

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	5
1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ.....	6
1.1 Описание предметной области.....	6
1.2 Обзор существующих решений	7
1.3 Постановка списка задач	10
1.4 Обоснование необходимости разработки системы.....	11
2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ.....	13
2.1 Архитектура системы.....	13
2.2 Используемые технологии.....	14
3 РАЗРАБОТКА.....	17
3.1 Клиент	17
3.2 Сервер	32
3.3 База данных	46
3.4 Развертывание.....	49
4 ТЕСТИРОВАНИЕ	51
5 РАСЧЕТ ТЕХНИКО-ЭКОНОМИЧЕСКИХ ПОКАЗАТЕЛЕЙ.....	57
5.1 Исходные данные для осуществления расчета	57
5.2 Расчет объема функций	57
5.3 Расчет полной себестоимости программного модуля	59
5.4 Расчет отпускной цены и чистой прибыли	65
ЗАКЛЮЧЕНИЕ.....	67
СПИСОК СОКРАЩЕНИЙ	68
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	69
ПРИЛОЖЕНИЕ А. ТЕКСТ ПРОГРАММЫ	

					<div>ДП.ИИ22.210586 – 05 81 00</div>			
Изм.	Лист	№ докум.	Подпись	Дата				
Разработал	Копанчук Е.Р.				<div>Разработка метода интеграции большой языковой модели для ана- лиза ответов пользователя в рамках профориентационного тестирования. Пояснительная записка</div>	Стади	Лист	Листов
Проверил	Михно Е.В.					Д	4	71
Реценз.	Парфомук С.И.					<div>УО БрГТУ</div>		
Н. контроль	Михно Е. В.							
Утвердил	Головки В. А.							

ВВЕДЕНИЕ

Профориентация является важным этапом в жизни каждого человека, поскольку выбор направления обучения и будущей профессиональной деятельности оказывает значительное влияние на карьеру и личную удовлетворённость. Успешный выбор профессии способствует раскрытию потенциала личности, удовлетворению её профессиональных амбиций и достижению гармонии в жизни. В свою очередь, неверное решение может привести к неудовлетворённости, низкой эффективности и необходимости изменения профессионального пути.

В современном мире существует множество систем и методик, направленных на помощь в выборе профессии. Профориентационные тесты, консультации с экспертами и образовательные платформы предлагают разнообразные подходы, начиная от анкетирования и анализа предпочтений до сложных алгоритмов, использующих современные технологии искусственного интеллекта. Однако, несмотря на их обширное распространение, многие из этих систем обладают ограниченной адаптивностью и не учитывают в полной мере индивидуальные особенности пользователей.

В рамках данной дипломной работы будет разработана MVP версия системы профориентационного тестирования, основанное на интеграции большой языковой модели. Применение LLM позволяет анализировать ответы пользователей на естественном языке, что помогает расширить возможности тестирования оценивать пользователя в соответствии с поставленными критериями.

Разрабатываемая система будет представлять собой веб-приложение, которое обеспечит интуитивно понятный интерфейс для разных категорий пользователей. Учащиеся смогут пройти тестирование и получить персонализированные рекомендации, а представители учебных заведений – управлять информацией о доступных специальностях и факультетах.

Настоящая работа будет направлена на реализацию процесса взаимодействия пользователя с системой и накоплении первичных данных для оценки предложенного в данной работе метода.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Описание предметной области

Профориентационное тестирование представляет собой комплекс мероприятий и инструментов, направленных на определение оптимальной профессиональной траектории личности. Основываясь на психологических, социологических и образовательных методиках, такие системы помогают абитуриентам и студентам оценить собственные интересы, способности и ценностные ориентиры. В условиях динамично меняющегося рынка труда и быстрого развития технологий адекватная профориентация становится критическим фактором, определяющим успешность профессиональной реализации и личностного развития.

Вот компактный ряд факторов, которые должен оценить тест на профориентацию.

- 1 Психологические особенности
 - 1.1 Индивидуальные интересы, склонности и увлечения
 - 1.2 Личностные качества, такие как коммуникабельность, стрессоустойчивость и уровень мотивации
- 2 Когнитивные способности
 - 2.1 Уровень интеллекта, аналитическое мышление, скорость обработки информации
 - 2.2 Творческий потенциал и способность к нестандартному решению задач
- 3 Ценностные ориентации и мотивация
 - 3.1 Приоритеты в выборе профессии. Материальная выгода, самореализация, статус
 - 3.2 Готовность к обучению и адаптация к изменениям
- 4 Профессиональные интересы
 - 4.1 Предпочтения в области деятельности (гуманитарная, техническая, творческая и т.д.)
 - 4.2 Осведомленность о профессиональных областях и карьерных траекториях
- 5 Социальные и внешние факторы
 - 5.1 Учет влияния окружения. Семья, друзья, тренды на рынке труда
 - 5.2 Доступность образования и ресурсов для освоения профессии
- 6 Сопоставление личных качеств с профессиональными требованиями
 - 6.1 Анализ соответствия между результатами теста и реальными

условиями работы в выбранной сфере

6.2 Оценка устойчивости интересов и способностей в долгосрочной перспективе

7 Актуальность и адаптация к изменениям

7.1 Учет современных тенденций, таких как цифровизация и автоматизация

7.2 Включение навыков будущего. Критическое мышление, работа с данными, креативность

8 Обратная связь и рекомендации

8.1 Генерация детализированных отчетов и индивидуальных рекомендаций

8.2 Возможность получения консультаций у экспертов.

Помимо этого нужно учитывать региональное нахождение абитуриента, его предпочтения в способе обучения, требования к учебному заведению. При выборе университета важно учитывать его репутацию и аккредитацию, качество образовательных программ, квалификацию преподавательского состава и доступ к современным ресурсам (лаборатории, библиотеки, общежития). Важны также финансовые аспекты: стоимость обучения, доступность грантов и стипендий. Стоит учитывать расположение университета, условия проживания и транспортную инфраструктуру. Немаловажны возможности для профессионального развития: стажировки, карьерные центры, связи с работодателями, а также международная деятельность (программы обмена и признание дипломов). Дополнительно стоит обратить внимание на студенческую жизнь, внеучебные активности и перспективы трудоустройства выпускников.

Современный рынок труда характеризуется высокой изменчивостью данных: многие профессии устаревают или трансформируются, а новые появляются с огромной скоростью. Это требует от систем профориентации постоянного обновления информации и адаптации рекомендаций. Важно учитывать также изменчивость востребованности профессий на рынке, вызванную глобализацией, автоматизацией и локальными экономическими трендами. Без учета этих факторов профориентационные тесты рискуют дать рекомендации, не соответствующие актуальным требованиям или перспективам карьерного развития.

1.2 Обзор существующих решений

В настоящее время в сети интернет есть множество вариаций тестов стремящихся решить нашу судьбу и подобрать будущую профессию. Вот

следующие категории.

Тесты в формате “Сможете ли вы стать ИТ-специалистом?”

Эти тесты фокусируются на оценке готовности пользователя к работе в сфере информационных технологий. Обычно они включают вопросы, связанные с базовыми знаниями программирования, математикой, логическим мышлением, а также с представлениями о повседневных задачах ИТ-специалистов. Часто такие тесты ориентированы на широкий круг пользователей и имеют целью популяризацию профессий в области технологий.

Плюсы.

- Доступность и простота. Не требуют специальной подготовки или глубоких знаний для прохождения.

Минусы.

- Низкий охват специальностей. По итогу результата теста пользователь отбрасывает либо находит одну профессию, что не подходит для выбора профессии с нуля.

- Ограниченная глубина анализа. Такие тесты чаще всего оценивают поверхностные навыки и не выявляют истинных способностей или предпочтений.

- Низкая персонализация. Рекомендации универсальны и не учитывают индивидуальные особенности пользователя.

- Устаревшие критерии. Тесты редко обновляются, что может привести к несовпадению с актуальными требованиями.

- Нет информации о месте освоения данной специальности.

Тест «16 типов личности» (MBTI)

Этот тест направлен на определение типа личности на основе теории Карла Юнга, выделяя предпочтения в восприятии информации, принятии решений и организации жизни. Тест включает четыре ключевые дихотомии [8].

- Экстраверсия (E) vs. Интроверсия (I). Ориентация на внешний или внутренний мир.

- Ощущение (S) vs. Интуиция (N). Работа с фактами или абстракциями.

- Мышление (T) vs. Чувствование (F). Логика или эмоции в принятии решений.

- Суждение (J) vs. Восприятие (P). Структурированность или гибкость в организации жизни.

Результатом является один из 16 типов личности, таких как ISTJ или ENFP, с детальным описанием особенностей, сильных и слабых сторон.

Плюсы.

- Универсальность. Подходит для анализа как личной жизни, так и профессиональной траектории.

- Детализированные результаты. Включают рекомендации по карьере и личностному развитию.
- Популярность и доступность. Множество адаптированных версий и интерпретаций.

Минусы.

- Нет информации о месте освоения данной специальности.
- Субъективность ответов. Результаты зависят от честности и самовосприятия участника.
- Ограниченная актуальность для выбора профессии. Тест больше описывает личные предпочтения, чем карьерные возможности.
- Стереотипизация. Типы личности могут восприниматься как ярлыки, ограничивающие развитие.
- Нет учета изменений в личностных качествах. Тест не учитывает развитие личности со временем.

Тест по методике Климова

Этот тест направлен на определение профессиональных склонностей и предпочтений, основанных на взаимодействии человека с разными объектами профессиональной деятельности. Климов выделяет пять типов профессий [14].

- 1 Человек – Природа. Интерес к работе с живыми существами и природой (эколог, ветеринар)
- 2 Человек – Техника. Предпочтение к работе с машинами и технологиями (инженер, программист)
- 3 Человек – Знаки. Склонность к абстрактным задачам, числам и алгоритмам (математик, экономист)
- 4 Человек – Человек. Ориентация на взаимодействие и помощь людям (педагог, психолог)
- 5 Человек – Художественный образ. Стремление к творчеству и самовыражению (дизайнер, музыкант).

Тест состоит из вопросов, определяющих, какой тип деятельности наиболее привлекателен для участника.

Плюсы.

- Простота. Подходит для всех возрастов, не требует подготовки.
- Объективность. Методика базируется на исследованиях психологии труда.
- Гибкость. Помогает выявить комбинированные склонности.

Минусы.

- Нет информации о месте освоения данной специальности.
- Ограниченность категорий. Может не учитывать современные

гибридные профессии.

- Отсутствие индивидуализации. Результаты зависят от интерпретации общих категорий.
- Устаревание. Разработка 20 века не всегда отражает актуальные изменения на рынке труда.

Методика Климова часто используется совместно с другими тестами, такими как тест Холланда, для получения более полной картины профессиональных предпочтений.

1.3 Постановка списка задач

Изучив существующие решения, я выделил ряд ключевых факторов, которых мне не хватает в вышеупомянутых тестах определяющих профориентацию.

Рассмотренные тесты.

- Не имеют информации о месте получения специальности.
- Дают узкое или нерелевантное представление о профессии, её востребованности на рынке.
- Опираются на субъективные критерии и качества.

Абитуриенты, не обладая полной информацией о возможных образовательных траекториях, нуждаются в системе, которая за короткое время сможет предложить рекомендации, основанные на их индивидуальных характеристиках, и предоставить исчерпывающие данные об образовательных учреждениях для более глубокого изучения. Исходя из этого, можно выделить следующие ключевые требования к системе.

1 Скорость получения предложений. Как пользователь я хотел бы не тратить много времени на прохождение теста, а в течении 10 минут уже ознакомиться с тем, что мне может предложить продукт

2 Ранжирование вместо исключения. Метрика теста должна сортировать специальности от самой подходящей пользователю до самой нерелевантной. Как ни крути тестирование не может иметь неоспоримую точность, поэтому нужно дать пользователю возможность выбирать среду подобранных вариантов

3 Расширяемость базы данных. Должна быть возможность добавления учебных заведений и изменения информации о них

4 Архитектура системы не должна быть зависима от формата тестирования. Создание наиболее точного и подходящего теста сложная итерационная задача, поэтому код системы должен быть написан так, чтобы

изменение теста касалось минимального количества компонентов

5 Специальности вместо профессий. Абитуриент ищет место куда поступать прежде всего, поэтому в результате нужно учитывать прежде всего учебное заведение и его особенности

6 Использование LLM. LLM имеют ряд недостатков при решении таких задач, например, они не имеют доступ к релевантной информации о специальностях, а также не имеют информации о пользователе и студентах подбираемых специальностей. С помощью использования других классических инструментов нужно закрыть эти недостатки.

Теперь давайте подытожим и посмотрим на функциональные требования к интерфейсу системы.

- 1 Регистрация пользователя студент
- 2 Регистрация пользователя университет
- 3 Редактирование профиля студент
- 4 Редактирование профиля университет
- 5 Просмотр информации об университете
- 6 Прохождение студентом теста
- 7 Просмотр результатов тестирования
- 8 Формирование рекомендаций на основе тестирования
- 9 Поиск по учебным учреждениям.

1.4 Обоснование необходимости разработки системы

Современный образовательный процесс сталкивается с рядом вызовов, обусловленных изменениями в социально-экономической и технологической среде. Среди ключевых факторов можно выделить. Динамичные изменения на рынке труда. Постоянное обновление требований к квалификации связано с автоматизацией процессов и появлением новых технологий. Абитуриенты вынуждены принимать решения в условиях неопределённости. Ускоренное развитие технологий. Новые профессии возникают быстрее, чем адаптируются образовательные программы, что создаёт разрыв между спросом и предложением. Недостаток поддержки. Абитуриенты испытывают трудности с доступом к достоверной информации о программах и перспективах трудоустройства. Часто предлагаемые решения не учитывают их индивидуальных особенностей. Возникает необходимость создания системы, персонализирующей профориентацию и предоставляющей качественную обратную связь.

Изм	Лист	№ докум	Подп.	Дата

ДП.ИИ22.210586 – 05 81 00

Лист

11

Роль больших языковых моделей (LLM). Интеграция LLM в профориентационные платформы открывает возможности для анализа ответов пользователей и формирования рекомендаций. Обработка естественного языка. LLM интерпретируют текстовые ответы, выявляют предпочтения и формируют рекомендации. Персонализация тестирования. Адаптивные алгоритмы уточняют индивидуальные особенности пользователя. Анализ данных. Модели обрабатывают данные о профессиях и учебных заведениях для интеграции в рекомендации. Однако, модели ограничены в знании специфики отдельных учебных заведений, что требует. Интеграции базы данных. Создание актуальной базы данных о специальностях и условиях поступления. Комбинированного подхода. Сочетание LLM с алгоритмами обработки структурированных данных.

Преимущества для образовательных учреждений.

Система также ориентирована на запросы университетов. Управление информацией. Учреждения смогут обновлять данные о программах и условиях поступления через удобный интерфейс.

Адресное взаимодействие. Персонализированные коммуникации с потенциальными студентами.

Укрепление связей. Система помогает устанавливать прочные связи с абитуриентами, способствуя осознанному выбору.

Универсальность и эффективность. Разрабатываемая система сочетает технологии искусственного интеллекта и методы обработки данных.

Поддержка выбора. Помогает учащимся осознанно выбирать образовательный путь с учётом их целей и возможностей.

Актуальная информация. Предоставляет полные данные о программах и перспективах трудоустройства.

Содействие университетам. Обеспечивает инструмент для управления информацией и взаимодействия с абитуриентами.

Таким образом, система удовлетворяет потребности учащихся в профориентации и способствует укреплению связей между образовательными учреждениями и их целевой аудиторией, отвечая на вызовы времени.

2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ

2.1 Архитектура системы

Архитектура программной системы представляет собой структуру, которая определяет взаимодействие её компонентов, их функциональное назначение и способы обмена данными. От качества проектирования архитектуры зависит масштабируемость, надежность, производительность и удобство эксплуатации системы. Для создания системы профориентационного тестирования архитектура играет ключевую роль, так как она должна обеспечивать эффективное взаимодействие пользователей, обработку данных и управление информацией.

Основными компонентами любой архитектуры являются клиент, сервер и база данных. Клиент – это приложение или интерфейс, через который пользователь взаимодействует с системой. Клиенты могут быть веб-приложениями, мобильными приложениями или десктопными программами. Сервер отвечает за обработку запросов клиентов, выполнение бизнес-логики и управление данными. Он является связующим звеном между клиентом и базой данных. База данных (БД) – это система хранения и управления структурированными данными, которые необходимы для работы системы, включая информацию о пользователях, учебных заведениях и результатах тестирования.

В данной главе будет рассмотрено проектирование архитектуры системы профориентационного тестирования. Особое внимание уделено роли каждого компонента, их взаимодействию, а также выбору технологий и инструментов, обеспечивающих эффективную работу и дальнейшее развитие системы.

В качестве клиента нашей системы будет выступать веб-сайт. Клиент будет реализовываться с помощью NextJs. В качестве библиотеки компонентов интерфейса используются MaterialUI и AntdDesign. Авторизация на стороне клиента и сервера будет реализована через NextAuth JWT токены. В качестве ORM используется Prisma. Для валидации форм используется React Hook Forms и Zod. Генерация id с помощью uuid [5].

В качестве сервера у нас будут выступать api роуты NextJs. В целом клиент и сервер представляют собой монолит. Код, который должен исполняться на сервере, такой как api роуты, остается на сервере, в то время как клиентский код отправляется и выполняется на стороне пользователя.

В качестве БД будет PostgreSQL.

Приложение на NextJS и База данных будут завернуты в отдельные Docker контейнеры. Сборка приложения будет осуществляться через Docker Compose [4].

2.2 Используемые технологии

Используемые технологии (рисунок 2.1):

- 1 Prisma ORM
- 2 Next.js
- 3 PostgreSQL
- 4 Material-UI
- 5 Ant Design
- 6 Redux Toolkit
- 7 PerplexityAI
- 8 NextAuth.js.

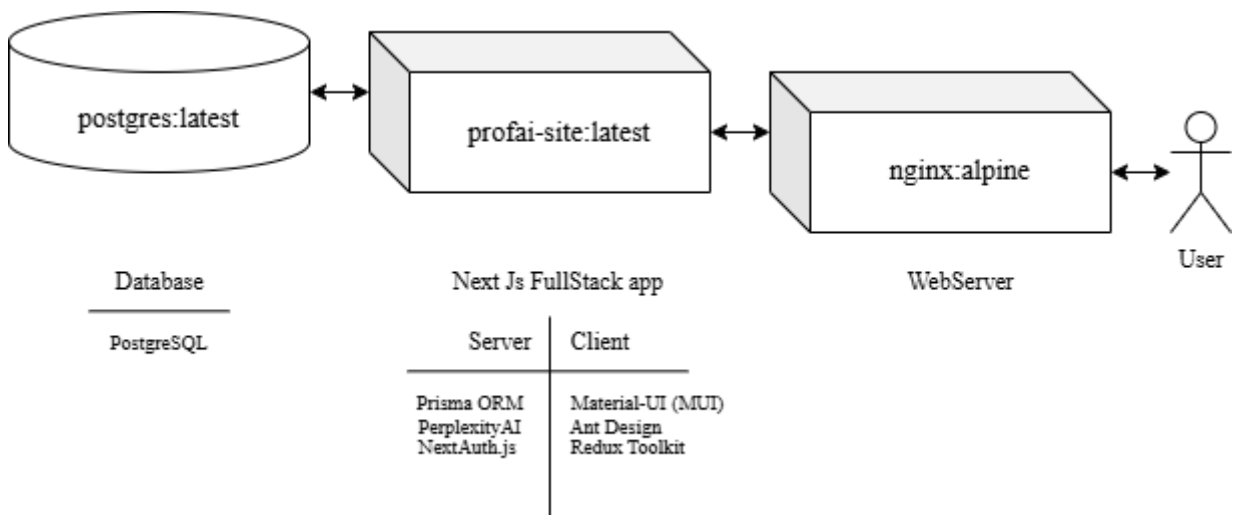


Рисунок 2.1 – Архитектура системы

Prisma ORM представляет собой современный инструмент для работы с базами данных в среде Node.js и TypeScript [12]. Он обеспечивает генерацию типобезопасного клиента на основании схемы данных, что значительно сокращает количество ошибок на этапе разработки и упрощает автодополнение в IDE. Благодаря удобному DSL для описания моделей данных Prisma позволяет легко мигрировать схему БД и поддерживать её в актуальном состоянии. Интеграция с PostgreSQL обеспечивает высокую производительность и надёжность при запросах к наиболее критичным таблицам. Использование Prisma в проекте ускоряет разработку бизнес-логики и улучшает читаемость кода.

Next.js – это React-фреймворк, который сочетает возможности серверного рендеринга, статической генерации и маршрутизации из коробки [1]. Он упрощает организацию файловой структуры проекта и позволяет создавать динамические маршруты для страниц и API-роутов без дополнительной конфигурации. Благодаря

поддержке Turbopack и встроенной оптимизации изображений, сборок и кода, Next.js обеспечивает высокую производительность и SEO-дружественность веб-приложения. Функциональность API-роутов облегчает реализацию серверной логики рядом с клиентским кодом, что соответствует архитектуре монолита. Широкое сообщество и развитая экосистема плагинов делают Next.js надёжным выбором для долгосрочных проектов.

PostgreSQL – это мощная объектно-реляционная система управления базами данных (СУБД) с поддержкой ACID-транзакций, расширяемости и богатого набора типов данных. Она заслужила репутацию стабильного и масштабируемого решения для приложений различного уровня сложности, от небольших MVP до корпоративных систем. Поддержка JSONB позволяет хранить и эффективно обрабатывать частично структурированные данные, что полезно для хранения динамичных пользовательских ответов. PostgreSQL отличается встроенными средствами резервного копирования и репликации, обеспечивающими высокую доступность. Использование этой СУБД гарантирует целостность данных и гибкость при изменении требований.

Material-UI (MUI) – это библиотека React-компонентов, реализующая принципы Material Design от Google [6]. Она предоставляет широкий набор готовых элементов интерфейса – кнопки, формы, таблицы и навигационные панели – с возможностью детальной кастомизации через тему. Интеграция MUI ускоряет разработку и обеспечивает единую стилистику приложения, а также поддержку адаптивного дизайна «из коробки». Благодаря модульной структуре можно подключать только необходимые компоненты, что уменьшает размер клиентского бандла. Использование Material-UI повышает удобство разработки и улучшает пользовательский опыт благодаря знакомым визуальным паттернам.

Ant Design – это корпоративная React-библиотека компонентов от Alibaba, ориентированная на создание сложных бизнес-интерфейсов [7]. Она включает в себя расширенные элементы, такие как таблицы с пагинацией, динамические формы, всплывающие уведомления и модальные окна, что ускоряет разработку административных панелей. Ant Design предоставляет продуманные шаблоны взаимодействия и систему дизайна, которая помогает создать единообразный и понятный UX. Совместное использование MUI и Ant Design позволяет выбрать оптимальный компонент под конкретную задачу, сохраняя консистентность стилей. Широкое сообщество и регулярные обновления делают Ant Design надёжным инструментом для коммерческих проектов.

Redux Toolkit – это официальное рекомендованное решение для эффективного управления состоянием в приложениях React. Оно упрощает конфигурацию хранилища, создаёт boilerplate-меньше код с помощью функций createSlice, createAsyncThunk и других утилит. Redux Toolkit гарантирует

иммутабельность состояния и предсказуемость обновлений, что критично для сложных интерфейсов и асинхронных операций. Интеграция с React-Redux обеспечивает простой доступ к состоянию и диспатчинг экшенов из компонентов. Благодаря этому инструменту легко масштабировать логику состояния при росте приложения и поддерживать чистую архитектуру.

PerplexityAI – это сервис на основе больших языковых моделей, который предоставляет API для генерации ответов на естественном языке и анализа текста [2]. В проекте он используется для интерактивной обработки открытых ответов пользователей и формирования рекомендаций на основе контекста. PerplexityAI умеет понимать сложные запросы, извлекать ключевые идеи и предоставлять развернутую информацию о профессиях и навыках. Интеграция с LLM позволяет системе давать более гибкие и обоснованные советы, чем классические тесты с фиксированными шкалами. Это расширяет возможности профориентации и повышает качество пользовательского опыта.

NextAuth.js – это решение для аутентификации в приложениях Next.js, поддерживающее стратегии JWT, OAuth-провайдеров и адаптеры для популярных баз данных [10]. В данном проекте NextAuth обеспечивает безопасную регистрацию и вход пользователей как со стороны клиентов, так и API-роутов. Использование адаптера Prisma упрощает хранение сессий и учётных записей в базе PostgreSQL. Благодаря встроенным механизмам обновления токенов и защите от CSRF, NextAuth.js гарантирует безопасность и корректную работу авторизации. Это позволяет быстро внедрить гибкую систему прав доступа и защиту личных данных.

3 РАЗРАБОТКА

3.1 Клиент

Страницы веб-сайта (рисунок 3.1):

- 1 Авторизация / Регистрация
- 2 Поиск среди специальностей
- 3 Профиль пользователя (абитуриент или студент)
- 4 Профиль учебного учреждения
- 5 Результат тестирования
- 6 Превью учреждения.

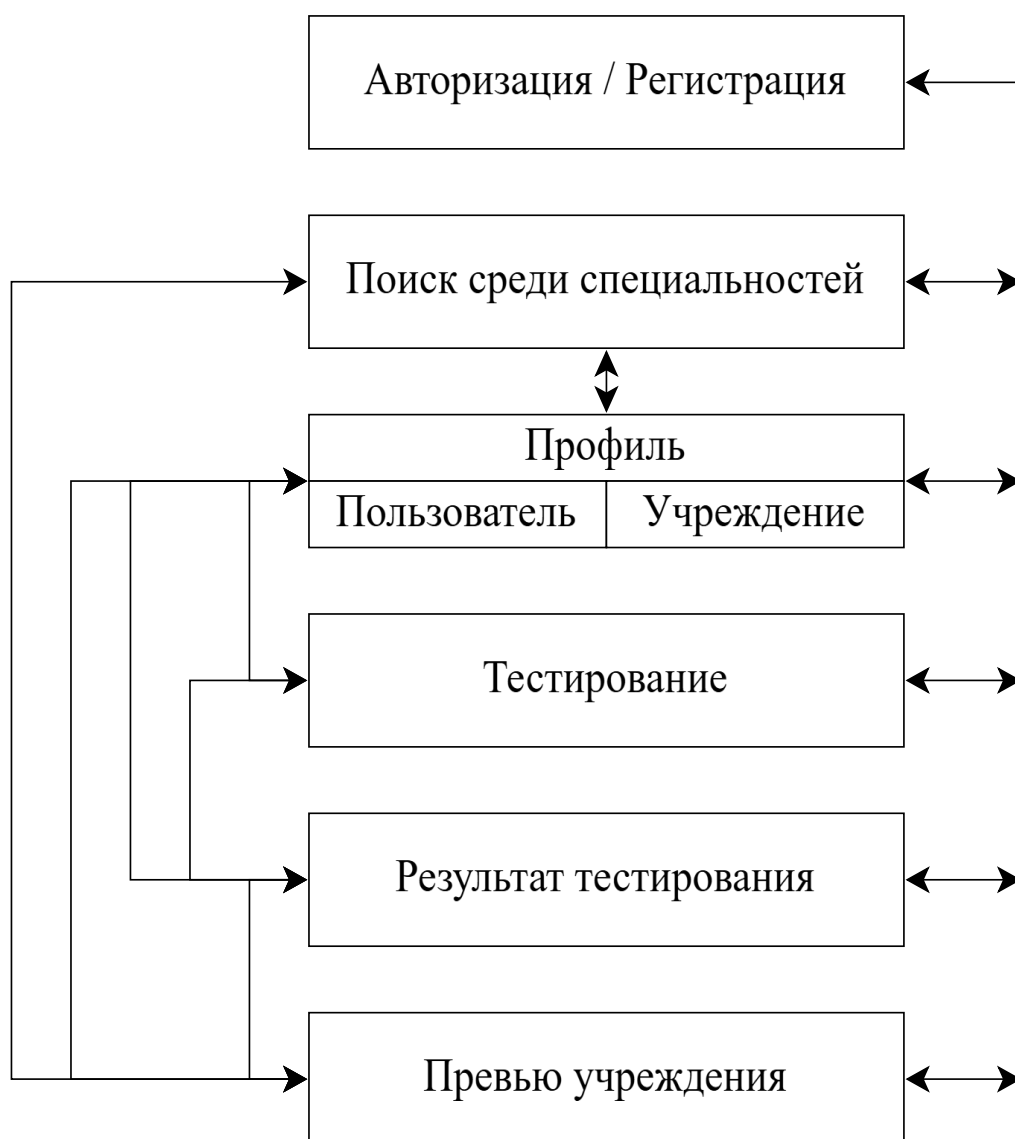


Рисунок 3.1 – Карта переходов между страницами сайта

Рисунок 3.2 – Форма авторизации

Страница авторизации с помощью которой происходит вход в систему (рисунок 3.2). Пользователь выбирает роль для своего аккаунта, в production версии эта возможность заблокирована. Далее пользователь заполняет свой email и придумывает пароль. После нажатия на кнопку “Войти” происходит валидация введенных данных. Пароль должен содержать не менее 6 символов, а email должен соответствовать стандартам библиотеки zod. Если данные некорректны пользователь видит следующие сообщения “Некорректный email” или “Пароль должен содержать не менее 6 символов” соответственно. Если же данные корректны, происходит попытка входа в систему. Если аккаунт с таким email уже существует сравниваются хэш значения паролей и при совпадении происходит авторизация. В случае, когда email не найден, создается новый пользователь. При любом из успешных вариантов происходит перенаправление на главную страницу либо на страницу, с которой был перенаправлен пользователь при попытке получения данных защищенных от неавторизованных пользователей. Также присутствует кнопка скрытия и открытия пароля.

Изм.	Лист	№ докум	Подп.	Дата

ДП.ИИ22.210586 – 05 81 00

Лист

18

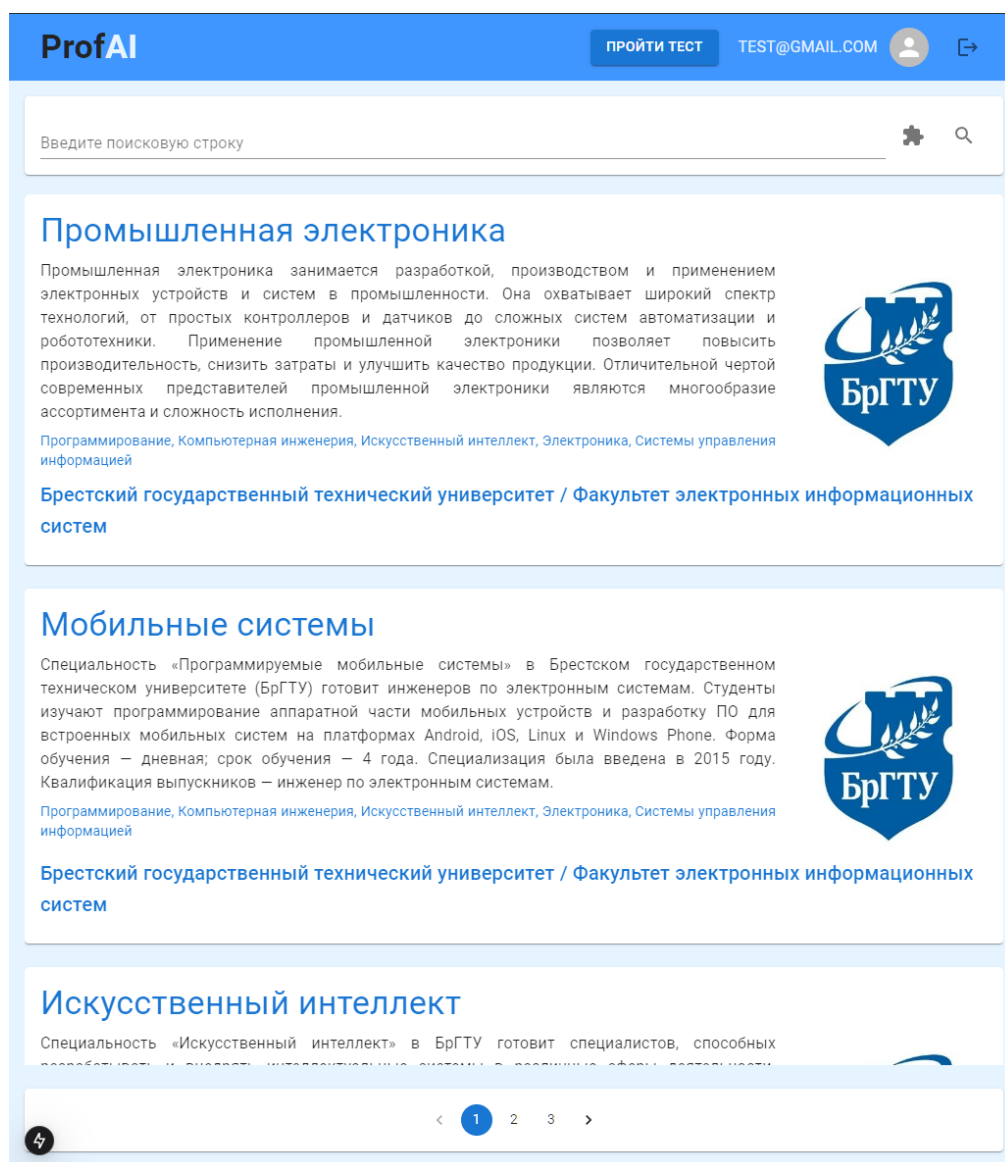


Рисунок 3.3 – Страница поиска среди специальностей

На этой странице (рисунок 3.3) пользователь имеет доступ к базе специальностей и может производить поиск среди них.

Компоненты.



Рисунок 3.4 – Шапка сайта

Шапка сайта (рисунок 3.4) – общий компонент для всех страниц, кроме авторизации. Слева расположен логотип-ссылка сайта, при нажатии на неё

происходит переход на главную страницу, то есть страницу поиска. Для аккаунтов с ролью “student” справа доступна кнопка пройти тестирование. При нажатии на эту кнопку происходит переход на страницу /test. Следующий активных

элемент это email и аватар пользователя, при нажатии на него происходит переход на страницы /profile/student или /profile/university в соответствии с ролью пользователя. Последний активный элемент шапки – кнопка выхода из аккаунта, при нажатии на неё удаляются cookies и происходит переход на страницу /auth для смены аккаунта.

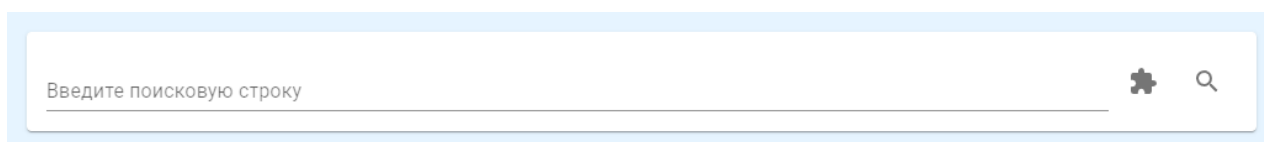


Рисунок 3.5 – Строка общего поиска

Элемент представляет собой элемент для текстового ввода и две кнопки. Текстовый ввод используется для определения строки, по которой будет выполнен общий поиск. При общем поиске выполняется запрос по роуту /api/search/query. Общий поиск (рисунок 3.5) происходит при нажатии на кнопку “Найти” с изображением лупы. При нажатии на кнопку “Расширенный поиск” у нас открывается форма расширенного поиска (рисунок 3.6). В ней при мы вводим строки в конкретные поля, где мы хотим найти вхождения. В верхней правой части присутствует кнопка “Сбросить”, при нажатии на которую происходит очистка формы. При нажатии на кнопку поиска происходит запрос на роут /api/search/extended. Строка общего поиска при открытой форме расширенного поиска становится неактивной. Чтобы перейти обратно к поиску с помощью общего запроса, нужно ещё раз нажать на кнопку “Расширенный поиск”.

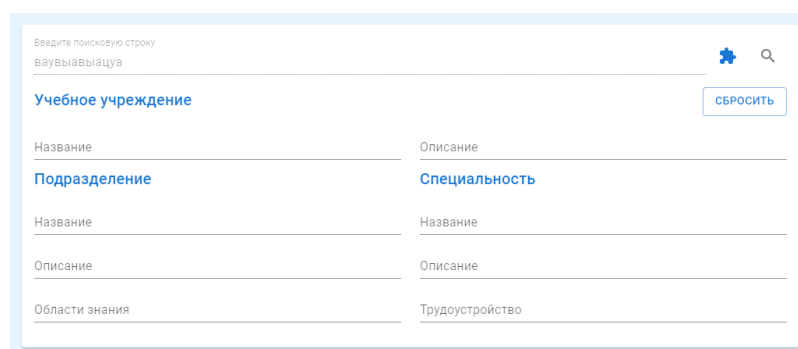


Рисунок 3.6 – Расширенный поиск

В этом элементе нам представлены карточки с краткой информацией специальностей соответствующих поисковому запросу. Если запрос пуст показываются все специальности присутствующие в базе данных.

Компонент для маршрутизации по списку специальностей, список отображает по десять штук, для перехода к следующей десятке, нужно сменить страницу на следующую.

ProfAI

ПРОЙТИ ТЕСТ
TEST@GMAIL.COM

Фамилия
Копанчук

Имя
Евгений

Отчество
Романович

Год рождения
2004

СОХРАНИТЬ

Автоматизированные системы обработки информации

Специальность направлена на проектирование, разработку и внедрение программно-аппаратных комплексов для обработки данных. Студенты изучают программирование, базы данных, сетевые технологии, кибербезопасность и искусственный интеллект. Обучение включает практику в IT-компаниях и работу с современными системами управления информацией. Выпускники способны создавать автоматизированные решения для бизнеса, промышленности и госструктур. Особое внимание уделяется защите информации и интеграции новых технологий в существующие системы.

Программирование, Компьютерная инженерия, Искусственный интеллект, Электроника, Системы управления информацией

Брестский государственный технический университет / Факультет электронных информационных систем

Критерии 76.4%

Тестирование 75%

AI 93.3%

← BACK

● ● ● ● ●

NEXT →

Искусственный интеллект

15:54 02.04.2025

AI

Тестирование

Point	AI (%)	Testing (%)
0	8.5	8.5
1	7.5	7.5
2	7.5	7.5
3	7.5	7.5
4	7.5	7.5
5	8.5	10.0
6	7.5	7.5
7	7.5	7.5
8	7.5	7.5
9	7.5	7.5
10	7.5	7.5
11	8.5	8.5

Финансовая стабильность

Востребованность на рынке

Работа с интересными задачами

Рисунок 3.7 – Страница профиль студента

В этой форме (рисунок 3.7) представлена информация необязательная для заполнения о студенте, при нажатии на форму сохранить происходит обновление информации в базе данных.

Этот компонент присутствует только при наличии хотя бы одного результата тестирования. На основе последнего пройденного теста, формируется 5 специальностей подобранных для пользователя. Компонент похож на карточку специальности, но в нижней части добавлено сходство в процентах между результатами тестирования пользователя и результатами тестирования студентов данной специальности. Расчет рекомендаций происходит по запросу с url адресом /api/search/suggestions.

					ДП.ИИ22.210586 – 05 81 00	Лист
Изм	Лист	№ докум	Подп.	Дата		21

Список карточек результатов тестирования, при нажатии на элемент происходит перенаправление на страницу /preview/test/[testId] с соответствующим id результатов тестирования. Тесты в списке отсортированы по дате по убыванию.

ProfAI KOPANCHUKE@GMAIL.COM

Учебное учреждение

БнГТУ
Главное фото

Максимальное количество изображений достигнуто (8).

Наименование
Брестский государственный технический университет

Описание
Белорусский государственный технологический университет (БГТУ) – ведущий вуз Республики Беларусь, основанный в 1930 году. Университет специализируется на подготовке инженеров для химической и лесной промышленности, специалистов в области издательского дела и полиграфии, а также проводит научные исследования в этих областях. Кроме того, БГТУ предлагает обучение по экономическим и компьютерным специальностям, готовит специалистов по автоматизации производственных процессов, машинам и аппаратам химической и лесной промышленности.

Официальный сайт https://www.bstu.by/	Количество учащихся 5000
---	-----------------------------

ДОБАВИТЬ

Рисунок 3.8 – Форма профиля учебного заведения. Учебное учреждение

Страница профиля учебного заведения представляет собой одну большую форму для заполнения информации об учебном заведении и его компонентах.

В этой секции (рисунок 3.8) можно выбрать до восьми фото включительно, а также фотографию, которая будет являться главной, выбрать порядок с помощью перетаскивания элементов. Вводятся данные, которые являются обязательными, а также можно добавить до трех своих полей введя наименование и значение поля. Добавление дополнительного поля происходит по нажатию кнопки добавить.

Подразделения

ФАКУЛЬТЕТ ЭЛЕКТРОННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

ЭКОНОМИЧЕСКИЙ ФАКУЛЬТЕТ

МАШИНОСТРОИТЕЛЬНЫЙ ФАИ > +

Наименование

Факультет электронных информационных систем

Описание

Факультет электронных информационных систем БрГТУ – это современное образовательное подразделение, которое готовит специалистов в области информационных технологий и электроники. Факультет предлагает широкий спектр специальностей, включая программирование, компьютерную инженерию и искусственный интеллект. Студенты получают глубокие знания и практические навыки, необходимые для успешной карьеры в быстро развивающейся IT-индустрии. Факультет активно сотрудничает с ведущими IT-компаниями, что обеспечивает студентам возможность стажировок и трудоустройства после окончания обучения.

Области знания

Программирование X

Компьютерная инженерия X

Искусственный интеллект X

Электроника X

Системы управления информацией X

Новая область

ДОБАВИТЬ

ДОБАВИТЬ

УДАЛИТЬ

Рисунок 3.9 – Форма профиля учебного заведения. Подразделение

В этой секции (рисунок 3.9) вводится информация об факультетах университета, для каждого факультета существует своя страница элемента Tab. Новую страницу можно создать по нажатию кнопки “Добавить новое подразделение” с инокой с символом “+”. Удаление подразделения происходит по нажатию кнопки “Удалить” и подтверждения действия в диалоговом окне. Информация “Области знания” представляет собой набор тегов, по которым можно узнать быстро направленность факультета. Также как и в предыдущей секции имеется возможность добавить свои поля для дополнительно информации о подразделении.



Брестский государственный технический университет

Белорусский государственный технологический университет (БГТУ) — ведущий вуз Республики Беларусь, основанный в 1930 году. Университет специализируется на подготовке инженеров для химической и лесной промышленности, специалистов в области издательского дела и полиграфии, а также проводит научные исследования в этих областях. Кроме того, БГТУ предлагает обучение по экономическим и компьютерным специальностям, готовит специалистов по автоматизации производственных процессов, машинам и аппаратам химической и лесной промышленности.

Официальный сайт учебного заведения

5000 человек

Факультет электронных информационных систем

Факультет электронных информационных систем БрГТУ — это современное образовательное подразделение, которое готовит специалистов в области информационных технологий и электроники. Факультет предлагает широкий спектр специальностей, включая программирование, компьютерную инженерию и искусственный интеллект. Студенты получают глубокие знания и практические навыки, необходимые для успешной карьеры в быстро развивающейся IT-индустрии. Факультет активно сотрудничает с ведущими IT-компаниями, что обеспечивает студентам возможность стажировок и трудоустройства после окончания обучения.

Программирование

Компьютерная инженерия

Искусственный интеллект

Электроника

Системы управления информацией

Рисунок 3.11 – Страница превью специальности. Описание учебного заведения и подразделения.

Промышленная электроника

Промышленная электроника занимается разработкой, производством и применением электронных устройств и систем в промышленности. Она охватывает широкий спектр технологий, от простых контроллеров и датчиков до сложных систем автоматизации и робототехники. Применение промышленной электроники позволяет повысить производительность, снизить затраты и улучшить качество продукции. Отличительной чертой современных представителей промышленной электроники являются многообразие ассортимента и сложность исполнения.

Специалисты в области промышленной электроники востребованы в Республике Беларусь, особенно в Минске, где регулярно открываются вакансии инженеров-электроников с конкурентоспособными заработными платами. Согласно данным сайта rabota.by, в Минске доступны десятки подобных вакансий с зарплатами от 1800 до 5000 рублей в месяц. Кроме того, Белорусский государственный университет информатики и радиоэлектроники (БГУИР) сообщает о наличии вакансий инженеров-электроников в различных подразделениях, что свидетельствует о высоком спросе на таких специалистов. Таким образом, выпускники по специальности "Промышленная электроника" имеют хорошие перспективы для трудоустройства в Беларуси.

Рисунок 3.12 – Страница превью специальности. Описание специальности

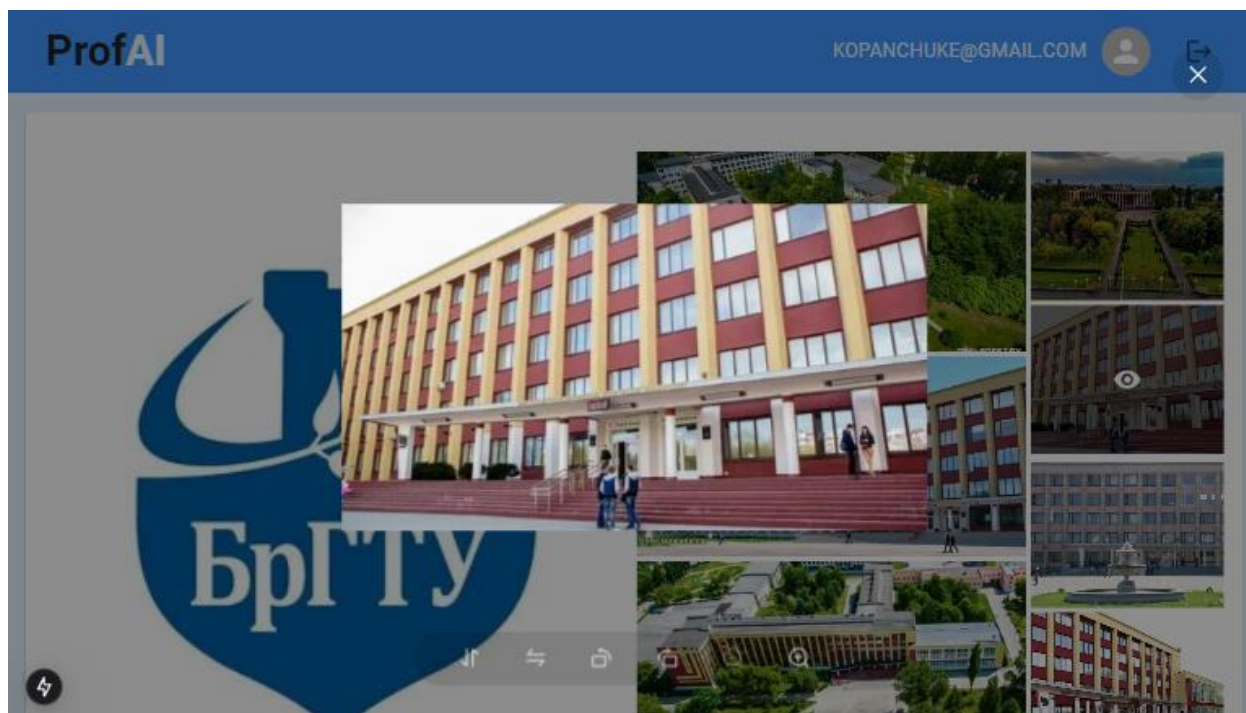


Рисунок 3.13 – Страница превью специальности. Просмотр фото

На странице результатов тестирования находится множество элементов, цель которых удобно подать рассчитанные метрики и показатели для пользователя. Сверху (рисунок 3.13) находится карточка специальности, она присутствует только в том случае, если пользователь проходя тест, указал, что является студентом данной специальности. Карточка интерактивная, нажав на неё можно перейти на страницу данной специальности с подробной информацией. Далее находится график результатов тестирования.

Изм.	Лист	№ докум	Подп.	Дата

ДП.ИИ22.210586 – 05 81 00

Лист

26



Искусственный интеллект

Специальность «Искусственный интеллект» в БрГТУ готовит специалистов, способных разрабатывать и внедрять интеллектуальные системы в различные сферы деятельности. Обучение включает изучение методов машинного обучения, нейронных сетей, анализа данных и компьютерного зрения. Студенты получают навыки создания программного обеспечения, способного анализировать большие объемы информации, принимать решения и обучаться на основе опыта. Выпускники могут работать в IT-компаниях, научно-исследовательских центрах, а также в организациях, использующих интеллектуальные системы для автоматизации процессов.

Программирование, Компьютерная инженерия, Искусственный интеллект, Электроника, Системы управления информацией

Брестский государственный технический университет / Факультет электронных информационных систем

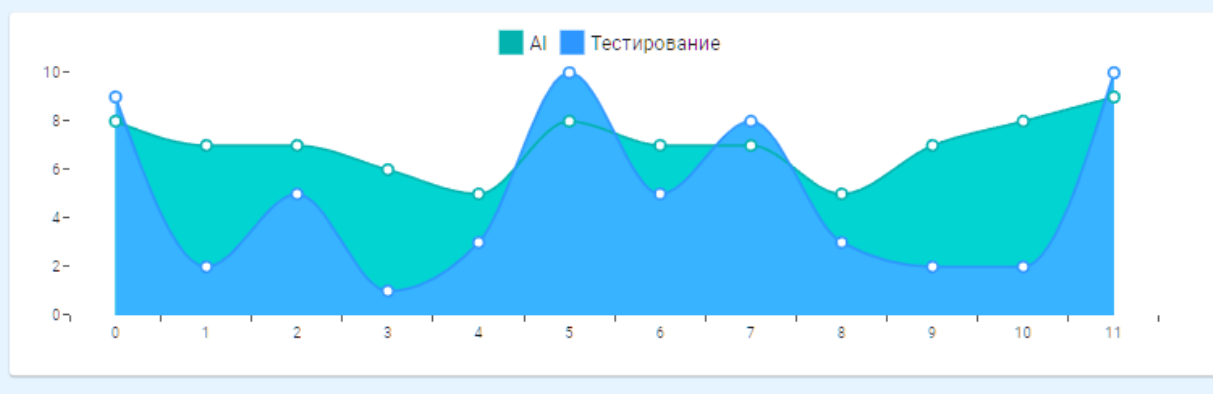








Рисунок 3.14 – Страница результата тестирования. Специальность студента и график показателей.

Далее мы видим таблицы (рисунок 3.15), в которых указаны ответы пользователя в колонке “Тестирование” и значения, которые оценил ИИ в колонке “AI”, а также средние показатели между ними в колонке “Среднее”. В таблице “Предпочтительные критерии” указаны первые шесть критериев, которые пользователь хотел бы видеть у своей будущей работы. Во второй таблице, “Менее интересные критерии”, указаны ещё шесть критериев, которые пользователь отнес в конец приоритетного списка, от самой менее интересной до более интересной.

Далее (рисунок 3.15) мы видим ответы пользователя на вопросы теста, по которым ИИ давал оценки, и такой же интерактивный список их пяти рассчитанных рекомендуемых специальностей.

№	Навык	Тестирование	AI	Среднее
0	Аналитическое мышление	9	8	8.5
1	Коммуникативные навыки	2	7	4.5
2	Творческое мышление	5	7	6
3	Физическая выносливость	1	6	3.5
4	Лидерство	3	5	4
5	Работа с техническим оборудованием	10	8	9
6	Организационные способности	5	7	6
7	Устойчивость к стрессу	8	7	7.5
8	Эмпатия	3	5	4
9	Мелкая моторика и точность	2	7	4.5
10	Быстрая реакция	2	8	5
11	Навыки самообучения	10	9	9.5

№	Предпочтительные критерии
	Финансовая стабильность
	Востребованность на рынке
	Работа с интересными задачами
	Пrestиж и уважение
	Вклад в будущее
	Возможность помогать людям







№	Менее интересные критерии
	Работа с людьми
	Связь с природой или окружающим миром
	Возможность путешествовать и работать
	Гибкость в управлении рабочим графиком
	Эстетическое удовольствие от работы
	Творческая реализация

Рисунок 3.15 – Страница результата тестирования. Таблицы показателей и топ критериев профессии

Чем вы любите заниматься в свободное время?

Я люблю играть в компьютерные игры, например, дота 2, разрабатывать сайты, люблю рисовать карандашами и красками по холсту, люблю гулять по улице, ходить в тренажерный зал.

Какая школьная дисциплина вам давалась легче всего и почему?

Математика. У меня был хороший учитель и мне нравилось решать задачи.

Какой свой навык или умение вы считаете самым сильным?

Думаю, что усидчивость, ведь, если я взялся за работу, я хочу сидеть за ней пока не закончу.

Каким критериям должна соответствовать работа вашей мечты?

Я хочу много зарабатывать, заниматься разработкой сайтов на React, удаленный режим работы, хорошую команду, с которой я смогу развиваться в программировании.

Вы предпочитаете работать в одиночку или в команде? Почему?

В команде, так как с командой можно реализовать более глобальные вещи, чем в одиночку.

Автоматизированные системы обработки информации

Специальность направлена на проектирование, разработку и внедрение программно-аппаратных комплексов для обработки данных. Студенты изучают программирование, базы данных, сетевые технологии, кибербезопасность и искусственный интеллект. Обучение включает практику в IT-компаниях и работу с современными системами управления информацией. Выпускники способны создавать автоматизированные решения для бизнеса, промышленности и госструктур. Особое внимание уделяется защите информации и интеграции новых технологий в существующие системы.

Программирование, Компьютерная инженерия, Искусственный интеллект, Электроника, Системы управления информацией

Брестский государственный технический университет / Факультет электронных информационных систем

Критерии 76.4%
Тестирование 75%
AI 93.3%

< BACK
● ● ● ● ●
NEXT >

Рисунок 3.16 – Страница результата тестирования. Ответы на вопросы и рекомендуемые специальности

Указание специальности (рисунок 3.17) используется для того, чтобы пользователь, который является студентом, могу указать свою специальность, его данные будут использоваться для улучшения системы и накапливаться в базе данных. При начале ввода пользователю предлагается автозаполнение соответствующее введенной подстроке.

Секция “Преимущества профессии” представляет собой сортируемый список. Пользователь располагает их в порядке своего предпочтения, что дает представление на что нужно основываться при подборе рабочего места.

ProfAI

ПРОЙТИ ТЕСТ

TEST@GMAIL.COM

Специальность

Я абитуриент

Если вы уже являетесь студентом, пожалуйста, укажите вашу специальность выше, это очень поможет нам в улучшении сервиса😊.

Преимущества профессии

Расположите характеристики будущей профессии в порядке важности по убыванию.

👤	1. Возможность помогать людям	^	▼
\$	2. Финансовая стабильность	^	▼
🔍	3. Востребованность на рынке	^	▼
🎨	4. Творческая реализация	^	▼
🧩	5. Работа с интересными задачами	^	▼
🌿	6. Связь с природой или окружающим миром	^	▼
👥	7. Работа с людьми	^	▼
📜	8. Престиж и уважение	^	▼
🏔️	9. Возможность путешествовать и работать	^	▼
🕒	10. Гибкость в управлении рабочим графиком	^	▼
📅	11. Вклад в будущее	^	▼
😊	12. Эстетическое удовольствие от работы	^	▼

Рисунок 3.17 – Страница прохождения тестирования. Указание специальности и список приоритетных критериев профессии.

Секция “Навыки” (рисунок 3.18). представляет собой список качеств, между которыми нужно распределить 60 очков, суть в том, что пользователь попадает в ситуацию, в которой не может превысить свои показатели во всем, поэтому он будет выбирать наиболее лучшие свои качества, чтобы повысить балл им. Пока все очки не будут распределены, форма тестирования не будет считаться заполненной.

Секция “Расскажите о себе” (рисунок 3.19). представляет собой обычный текстовые ответы на вопросы, именно здесь нам и понадобится LLM, чтобы дать оценку пользователю по ответам на естественном языке.

Кнопка отправить делает запрос по роуту /api/ai/evaluate, который при успешном результате делает перенаправление на страницу просмотра результата тестирования.

Навыки

Распределите очки навыков так, чтобы они максимально соответствовали вам.

Вы распределили все очки (60/60).

1. Аналитическое мышление

2. Коммуникативные навыки

3. Творческое мышление

4. Физическая выносливость

5. Лидерство

6. Работа с техническим оборудованием

7. Организационные способности

8. Устойчивость к стрессу

9. Эмпатия

10. Мелкая моторика и точность

11. Быстрая реакция

12. Навыки самообучения

Рисунок 3.18 – Страница прохождения тестирования. Распределение навыков пользователя.

Расскажите о себе

Максимально развернуто ответьте на следующие вопросы.

Чем вы любите заниматься в свободное время?

Какая школьная дисциплина вам давалась легче всего и почему?

Какой свой навык или умение вы считаете самым сильным?

Каким критериям должна соответствовать работа вашей мечты?

Вы предпочитаете работать в одиночку или в команде? Почему?

ОТПРАВИТЬ

Рисунок 3.19 – Страница прохождения тестирования. Ответы на вопросы для ИИ и кнопка расчета результатов тестирования.

3.2 Сервер

POST /api/ai/evaluate.

Входные данные. текстовое описание ответов пользователя на вопросы профориентации.

Описание. формирует промпт с критериями оценки, отправляет его в API Perplexity для получения текстового результата, извлекает числовые оценки по 12 критериям из ответа, проверяет их корректность.

Возвращаемые значения. При успешном выполнении возвращает статус 200, текст ответа от AI и массив из 12 числовых оценок. В случае ошибки возвращает статус 400 или 500 с соответствующим описанием проблемы.

GET /api/ai/test/[specialtyId].

Входные данные. Идентификатор специальности.

Описание. Запрашивает данные о специальности, факультете и университете, используя их идентификаторы, генерирует JSON-запрос для модели AI на основе описания, переданного в промпт, а затем создаёт тест в базе данных с использованием возвращённых данных.

Возвращаемые значения.

- При успешном создании теста возвращается статус 200 и объект теста с данными о преимуществах, навыках, вопросах и связанными сущностями.
- В случае ошибок возвращаются статусы 400 (неверные данные или проблемы с созданием теста) или 500 (ошибки на стороне сервера).

[GET, POST] /api/auth/[...nextauth].

Входные данные. Запрос принимает параметры аутентификации пользователя, такие как данные для входа (например, email и пароль) или данные провайдера OAuth.

Описание. Обработывает аутентификацию пользователей, включая вход, выход, колбэки от OAuth-провайдеров, проверку сессий и получение данных текущего пользователя.

Возвращаемые значения.

- При успешной аутентификации возвращает статус 200 и объект сессии, содержащий информацию о пользователе.
- При ошибке возвращает соответствующий код состояния (например, 401 для неудачной аутентификации) и сообщение об ошибке.

GET /api/extraFacultyInfo/[extraFacultyInfoId].

Входные данные. Идентификатор дополнительной информации о факультете.

Описание. Запрашивает данные дополнительной информации о факультете из базы данных по указанному идентификатору.

Возвращаемые значения.

- При успешном запросе возвращается статус 200 и объект с дополнительной информацией о факультете.
- В случае ошибок возвращается статус 400 (неверный идентификатор или отсутствие данных) или 500 (ошибка сервера).

POST /api/extraFacultyInfo/[extraFacultyInfoId].

Входные данные. Идентификатор дополнительной информации о факультете и новые данные для обновления.

Описание. Обновляет данные дополнительной информации о факультете в базе данных по указанному идентификатору.

Возвращаемые значения.

- При успешном обновлении возвращается статус 200 и обновлённый объект с дополнительной информацией о факультете.
- В случае ошибок возвращается статус 400 (неверный идентификатор или данные) или 500 (ошибка сервера).

DELETE /api/extraFacultyInfo/[extraFacultyInfoId].

Входные данные. Идентификатор дополнительной информации о факультете.

Описание. Удаляет данные дополнительной информации о факультете из базы данных по указанному идентификатору.

Возвращаемые значения.

- При успешном удалении возвращается статус 200 и объект с информацией о выполненном удалении.
- В случае ошибок возвращается статус 400 (неверный идентификатор) или 500 (ошибка сервера).

POST /api/extraFacultyInfo.

Входные данные. Данные для создания дополнительной информации о факультете, включая идентификатор факультета.

Описание. Создаёт новую запись в базе данных для дополнительной информации о факультете, связывая её с указанным факультетом.

Возвращаемые значения.

- При успешном создании возвращается статус 200 и объект с данными созданной записи.
- В случае ошибок возвращается статус 400 (некорректные данные) или 500 (ошибка сервера).

GET /api/extraSpecialtyInfo/[extraSpecialtyInfoId].

Входные данные. Идентификатор дополнительной информации о специальности.

Описание. Запрашивает данные дополнительной информации о специальности из базы данных по указанному идентификатору.

Возвращаемые значения.

- При успешном выполнении возвращается статус 200 и объект с данными дополнительной информации о специальности.
- В случае ошибок возвращается статус 400 (некорректный идентификатор или данные не найдены) или 500 (ошибка сервера).

POST /api/extraSpecialtyInfo/[extraSpecialtyInfoId].

Входные данные. Идентификатор дополнительной информации о специальности и данные для обновления.

Описание. Обновляет запись в базе данных, связанной с указанным идентификатором, новыми данными, переданными в запрос.

Возвращаемые значения.

- При успешном обновлении возвращается статус 200 и объект с обновлёнными данными.
- В случае ошибок возвращается статус 400 (некорректный идентификатор или данные) или 500 (ошибка сервера).

DELETE /api/extraSpecialtyInfo/[extraSpecialtyInfoId].

Изм	Лист	№ докум	Подп.	Дата

ДП.ИИ22.210586 – 05 81 00

Лист

34

Входные данные. Идентификатор дополнительной информации о специальности.

Описание. Удаляет запись из базы данных, связанную с указанным идентификатором.

Возвращаемые значения.

- При успешном удалении возвращается статус 200 и объект с данными удалённой записи.
- В случае ошибок возвращается статус 400 (некорректный идентификатор или данные не найдены) или 500 (ошибка сервера).

POST /api/extraSpecialtyInfo.

Входные данные. Данные дополнительной информации о специальности, включая идентификатор специальности.

Описание. Создаёт новую запись дополнительной информации о специальности в базе данных, связывая её с указанной специальностью по идентификатору.

Возвращаемые значения.

- При успешном создании возвращается статус 200 и объект с данными дополнительной информации о специальности.
- В случае ошибок возвращается статус 400 (неверные данные или проблемы с созданием) или 500 (ошибка на стороне сервера).

GET /api/extraUniversityInfo/[extraUniversityInfoId].

Входные данные. Идентификатор дополнительной информации о университете.

Описание. Запрашивает данные дополнительной информации о университете по указанному идентификатору.

Возвращаемые значения.

- При успешном получении данных возвращается статус 200 и объект с данными дополнительной информации о университете.
- В случае ошибок возвращается статус 400 (неверные данные или отсутствие информации с таким идентификатором) или 500 (ошибка на стороне сервера).

POST /api/extraUniversityInfo/[extraUniversityInfoId].

Входные данные. Данные дополнительной информации о университете, включая идентификатор дополнительной информации о университете.

Описание. Обновляет данные дополнительной информации о университете, связанные с указанным идентификатором.

Возвращаемые значения.

- При успешном обновлении возвращается статус 200 и обновлённый объект дополнительной информации о университете.

- В случае ошибок возвращается статус 400 (неверные данные или отсутствие информации с таким идентификатором) или 500 (ошибка на стороне сервера).

DELETE /api/extraUniversityInfo/[extraUniversityInfoId].

Входные данные. Идентификатор дополнительной информации о университете.

Описание. Удаляет запись дополнительной информации о университете по указанному идентификатору.

Возвращаемые значения.

- При успешном удалении возвращается статус 200 и объект с удалённой дополнительной информацией о университете.

- В случае ошибок возвращается статус 400 (неверные данные или отсутствие информации с таким идентификатором) или 500 (ошибка на стороне сервера).

POST /api/extraUniversityInfo.

Входные данные. Данные дополнительной информации о университете, включая идентификатор университета.

Описание. Создаёт новую запись дополнительной информации о университете, связывая её с указанным университетом по идентификатору.

Возвращаемые значения.

- При успешном создании возвращается статус 200 и объект с данными дополнительной информации о университете.

- В случае ошибок возвращается статус 400 (неверные данные или проблемы с созданием) или 500 (ошибка на стороне сервера).

GET /api/faculty/[facultyId]/detailed.

Входные данные. Идентификатор факультета.

Описание. Запрашивает полную информацию о факультете, включая основные данные, дополнительную информацию и список его специальностей вместе с их дополнительными данными.

Возвращаемые значения.

- При успешном выполнении возвращается статус 200 и объект с подробной информацией о факультете.

- В случае ошибок возвращается статус 400 (неверный идентификатор или факультет не найден) или 500 (ошибка на стороне сервера).

GET /api/faculty/[facultyId].

Входные данные. Идентификатор факультета.

Описание. Запрашивает информацию о факультете, используя переданный идентификатор. Возвращает данные о факультете.

Возвращаемые значения.

- При успешном запросе возвращается статус 200 и объект с данными о факультете.

- В случае ошибок возвращаются статусы 400 (неверный идентификатор факультета или факультет не найден) или 500 (ошибка на сервере).

POST /api/faculty/[facultyId].

Входные данные. Данные для обновления факультета.

Описание. Обновляет информацию о факультете, используя переданный идентификатор и данные из запроса.

Возвращаемые значения.

- При успешном обновлении возвращается статус 200 и объект с обновленными данными факультета.

- В случае ошибок возвращаются статусы 400 (неверный идентификатор или проблемы при обновлении данных) или 500 (ошибка на сервере).

DELETE /api/faculty/[facultyId].

Входные данные. Идентификатор факультета.

Описание. Удаляет факультет из базы данных по переданному идентификатору.

Возвращаемые значения.

- При успешном удалении возвращается статус 200 и объект с удаленными данными факультета.

- В случае ошибок возвращаются статусы 400 (неверный идентификатор факультета или факультет не найден) или 500 (ошибка на сервере).

POST /api/faculty.

Входные данные. Данные факультета, включая идентификатор университета.

Описание. Создает новый факультет с переданными данными и связывает его с университетом по идентификатору университета.

Возвращаемые значения.

- При успешном создании факультета возвращается статус 200 и объект с данными нового факультета.

- В случае ошибок возвращаются статусы 400 (ошибка при создании факультета или неверные данные) или 500 (ошибка на сервере).

POST /api/search/extended.

Входные данные. Данные для расширенного поиска, включая параметры для университета, факультета и специальности (названия, описания, области знаний и т. д.).

Описание. Выполняет расширенный поиск по университетам, факультетам и специальностям, фильтруя результаты на основе переданных параметров (например,

Изм	Лист	№ докум	Подп.	Дата

ДП.ИИ22.210586 – 05 81 00

Лист

37

названия и описания). Поиск учитывает данные о вузах, факультетах и специализациях и возвращает отфильтрованные результаты.

Возвращаемые значения.

- При успешном выполнении запроса возвращается статус 200 и список результатов поиска, сжимаемых и фильтруемых в соответствии с переданными данными.
- В случае ошибок возвращается статус 400 (ошибка в данных для поиска) или 500 (ошибка на сервере).

POST /api/search/query.

Входные данные. Строка запроса для поиска, которая может быть использована для фильтрации информации по университетам, факультетам и специальностям.

Описание. Выполняет поиск по университетам, факультетам и специальностям, фильтруя результаты по строке запроса. При отсутствии запроса возвращает все доступные данные. Запрос может быть выполнен по ключевым словам, присутствующим в названиях и описаниях университетов, факультетов и специальностей.

Возвращаемые значения.

- При успешном выполнении запроса возвращается статус 200 и отфильтрованный и сжатый результат поиска.
- В случае ошибок возвращается статус 400 (ошибка при получении данных) или 500 (ошибка на сервере).

GET /api/search/suggestions/[testId].

Описание. Этот API-метод предоставляет рекомендации для теста на основе сравнения характеристик теста, определённых в параметре testId, с другими тестами. Рекомендации основываются на совпадениях в benefits, skills, и llmSkills и выводятся для 5 наиболее подходящих тестов.

Входные данные. testId – идентификатор теста, для которого необходимо найти рекомендации.

Возвращаемые значения.

- Статус 200 с отфильтрованными и сжатыми рекомендациями, включающими данные о специальности, факультете и университете для каждого из рекомендованных тестов.
- Статус 400, если testId некорректен или тест не найден.
- Статус 500, если произошла ошибка на сервере.

Методы сравнения.

compareBenefits. Сравнивает benefits двух тестов, вычисляя их различия.

compareSkills. Сравнивает skills двух тестов.

compareTests. Сравнивает указанный тест с другими и оценивает их по совокупности факторов.

GET /api/specialty/[specialtyId]/detailed.

Входные данные. specialtyId – идентификатор специальности, для которой необходимо получить подробную информацию.

Описание. Этот API-метод предоставляет детализированные данные о специальности, включая дополнительную информацию и связанный тест, если он есть.

Возвращаемые значения.

- Статус 200 с подробными данными о специальности, включая информацию о дополнительной информации и тесте, если он связан с этой специальностью.

- Статус 400, если specialtyId некорректен или специальность не найдена.
- Статус 500, если произошла ошибка на сервере.

GET /api/specialty/[specialtyId].

Входные данные. Идентификатор специальности.

Описание. Запрашивает из базы данных информацию о специальности по указанному идентификатору.

Возвращаемые значения.

- При успешном запросе возвращается статус 200 и объект специальности.
- В случае ошибок возвращаются статус 400 (неверный идентификатор или специальность не найдена) или 500 (ошибка на сервере).

POST /api/specialty/[specialtyId].

Входные данные. Идентификатор специальности и новые данные для её обновления.

Описание. Обновляет в базе данных запись специальности по указанному идентификатору, используя переданные данные.

Возвращаемые значения.

- При успешном обновлении возвращается статус 200 и обновлённый объект специальности.
- В случае ошибок возвращаются статус 400 (неверный идентификатор или некорректные данные) или 500 (ошибка на сервере).

DELETE /api/specialty/[specialtyId].

Входные данные. Идентификатор специальности.

Описание. Удаляет из базы данных запись специальности по указанному идентификатору.

Возвращаемые значения.

- При успешном удалении возвращается статус 200 и объект удалённой специальности.

- В случае ошибок возвращаются статус 400 (неверный идентификатор или специальность не найдена) или 500 (ошибка на сервере).

POST /api/specialty.

Входные данные. Данные специальности, включая идентификатор факультета.

Описание. Создаёт новую запись специальности в базе данных и связывает её с указанным факультетом.

Возвращаемые значения.

- При успешном создании возвращается статус 200 и объект созданной специальности.

- В случае ошибок возвращается статус 400 (некорректные входные данные) или 500 (ошибка сервера).

GET /api/student/[studentId]/detailed.

Входные данные. Идентификатор студента.

Описание. Запрашивает детализированные данные о студенте, включая информацию о связанных тестах, используя переданный идентификатор студента.

Возвращаемые значения.

- При успешном запросе возвращается статус 200 и объект студента с дополнительными данными о тестах.

- В случае ошибок возвращается статус 400 (некорректный идентификатор студента или отсутствие данных) или 500 (ошибка сервера).

GET /api/student/[studentId].

Входные данные. Идентификатор студента.

Описание. Получает из базы данных запись студента по указанному идентификатору.

Возвращаемые значения.

- При успешном запросе возвращается статус 200 и объект студента.

- В случае ошибок возвращается статус 400 (некорректный идентификатор или студент не найден) или 500 (ошибка на стороне сервера).

POST /api/student/[studentId].

Входные данные. Идентификатор студента и новые данные для его профиля.

Описание. Обновляет в базе данных существующую запись студента, используя переданные данные.

Возвращаемые значения.

- При успешном обновлении возвращается статус 200 и объект обновлённого студента.

- В случае ошибок возвращается статус 400 (некорректные данные или студент не найден) или 500 (ошибка на стороне сервера).

DELETE /api/student/[studentId].

Входные данные. Идентификатор студента.

Описание. Удаляет из базы данных запись студента по указанному идентификатору.

Возвращаемые значения.

- При успешном удалении возвращается статус 200 и объект удалённого студента.

- В случае ошибок возвращается статус 400 (некорректный идентификатор или студент не найден) или 500 (ошибка на стороне сервера).

POST /api/student.

Входные данные. Данные студента, включая его идентификатор и идентификатор пользователя, с которым он связан.

Описание. Создаёт нового студента в базе данных, используя переданные данные. Студент связывается с существующим пользователем через его идентификатор.

Возвращаемые значения.

- При успешном создании студента возвращается статус 200 и объект нового студента.

- В случае ошибок возвращается статус 400 (неверные данные или ошибка при создании студента) или 500 (ошибка на стороне сервера).

GET /api/test/[testId]/detailed.

Входные данные. Идентификатор теста.

Описание. Запрашивает подробную информацию о тесте, включая связанные с ним данные о студенте и специальности, используя переданный идентификатор теста.

Возвращаемые значения.

- При успешном запросе возвращается статус 200 и объект теста с дополнительными данными о студенте и специальности.

- В случае ошибок возвращается статус 400 (неверный идентификатор теста или тест не найден) или 500 (ошибка на стороне сервера).

GET /api/test/[testId].

Входные данные. идентификатор теста.

Описание. запрашивает из базы данных запись теста по указанному идентификатору.

Возвращаемые значения. При успешном запросе возвращается статус 200 и объект теста; при неверном идентификаторе или отсутствии записи – статус 400; при внутренней ошибке – статус 500.

POST /api/test/[testId].

Входные данные. Идентификатор теста и новые данные для обновления.

Описание. Обновляет существующую запись теста в базе данных по переданному идентификатору, применяя переданные данные.

Возвращаемые значения. При успешном обновлении возвращается статус 200 и обновлённый объект теста; при неверном идентификаторе или некорректных данных – статус 400; при внутренней ошибке – статус 500.

DELETE /api/test/[testId].

Входные данные. Идентификатор теста.

Описание. Удаляет запись теста из базы данных по указанному идентификатору.

Возвращаемые значения. При успешном удалении возвращается статус 200 и объект удалённого теста; при неверном идентификаторе или отсутствии записи – статус 400; при внутренней ошибке – статус 500.

GET /api/test/detailed/with/[studentId].

Входные данные. Идентификатор студента.

Описание. Получает из базы данных все тесты, связанные с указанным студентом, включая информацию о самом студенте и соответствующей специальности для каждого теста.

Возвращаемые значения.

- При успешном выполнении возвращается статус 200 и массив объектов тестов с подробными данными.
- В случае ошибок возвращается статус 400 (неверный идентификатор студента или отсутствие тестов) или 500 (ошибка на стороне сервера).

GET /api/test/detailed.

Входные данные. Нет входных параметров.

Описание. Запрашивает из базы данных все тесты, включая связанные данные о студентах и специальностях.

Возвращаемые значения.

- При успешном выполнении возвращается статус 200 и массив объектов тестов с подробной информацией.
- В случае ошибки возвращается статус 400 (если набор тестов не найден) или 500 (ошибка на стороне сервера).

POST /api/test.

Входные данные. Данные теста, включая дату прохождения, преимущества, навыки, вопросы, навыки LLM, идентификатор студента и необязательный идентификатор специальности.

Описание. Создает новый тест с переданными данными, связывая его с указанным студентом и, при необходимости, с указанной специальностью.

Возвращаемые значения.

- При успешном создании теста возвращается статус 200 и объект созданного теста.

- В случае ошибки возвращается статус 400 (если данные теста некорректны или ошибка при создании теста) или 500 (ошибка на сервере).

GET /api/university/[universityId]/detailed.

Входные данные. Идентификатор университета.

Описание. Запрашивает данные об университете, включая связанные факультеты, специальности и дополнительную информацию.

Возвращаемые значения.

- При успешном выполнении возвращается статус 200 и объект университета с детальной информацией.

- В случае ошибки возвращается статус 400 (если университет с указанным идентификатором не найден) или 500 (ошибка на стороне сервера).

GET /api/university/[universityId].

Входные данные. Идентификатор университета.

Описание. Запрашивает данные университета по указанному идентификатору.

Возвращаемые значения.

- При успешном выполнении возвращается статус 200 и объект университета.

- В случае ошибки возвращается статус 400 (если университет с указанным идентификатором не найден) или 500 (ошибка на стороне сервера).

POST /api/university/[universityId].

Входные данные. Данные для обновления университета.

Описание. Обновляет данные университета по указанному идентификатору с переданными в запросе значениями.

Возвращаемые значения.

- При успешном обновлении возвращается статус 200 и обновлённый объект университета.

- В случае ошибки возвращается статус 400 (если университет с указанным идентификатором не найден или данные некорректны) или 500 (ошибка на стороне сервера).

DELETE /api/university/[universityId].

Входные данные. Идентификатор университета.

Описание. Удаляет университет с указанным идентификатором из базы данных.

Возвращаемые значения.

- При успешном удалении возвращается статус 200 и данные об удалённом университете.

- В случае ошибки возвращается статус 400 (если университет с указанным идентификатором не найден) или 500 (ошибка на стороне сервера).

GET /api/university/detailed.

Входные данные. Нет входных данных.

Описание. Запрашивает полные данные обо всех университетах, включая их факультеты, специальности и дополнительную информацию.

Возвращаемые значения.

- При успешном выполнении возвращается статус 200 и массив объектов университетов с вложенными данными о факультетах, специальностях и дополнительной информации.

- В случае ошибки возвращается статус 400 (если данные не могут быть получены) или 500 (ошибка на стороне сервера).

POST /api/university.

Входные данные. Данные университета, включая идентификатор университета и идентификатор пользователя.

Описание. Создаёт новый университет в базе данных, связывая его с пользователем по переданному идентификатору.

Возвращаемые значения.

- При успешном создании возвращается статус 200 и объект университета, созданный в базе данных.

- В случае ошибки возвращается статус 400 (некорректные данные или ошибка создания) или 500 (ошибка на стороне сервера).

GET /api/user/[userId]/detailed.

Входные данные. Идентификатор пользователя.

Описание. Запрашивает подробные данные о пользователе, включая связанную информацию о студенте и тестах, а также о его университете, факультетах, специальностях и дополнительной информации.

Возвращаемые значения.

- При успешном выполнении возвращается статус 200 и объект с подробной информацией о пользователе.

- В случае ошибки возвращается статус 400 (некорректный идентификатор пользователя или пользователь не найден) или 500 (ошибка на стороне сервера).

GET /api/user/[userId].

Входные данные. Идентификатор пользователя.

Описание. Запрашивает данные пользователя по указанному идентификатору.

Возвращаемые значения.

- При успешном выполнении возвращается статус 200 и объект с данными пользователя.

- В случае ошибки возвращается статус 400 (некорректный идентификатор или пользователь не найден) или 500 (ошибка на стороне сервера).

POST /api/user/[userId].

Входные данные. Идентификатор пользователя и обновлённые данные пользователя.

Описание. Обновляет данные существующего пользователя с указанным идентификатором на основе переданных данных.

Возвращаемые значения.

- При успешном обновлении возвращается статус 200 и объект с обновлёнными данными пользователя.

- В случае ошибки возвращается статус 400 (некорректный идентификатор или данные) или 500 (ошибка на стороне сервера).

DELETE /api/user/[userId].

Входные данные. Идентификатор пользователя.

Описание. Удаляет пользователя с указанным идентификатором.

Возвращаемые значения.

- При успешном удалении возвращается статус 200 и объект с удалёнными данными пользователя.

- В случае ошибки возвращается статус 400 (некорректный идентификатор или пользователь не найден) или 500 (ошибка на стороне сервера).

POST /api/user.

Входные данные. Данные нового пользователя.

Описание. Создаёт нового пользователя на основе переданных данных.

Возвращаемые значения.

- При успешном создании возвращается статус 200 и объект с данными созданного пользователя.

- В случае ошибки возвращается статус 400 (некорректные данные) или 500 (ошибка на стороне сервера).

3.3 База данных

Для реализации функционала веб-приложения была разработана реляционная база данных с использованием PostgreSQL. В основе структуры базы данных лежит несколько

связанных между собой таблиц, каждая из которых отвечает за хранение информации, необходимой для работы системы.

Для демонстрации структуры базы данных приведена схема с указанием таблиц (рисунок 3.20), их атрибутов и связей, отражающая логические отношения между сущностями и обеспечивающая целостность данных.

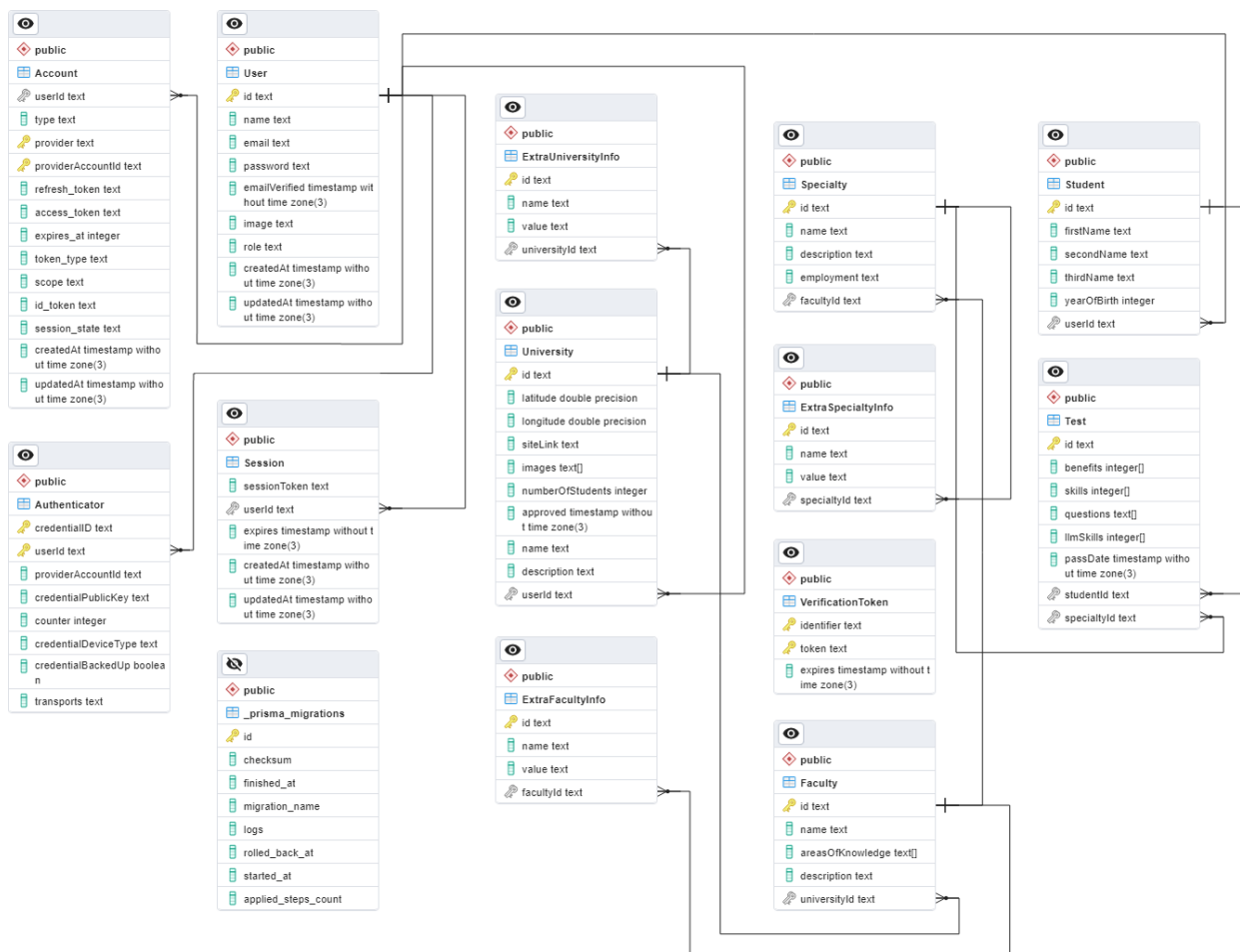


Рисунок 3.20 – ERD диаграмма базы данных.

Ниже представлено описание структуры базы данных.

User (Пользователь). Таблица для хранения данных всех пользователей системы.

Поля.

- id – уникальный идентификатор пользователя (генерируется с

использованием cuid()).

- name – имя пользователя (опционально).
- email – уникальный адрес электронной почты.
- password – хэшированный пароль (опционально).
- emailVerified – дата подтверждения электронной почты.

- image – URL изображения профиля (опционально).
- role – роль пользователя в системе (например, студент, университет).
- createdAt – дата создания записи.
- updatedAt – дата последнего обновления записи.

Связи.

- Один пользователь может быть связан с несколькими аккаунтами (Account[]) и сессиями (Session[]).

- Пользователь может быть студентом (Student) или представителем университета (University).

Student (Студент). Таблица для хранения информации о студентах.

Поля.

- id – уникальный идентификатор студента.
- firstName, secondName, thirdName – персональные данные (имя, фамилия, отчество).
- yearOfBirth – год рождения.

Связи.

- Связан с таблицей User (опционально).
- Один студент может проходить несколько тестов (Test[]).

University (Университет). Таблица для хранения данных об образовательных учреждениях.

Поля.

- id – уникальный идентификатор университета.
- latitude, longitude – координаты университета.
- siteLink – ссылка на официальный сайт.
- images – массив URL изображений университета.
- numberOfStudents – общее количество студентов.
- approved – дата подтверждения университета в системе.
- name – название университета.
- description – описание университета.

Связи.

- Университет может иметь несколько факультетов (Faculty[]) и дополнительные данные (ExtraUniversityInfo[]).

- Связан с пользователем через поле userId.

Faculty (Факультет). Таблица для хранения информации о факультетах университетов.

Поля.

- id – уникальный идентификатор факультета.
- name – название факультета.

- areasOfKnowledge – массив областей знаний, представленных на факультете.

- description – описание факультета.

Связи.

- Факультет относится к конкретному университету (University).

- Имеет связи со специальностями (Specialty[]) и дополнительной информацией (ExtraFacultyInfo[]).

Specialty (Специальность). Таблица для хранения информации о направлениях подготовки.

Поля.

- id – уникальный идентификатор специальности.

- name – название специальности.

- description – описание специальности.

- employment – информация о трудоустройстве.

Связи.

- Специальность относится к факультету (Faculty).

- Связана с тестами (Test[]) и дополнительной информацией (ExtraSpecialtyInfo[]).

Test (Тест). Таблица для хранения данных о тестировании студентов.

Поля.

- id – уникальный идентификатор теста.

- benefits, skills, llmSkills – массивы числовых данных о результатах теста.

- questions – массив вопросов теста.

- passDate – дата прохождения теста.

Связи.

- Связан со студентом (Student) и, опционально, со специальностью (Specialty).

ExtraUniversityInfo, ExtraFacultyInfo, ExtraSpecialtyInfo. Таблицы для хранения дополнительных данных об университетах, факультетах и специальностях. Содержат поля name (название), value (значение) и ссылки на основную таблицу (universityId, facultyId, specialtyId).

Account, Session, VerificationToken, Authenticator. Таблицы для работы с аутентификацией и авторизацией.

3.4 Развертывание

Развёртывание системы профориентационного тестирования включает использование технологий контейнеризации Docker и облачной инфраструктуры RuVDS [3]. В данном разделе описаны основные этапы создания Docker-образов, настройки сервера, доставки и запуска системы [11].

Сервер для размещения приложения предоставлен облачным провайдером RuVDS. Его характеристики.

Дата-центр. Rucloud (Россия, Королёв)

Операционная система. Ubuntu 22.04 LTS (ENG)

Конфигурация. 1x2.2 ГГц CPU, 1 ГБ RAM, 20 ГБ HDD RAID

IP-адрес. 194.87.248.134

Доступ. SSH готовность

Эта конфигурация достаточно проста, но достаточна для развёртывания MVP проекта. Сервер использует статический IP-адрес для доступа извне.

Для системы были созданы три Docker-образа.

1 nginx:alpine Базовый образ Nginx версии Alpine. Используется для обработки HTTP-запросов и проксирования на backend. Файл конфигурации Nginx расположен по пути ./nginx/nginx.conf.

2 postgres:latest Официальный образ PostgreSQL. Пользователь: profai_user. Пароль: profai_password. База данных: profai_db. Для начальной инициализации базы данных используется скрипт ./db/init.sql.

3 profai-site:latest Собственный образ приложения, построенный на основе Next.js. Включает серверную часть (API маршруты) и клиентскую часть (React). Используемые ENV-переменные: DATABASE_URL: URL подключения к PostgreSQL. NEXTAUTH_URL: URL для NextAuth. AUTH_SECRET: секретный ключ для NextAuth. Сборка: Устанавливаются зависимости (npm install). Выполняется сборка (npm run build) [12]. При запуске выполняются миграции Prisma и старт приложения.

Создание образов локально.

```
docker save -o nginx.tar nginx:alpine
docker save -o postgres.tar postgres:latest
docker save -o profai-site.tar profai-site:latest
```

Передача на сервер.

```
scp nginx.tar postgres.tar profai-site.tar root@194.87.248.134:/home
```

Распаковка и запуск.

```
cd /home/profai
docker load < nginx.tar
docker load < postgres.tar
docker load < profai-site.tar
docker compose -f ./docker-compose-deploy.yml up
```

/home/profai — директория, содержащая все файлы проекта.

docker-compose-deploy.yml — файл для запуска Docker Compose.

nginx/nginx.conf — конфигурация Nginx.

db/init.sql — скрипт инициализации базы данных.

site/.env и site/.env.development — файлы переменных окружения.

Nginx используется как прокси-сервер для маршрутизации запросов. Основные настройки. Прослушивание порта 80. Проксирование запросов на контейнер с приложением (порт 3000). Настройка заголовков для поддержки WebSocket и предотвращения кэширования.

Конфигурация расположена в файле nginx.conf.

```
server {
    listen 80;
    server_name localhost;

    location / {
        proxy_pass http://site:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

Как это работает.

- Инициализация. Система запускается через docker compose up, создавая три контейнера: сайт, базу данных и Nginx.
- Пользователи отправляют HTTP-запросы, которые обрабатываются Nginx.
- Nginx перенаправляет запросы на приложение (Next.js).
- Приложение взаимодействует с PostgreSQL через Prisma для чтения и записи данных.
- Управление. Контейнеры автоматически перезапускаются при сбоях благодаря параметру restart: unless-stopped в docker-compose.yml.

4 ТЕСТИРОВАНИЕ

Основной поток действий пользователя с ролью student.

- 1 Регистрация на сайте
- 2 Переход по кнопке “Пройти тестирование”
- 3 Заполнение формы
- 4 Нажатие на кнопку “Отправить”
- 5 Просмотр результатов тестирования
- 6 Переход в профиль пользователя
- 7 Редактирование профиля пользователя и сохранение.

Создадим новый аккаунт используя email test@gmail.com и пароль 12345678. Выбираем роль “учащийся” и вводим необходимые данные, нажимаем кнопку “Войти” (рисунок 4.1).

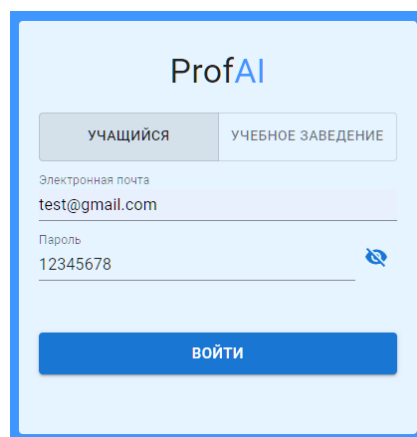


Рисунок 4.1 – Создание нового аккаунта

Успешно проходя регистрацию попадаем на страницу поиска по специальностям. В верхней правой части сайта (рисунок 4.2) видим кнопку “Пройти тест”, нажимаем для перехода на страницу проведения теста.



Рисунок 4.2 – Шапка страницы поиска по специальностям.

Нас перенаправляет на страницу тестирования, заполняем необходимые формы и в низу сайта мы видим кнопку “Отправить”, которую нажимаем. После этого появляется колесо загрузки, которое пропадет после завершения расчета результатов и нас перенаправит на страницу просмотра результатов тестирования (рисунок 4.3 и 4.4).



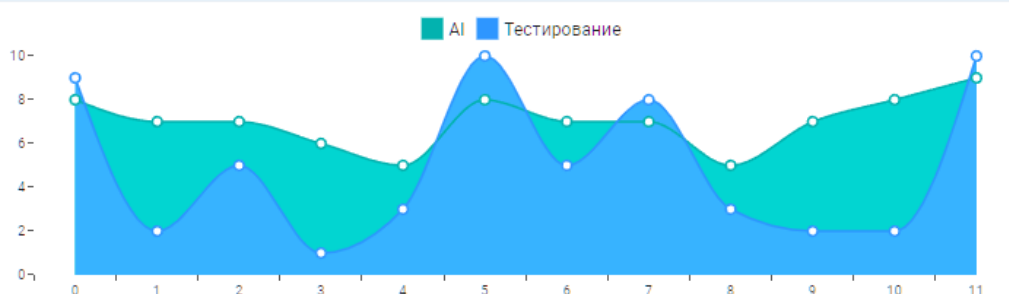
Искусственный интеллект

Специальность «Искусственный интеллект» в БрГТУ готовит специалистов, способных разрабатывать и внедрять интеллектуальные системы в различные сферы деятельности. Обучение включает изучение методов машинного обучения, нейронных сетей, анализа данных и компьютерного зрения. Студенты получают навыки создания программного обеспечения, способного анализировать большие объемы информации, принимать решения и обучаться на основе опыта. Выпускники могут работать в IT-компаниях, научно-исследовательских центрах, а также в организациях, использующих интеллектуальные системы для автоматизации процессов.



Программирование, Компьютерная инженерия, Искусственный интеллект, Электроника, Системы управления информацией

Брестский государственный технический университет / Факультет электронных информационных систем



№	Навык	Тестирование	AI	Среднее
0	Аналитическое мышление	9	8	8.5
1	Коммуникативные навыки	2	7	4.5
2	Творческое мышление	5	7	6
3	Физическая выносливость	1	6	3.5
4	Лидерство	3	5	4
5	Работа с техническим оборудованием	10	8	9
6	Организационные способности	5	7	6
7	Устойчивость к стрессу	8	7	7.5
8	Эмпатия	3	5	4
9	Мелкая моторика и точность	2	7	4.5
10	Быстрая реакция	2	8	5
11	Навыки самообучения	10	9	9.5

Рисунок 4.3 – Страница результатов тестирования. Численные данные тестирования.

№	Предпочтительные критерии	№	Менее интересные критерии
1	Финансовая стабильность	1	Работа с людьми
2	Востребованность на рынке	2	Связь с природой или окружающим миром
3	Работа с интересными задачами	3	Возможность путешествовать и работать
4	Престиж и уважение	4	Гибкость в управлении рабочим графиком
5	Вклад в будущее	5	Эстетическое удовольствие от работы
6	Возможность помогать людям	6	Творческая реализация

Чем вы любите заниматься в свободное время?
Я люблю играть в компьютерные игры, например, дота 2, разрабатывать сайты, люблю рисовать карандашами и красками по холсту, люблю гулять по улице, ходить в тренажерный зал.

Какая школьная дисциплина вам давалась легче всего и почему?
Математика. У меня был хороший учитель и мне нравилось решать задачи.

Какой свой навык или умение вы считаете самым сильным?
Думаю, что усидчивость, ведь, если я взялся за работу, я хочу сидеть за ней пока не закончу.

Каким критериям должна соответствовать работа вашей мечты?
Я хочу много зарабатывать, заниматься разработкой сайтов на React, удаленный режим работы, хорошую команду, с которой я смогу развиваться в программировании.

Вы предпочитаете работать в одиночку или в команде? Почему?
В команде, так как с командой можно реализовать более глобальные вещи, чем в одиночку.

Рисунок 4.4 – Страница результатов тестирования. Ответы на естественном языке.

Ответы на естественном языке мы подаём модели Perplexity на вход для формирования вектора оценки навыков пользователя с помощью данной инструкции:

`Сейчас я дам тебе ответы пользователя на вопросы по профориентации, твоя задача сделать целочисленную оценку от 0 до 10 по следующим критериям:

1. Аналитическое мышление

Описание: способность анализировать данные, находить закономерности, прогнозировать результаты и принимать решения на основе логики.

Оценка 0: Не способен интерпретировать сложные данные, часто упускает важные детали.

Оценка 5: Может анализировать данные при наличии чётких инструкций и решать стандартные задачи.

Оценка 10: Умеет работать с большими массивами данных, находит взаимосвязи даже в сложных условиях.

2. Коммуникативные навыки

Описание: умение ясно выражать свои мысли, слушать и эффективно взаимодействовать с окружающими.

Оценка 0: С трудом доносит информацию, часто вызывает недопонимание.
 Оценка 5: Умеет поддерживать разговор в обычных ситуациях, но испытывает сложности с убеждением.
 Оценка 10: Легко находит общий язык, убеждает в своих идеях, выступает перед большой аудиторией.

3. Творческое мышление
 Описание: способность генерировать новые идеи и нестандартные решения.
 Оценка 0: Не склонен к оригинальным решениям, следует только шаблонам.
 Оценка 5: Может предложить свежую идею в рамках ограниченных задач.
 Оценка 10: Постоянно создаёт инновационные идеи, вдохновляя других.

4. Физическая выносливость
 Описание: способность выдерживать физические нагрузки длительное время.
 Оценка 0: Устаёт быстро, не может выполнять тяжёлую физическую работу.
 Оценка 5: Может справляться с умеренными нагрузками, требующими регулярного перерыва.
 Оценка 10: Способен выдерживать длительные нагрузки без снижения эффективности.

5. Лидерство
 Описание: умение брать на себя ответственность, организовывать и мотивировать команду.
 Оценка 0: Избегает ответственности, не умеет вести за собой.
 Оценка 5: Может руководить командой при наличии ясного плана.
 Оценка 10: Легко берёт ответственность за сложные проекты, вдохновляет других.

6. Работа с техническим оборудованием
 Описание: умение работать с инструментами, машинами или специализированными устройствами.
 Оценка 0: Не знает, как пользоваться даже простым оборудованием.
 Оценка 5: Справляется с настройкой и использованием техники при инструктаже.
 Оценка 10: Легко разбирается с новым оборудованием и обучает других.

7. Организационные способности
 Описание: умение планировать, распределять задачи и управлять временем.
 Оценка 0: Неорганизован, забывает о сроках и обязательствах.
 Оценка 5: Умеет организовать небольшие процессы и контролировать их выполнение.
 Оценка 10: Эффективно управляет сложными проектами с несколькими этапами.

8. Устойчивость к стрессу
 Описание: способность сохранять работоспособность в условиях давления.
 Оценка 0: Легко теряется в сложных ситуациях, не может продолжать работу.
 Оценка 5: Сохраняет спокойствие в большинстве рабочих ситуаций.
 Оценка 10: Легко работает под давлением, принимает взвешенные решения.

9. Эмпатия
 Описание: способность понимать чувства других людей и сопереживать им.
 Оценка 0: Не замечает эмоций окружающих, избегает общения.
 Оценка 5: Понимает эмоциональное состояние других, но не всегда готов помочь.
 Оценка 10: Всегда замечает эмоциональные изменения и поддерживает окружающих.

10. Мелкая моторика и точность
 Описание: способность выполнять точные движения с использованием мелкой моторики.
 Оценка 0: Плохо справляется с мелкими деталями.
 Оценка 5: Уверенно выполняет задачи, требующие средней точности.
 Оценка 10: Выполняет сложные задачи с высокой точностью.

11. Быстрая реакция
 Описание: способность быстро реагировать на изменения и принимать решения.

Оценка 0: Долго думает, замедляет процесс работы.

Оценка 5: Реагирует быстро в большинстве случаев, но допускает ошибки в стрессе.

Оценка 10: Всегда мгновенно реагирует и принимает точные решения.

12. Навыки самообучения

Описание: способность самостоятельно осваивать новую информацию и применять её на практике.

Оценка 0: Не проявляет интереса к обучению, требует постоянного руководства.

Оценка 5: Осваивает новые навыки при наличии руководства.

Оценка 10: Быстро и самостоятельно учится новому, внедряя это в работу.

Формат вывода ответа следующий, ты пишешь название критерия, замет через дефис оценку, которую ты дал этому критерию в соответствии с описанием пользователя, например:

Аналитическое мышление - 10

Коммуникативные навыки - 3

Творческое мышление - 4

Физическая выносливость - 9

Лидерство - 1

Работа с техническим оборудованием - 4

Организационные способности - 6

Устойчивость к стрессу - 7

Эмпатия - 5

Мелкая моторика и точность - 2

Быстрая реакция - 5

Навыки самообучения - 1

Вот описание пользователя: \${text}`

Алгоритм сравнения результатов теста пользователя с результатами накопленными ранее и формирование набора из пяти рекомендуемых специальностей:

1 Выбор целевого теста. Выбирается тест пользователя, для которого нужно найти подходящие рекомендации

2 Получение данных для сравнения. Из базы данных извлекаются тесты, связанные со специальностями, за исключением целевого теста

3 Сравнение преимуществ. Для каждого теста из выборки. Анализируются списки преимуществ из целевого теста и текущего теста. Вычисляется разница в позициях совпадающих преимуществ. На основе этих различий определяется показатель близости по преимуществам

4 Сравнение навыков. Сравняются значения навыков между целевым тестом и текущим. Разница между каждым навыком суммируется. Полученная сумма нормализуется для определения показателя близости. Аналогично сравниваются навыки, рассчитанные языковой моделью

5 Сохранение результатов. Для каждого теста сохраняются рассчитанные показатели близости по преимуществам и навыкам, а также связанная с ним специальность

6 Агрегация результатов по специальностям. Результаты группируются

по специальностям. Для каждой из них. Суммируются показатели близости по преимуществам и навыкам. Подсчитывается количество тестов, относящихся к специальности

7 Расчет средних значений. Для каждой специальности рассчитываются средние показатели по преимуществам и навыкам. Эти значения складываются для определения общего показателя соответствия

8 Сортировка специальностей. Специальности упорядочиваются по общему показателю соответствия. Первые в списке – наиболее подходящие

9 Выбор рекомендаций. Из отсортированного списка выбираются пять специальностей с наилучшими показателями.

По итогам сравнения данных пройденного теста с результатами тестов других пользователей мы получаем оценки сходства и пять наиболее подходящих факультетов (см. таблицу 4.1).

Таблица 4.1 – Рекомендованные специальности.

Специальность	Критерии, %	Тестирование, %	LLM, %
Автоматизированные системы обработки информации	76.4	75	93.3
Мобильные системы	72.2	77.5	90.8
Техническая эксплуатация автомобилей	77.8	73.3	93.3
Специальность	Критерии, %	Тестирование, %	LLM, %
Бухгалтерский учет, анализ и аудит	75	80	86.7
Финансы и кредит	86.1	73.3	81.7

Рекомендации рассчитываются по косинусному расстоянию между тестом показателями и показателями тестирования студентов данных специальностей, после чего специальности ранжируются от самого подходящего по сумме оценок до менее подходящего. При создании новой специальности ИИ создает мок-тест, используемый для расчетов пока статистика по специальности не будет полной.

По итогу тестирования весь функционал основного потока действий пользователя работает корректно.

5 РАСЧЕТ ТЕХНИКО-ЭКОНОМИЧЕСКИХ ПОКАЗАТЕЛЕЙ

5.1 Исходные данные для осуществления расчета

Любая разработка программного обеспечения, веб-приложений или их модулей требует финансовых, временных и человеческих ресурсов.

Задачей данного дипломного проекта является разработка веб-приложения для профориентационного тестирования, включая использование крупной языковой модели для анализа ответов пользователей. В процессе разработки применялись современные технологии, такие как Next.js, TypeScript, Prisma, PostgreSQL, и интеграция с крупными языковыми моделями (LLM) [13].

Разработка программного продукта предусматривает проведение всех этапов проектирования, включая создание архитектуры базы данных, серверной и клиентской частей приложения, а также оптимизацию взаимодействия с языковыми моделями. Проект относится к первой группе сложности.

Последовательность расчетов:

- 1 Расчёт объёма функций программных модулей
- 2 Расчёт полной себестоимости разработки функционала анализа данных с использованием языковой модели
- 3 Расчёт цены и прибыли по разработанному программному продукту.

5.2 Расчет объема функций

Наименование проекта: «Разработка метода интеграции крупной языковой модели для анализа ответов пользователей в рамках профориентационного тестирования».

Среда разработки: Visual Studio Code, Next.js.

Определение общего объема системы

Общий объем системы (Vo) определяется исходя из количества и объема функций, реализуемых программой, по формуле (5.1):

$$V_0 = \sum_{i=1}^n V_i,$$
 (5.1)

где V_0 – общий объем системы,
 V_i – объем отдельной функции системы,
 n – общее число функций.

Расчёт уточнённого объёма системы

На основании анализа структуры проекта и функциональности приложения объем каждой функции определяется с учетом её реализации. Общий уточнённый объем системы (V_y) рассчитывается по формуле (5.2):

$$V_y = \sum_{i=1}^n V_{yi}, \quad (5.2)$$

где: V_{yi} – уточнённый объем отдельной функции в строках исходного кода (см. таблицу 5.1).

Таблица 5.1 – Перечень и объем функций программного обеспечения

Код функции	Наименование (содержание) функции	Объем функции (LOC)	
		По каталогу V_i	Уточнённый V_{yi}
101	Авторизация и управление учетными записями пользователей	559	432
201	Управление профилями (студент/вуз)	751	599
301	Формирование тестов на основе интеграции языковой модели	590	487
401	Анализ ответов тестируемых и предоставление результатов	671	556
501	Обработка запросов поиска информации о вузах, специальностях и факультетах	755	645
601	Отображение детальной информации о вузах, факультетах, специальностях	977	837
701	Генерация рекомендаций по профориентации на основе результатов тестирования	579	487
801	Управление дополнительной информацией о вузах, факультетах и специальностях	542	432
901	Управление пользовательским интерфейсом (интерактивные компоненты и провайдеры)	551	432
1001	Реализация вспомогательных утилит для обработки данных	675	556
1101	Управление состоянием приложения	47	38
1201	Модуль конфигурации приложения	115	94

Продолжение таблицы 5.1

Код функции	Наименование (содержание) функции	Объем функции (LOC)	
		По каталогу V_i	Уточнённый V_{yi}
1301	Реализация API для взаимодействия с базой данных и внешними модулями	1763	1628
ИТОГО		7893	7275

На основании анализа общего объема системы уточнённый объем системы (V_y) составляет 7275 LOC, что соответствует полному объему проекта и его функциональным требованиям.

5.3 Расчет полной себестоимости программного модуля

Стоимостная оценка программного средства у разработчика предполагает составление сметы затрат, включающие следующие статьи расходов:

- 1 Отчисления на социальные нужды ($P_{соц}$)
- 2 Материалы и комплектующие изделия (P_M)
- 3 Спецоборудование (P_C)
- 4 Машинное время ($P_{МВ}$)
- 5 Расходы на научные командировки ($P_{НК}$)
- 6 Прочие прямые расходы ($P_{пр}$)
- 7 Накладные расходы ($P_{НР}$)
- 8 Затраты на освоение и сопровождение программного средства (P_O и $P_{со}$).

Полная себестоимость (C_{Π}) разработки программного продукта рассчитывается как сумма расходов по всем статьям с учётом рыночной стоимости аналогичных продуктов.

Основной статьёй расходов на создание программного продукта является заработная плата проекта (основная и дополнительная) разработчиков (исполнителей) ($ЗП_{осн} + ЗП_{доп}$), в число которых принято включать инженеров–программистов, руководителей проекта, системных архитекторов, дизайнеров, разработчиков баз данных и других специалистов, необходимых для решения специальных задач в команде.

Расчёт заработной платы разработчиков программного продукта начинается с определения:

1 Продолжительности времени разработки ($\Phi_{рв}$), которое устанавливается студентом экспертным путём с учётом сложности, новизны ПО и фактически затраченного времени. В данном дипломном проекте $\Phi_{рв}$ – 3 месяца

2 Количества разработчиков программного обеспечения.

В данном дипломном проекте будет разработчик: инженер–программист. Заработная плата разработчиков определятся как сумма основной и дополнительной заработной платы всех исполнителей.

Основная заработная плата $ЗП_{осн}$ каждого исполнителя определяется по формуле (5.3).

$$ЗП_{осн} = Т_{бтс} \cdot \frac{T_k}{22} \cdot \Phi_{рв} \cdot K_{пр}, \quad (5.3)$$

где $T_{бтс}$ – размер базовой ставки, на текущий момент равный 270 бел.руб.;

T_k – тарифный коэффициент согласно разряду исполнителя; 22 – среднее количество рабочих дней в месяце;

$\Phi_{рв}$ – фонд рабочего времени исполнителя (продолжительность разработки программного модуля), дни;

$K_{пр}$ – коэффициент премии, $K_{пр} = 1,5$.

Таким образом тарифный коэффициент согласно 11 разряду инженера–программиста $T_k = 1,91$. Продолжительность разработки программного продукта – 90 дней, количество рабочих дней в месяце на текущий год составляет 22 день. Рассчитаем основную заработную плату инженера–программиста, согласно формуле 5.3:

$$ЗП_{осн} = 270 \cdot \frac{1,91}{22} \cdot 90 \cdot 1,5 = 3\,164,52 \text{ (белорусских рубля)}$$

Дополнительная заработная плата $ЗП_{доп}$ каждого исполнителя рассчитывается от основной заработной платы по формуле (5.4).

$$ЗП_{доп} = ЗП_{осн} \cdot \frac{H_{доп. зп}}{100\%}, \quad (5.4)$$

где $H_{доп. зп}$ – надбавка дополнительной заработной платы, равная 15%. Рассчитаем дополнительную заработную плату инженера–программиста, согласно формуле 6.3:

$$ЗП_{доп} = 3\,164,52 \cdot \frac{15\%}{100\%} = 474,68 \text{ (белорусских рубля)}.$$

Результаты вычислений внесём в таблицу 5.2.

Таблица 5.2 – Расчет заработной платы

Категория работников	Разряд	Тарифны коэффициент (Т _к)	Фэфф.р.в. (дн.)	Коэффициент премии (К _{пр})	Заработная плата, бел.руб.		
					Основная	Дополнительна	Всего
Инженер–программист	11	1,91	90	1,5	3 164,52	474,68	3 639,2
Итого	–	–	–	–	3 164,52	474,68	3 639,2

Таким образом, как видно из таблицы, заработная плата инженера–программиста составляет 3 639,2 бел. руб.

Отчисления на социальные нужды Р_{соц} определяются по формуле (5.5) в соответствии с действующим законодательством по нормативу.

$$P_{\text{соц}} = (3П_{\text{осн}} + 3П_{\text{доп}}) \cdot \frac{35\%}{100\%}, \quad (5.5)$$

Рассчитаем отчисления на социальные нужды:

$$P_{\text{соц}} = (3\,164,52 + 474,68) \cdot \frac{35\%}{100\%} = 1\,273,72 \text{ (белорусских рубля).}$$

Расходы на материалы и комплектующие Р_м отражают расходы на магнитные носители, бумагу, красящие ленты и другие материалы, необходимые для разработки программного продукта. Норма расхода материалов в суммарном выражении определяется в процентах к основной заработной плате (5.6).

$$P_{\text{м}} = 3П_{\text{осн}} \cdot \frac{H_{\text{мз}}}{100\%}, \quad (5.6)$$

где H_{мз} – норма расхода материалов от основной заработной платы, в процентах. Тогда рассчитаем расходы на материалы и комплектующий, приняв H_{мз} равным 3%:

$$P_{\text{м}} = 3\,164,52 \cdot \frac{3\%}{100\%} = 94,94 \text{ (белорусских рубля).}$$

Расходы на спецоборудование P_c включают затраты на приобретение технических и программных средств специального назначения, необходимых для разработки методического пособия, включая расходы на проектирование, изготовление, отладку и другое.

В данном дипломном проекте для разработки системы профориентационного тестирования с использованием LLM приобретение какого-либо спецоборудования не предусматривалось. Так как спецоборудование не было приобретено, расходы равны нулю.

Расходы на машинное время P_{MB} включают оплату машинного времени, необходимого для разработки и отладки программного продукта. Они определяются в машино-часах по нормативам на 100 строк исходного кода машинного времени. P_{MB} определяется по формуле (5.7).

$$P_{MB} = C_{MBi} \cdot \frac{V_0}{100} \cdot H_{MB}, \quad (5.7)$$

где C_{MBi} – цена одного машинного часа (70 бел. руб.);

V_0 – уточнённый общий объём машинного кода;

H_{MB} – норматив расхода машинного времени на отладку 100 строк кода в машино-часах. Принимается в размере 0,7.

Рассчитаем расходы на машинное время

$$P_{MB} = 70 \cdot \frac{7275}{100} \cdot 0,7 = 3\,564,75 \text{ (белорусских рубля)}.$$

Расходы по статье «Научные командировки» $P_{НК}$ берутся либо по смете научных командировок, разрабатываемой на предприятии, либо в процентах от основной заработной платы исполнителей (10–15%). Так как в данном проекте научные командировки не предусмотрены, данная статья не рассчитывается.

Прочие затраты $P_{пр}$ включают затраты на приобретение специальной научно-технической информации и специальной литературы. Определяются по нормативу в процентах к основной заработной плате исполнителей. Так как специальная научно-техническая информация и специальная литература не приобреталась, то данная статья не рассчитывается.

Затраты на накладные расходы $P_{НР}$ связаны с содержанием вспомогательных хозяйств, а также с расходами на общехозяйственные нужды. Определяется по нормативу в процентах к основной заработной плате по формуле (5.8).

$$P_{НР} = \frac{H_{НР}}{100\%} \cdot 3P_{осн}, \quad (5.8)$$

где $H_{\text{нр}}$ – норматив накладных расходов.

В данном дипломном проекте норматив накладных расходов равен 45%, поэтому затраты на накладные расходы равны:

$$P_{\text{нр}} = \frac{45\%}{100\%} \cdot 3\,164,52 = 1\,604,1375 \text{ (белорусских рублей).}$$

Сумма вышеперечисленных расходов по статьям СЗ на программный продукт служит исходной базой для расчёта затрат на освоение и сопровождение программного продукта. Они рассчитываются по формуле (5.9).

$$СЗ = ЗП_{\text{осн}} + ЗП_{\text{доп}} + P_{\text{соц}} + P_{\text{м}} + P_{\text{с}} + P_{\text{мв}} + P_{\text{нк}} + P_{\text{пр}} + P_{\text{нр}}, \quad (5.9)$$

Тогда, $СЗ = ЗП_{\text{осн}} + ЗП_{\text{доп}} + P_{\text{соц}} + P_{\text{м}} + P_{\text{с}} + P_{\text{мв}} + P_{\text{нк}} + P_{\text{пр}} + P_{\text{нр}} = 10\,176,75$ (белорусских рубля).

Организация–разработчик участвует в освоении программного продукта и несёт соответствующие затраты, на которые составляется смета, оплачиваемая заказчиком по договору. Затраты на освоение P_o определяются по установленному нормативу от суммы затрат по формуле (5.10).

$$P_o = СЗ \cdot \frac{H_o}{100\%}, \quad (5.10)$$

где $СЗ$ – сумма вышеперечисленных расходов по статьям на разработку программного продукта;

H_o – установленный норматив затрат на освоение. Для данного дипломного проекта принимается равной 5%.

Рассчитаем затраты на освоение продукта:

$$P_o = 10\,176,75 \cdot \frac{5\%}{100\%} = 508,837375 \text{ (белорусских рубля)}$$

Организация–разработчик осуществляет сопровождение программного продукта и несёт расходы, которые оплачиваются заказчиком в соответствии с договором и сметой на сопровождение. Расходы на сопровождение $P_{\text{со}}$ рассчитываются по формуле (5.11).

$$P_{\text{со}} = СЗ \cdot \frac{H_{\text{со}}}{100\%}, \quad (5.11)$$

где $H_{\text{со}}$ – установленный норматив затрат на сопровождение программного продукта. Для данного дипломного проекта принимается равным 5%.

Рассчитаем расходы на сопровождение:

$$P_{\text{со}} = 10\,176,75 \cdot \frac{5\%}{100\%} = 508,84 \text{ (белорусских рубля).}$$

Полная себестоимость (СП) разработки программного продукта рассчитывается как сумма расходов по всем статьям. Она определяется по формуле (5.12).

$$СП = СЗ + Р_о + Р_{со}, \quad (5.12)$$

Рассчитаем полную себестоимость продукта:

$$СП = 10\,176,75 + 508,84 + 508,84 = 11\,194,43$$

(белорусских рубля). Результаты вычислений занесём в таблицу 5.3.

Таблица 5.3 – Себестоимость программного продукта

Наименование статей затрат	Норматив, %	Сумма затрат, бел.руб.
Заработная плата, всего	—	3 639,2
Основная заработная плата	—	3 164,52
Дополнительная заработная плата	—	474,68
Наименование статей затрат	Норматив, %	Сумма затрат, бел.руб.
Отчисления на социальные нужды	35%	1 273,72
Спецоборудование	Не применялось	—
Материалы и комплектующие изделия	4%	94,94
Машинное время	—	3 564,75
Научные командировки	Не планировались	—
Прочие затраты	Не применялись	—
Накладные расходы	45%	1 604,1375
Сумма затрат	—	10 176,75
Затраты на освоение	5%	508,84
Затраты на сопровождение	5%	508,84
Полная себестоимость	—	11 194,43

В результате всех расчётов полная себестоимость программного продукта составила 11 194,43 бел.руб.

5.4 Расчет отпускной цены и чистой прибыли

Для определения цены программного продукта необходимо рассчитать плановую прибыль П, которая рассчитывается по формуле (5.13).

$$П = СП \cdot \frac{R}{100\%}, \quad (5.13)$$

где СП – полная себестоимость программного модуля, бел. руб; R – уровень рентабельности программного модуля.

В данном дипломном проекте уровень рентабельности равен 20%. Рассчитаем прибыль от реализации:

$$П = 11\,194,43 \cdot \frac{20\%}{100\%} = 2\,238,89 \text{ (белорусских рубля)}$$

После расчета прибыли от реализации по формуле (5.14) определяется прогнозируемая цена программного продукта без налогов Ц_п.

$$Ц_{п} = СП + П, \quad (5.14)$$

Рассчитаем цену программного продукта без налогов:

$$Ц_{п} = 11\,194,43 + 2\,238,89 = 13\,433,32 \text{ (белорусских рубля)}.$$

Отпускная цена Ц_о (цена реализации) программного продукта включает налог на добавленную стоимость и рассчитывается по формуле (5.15).

$$Ц_{о} = СП + П + НДС_{пп}, \quad (5.15)$$

где НДС_{пп} – налог на добавленную стоимость для программного продукта. Для данного программного продукта НДС_{пп} рассчитывается по формуле (5.16).

$$НДС_{пп} = Ц_{п} \cdot \frac{НДС}{100\%}, \quad (5.16)$$

где НДС – налог на добавленную стоимость. В настоящее время он составляет 20%.

Рассчитаем НДС_{пп} и отпускную цену:

$$НДС_{пп} = 11\,194,43 \cdot \frac{20\%}{100\%} = 2\,238,87 \text{ (белорусских рубля)}.$$

$$Ц_0 = 11\,194,43 + 2\,238,89 + 2\,238,87 = 15\,672,19 \text{ (белорусских рубля).}$$

Прибыль от реализации программного продукта за вычетом налога на прибыль является чистой прибылью ПЧ. Чистая прибыль остаётся организации–разработчику и представляет собой экономический эффект от создания нового программного продукта. Она рассчитывается по формуле (5.17).

$$ПЧ = П \cdot \frac{Н_{п}}{100\%}, \quad (5.17)$$

где $Н_{п}$ – ставка налога на прибыль. В настоящее время он равен 20%.

Рассчитаем чистую прибыль:

$$ПЧ = 2\,238,89 \cdot \left(1 - \frac{20\%}{100\%}\right) = 1\,791,11 \text{ (белорусских рубля).}$$

Результаты расчётов цены и прибыли по программному продукту сведены в таблицу 5.4.

В ходе произведенных расчетов определены основные экономические показатели:

- 1 Полная себестоимость – 11 194,43 бел. руб.
- 2 Прогнозируемая цена – 13 433,32 бел. руб.
- 3 Чистая прибыль – 1 791,11 бел. руб.

Таблица 5.4. – Расчёт отпускной цены и чистой прибыли программного модуля

Наименование статей затрат	Норматив, %	Сумма затрат, бел. руб.
Полная себестоимость	–	11 194,43
Прибыль	20	2 238,89
Цена без НДС	–	8 955,56
НДС	20	2 238,87
Отпускная цена	–	15 672,19
Налог на прибыль	20	447,78
Чистая прибыль	–	1 791,11

Разработанный программный продукт ориентирован на автоматизацию процесса профессиональной ориентации учащихся. Рассчитанная отпускная цена приложения составляет 15 672,19 белорусских рубля. Продукт способен конкурировать с существующими аналогами за счёт высокой масштабируемости, удобного интерфейса и адекватной стоимости.

ЗАКЛЮЧЕНИЕ

В процессе разработки дипломного проекта была создана система профориентационного тестирования, основанная на интеграции методов искусственного интеллекта и современных технологий обработки данных. Основной целью проекта стало предоставление абитуриентам инструментов для выбора образовательной траектории с учётом их индивидуальных особенностей. Система учитывает недостатки существующих решений, таких как отсутствие информации о местах получения специальности, узость рекомендаций и субъективность критериев.

По итогам работы был решен следующий ряд задач:

1 Разработана структура системы, поддерживающая гибкость и расширяемость базы данных об учебных заведениях и специальностях

2 Реализованы алгоритмы ранжирования специальностей на основе индивидуальных характеристик пользователя, что позволяет предоставлять релевантные и разносторонние рекомендации

3 Интегрированы классические методы анализа данных и крупная языковая модель (LLM) для повышения точности рекомендаций и минимизации субъективности

4 Создан интерфейс, включающий функционал для студентов и учебных заведений: регистрация, редактирование профилей, тестирование, рекомендации и поиск

5 Обеспечена возможность быстрого обновления тестов без изменений основной архитектуры.

Также данная система имеет ряд преимуществ над другими методами, упомянутыми в этой работе, а именно:

1 Объективность, так как система опирается на собранную статистику, а оценка LLM происходит в равных условиях

2 Конкретика рассчитанных результатов. Система выбирает специальность среди релевантных учебных заведений, предлагая существующие варианты, а не абстрактную профессию

3 Централизованность и расширяемость. Данная система может пополняться и в перспективе стать агрегатором информации об учебных заведениях.

Таким образом, выполненная работа предоставляет пользователям инструмент для принятия осознанного решения о выборе образовательной траектории, что делает систему значимым вкладом в развитие профориентационного тестирования.

СПИСОК СОКРАЩЕНИЙ

- MVP – Minimum Viable Product (Минимально жизнеспособный продукт).
LLM – Large Language Model (Большая языковая модель).
API – Application Programming Interface (Интерфейс программирования приложений).
БД – База данных.
СУБД – Система управления базами данных.
ORM – Object-Relational Mapper (Объектно-реляционное отображение).
JWT – JSON Web Token.
CSS – Cascading Style Sheets.
UX – User Experience (Пользовательский опыт).
SEO – Search Engine Optimization (Поисковая оптимизация).
HTTP – Hypertext Transfer Protocol.
ACID – Atomicity, Consistency, Isolation, Durability (Атомарность, Согласованность, Изолированность, Долговечность).
JSON – JavaScript Object Notation.
DSL – Domain-Specific Language (Предметно-ориентированный язык).
IDE – Integrated Development Environment (Интегрированная среда разработки).
CSRF – Cross-Site Request Forgery (Межсайтовая подделка запросов).
SQL – Structured Query Language (Структурированный язык запросов).
RuVDS – Российский виртуальный выделенный сервер (название хостинг-провайдера).
LTS – Long Term Support (Долгосрочная поддержка).
RAM – Random Access Memory (Оперативная память).
HDD – Hard Disk Drive (Жесткий диск).
RAID – Redundant Array of Independent Disks (Избыточный массив независимых дисков).
IP – Internet Protocol (Интернет-протокол).
SSH – Secure Shell (Безопасная оболочка).
ENV – Environment Variable (Переменная окружения).
URL – Uniform Resource Locator (Унифицированный указатель ресурса).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Next.js [Электронный ресурс]. Режим доступа: <https://nextjs.org/> Дата доступа: 27.04.2025.
2. Perplexity AI [Электронный ресурс]. Режим доступа: <https://www.perplexity.ai/>. Дата доступа: 27.04.2025.
3. RUVDS [Электронный ресурс]. Режим доступа: <https://ruvds.com/ru-rub>. Дата доступа: 27.04.2025.
4. Docker [Электронный ресурс]. Режим доступа: <https://www.docker.com/>. Дата доступа: 27.04.2025.
5. React Hook Form [Электронный ресурс]. Режим доступа: <https://react-hook-form.com>. Дата доступа: 27.04.2025.
6. Material-UI (MUI) [Электронный ресурс]. Режим доступа: <https://mui.com/>. Дата доступа: 27.04.2025.
7. Ant Design [Электронный ресурс]. Режим доступа: <https://ant.design/>. Дата доступа: 27.04.2025.
8. 16 Personalities [Электронный ресурс]. Режим доступа: <https://www.16personalities.com/ru/tipy-lichnosti>. Дата доступа: 27.04.2025.
9. NextAuth.js [Электронный ресурс]. Режим доступа: <https://next-auth.js.org/>. Дата доступа: 27.04.2025.
10. Docker и контейнеризация на практике [Электронный ресурс]. Режим доступа: <https://habr.com/ru/articles/896024/>. Дата доступа: 27.04.2025.
11. Prisma [Электронный ресурс]. Режим доступа: <https://www.prisma.io/> – Дата доступа: 27.04.2025.
12. npm [Электронный ресурс]. Режим доступа: <https://www.npmjs.com/> – Дата доступа: 27.04.2025.
13. TypeScript [Электронный ресурс]. Режим доступа: <https://www.typescriptlang.org/>. Дата доступа: 27.04.2025.
14. Тест Климова [Электронный ресурс]. Режим доступа: <https://www.profguide.io/test/klimov.html>. Дата доступа: 27.04.2025.