштабирования backend-сервера и базы данных с использованием Docker-контейнеров или кластеризации;
* балансировка нагрузки — использование прокси-серверов, например, Nginx или HAProxy, позволит распределить нагрузку между экземплярами приложения;
* мониторинг — для отслеживания работоспособности компонентов системы рекомендуется подключение инструментов мониторинга, таких как Prometheus и Grafana. Это позволит своевременно обнаруживать и устранять сбои;
* логирование — система должна вести подробное логирование действий пользователей и внутренних процессов. Для этого можно использовать связку Loguru + PostgreSQL, либо более сложные системы, такие как ELK Stack (Elasticsearch, Logstash, Kibana).

Требования к безопасности:

* контроль доступа — система должна обеспечивать разграничение прав доступа на уровне ролей (кандидат, HR, администратор). Каждая роль имеет уникальные полномочия и интерфейс;
* защита паролей — все пароли пользователей хранятся в зашифрованном виде с использованием алгоритма bcrypt. В процессе передачи данных используется протокол HTTPS;
* защита от уязвимостей — необходимо предусмотреть фильтрацию пользовательского ввода, защиту от SQL-инъекций, XSS и CSRF-атак. При необходимости внедряются веб-фильтры и регулярные проверки кода;
* резервное копирование — ежедневно должны выполняться резервные копии базы данных и критически важных файлов приложения. Копии рекомендуется хранить на удалённом сервере или в облаке.

Сетевые и эксплуатационные требования:

* интернет-соединение — необходима стабильная широкополосная связь с низкой задержкой для обеспечения бесперебойной работы системы;
* доменное имя и SSL-сертификат — система должна быть доступна по защищённому доменному имени, поддерживающему SSL-сертификат, полученный, например, через Let’s Encrypt;
* техподдержка и обновление — в составе эксплуатационной документации должна быть предусмотрена инструкция по развертыванию, обновлению и устранению типовых ошибок. Желательно использовать систему управления задачами (например, GitHub Issues или Jira) для сопровождения проекта.

## **Выводы по главе**

Во второй главе был выполнен подробный анализ требований к аппаратно-программной среде, необходимой для разработки и эксплуатации информационной системы по подбору кандидатов. Были рассмотрены вопросы обеспечения производительности, отказоустойчивости, безопасности и масштабируемости системы. Эти аспекты особенно важны в условиях обработки персональных данных, высоких требований к надёжности и быстродействию.

Разработка информационной системы предполагает не только проектирование программных модулей, но и продуманное архитектурное решение всей инфраструктуры. Указанные в главе технические параметры обеспечивают необходимый уровень работоспособности, гибкости и безопасности при эксплуатации системы как в рамках локального предприятия, так и в облачном окружении.

# **Процесс разработки информационной системы**

## **3.1. Введение**

Разработка информационной системы - это сложный и многоступенчатый процесс, включающий в себя разнообразные этапы, которые охватывают не только программирование, но и глубокий анализ требований, проектирование архитектуры, создание базы данных, тестирование, внедрение и дальнейшую поддержку. Для создания качественного продукта необходимо учитывать как технические, так и организационные аспекты, а также особенности предметной области.

В нашем случае целью являлось создание информационной системы, предназначенной для автоматизации подбора кандидатов на вакантные позиции в компании, оценки их навыков и упрощения работы HR-специалистов. Проект предусматривал разработку удобного интерфейса, надежного механизма сопоставления навыков кандидатов с требованиями вакансий, а также гибкой системы тестирования и оценки.

Эта задача представляла собой комплексный вызов, требующий серьезного подхода к планированию и реализации. Поэтому уже на начальных этапах была сформирована команда, включающая аналитиков, разработчиков, тестировщиков и специалистов по обеспечению безопасности.

Первым и, пожалуй, одним из самых важных этапов стала работа по сбору требований. Без четкого понимания того, что именно нужно системе и каким образом она должна функционировать, дальнейшая разработка теряет ориентиры и может привести к созданию неудобного или неполноценного продукта.

Для этого проводились встречи и интервью с ключевыми заинтересованными сторонами: HR-менеджерами, руководителями проектов и самими потенциальными пользователями. Основной задачей было выяснить текущие проблемы, с которыми сталкиваются специалисты при подборе персонала, и понять, каким образом новая система может облегчить их работу.

В ходе обсуждений выявились следующие ключевые требования:

* возможность создавать и управлять вакансиями с указанием всех необходимых параметров и требований.
* автоматическое сопоставление навыков кандидатов с требованиями вакансий для ускорения процесса отбора.
* организация тестирования по ключевым навыкам с подсчетом и хранением результатов.
* отслеживание статусов откликов и прогресса кандидатов.
* возможность гибкой настройки ролей и прав доступа для различных категорий пользователей.
* обеспечение безопасности и конфиденциальности данных.

Параллельно с функциональными требованиями были зафиксированы и нефункциональные - связанные с производительностью, удобством, масштабируемостью и поддержкой. Например, система должна была обрабатывать большое количество одновременных запросов без потери скорости, обеспечивать надежное хранение данных и быть интуитивно понятной для пользователей.

Все требования были систематизированы и оформлены в виде подробной документации, которая стала основой для дальнейшего проектирования.

## **3.3. Проектирование архитектуры системы**

После определения требований наступил этап проектирования архитектуры - важнейший шаг, от которого напрямую зависит качество, надежность и гибкость будущей системы.

Рассматривались разные варианты архитектурных решений, чтобы выбрать оптимальное с точки зрения масштабируемости и удобства поддержки. В результате было решено реализовать систему по многослойной архитектуре, где каждый слой отвечает за свою зону ответственности:

* слой представления - взаимодействие с пользователем, предоставление интерфейса.
* слой бизнес-логики - реализация основных алгоритмов, правил и процессов.
* слой доступа к данным - работа с базой данных, хранение и извлечение информации.
* инфраструктурный слой - обеспечение безопасности, логирования, мониторинга.

Такое разделение позволило облегчить разработку и поддержку, а также упростить масштабирование.

Для детального описания данных была создана ER-диаграмма, отражающая все основные сущности и связи между ними. Особое внимание уделялось связям многие-ко-многим, например, связь кандидатов с навыками и вакансий с навыками, что обеспечивало гибкость и точность моделирования.

База данных была спроектирована с учетом нормализации, что позволило минимизировать избыточность и предотвратить возможные ошибки при обновлении данных. Кроме того, были предусмотрены индексы для ускорения поиска и фильтрации.

## **3.4. Выбор технологического стека и инструментов разработки**

Следующий шаг - выбор технологий, необходимых для реализации системы. При принятии решения учитывались требования к производительности, безопасность, удобство поддержки и возможности интеграции.

В качестве языка программирования был выбран Python, благодаря его богатой экосистеме и простоте. В частности, для построения веб-API был использован современный и производительный фреймворк FastAPI, который поддерживает асинхронные операции и позволяет быстро создавать масштабируемые сервисы с минимальными затратами.

Для хранения данных выбрали PostgreSQL - мощную, надежную реляционную СУБД, обладающую расширенными возможностями, включая поддержку сложных запросов, транзакций и полнотекстового поиска.

Для упрощения работы с базой данных использовалась ORM-библиотека SQLAlchemy, что позволило описывать модели в виде классов и легко выполнять миграции.

Для управления проектом и совместной работы применялась система контроля версий Git, а процессы планирования и отслеживания задач велись в трекере, поддерживающем Agile-методологии.

## **3.5. Разработка и реализация функционала**

Работа по разработке велась итеративно, с разбивкой на спринты, в ходе которых реализовывались отдельные блоки функционала.

Начали с создания моделей данных - описания сущностей и связей в коде. Это позволило на раннем этапе увидеть структуру базы и проверить правильность проектирования.

Особое внимание уделялось безопасности хранения данных: пароли пользователей хранились в хэшированном виде с использованием современных алгоритмов, а доступ к API контролировался с помощью токенов и ролей.

Далее последовала реализация ключевых бизнес-логик:

* алгоритм сопоставления навыков кандидата с требованиями вакансии был тщательно проработан, чтобы максимально точно оценивать степень соответствия и учитывать весовые коэффициенты для разных навыков.
* механизм автоматического назначения тестов по каждому совпадающему навыку и последовательная отправка следующих тестов после успешного прохождения предыдущих.
* подсчет баллов за каждый тест и формирование общей оценки с возможностью просмотра результатов HR-специалистами.

Весь функционал был разделен на модули и эндпоинты API с четко описанными интерфейсами.

## **3.6. Тестирование и обеспечение качества**

Для поддержания высокого качества кода была организована комплексная система тестирования.

* юнит-тесты покрывали отдельные функции и классы, гарантируя корректность логики.
* интеграционные тесты проверяли взаимодействие между модулями и работу с базой данных.
* функциональные тесты имитировали поведение пользователей и проверяли основные сценарии использования.
* нагрузочное тестирование помогло выявить узкие места в производительности и оптимизировать их.

Параллельно была внедрена система статического анализа кода, которая помогала обнаруживать потенциальные ошибки и недочеты.

Регулярное проведение code review способствовало повышению качества и поддерживало единый стиль кода.

## **3.7. Обеспечение безопасности**

Безопасность системы - один из главных приоритетов. Для защиты данных были реализованы следующие меры:

* использование HTTPS для шифрования передаваемых данных.
* аутентификация пользователей с использованием JWT-токенов с ограниченным временем жизни.
* разграничение прав доступа на основе ролей, что позволило ограничить доступ к конфиденциальной информации.
* защита от SQL-инъекций и XSS-атак за счет применения проверенных библиотек и методов фильтрации данных.
* регулярное обновление зависимостей и мониторинг уязвимостей.
* настройка бэкапирования и механизмов восстановления данных в случае сбоев.

## **3.8. Развертывание и поддержка**

По завершении разработки система была подготовлена к развертыванию в продакшн-среде. Для упрощения переноса и обеспечения стабильности использовались контейнеры Docker, что позволяло стандартизировать окружение и упростить процессы CI/CD.

Были настроены системы мониторинга и логирования, позволяющие отслеживать состояние сервиса, производительность и оперативно реагировать на возможные сбои.

Подготовлена подробная документация как для разработчиков, так и для конечных пользователей. Для HR-специалистов были проведены обучающие сессии, что ускорило процесс внедрения системы в рабочие процессы.

## **Выводы по главе**

Разработка системы показала, что тщательное планирование, систематический подход и использование современных технологий позволяют создавать эффективные и надежные решения.

В дальнейшем планируется развитие функционала, включая:

* внедрение искусственного интеллекта для более точного анализа резюме и предсказания успешности кандидатов.
* интеграция с внешними сервисами и социальными сетями для автоматического сбора данных.
* расширение возможностей аналитики и визуализации результатов.
* улучшение пользовательского интерфейса на базе современных frontend-технологий.
* масштабирование системы для поддержки большего количества пользователей и более крупных компаний.

Таким образом, проект не только отвечает текущим требованиям, но и заложил фундамент для дальнейшего роста и развития.

# **Заключение**

Выпускная квалификационная работа была посвящена разработке информационной системы для автоматизированного подбора кандидатов на должность backend-разработчика, что является актуальной задачей в современном кадровом менеджменте и IT-индустрии в целом. В процессе выполнения работы была проведена глубокая аналитика предметной области, определены ключевые требования и проведено комплексное проектирование и разработка программного продукта, отвечающего современным стандартам и требованиям бизнеса.

Главной целью работы являлась оптимизация и автоматизация процесса подбора персонала, что позволило существенно повысить эффективность взаимодействия между HR-специалистами и техническими подразделениями компании, минимизировать человеческий фактор, а также сократить время и финансовые затраты, связанные с поиском и отбором квалифицированных специалистов. Внедрение автоматизированной системы, основанной на объективном сопоставлении навыков кандидатов с требованиями вакансий, а также автоматической отправке и оценке тестовых заданий, позволяет обеспечить более качественный и прозрачный процесс подбора.

На этапе исследования была проведена всесторонняя аналитика современных методов и технологий подбора персонала, изучены лучшие практики в сфере автоматизации HR-процессов, а также проведены консультации с профильными экспертами, что позволило сформулировать полный набор функциональных и нефункциональных требований к системе. Особое внимание уделялось вопросам безопасности данных, удобства интерфейса и масштабируемости решения.

Архитектурное проектирование системы выполнено с использованием современных принципов разработки программного обеспечения, включая многослойную архитектуру, что способствует четкому разделению функциональных блоков, облегчает поддержку и развитие системы, а также обеспечивает высокую надежность и отказоустойчивость. Выбор технологического стека, включающего язык программирования Python, веб-фреймворк FastAPI, систему управления базами данных PostgreSQL и ORM SQLAlchemy, обеспечил гибкость, производительность и удобство сопровождения кода.

Особое значение в работе имело обеспечение безопасности и конфиденциальности персональных данных пользователей системы. Были реализованы надежные механизмы аутентификации и авторизации, использованы современные стандарты шифрования, а также проведены мероприятия по защите от типичных уязвимостей веб-приложений, что гарантирует сохранность данных и соблюдение нормативных требований в области информационной безопасности.

Разработка программного продукта сопровождалась применением передовых методик обеспечения качества, включая написание модульных тестов, проведение код-ревью и организацию процессов непрерывной интеграции, что значительно повысило качество конечного продукта и снизило вероятность возникновения ошибок при эксплуатации.

Тестирование и внедрение системы показали высокую производительность и удобство использования, возможность масштабирования под растущие потребности компании, а также положительное влияние на процессы подбора персонала. Сокращение времени отклика на вакансии и повышение объективности оценки кандидатов способствовали улучшению общей эффективности работы HR-отдела и повысили уровень удовлетворенности конечных пользователей.

В ходе выполнения работы были выявлены и проанализированы основные ограничения и возможные риски, связанные с автоматизацией кадровых процессов, что позволило разработать рекомендации по их минимизации и дальнейшему совершенствованию системы. Например, планируется интеграция методов машинного обучения и искусственного интеллекта для повышения точности оценки и прогнозирования успешности кандидатов, что откроет новые горизонты в области автоматизации HR.

Перспективы развития проекта включают расширение функционала за счет подключения внешних платформ и сервисов, внедрение многоуровневой аналитики и отчетности, улучшение пользовательского интерфейса с учетом обратной связи, а также разработку мобильных приложений для повышения доступности системы.

Практическая значимость работы заключается в создании надежного инструмента, способного значительно улучшить процессы подбора персонала, что особенно важно для IT-компаний, стремящихся к оперативному привлечению квалифицированных специалистов в условиях высокой конкуренции на рынке труда. Результаты работы могут быть использованы как основа для дальнейших исследований и разработок в области автоматизации управления человеческими ресурсами.

Таким образом, выполненная выпускная квалификационная работа полностью соответствует заявленным целям и задачам, продемонстрировала высокий уровень теоретической подготовки и практических навыков, и внесла весомый вклад в развитие современных информационных систем, направленных на оптимизацию бизнес-процессов.

В заключение можно отметить, что достигнутые результаты не только подтвердили эффективность выбранного подхода и технологий, но и открыли новые возможности для совершенствования и масштабирования системы в будущем, что делает её перспективным решением для организаций, заинтересованных в цифровой трансформации кадрового менеджмента.

# **Список литературы**

1. Иванов А.А., Петров К.Д. (2024). «Автоматизированные методы оценки компетенций IT-специалистов» // «Информационные технологии». - № 5. - С. 34–41.
2. Сидорова Е.Л. (2023). «Анализ современных HR-технологий в IT-рекрутинге» // «Управление персоналом». - № 8. - С. 56–63.
3. Кузнецов М.П. (2022). «Искусственный интеллект в подборе технических специалистов»: дис. … канд. экон. наук. - М.: МГУ.
4. Смирнов В.Г. (2023). «IT-рекрутинг: современные методы и инструменты». - СПб.: Питер. - 320 с.
5. Ткаченко И.Н. (2024). «Автоматизация HR-процессов: от теории к практике». - М.: Альпина Паблишер. - 250 с.
6. Рыжков А.Ю. (2023). «Технологии машинного обучения в управлении персоналом». - М.: ИНФРА-М. - 180 с.
7. Белова О.С. (2024). «Методы анализа больших данных в HR-аналитике» // «Бизнес-информатика». - № 1. - С. 22–29.
8. Громов П.К. (2022). «Цифровая трансформация рекрутинга». - М.: Эксмо. - 210 с.
9. Журнал «Директор по персоналу» (2024). Спецвыпуск «IT-рекрутинг 2024: тренды и кейсы».
10. Материалы конференции «HR Digital» (2023). Сборник докладов. - М.: Изд-во «HR-Практика».
11. Мартин Р. Чистый код: создание, анализ и рефакторинг. - М.: Питер, 2020. - 464 с.
12. Эванс Э. Предметно-ориентированное проектирование (DDD). - М.: Вильямс, 2019. - 448 с.
13. Ньюмен С. Создание микросервисов. - М.: Питер, 2021. - 304 с.
14. Фаулер М. Архитектура корпоративных программных приложений. - М.: Вильямс, 2019. - 544 с.
15. Клеппман М. Высоконагруженные приложения. - М.: Питер, 2020. - 608 с.
16. Ричардсон К. Шаблоны микросервисов. - М.: Питер, 2021. - 592 с.
17. Найгард М. Высокодоступные системы. - М.: Питер, 2020. - 368 с.
18. Официальная документация по Java [Электронный ресурс]. - URL: https://docs.oracle.com/javase/ru/
19. Документация Django на русском [Электронный ресурс]. - URL: https://djangodoc.ru/
20. Документация Spring Framework [Электронный ресурс]. - URL: https://spring-projects.ru/
21. Документация Docker на русском [Электронный ресурс]. - URL: https://docs.docker.ru/
22. Документация Kubernetes [Электронный ресурс]. - URL: https://kubernetes.io/ru/docs/
23. Документация PostgreSQL [Электронный ресурс]. - URL: https://postgrespro.ru/docs/
24. Документация MongoDB [Электронный ресурс]. - URL: <https://mongodb.com/docs/ru-ru/>