**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.

## **Вывод 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.
* спецификации API.
* макеты пользовательских интерфейсов.

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



## **Логическое моделирование предметной области**

Логическим моделированием предметной области назвается процесс глубокого анализа закономерностей развития проектируемой системы. На его основе создаются формальные модели-образы, которые позволяют чётко отделить этап проектирования от непосредственного написания кода. Такой подход делает разработку более осмысленной и структурированной.

Ключевое преимущество логического моделирования - значительное повышение эффективности всего процесса разработки. За счёт детальной проработки архитектуры на ранних этапах сокращаются сроки реализации проекта, уменьшается количество потенциальных ошибок. Кроме того, продуманная модульная структура позволяет повторно использовать компоненты в других проектах.

Для веб-приложений особенно важны два типа визуализации: диаграммы вариантов использования и диаграммы классов. Первые наглядно показывают, как пользователи взаимодействуют с системой. Вторые детально описывают структуру данных и взаимосвязи между ними.

В результате логическое моделирование превращает хаотичный процесс разработки в управляемый workflow. Каждое решение принимается осознанно, на основе проработанной концептуальной модели. Это приводит к созданию более надёжных, масштабируемых и адаптируемых программных продуктов.

### **Выбор средств описания логической модели**

Логическое моделирование предметной области представляет собой процесс анализа структуры данных, бизнес-процессов и взаимодействий между компонентами системы. Его цель — создать четкую схему работы ИС до этапа программирования, что позволяет:

* ускорить разработку за счет предварительного проектирования.
* снизить количество ошибок благодаря детализированным требованиям.
* обеспечить модульность и возможность повторного использования компонентов.

Для описания логической модели разрабатываемой информационной системы используются:

* диаграмма вариантов использования - для определения функциональных возможностей и взаимодействия пользователей с системой.
* диаграмма классов - для отображения структуры данных и их взаимосвязей.
* диаграмма последовательности - для визуализации сценариев взаимодействия между объектами системы.
  + 1. **Описание логической модели**

Диаграмма вариантов использования представляет собой важнейший инструмент концептуального моделирования, который позволяет систематизировать и визуализировать функциональные требования к разрабатываемой системе. Ее основное назначение заключается в формализации взаимодействия между внешними акторами и системой через определение набора предоставляемых услуг или функций. Эта диаграмма создает абстрактное, но при этом достаточно полное представление о возможностях системы, фокусируясь на том, что система должна делать, а не на том, как она это делает.

Ключевыми элементами данной диаграммы являются акторы и варианты использования. Акторы символизируют роли, которые взаимодействуют с системой, причем эти роли могут принадлежать как людям, так и другим системам. Варианты использования отражают отдельные функциональные возможности, предоставляемые системой в ответ на запросы акторов. Связи между этими элементами показывают, какие акторы могут инициировать те или иные варианты использования. Особую выразительность диаграмме придают различные типы отношений между самими вариантами использования, которые позволяют моделировать сложные сценарии взаимодействия.

Главное преимущество диаграммы вариантов использования заключается в ее способности служить эффективным средством коммуникации между различными участниками процесса разработки. Она обеспечивает общий язык для обсуждения функциональных требований, понятный как техническим специалистам, так и бизнес-пользователям. При этом диаграмма остается достаточно абстрактной, чтобы не ограничивать варианты технической реализации, но при этом достаточно конкретной, чтобы четко обозначить границы проектируемой системы и ее основные возможности.

На этапе анализа требований диаграмма вариантов использования помогает выявить полноту и непротиворечивость функциональных требований, обнаружить возможные пробелы в описании системы. В процессе проектирования она служит отправной точкой для создания более детальных моделей и спецификаций. При этом важно понимать, что диаграмма вариантов использования не предназначена для описания внутренней логики системы или последовательности действий - эти аспекты относятся к другим видам диаграмм.

Диаграмма вариантов использования представлена на рисунке 1.

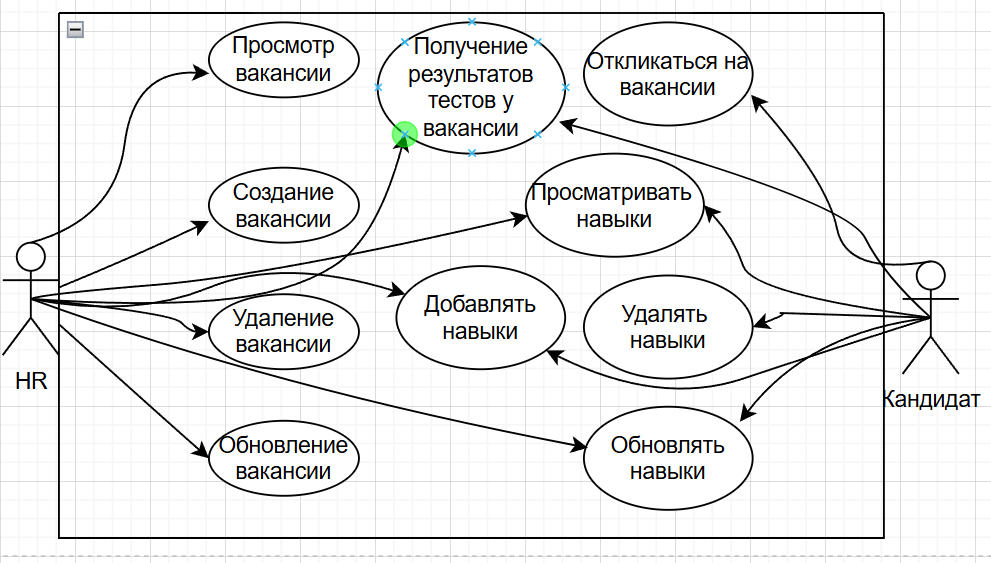


Рисунок 1 - Диаграмма использования

В таблице 2 отражена характеристика прецедентов, которые описывают взаимодействие системы и пользователя.

Таблица 2 - Краткая характеристика прецедентов

|  |  |
| --- | --- |
| Прецедент | Краткая характеристика |
| Создание вакансии | Пользователь создаёт вакансию |
| Просмотр вакансии | Пользователь просматривает вакансию |
| Изменение вакансии | Пользователь редактирует вакансию |
| Удаление вакансии | Пользователь удаляет вакансию |
| Получение списка навыков | Пользователь просматривает список навыков |
| Добавление списка навыков | Пользователь добавляет список навыков |
| Удаление навыков | Пользователь удаляет навыки |
| Регистрация | Пользователь регистрируется |
| Авторизация | Пользователь авторизуется |
| Редактирование профиля | Пользователь редактирует профиль |
| Прохождение тестов | Пользователь проходит тесты по навыкам |

Диаграмма классов представляет собой фундаментальный инструмент объектно-ориентированного моделирования, который формализует статическую структуру проектируемой системы через описание классов и взаимосвязей между ними. Эта диаграмма отражает архитектурный каркас системы, где каждый класс выступает как абстракция некоторой сущности предметной области, инкапсулирующей данные и поведение. Ее основное назначение заключается в четком определении структурных компонентов системы и отношений между ними до этапа непосредственной реализации.

Ключевыми элементами диаграммы классов являются непосредственно классы, интерфейсы и различные типы отношений. Класс изображается в виде прямоугольника, разделенного на три секции: название класса, список атрибутов и перечень методов. Атрибуты характеризуют состояние объекта, а методы определяют его поведение. Интерфейсы представляют собой абстрактные описания функциональности, которые могут быть реализованы классами. Отношения между элементами включают ассоциации, агрегации, композиции и наследование, каждое из которых имеет строго определенную семантику и графическое обозначение.

Особую значимость диаграмма классов приобретает благодаря своей способности моделировать сложные структурные взаимосвязи в системе. Отношения наследования позволяют выстраивать иерархии классов, способствуя повторному использованию кода и реализации полиморфизма. Ассоциативные связи отражают структурные взаимосвязи между объектами разных классов. Агрегация и композиция специфицируют отношения "часть-целое" с различной степенью зависимости между составляющими. Все эти элементы в совокупности создают полную картину организации данных и поведения в системе.

Основная ценность диаграммы классов заключается в ее универсальности как инструмента проектирования. Она служит важным связующим звеном между концептуальным моделированием предметной области и физической реализацией системы в коде. На этапе проектирования диаграмма помогает выявить избыточность или недостаточность структуры данных, обнаружить потенциальные проблемы архитектуры и оптимизировать организацию классов. В процессе разработки она становится наглядным справочником структуры системы, а при сопровождении - ценным документом, облегчающим анализ и модификацию кода.

Диаграмма классов представлена на рисунке 2.

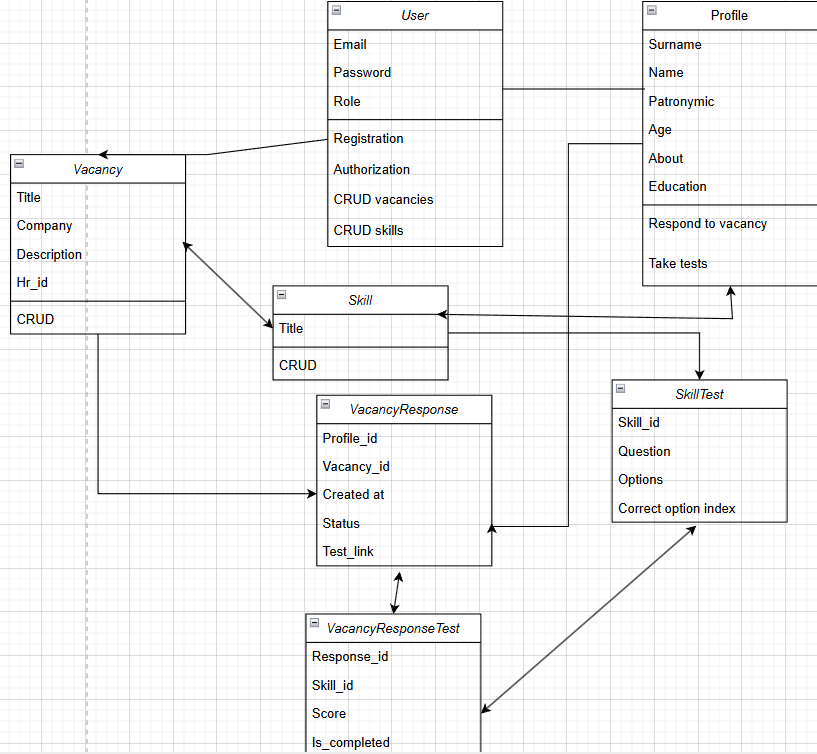


Рисунок 2 - Диаграмма классов

На диаграмме представлены следующие классы:

* класс «User» содержит атрибуты: Email, Password, Role, а также методы: регистрация, авторизация, управление вакансиями и управление навыками. Класс отвечает за авторизацию пользователей и определение их роли.
* класс «Profile» содержит атрибуты: Surname, Name , Patronymic , Age, About, Education. Пользователь может откликаться на вакансии и проходить тесты. Данный класс представляет расширенные данные пользователя, связанные с его анкетой.
* класс «Vacancy» содержит атрибуты: Title, Company, Description, Hr\_id. Предусмотрены методы CRUD для управления вакансиями. Связан с классом User, т.е. пользователь-HR создает вакансии.
* класс «Skill» содержит атрибут Title и методы для CRUD-операций. Навык используется как характеристика, необходимая для вакансий и тестов.
* класс «SkillTest» содержит атрибуты: Skill\_id, Question (вопрос), Options, Correct option index. Используется для создания тестов на проверку навыков.
* класс «VacancyResponse» содержит атрибуты: Profile\_id, Vacancy\_id, Created at, Status, Test\_link. Представляет отклик соискателя на вакансию и связан с классами Profile и Vacancy.
* класс «VacancyResponseTest» содержит атрибуты: Response\_id, Skill\_id, Score, Is\_completed. Связан с результатами тестирования соискателя по определенным навыкам в рамках отклика на вакансию.

Связи между классами:

* + класс «User» управляет вакансиями и навыками.
  + класс «Vacancy» связан с классом «User» через атрибут Hr\_id.
  + класс «Profile» связан с классом «User» и участвует в откликах на вакансии.
  + класс «VacancyResponse» связывает анкету пользователя с конкретной вакансией.
  + класс «SkillTest» относится к конкретному навыку (Skill).
  + класс «VacancyResponseTest» хранит результаты прохождения тестов и связан с откликами на вакансии и навыками.

Таким образом, были определены основные классы, их атрибуты и взаимосвязи: от регистрации и создания вакансий до откликов, тестирования навыков и анализа результатов.

Диаграмма последовательности представляет собой важнейший инструмент моделирования динамического поведения системы, который детально описывает взаимодействие объектов в процессе выполнения конкретного сценария. Эта диаграмма особенно ценна при проектировании сложных алгоритмов и бизнес-процессов, так как позволяет наглядно проследить последовательность сообщений между объектами с учетом временной оси. Основное назначение диаграммы последовательности заключается в точной спецификации порядка взаимодействия компонентов системы, что делает ее незаменимой при разработке сложной логики взаимодействия между различными частями системы.

Ключевыми элементами диаграммы являются объекты-участники взаимодействия, представленные в верхней части диаграммы, и их линии жизни, тянущиеся вертикально вниз. Линии жизни демонстрируют период активности объектов в рамках рассматриваемого сценария. Сообщения между объектами изображаются стрелками различного вида, каждая из которых имеет строго определенную семантику. Синхронные сообщения, показанные сплошными стрелками с заполненными наконечниками, подразумевают ожидание ответа, тогда как асинхронные сообщения, обозначаемые сплошными стрелками с линейными наконечниками, не требуют немедленного ответа. Особое значение имеют возвратные сообщения, которые хотя часто и опускаются для упрощения диаграммы, но могут быть важны для полного понимания процесса взаимодействия.

Важной особенностью диаграммы последовательности является возможность показать активационные полосы - прямоугольники на линиях жизни, обозначающие периоды непосредственной активности объектов при обработке сообщений. Эти полосы наглядно демонстрируют, какие объекты в какие моменты времени выполняют те или иные операции. Для сложных сценариев диаграмма позволяет использовать различные конструкции управления потоком выполнения, такие как альтернативные фрагменты (alt), циклические фрагменты (loop) и параллельные выполнения (par). Эти элементы значительно расширяют выразительные возможности диаграммы, позволяя моделировать сложные логические ветвления в рамках единого сценария.

Основная ценность диаграммы последовательности заключается в ее способности точно специфицировать поведенческие аспекты системы на уровне взаимодействия объектов. Она служит важным связующим звеном между требованиями к системе и ее фактической реализацией, позволяя разработчикам заранее проработать сложные сценарии взаимодействия. В процессе проектирования диаграмма помогает выявить потенциальные проблемы в логике взаимодействия компонентов, такие как избыточные сообщения, тупиковые ситуации или неоптимальные последовательности вызовов. При этом важно понимать, что диаграмма последовательности фокусируется именно на порядке взаимодействия объектов, а не на их внутренней структуре или алгоритмах обработки.  
Диаграмма последовательностей авторизации представлена на рисунке 3.

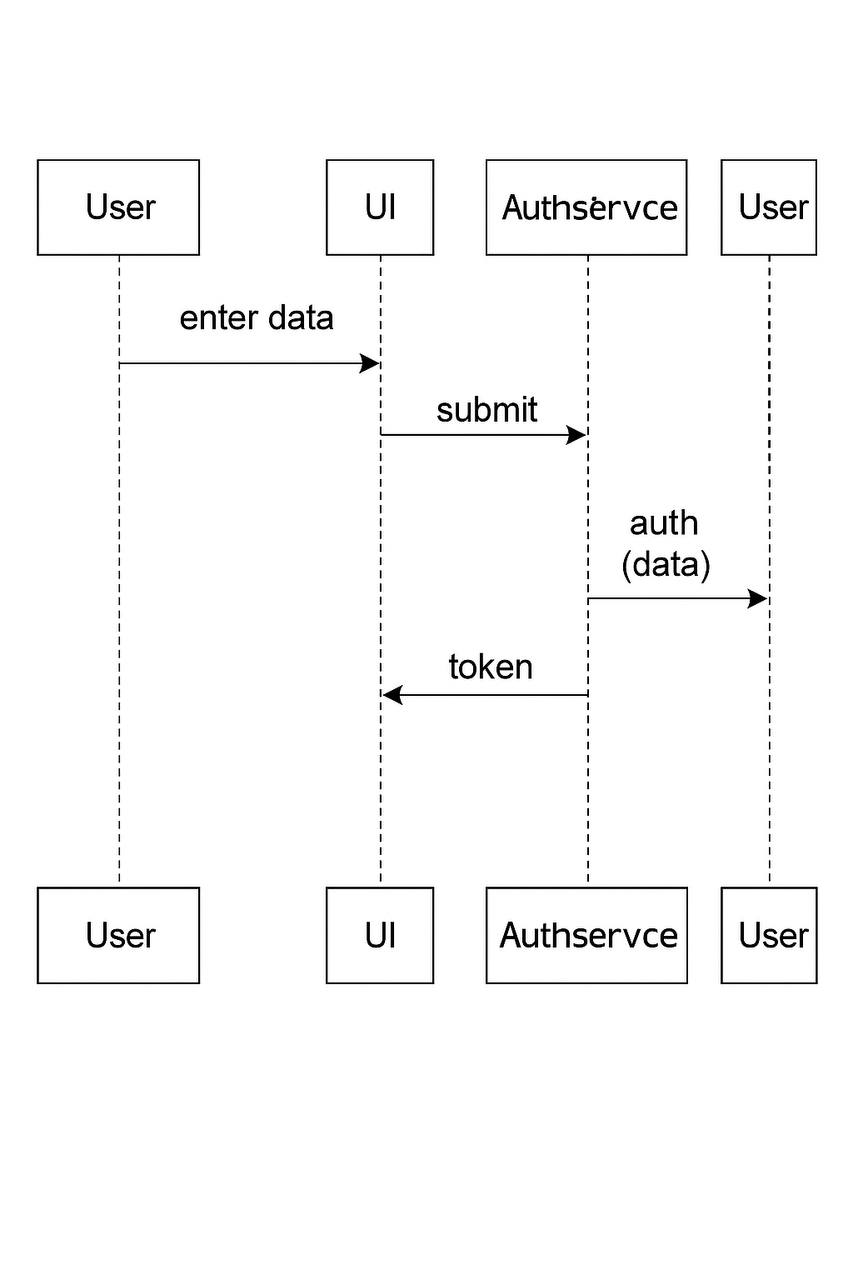


Рисунок 3 - Авторизация пользователя

Диаграмма последовательностей регистрации представлена на рисунке 4.

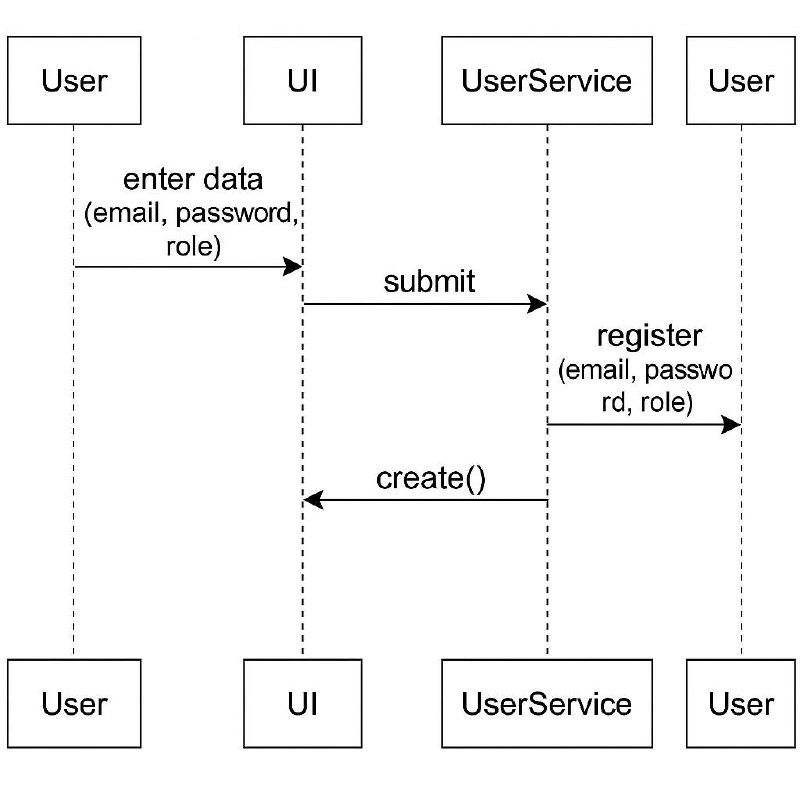


Рисунок 4 - Регистрация пользователя

## **Физическое моделирование системы**

Физическое моделирование представляет собой фундаментальный этап в разработке программного обеспечения, когда концептуальные проектные решения трансформируются в конкретную техническую реализацию. Этот процесс требует глубокого анализа множества аспектов, начиная от выбора архитектурного подхода и заканчивая детальной настройкой каждого компонента системы. В данном проекте реализована трехуровневая архитектура, которая обеспечивает оптимальный баланс между производительностью, масштабируемостью и безопасностью, что особенно важно для современных информационных систем, работающих под высокой нагрузкой.

Клиентский уровень системы построен на базе современных веб-технологий - HTML5, CSS3 и Vanilla JavaScript. Такой выбор обусловлен стремлением обеспечить максимальную производительность и отзывчивость пользовательского интерфейса, а также полный контроль над DOM. Отказ от использования тяжеловесных фреймворков позволяет минимизировать время загрузки и выполнения кода, что особенно важно для пользователей с медленным интернет-соединением или слабыми устройствами. Чистый JavaScript без дополнительных абстракций дает разработчику полный контроль над поведением приложения и позволяет создавать высокооптимизированные решения, хотя и требует более тщательной проработки архитектуры кода.

Серверный уровень реализован на Python с использованием фреймворка FastAPI, который сочетает в себе высокую производительность, удобство разработки и богатый функционал. FastAPI был выбран благодаря его асинхронной природе, автоматической генерации документации и встроенной поддержке валидации данных. Python как язык программирования обеспечивает читаемость кода, богатую экосистему библиотек и хорошую поддерживаемость проекта. Асинхронная модель выполнения позволяет эффективно обрабатывать множество одновременных запросов, что критически важно для современных веб-приложений. Встроенная система валидации данных на основе Pydantic значительно сокращает количество потенциальных ошибок и упрощает разработку API.

Для хранения данных выбрана PostgreSQL - надежная и функциональная реляционная СУБД. Ее преимущества включают поддержку сложных запросов, расширяемость через дополнительные модули, а также развитые механизмы обеспечения целостности данных. PostgreSQL отлично подходит для систем с интенсивной нагрузкой и сложной бизнес-логикой. Возможность создания сложных индексов, использование оконных функций и поддержка JSON-данных делают эту СУБД универсальным решением для большинства задач. Настроенные механизмы репликации и резервного копирования обеспечивают надежность хранения данных и минимальное время восстановления в случае сбоев.

Вся система контейнеризирована с помощью Docker, что обеспечивает идентичность окружений на всех этапах разработки и упрощает процесс развертывания. Docker позволяет легко масштабировать систему, управлять зависимостями и изолировать различные сервисы друг от друга. Использование контейнеров значительно упрощает процесс развертывания приложения в различных средах - от локальной машины разработчика до облачного кластера. Особое внимание уделено вопросам безопасности: реализована JWT-аутентификация, валидация всех входящих данных, защита от распространенных веб-уязвимостей. Все контейнеры построены на минимальных базовых образах, что уменьшает потенциальную поверхность для атак и снижает нагрузку на систему.

Производительность системы оптимизирована за счет использования асинхронных операций и грамотного проектирования базы данных. Реализованы механизмы для обработки фоновых задач через Celery, что позволяет разгрузить основной поток выполнения и улучшить отзывчивость системы. Архитектура системы предусматривает возможность горизонтального масштабирования для обработки возрастающей нагрузки. Все критические компоненты системы протестированы под нагрузкой, что позволяет гарантировать стабильную работу даже в пиковые периоды активности. Мониторинг ключевых метрик производительности реализован с помощью встроенных средств FastAPI и дополнительных инструментов профилирования.

Такой комплексный подход к физическому моделированию обеспечивает не только эффективную реализацию текущих требований, но и создает прочный фундамент для будущего развития системы. Гибкость выбранных решений позволяет относительно легко добавлять новые функции, адаптироваться к изменяющимся бизнес-потребностям и технологическим трендам. Система демонстрирует отличные показатели производительности при сохранении высокой надежности и безопасности, что подтверждается нагрузочным тестированием и анализом архитектуры. Дальнейшее развитие системы может включать внедрение дополнительных модулей, интеграцию с внешними сервисами и оптимизацию работы с большими объемами данных.

HTML является фундаментальной технологией веб-разработки и стандартным языком разметки для создания веб-страниц и приложений. Этот язык предоставляет структуру контенту в интернете, определяя элементы страницы и их взаимное расположение. HTML работает в связке с CSS для стилизации и JavaScript для интерактивности, образуя триаду основных веб-технологий.

Основное преимущество HTML заключается в его простоте и доступности для изучения. Язык использует интуитивно понятную систему тегов, которые оборачивают содержимое, придавая ему смысловое значение. Например, теги заголовков, параграфов, списков и таблиц позволяют логически организовать информацию на странице. HTML поддерживает все современные браузеры, что гарантирует кросс-платформенную совместимость.

Современная спецификация HTML5 привнесла значительные улучшения по сравнению с предыдущими версиями. Она включает семантические элементы, которые лучше описывают структуру документа. Эти элементы помогают поисковым системам понимать содержание страницы и улучшают доступность для пользователей с ограниченными возможностями. HTML5 также добавил встроенную поддержку мультимедиа, устраняя необходимость в сторонних плагинах для воспроизведения видео и аудио.

Важной особенностью HTML является его способность интегрироваться с различными API браузера. Это позволяет веб-страницам получать доступ к геолокации, работать с локальным хранилищем, выполнять фоновые задачи и устанавливать двустороннее соединение с сервером. HTML поддерживает адаптивный дизайн через метатеги и медиа-запросы, что делает сайты доступными на устройствах с любым размером экрана.

Язык постоянно развивается, добавляя новые возможности при сохранении обратной совместимости. HTML-документы легко проверяются на валидность с помощью специальных инструментов, что помогает поддерживать высокое качество кода. Благодаря своей универсальности и широкой поддержке, HTML остается незаменимым инструментом для создания веб-контента, обеспечивая стабильную основу для современных интернет-технологий.

CSS3 очень сильно повлиял на оформления веб-страниц. Среди ключевых преимуществ:

* гибкие системы макетов, которые позволяют создавать адаптивные интерфейсы без сложных хаков.
* продвинутые анимации и трансформации, заменяющие JavaScript-анимации в большинстве случаев.
* переменные, упрощающие поддержку единого стиля в больших проектах.
* медиа-запросы, обеспечивающие отзывчивый дизайн под любые устройства.

CSS3 также оптимизирует производительность, так как браузеры рендерят стили на уровне движка, а не через JavaScript, что снижает нагрузку на процессор.

JavaScript представляет собой высокоуровневый, интерпретируемый язык программирования, который стал неотъемлемой частью современной веб-разработки. Изначально созданный для добавления интерактивности на веб-страницы, JavaScript эволюционировал в мощный мультипарадигменный язык, используемый как на клиентской, так и на серверной стороне.

Ключевой особенностью JavaScript является его полная интеграция с веб-браузерами. Как единственный язык программирования, поддерживаемый всеми современными браузерами без необходимости компиляции или дополнительных плагинов, он позволяет создавать динамически обновляемый контент, обрабатывать пользовательские события и асинхронно взаимодействовать с серверами. Реализация через движки V8, SpiderMonkey и JavaScriptCore обеспечивает высокую производительность выполнения кода.

Современный JavaScript включает множество улучшений:

* классы и модули для лучшей организации кода
* стрелочные функции для лаконичного синтаксиса
* промисы и async/await для удобной работы с асинхронностью
* деструктуризацию и операторы расширения
* типизированные массивы для работы с бинарными данными

JavaScript поддерживает несколько парадигм программирования: объектно-ориентированную, функциональную и императивную. Это делает язык гибким и подходящим для различных стилей разработки. С появлением Node.js JavaScript вышел за пределы браузеров, позволив создавать серверные приложения, утилиты командной строки и даже настольные приложения.

Экосистема JavaScript включает огромное количество библиотек и фреймворков. React, Angular и Vue.js доминируют во фронтенд-разработке, тогда как Express, Nest.js и Koa популярны для бэкенда. Системы сборки и пакетные менеджеры облегчают разработку сложных приложений.

JavaScript продолжает развиваться, ежегодно получая новые возможности, сохраняя при этом обратную совместимость. Его универсальность, богатая экосистема и повсеместная поддержка делают JavaScript одним из самых востребованных языков программирования в мире.

Python представляет собой высокоуровневый интерпретируемый язык программирования общего назначения, который сочетает в себе простоту изучения с мощными возможностями. Созданный с акцентом на читаемость кода, Python использует интуитивно понятный синтаксис с минимальным количеством служебных символов, что делает его идеальным выбором как для начинающих, так и для опытных разработчиков.

* основные преимущества Python проистекают из его философии:
* чистый и выразительный синтаксис с обязательными отступами
* динамическая типизация с сильной проверкой типов
* автоматическое управление памятью
* поддержка нескольких парадигм программирования
* богатая стандартная библиотека

Python демонстрирует исключительную универсальность в различных областях применения:

* веб-разработка с использованием Django, Flask и FastAPI
* научные вычисления и анализ данных через NumPy, Pandas и SciPy
* машинное обучение и искусственный интеллект с TensorFlow и PyTorch
* автоматизация и написание скриптов
* разработка графических интерфейсов с помощью Tkinter и PyQt
* создание сетевых приложений и работа с API

Современные реализации Python (CPython, PyPy, Jython) обеспечивают высокую производительность и кроссплатформенность. Виртуальные окружения и система управления пакетами pip позволяют эффективно работать с зависимостями. Поддержка асинхронного программирования через asyncio делает Python конкурентоспособным для создания высоконагруженных приложений.

Экосистема Python включает более 300 тысяч пакетов в индексе PyPI, охватывающих практически все возможные области применения. Язык активно развивается, сохраняя обратную совместимость и ежегодно добавляя новые возможности. Его используют такие технологические гиганты как Google, NASA, Instagram и Spotify.

Благодаря сочетанию простоты, мощности и универсальности Python продолжает оставаться одним из самых популярных языков программирования, занимая лидирующие позиции в рейтингах TIOBE, Stack Overflow и GitHub. Его сообщество разработчиков — одно из самых активных и дружелюбных в индустрии, что способствует постоянному совершенствованию языка и его экосистемы.

FastAPI - это современный веб-фреймворк для создания API на Python, который сочетает в себе высокую производительность, простоту разработки и строгую типизацию данных. Построенный на основе Starlette для обработки запросов и Pydantic для валидации данных, этот фреймворк идеально подходит для разработки как небольших сервисов, так и сложных высоконагруженных систем. Одной из ключевых особенностей FastAPI является его асинхронная природа, которая позволяет эффективно обрабатывать множество одновременных подключений без блокировки основного потока выполнения, что особенно важно для современных веб-приложений с высокой нагрузкой. Фреймворк автоматически генерирует подробную интерактивную документацию API в форматах Swagger UI и ReDoc, основываясь на типах данных и аннотациях в коде, что значительно ускоряет процесс разработки и взаимодействия между фронтендом и бэкендом. Встроенная система валидации данных через Pydantic обеспечивает строгую типизацию входных и выходных параметров, автоматически преобразуя данные между Python-объектами и JSON, а также выдавая понятные сообщения об ошибках при несоответствии ожидаемым типам.

FastAPI поддерживает все возможности современного Python, включая асинхронные функции, type hints и dependency injection, что делает код более читаемым, поддерживаемым и менее подверженным ошибкам. Фреймворк имеет модульную архитектуру и легко интегрируется с различными базами данных через SQLAlchemy, Tortoise ORM или асинхронные драйверы, а также поддерживает аутентификацию через OAuth2, JWT-токены и другие механизмы безопасности. Благодаря использованию стандартов OpenAPI и JSON Schema, API, созданные на FastAPI, легко интегрируются с другими сервисами и клиентскими приложениями. Производительность FastAPI сравнима с Node.js и Go, при этом сохраняя все преимущества экосистемы Python, включая богатую коллекцию библиотек для машинного обучения, анализа данных и научных вычислений. Фреймворк активно развивается, имеет подробную документацию и большое сообщество разработчиков, что делает его надежным выбором для production-решений.

В отличие от традиционных Python-фреймворков, FastAPI изначально проектировался с учетом современных требований к веб-разработке, поддерживая WebSockets, GraphQL, фоновые задачи и другие передовые технологии. Его минималистичный, но выразительный синтаксис позволяет быстро создавать функциональные конечные точки, одновременно обеспечивая высокий уровень безопасности и производительности. FastAPI особенно хорошо подходит для микросервисной архитектуры, облачных решений и систем, где важны скорость отклика и эффективное использование ресурсов. Благодаря своей гибкости, производительности и удобству разработки, FastAPI заслуженно считается одним из лучших фреймворков для создания API на Python, объединяя в себе лучшие практики веб-разработки с мощью и простотой языка Python.

PostgreSQL - это мощная, объектно-реляционная система управления базами данных с открытым исходным кодом, которая сочетает в себе надежность корпоративных решений и гибкость современных технологий. Одним из ключевых преимуществ PostgreSQL является ее строгое соответствие принципам ACID (Atomicity, Consistency, Isolation, Durability), что гарантирует целостность данных даже в условиях высокой нагрузки или системных сбоев. Эта СУБД поддерживает сложные запросы, включая рекурсивные запросы и оконные функции, что делает ее идеальным выбором для аналитических задач и систем с интенсивной обработкой данных. PostgreSQL отличается расширяемой архитектурой — разработчики могут создавать собственные типы данных, операторы и функции на различных языках программирования, включая PL/pgSQL, Python, JavaScript и даже R. Система предлагает встроенную поддержку JSON и JSONB, что позволяет эффективно работать с полуструктурированными данными, сочетая преимущества реляционных и документоориентированных подходов.

Важной особенностью является развитая система индексов, включая B-деревья, GIN, GiST и BRIN, которые оптимизируют выполнение запросов для различных сценариев использования. PostgreSQL обеспечивает высокий уровень безопасности благодаря механизмам ролевого доступа, шифрованию соединений и детальному управлению правами пользователей. Система поддерживает репликацию (как синхронную, так и асинхронную), что позволяет создавать отказоустойчивые кластерные решения. Для работы с пространственными данными существует расширение PostGIS, превращающее PostgreSQL в полноценную геоинформационную систему.

Производительность PostgreSQL может масштабироваться за счет параллельного выполнения запросов и эффективного использования ресурсов многоядерных процессоров. Сообщество разработчиков постоянно совершенствует систему, добавляя новые функции, такие как декларативное секционирование таблиц, улучшенная обработка временных рядов и поддержка распределенных транзакций. PostgreSQL интегрируется со всеми популярными языками программирования и фреймворками, а его открытая лицензия позволяет свободно использовать систему в коммерческих проектах без ограничений. Благодаря сочетанию надежности, функциональности и производительности, PostgreSQL является оптимальным выбором для сложных корпоративных приложений, веб-сервисов с высокой нагрузкой и систем обработки больших объемов структурированных данных.

Docker представляет собой революционную платформу для контейнеризации приложений, которая коренным образом изменила подход к разработке и развертыванию программного обеспечения. В основе Docker лежит технология контейнеризации, позволяющая запускать изолированные процессы на уровне операционной системы, что обеспечивает высокую производительность и эффективность использования ресурсов. Ключевым преимуществом Docker является его способность создавать легковесные и самодостаточные контейнеры, содержащие все необходимое для работы приложения. Это устраняет проблему различий между средами выполнения, так как контейнер гарантированно работает одинаково на любой поддерживающей системе.

Docker использует многослойную файловую систему, что позволяет эффективно управлять образами и ускорять процесс развертывания. Одной из мощных возможностей является Docker Compose, который позволяет описывать многоконтейнерные приложения и управлять ими как единым целым. Это особенно полезно для сложных систем, состоящих из нескольких взаимосвязанных сервисов. Интеграция с Kubernetes предоставляет инструменты для оркестрации контейнеров, позволяя автоматизировать развертывание и масштабирование распределенных систем.

Важным аспектом является экосистема Docker, включающая репозиторий готовых образов, где можно найти официальные образы для большинства популярных технологий, что значительно ускоряет процесс разработки. С точки зрения безопасности Docker предоставляет механизмы изоляции контейнеров и контроля ресурсов. Для разработчиков предлагаются удобные инструменты для отладки и мониторинга работы контейнеров.

В производственной среде Docker интегрируется с системами непрерывной интеграции, позволяя автоматизировать весь процесс разработки. Особенностью является кроссплатформенность - он работает на различных операционных системах и поддерживается всеми крупными облачными провайдерами. Для управления конфигурацией Docker предоставляет специальные механизмы. Оптимизация образов достигается за счет использования многоступенчатых сборок и выбора минимальных базовых образов.

Docker поддерживает различные сетевые драйверы, что позволяет гибко настраивать взаимодействие между контейнерами. В современных практиках Docker стал фактическим стандартом, обеспечивающим согласованность сред разработки и production. Его модульная архитектура идеально подходит для микросервисных приложений, где каждый сервис может быть упакован в отдельный контейнер. Благодаря своей гибкости и производительности, Docker остается одной из самых востребованных технологий в современной разработке программного обеспечения.

Alembic представляет собой профессиональную систему управления миграциями, созданную специально для работы с SQLAlchemy. Этот инструмент стал стандартом де-факто для Python-разработчиков, работающих с реляционными базами данных. Основная задача Alembic — обеспечить контролируемое и предсказуемое изменение схемы базы данных по мере развития приложения. В отличие от многих аналогов, Alembic не пытается быть универсальным решением, а предлагает глубокую интеграцию именно с экосистемой SQLAlchemy, что делает его особенно ценным для сложных проектов.

Работа с Alembic строится вокруг концепции явных миграций. Каждое изменение структуры базы данных описывается в виде отдельного Python-скрипта, содержащего функции upgrade и downgrade. Такой подход обеспечивает полную прозрачность изменений и возможность точного контроля над процессом эволюции базы данных. Особенно мощной особенностью является функция автогенерации миграций, которая анализирует различия между моделями SQLAlchemy и текущим состоянием базы данных, значительно сокращая объем ручной работы.

Alembic поддерживает все основные реляционные СУБД, включая PostgreSQL, MySQL, SQLite и Oracle, сохраняя при этом единый интерфейс работы. Инструмент предоставляет развитые механизмы для работы с транзакциями, ветвлением миграций и обработкой данных при изменении схемы. Встроенная система версионирования позволяет точно отслеживать состояние базы данных на любом окружении, что критически важно для командной разработки и промышленного развертывания приложений.

Pydantic установил новые стандарты работы с данными в Python-экосистеме. Эта библиотека коренным образом изменила подход к валидации и парсингу данных, предложив элегантное решение на основе аннотаций типов. Pydantic V2, с его Rust-ядром, обеспечивает невероятную производительность, делая валидацию данных практически бесплатной с точки зрения вычислительных ресурсов. Библиотека стала неотъемлемой частью современных Python-фреймворков, особенно FastAPI, где используется как стандартная система валидации запросов и ответов.

Основная сила Pydantic заключается в его простоте и выразительности. Модели данных описываются как обычные Python-классы с аннотациями типов, а вся сложная логика валидации скрыта под капотом. Библиотека поддерживает все стандартные типы Python, а также предлагает богатый набор специализированных типов для работы с email-адресами, URL, UUID и другими распространенными форматами данных. Особенно ценным является автоматическое преобразование данных между различными представлениями — например, из JSON в Python-объекты и обратно.

Pydantic нашел применение в самых разных областях — от валидации API-запросов до управления конфигурацией приложений. Его способность автоматически генерировать JSON-схемы делает его идеальным инструментом для разработки веб-API. Встроенная поддержка .env-файлов через класс BaseSettings значительно упрощает работу с конфигурацией. Расширенные возможности, такие как кастомные валидаторы, динамические модели и наследование схем, позволяют решать даже самые сложные задачи работы с данными, сохраняя при этом чистоту и читаемость кода.

## **Проектирование базы данных**

Проектирование базы данных представляет собой ключевой этап в процессе создания информационной системы, так как именно от архитектуры базы зависит стабильность, производительность, масштабируемость и удобство поддержки всей системы. Грамотно спроектированная база данных обеспечивает корректное хранение информации, её целостность, минимизирует избыточность и обеспечивает высокую скорость доступа к данным. В рамках данного проекта была разработана база данных, поддерживающая работу автоматизированной системы подбора кандидатов на должность backend-разработчика.

### **2.3.1 Общая характеристика предметной области**

Предметная область охватывает процессы подбора технических специалистов с упором на backend-разработчиков. Основными действующими лицами являются:

* кандидат - человек, желающий получить работу, обладающий набором профессиональных навыков;
* hr-менеджер - сотрудник компании, создающий вакансии и анализирующий отклики кандидатов;
* информационная система - посредник, позволяющий связывать вакансии с кандидатами и обеспечивающий автоматизированную проверку на соответствие.

Работа системы строится на взаимодействии следующих объектов:

* профиль кандидата содержит информацию о навыках, опыте и предпочтениях;
* вакансия содержит описание позиции и список обязательных навыков;
* отклик содержит результат анализа соответствия навыков кандидата требованиям вакансии;
* тесты позволяют проверить реальные знания кандидата по каждой совпавшей компетенции.

Таким образом, база данных должна моделировать все указанные объекты и отношения между ними.

### **2.3.2 Подход к проектированию базы данных**

Проектирование базы данных выполнялось с учётом следующих принципов:

* абстрагирование - отбор только тех сущностей, которые действительно отражают бизнес-логику;
* нормализация - разбиение данных на логически независимые сущности для исключения дублирования;
* соблюдение ограничений целостности - для недопущения логических ошибок (например, отклик без связанной вакансии);
* минимизация связности - для повышения гибкости и снижения риска каскадных изменений;
* логическое и физическое разделение проектирования - сначала создаётся концептуальная модель, затем она трансформируется в реализацию через ORM;
* использование асинхронного доступа - для масштабируемости и высокой производительности системы.

### **2.3.3 Список сущностей и их функциональное назначение**

В результате анализа предметной области были выделены следующие сущности:

* пользователь - представляет собой любую зарегистрированную единицу в системе (hr или кандидат). Хранит авторизационные и контактные данные.
* кандидат - расширение пользователя, обладающее профилем с навыками и историей откликов.
* hr - расширение пользователя, имеющее доступ к управлению вакансиями.
* вакансия - описание открытой должности, включающее требования, описание компании и перечень нужных навыков.
* навык - отдельная единица профессиональной квалификации. Каждый навык может быть связан с несколькими вакансиями и профилями.
* отклик - действие, инициированное кандидатом, представляющее интерес к вакансии. Хранит процент совпадения, дату отклика, статус.
* тест - набор заданий, автоматически формируемых по каждому навыку при достаточном совпадении между кандидатом и вакансией.
* результат теста - данные о попытке прохождения теста: завершённость, оценка, дата выполнения.
* вопрос - элемент теста, включающий текст вопроса, варианты ответов и правильный ответ.
* профиль кандидата - сущность, хранящая список связанных с кандидатом навыков.
* связь "вакансия - навык" - реализует отношение "многие ко многим" между вакансиями и навыками.
* связь "профиль - навык" - позволяет указать, какие навыки есть у кандидата.
* связь "отклик - тест" - привязывает отклик к отправленным тестам.

### **2.3.4 Структура и типы связей между сущностями**

В проектируемой базе данных реализуются следующие типы связей:

* один к одному - между пользователем и профилем (кандидат или hr);
* один ко многим - между hr и вакансиями, между вакансией и откликами;
* многие ко многим - между вакансиями и навыками, между профилями кандидатов и навыками;
* один ко многим - между откликом и тестами, между тестом и вопросами.

Реализация отношений "многие ко многим" осуществляется посредством промежуточных таблиц, каждая из которых содержит внешние ключи на связываемые сущности и, при необходимости, дополнительные атрибуты.

### **2.3.5 Обоснование выбора атрибутов и типов данных**

Для каждой сущности были определены ключевые атрибуты, удовлетворяющие следующим критериям:

* минимальность - каждый атрибут содержит атомарное значение;
* уникальность - первичные ключи обеспечивают однозначную идентификацию каждой записи;
* согласованность - каждый атрибут имеет чётко определённый тип данных, соответствующий своей природе;
* нормализованность - отсутствует повторение данных между таблицами.

Примеры:

* у пользователя: идентификатор (integer), электронная почта (varchar), дата регистрации (timestamp), роль (enum);
* у вакансии: идентификатор (integer), название (varchar), описание (text), идентификатор hr (foreign key);
* у отклика: идентификатор (integer), процент совпадения (float), статус (enum), внешний ключ на кандидата и вакансию;
* у навыка: уникальное название (varchar);
* у теста: идентификатор (integer), связан с откликом и навыком, статус выполнения (boolean), оценка (float);
* у вопроса: текст вопроса (text), список опций (jsonb), индекс правильного ответа (integer).

В таблице 3 описаны сущности и их атрибуты.

Таблица 3 - Сущности и атрибуты базы данных

|  |  |  |
| --- | --- | --- |
| Имя сущности | Описание | Тип |
| Пользователь | id пользователя | primary key |
|  | email | текстовый |
|  | hashed\_password | текстовый |
|  | роль | текстовый |
| Кандидат | id профиля кандидата | primary key |
|  | id пользователя | внешний ключ |
| HR | id профиля HR | primary key |
|  | id пользователя | внешний ключ |
| Вакансия | id вакансии | primary key |
|  | заголовок | текстовый |
|  | компания | текстовый |
|  | описание | текстовый |
|  | id HR-профиля | внешний ключ |
| Навык | id навыка | primary key |
|  | название навыка | текстовый |
| Навыки вакансии | id вакансии | внешний ключ |
|  | id навыка | внешний ключ |
| Навыки кандидата | id профиля кандидата | внешний ключ |
|  | id навыка | внешний ключ |
| Отклик | id отклика | primary key |
|  | id профиля кандидата | внешний ключ |
|  | id вакансии | внешний ключ |
|  | статус | текстовый |
|  | процент совпадения | числовой |
| Тест отклика | id теста | primary key |
|  | id отклика | внешний ключ |
|  | id навыка | внешний ключ |
|  | завершён | логический |
|  | оценка | числовой |
| Вопрос | id вопроса | primary key |
|  | id навыка | внешний ключ |
|  | текст вопроса | текстовый |
|  | варианты ответов | json |
|  | индекс правильного ответа | целочисленный |

### **2.3.6 Хранение вложенных структур**

Особенностью проектируемой базы данных является необходимость хранения списков (варианты ответов в вопросах). Для этого в PostgreSQL используется тип данных jsonb, позволяющий:

* сохранять списки строк и структур в виде одного поля;
* осуществлять фильтрацию, индексирование и извлечение значений;
* обеспечивать гибкость хранения и передачи данных в API.

### **2.3.7 Обеспечение целостности данных**

Целостность базы данных обеспечивается следующими механизмами:

* первичные ключи на каждой сущности;
* внешние ключи между связанными таблицами;
* каскадные удаления в ассоциативных таблицах;
* ограничения уникальности;
* ограничения not null на обязательных полях;
* использование перечислений (enum) для полей статуса, ролей и категорий.

### **2.3.8 Масштабируемость и производительность**

Архитектура базы данных спроектирована с учётом следующих требований к масштабируемости:

* использование индексов на часто запрашиваемые поля (например, email, внешние ключи);
* нормализованная модель, позволяющая расширять систему без дублирования данных;
* асинхронный доступ к базе, позволяющий обрабатывать множество параллельных запросов;
* кэширование данных на уровне приложения для уменьшения нагрузки на БД;
* потенциальная возможность горизонтального масштабирования базы за счёт использования реплик.

### **2.3.9 Интеграция с backend и API**

Все модели базы данных реализованы с использованием SQLAlchemy ORM и сопровождаются Pydantic-схемами. Это обеспечивает:

* строгую типизацию данных;
* автоматическую валидацию при получении и отправке данных через API;
* удобную сериализацию и десериализацию;
* однозначную связь между логической моделью и структурой таблиц.

### **2.3.10 Перспективы расширения модели данных**

Концепция проектирования предусматривает возможность расширения базы данных без необходимости полного рефакторинга. Например:

* добавление сущностей "комментарии", "собеседования", "приглашения";
* хранение резюме кандидатов;
* внедрение версионирования вакансий;
* автоматическое логирование всех действий пользователей.

Такая гибкость достигается за счёт соблюдения принципов нормализации и слабой связности компонентов.

В результате проектирования была создана полнофункциональная база данных, удовлетворяющая всем требованиям предметной области. Она обеспечивает корректное моделирование бизнес-процессов, устойчивость к ошибкам, высокую производительность и гибкость. База данных является надёжной основой для функционирования всей информационной системы, а её структура обеспечивает лёгкость сопровождения и масштабирования проекта.

## **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.

## **Вывод 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.

## **Вывод 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/>