**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ 4**](#_Toc179487093)

[**1 СИСТЕМНЫЙ АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 5**](#_Toc179487094)

[1.1 Анализ предметной области 5](#_Toc179487095)

[**2 РАЗРАБОТКА 7**](#_Toc179487103)

[2.1 Интерфейс и взаимодействие: 7](#_Toc179487104)

[2.2 Структура базы данных 17](#_Toc179487105)

[2.3 Что планируется доработать 20](#_Toc179487105)

[**ЗАКЛЮЧЕНИЕ 22**](#_Toc179487108)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 23**](#_Toc179487109)

# ВВЕДЕНИЕ

Выбор подходящего учебного направления – один из ключевых этапов в жизни каждого студента, от которого зависит его будущая профессиональная реализация. В последние годы профориентационные системы, предлагающие студентам и школьникам помощь в определении своей профессиональной траектории, получили широкое распространение. Такие системы позволяют учащимся оценить свои профессиональные склонности и интересы, выявить сильные и слабые стороны, и, исходя из этих данных, выбрать наиболее подходящее направление обучения. Однако традиционные подходы к профориентации часто оказываются недостаточно точными, поскольку базируются на ограниченных и обобщенных методиках оценки, которые не учитывают индивидуальные особенности каждого студента.

С появлением и развитием технологий глубокого обучения и языковых моделей (LLM), возможности профориентационного тестирования значительно расширились. Использование LLM позволяет анализировать ответы пользователей на более глубоком уровне, выявляя тонкие аспекты их профессиональных предпочтений и личных качеств. Применение LLM делает возможным формирование точного профиля пользователя, сопоставимого с особенностями различных образовательных программ и факультетов, что позволяет повысить точность рекомендаций.

В данном проекте представлена разработка системы профориентационного тестирования с применением LLM, которая нацелена на помощь в выборе одного из пяти факультетов Брестского государственного технического университета. Система реализована в виде веб-приложения, где пользователи проходят тест из 25 вопросов. Ответы анализируются с помощью локальной LLM, что позволяет более точно определить подходящий факультет, а также предоставляет подробные объяснения результатов, позволяя пользователю понять, как его ответы связаны с полученной рекомендацией.

# 1 СИСТЕМНЫЙ АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

**1.1 Анализ предметной области**

Профориентационное тестирование направлено на помощь школьникам и студентам в выборе учебного направления и будущей карьеры. Основная цель такого тестирования – выявление предпочтений, способностей и личностных качеств человека, которые соответствуют определённым профессиям и специальностям. Правильно подобранное направление обучения позволяет раскрыть потенциал студента и увеличить его шансы на успешное трудоустройство в будущем.

Существуют различные методы профориентационного тестирования, включающие психологические опросники, анкеты на выявление интересов и тесты на оценку профессиональных склонностей. Эти подходы ориентированы на анализ таких качеств, как аналитическое и креативное мышление, коммуникативные навыки, склонность к работе в команде, уровень физической активности и другие важные характеристики.

Классические методы профориентационного тестирования включают в себя проверенные временем подходы и инструменты, разработанные для выявления интересов, способностей и склонностей человека к определённым профессиональным сферам. Вот основные из них:

Психологические опросники и анкеты на выявление интересов. Опросники, такие как методики Джона Холланда или опросники о профессиональных интересах, позволяют определить профессиональные предпочтения человека, то есть сферы, к которым он испытывает особый интерес. В основе таких опросников лежит предположение, что человеку легче развиваться и быть успешным в деятельности, которая его увлекает. Например, результаты тестов Холланда распределяют участников по типам личности, таким как реалистический, исследовательский, артистический, социальный, предпринимательский и конвенциональный, что помогает уточнить подходящую сферу занятости.

Тесты на оценку профессиональных склонностей и способностей. Эти тесты оценивают уровни развития определённых навыков, таких как аналитическое мышление, внимание к деталям, коммуникативные или лидерские качества, что позволяет определить, какие области могут подходить человеку. Например, тесты на когнитивные способности выявляют силу в таких областях, как математическое или вербальное мышление, что может помочь в выборе профессий, требующих аналитических или языковых навыков.

Методики на оценку личностных качеств. В рамках профориентации также часто используются тесты на личностные качества, такие как 16-факторный личностный опросник Кэттелла или MBTI (Майерс-Бриггс). Эти тесты помогают понять, как человек взаимодействует с окружающим миром, принимает решения, предпочитает ли структурированную деятельность или гибкие задачи. Результаты таких тестов показывают, как черты характера могут способствовать или мешать успешной работе в тех или иных профессиях.

Практические упражнения и кейс-задания. В некоторых профориентационных тестированиях используются упражнения, имитирующие реальные профессиональные задачи. Это может быть решение логических задач для будущих программистов или кейс-задания на общение для тех, кто склонен к работе с людьми. Подобные методы позволяют выявить способности человека в реальных условиях, и такие данные помогают сопоставить его с профессиями, требующими аналогичных навыков.

Эти традиционные методы широко применяются и сегодня, так как дают возможность объективно оценить предпочтения и склонности человека, не прибегая к сложным технологиям. Однако их результаты ограничены формальными рамками и могут не учитывать тонких психологических и социальных факторов, проявляющихся в индивидуальном поведении.

В современных профориентационных системах классические методы могут быть усилены искусственным интеллектом и большими языковыми моделями (LLM), что позволяет более точно анализировать личностные черты, предпочтения и даже скрытые интересы человека. Например, LLM могут не только интерпретировать текстовые ответы, но и создавать связи между личностными характеристиками и особенностями разных профессий, обеспечивая более точные и детализированные результаты профориентации.

Основная цель системы профориентационного тестирования – помочь абитуриентам определить специальность, которая наиболее соответствует их личным качествам, интересам и профессиональным наклонностям. Система должна проанализировать ответы на серию вопросов, выявить ключевые черты, сопоставить их с требованиями различных факультетов и предложить наиболее подходящее учебное направление.

**Определение требований к системе**

1. **Точность и обоснованность рекомендаций.** Система должна обеспечивать высокую статистическую достоверность рекомендаций. Для этого используются методы анализа данных, выявляющие явные и скрытые признаки, связанные со склонностями и способностями абитуриента.
2. **Простота и удобство использования.** Интерфейс должен быть интуитивно понятен, чтобы тестирование было комфортным и доступным. Это включает логичную структуру вопросов, простую навигацию и возможность быстрого завершения этапов теста.
3. **Интерактивность и визуализация результатов.** Результаты тестирования должны представляться наглядно (графики, схемы), показывая уровни развития различных профессиональных качеств пользователя и их сопоставление с характеристиками факультетов.
4. **Гибкость анализа ответов.** Система должна анализировать текстовые ответы пользователей, выявляя нюансы и особенности с помощью методов на основе больших языковых моделей (LLM). Это позволит формировать уникальный дескриптор пользователя, включающий его ключевые характеристики и сопоставлять их с требованиями факультетов.

**Ожидаемый результат**

Разработанная система должна помочь пользователю осознанно выбрать факультет, максимально подходящий к его личностным характеристикам и профессиональным склонностям, а также предоставить прозрачные и понятные обоснования этому выбору.

# 2 РазРАБОТКА

**2.1 Интерфейс и взаимодействие**

Система профориентационного тестирования предназначена для абитуриентов, чтобы определить наиболее подходящий факультет на основе их ответов на тестовые вопросы. Она представлена как веб-приложение, где серверная и клиентская части выполняют разные, но взаимодополняющие функции.

**Список страниц:**

**(opened)\auth**

На странице авторизации пользователь может выбрать свою роль — учащегося или учебного заведения, ввести email и пароль для входа, а также при необходимости показать или скрыть пароль. Если данные введены некорректно, появятся подсказки с ошибками. После заполнения формы можно отправить данные и войти в систему. Всё сделано просто и удобно для разных категорий пользователей. (Рисунок 2.1)

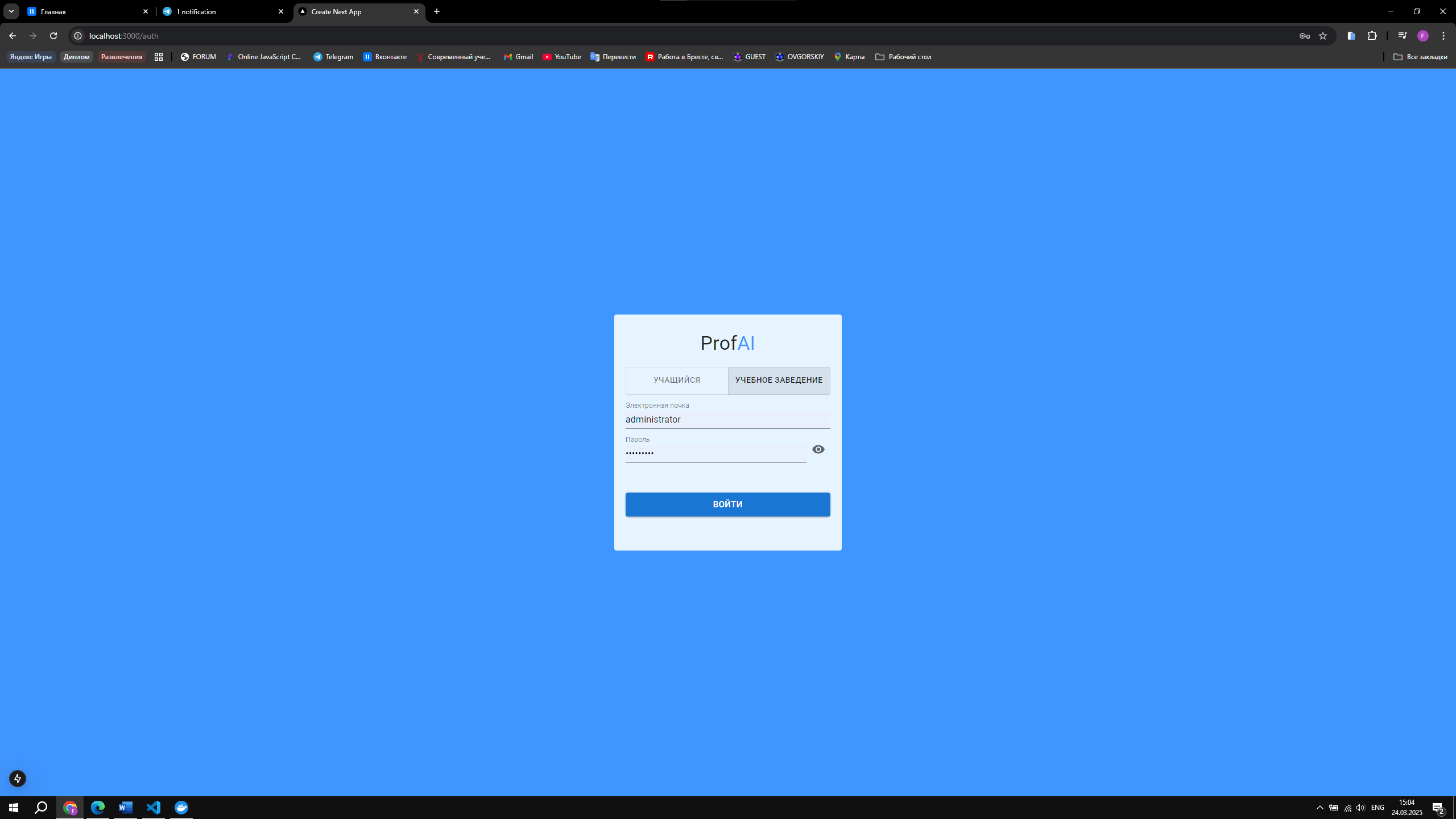


Рисунок 2.1 – AuthPage

**(protected)\preview\university[universityId]\faculty[facultyId]\specialty[specialtyId]**

На этой странице можно подробно изучить информацию об учебном заведении, его факультетах и доступных специальностях. Пользователь видит описание университета, количество студентов, ссылку на официальный сайт, данные о направлениях факультета и их особенностях, а также детали выбранной специальности, включая возможности трудоустройства. Всё представлено структурировано и удобно для восприятия. (Рисунок 2.2)

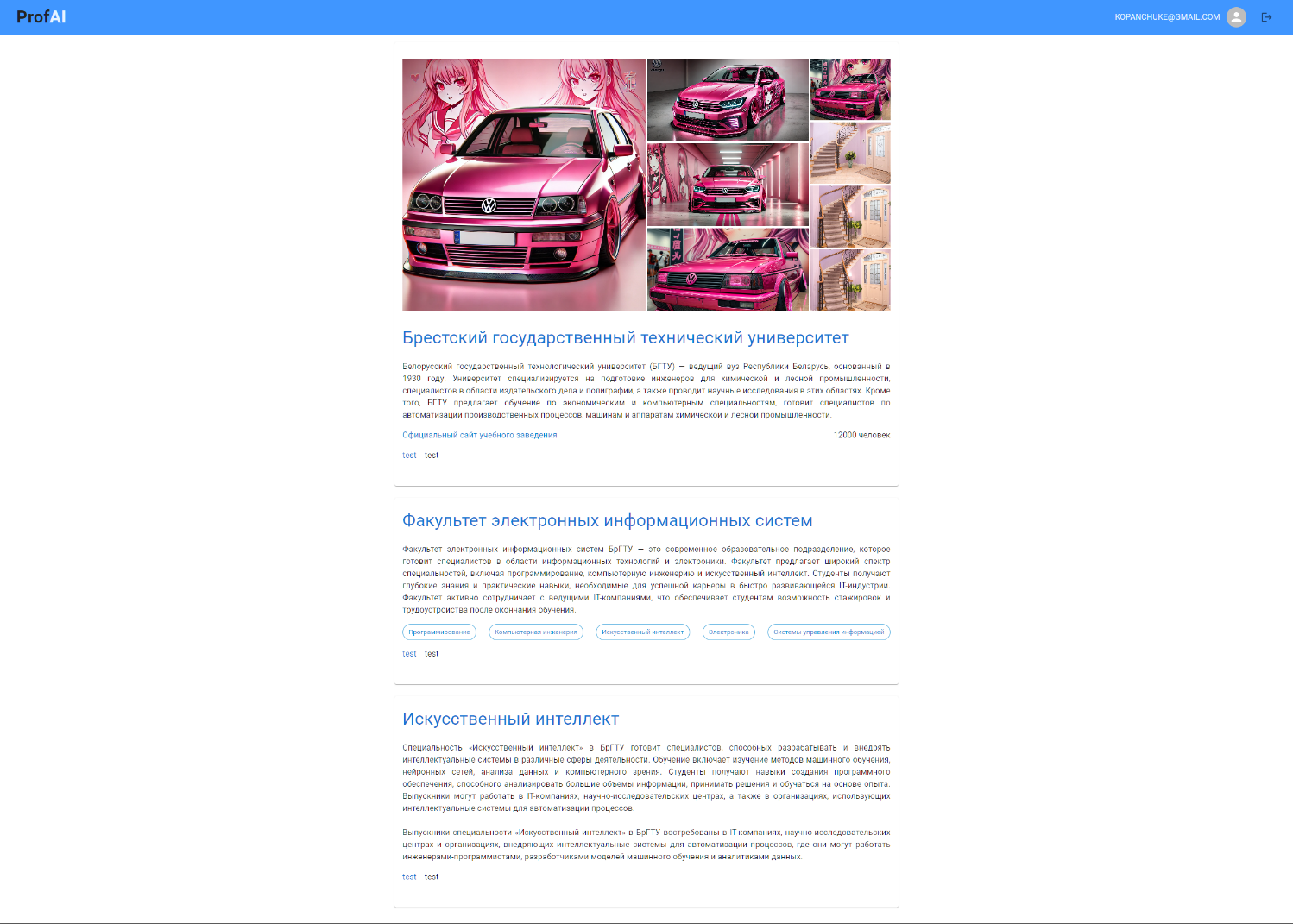


Рисунок 2.2 – PreviewSpecialtyPage

**(protected)\profile\student**

Форма позволяет пользователю (студенту) редактировать и сохранять свою личную информацию, такую как фамилия, имя, отчество и год рождения. При загрузке данные автоматически подгружаются из профиля, а в случае отсутствия данных показываются placeholders. После внесения изменений пользователь может сохранить обновлённые данные. (Рисунок 2.3)

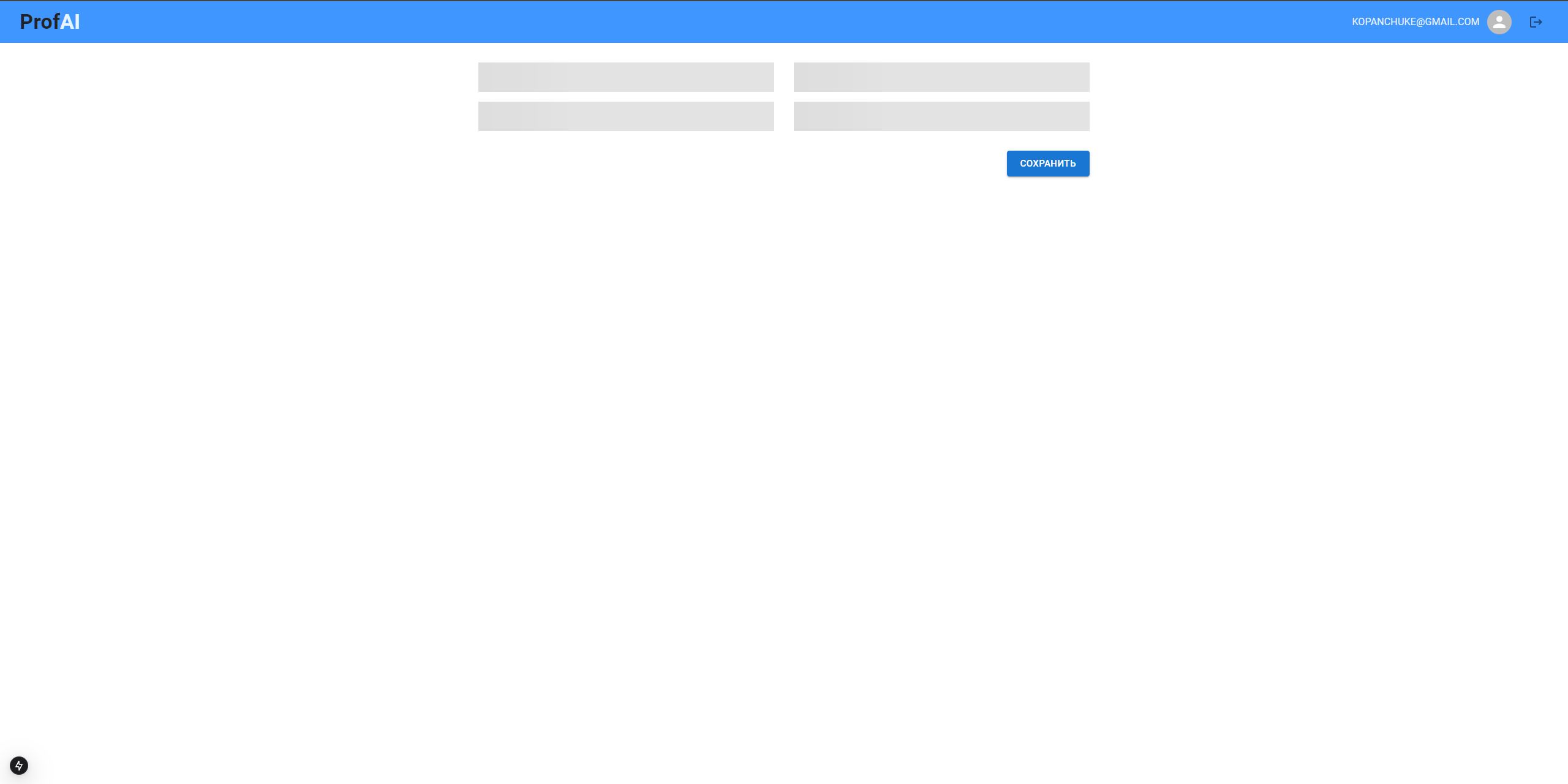
****

Рисунок 2.3 – ProfileStudent

**(protected)\profile\university**

Форма редактирования информации об учебном заведении предоставляет удобный интерфейс для управления основными данными, такими как название, описание, сайт, изображения и дополнительная информация, а также подразделениями (факультетами и специальностями). Пользователь может загружать изображения, добавлять и редактировать факультеты и специальности, обновлять описание и контактные данные учебного заведения. Все изменения сохраняются через интеграцию с API сервиса пользователей. (Рисунок 2.4)

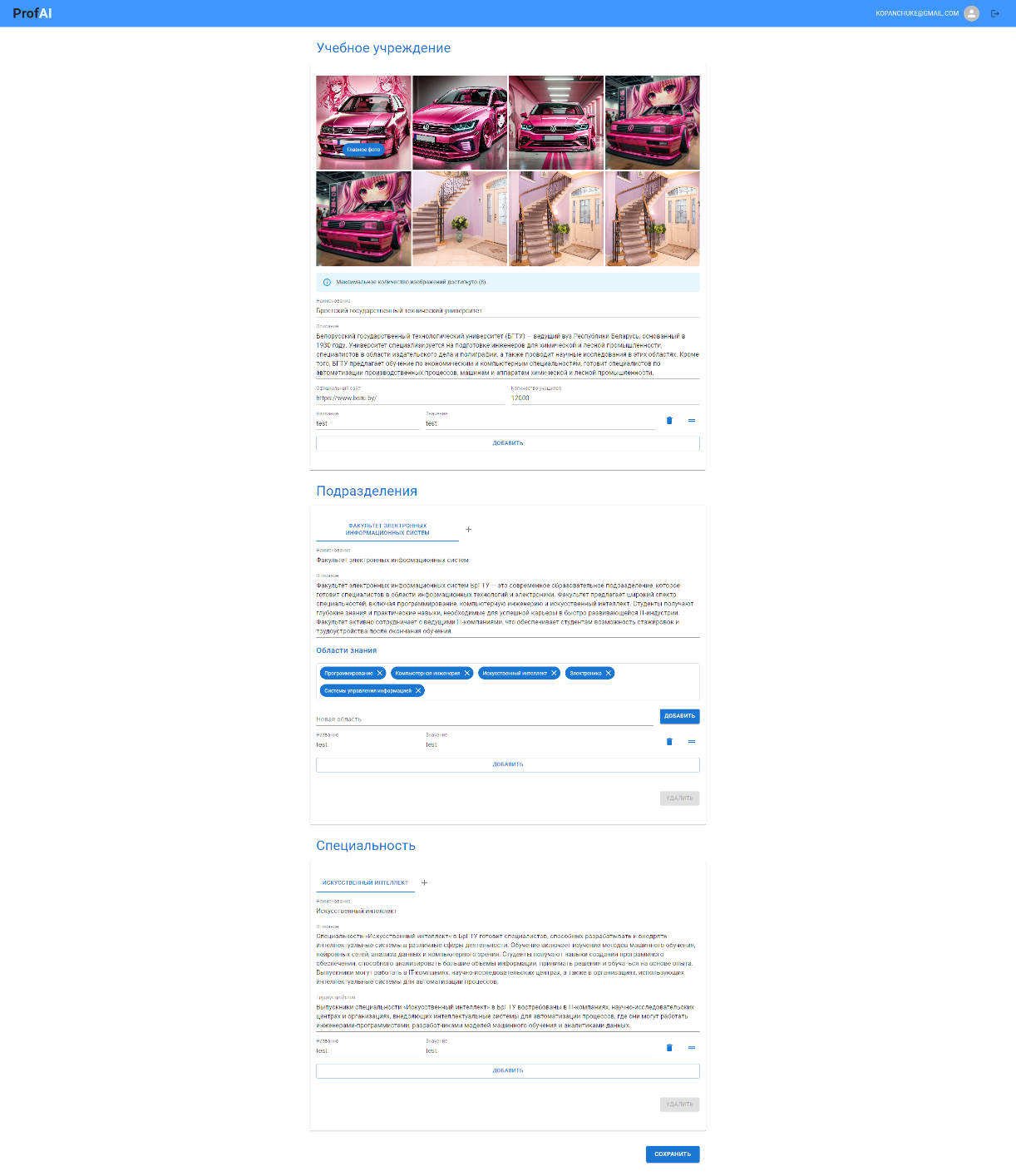


Рисунок 2.4 – ProfileUniversity

**(protected)\test**

Пользователь может выбрать специальность, расставить преимущества профессии в порядке приоритетности, распределить доступные очки навыков в соответствии со своими способностями и ответить на вопросы о себе в свободной форме. Это помогает лучше понять свои сильные стороны, предпочтения и сформировать представление о будущей профессии. (Рисунок 2.5)

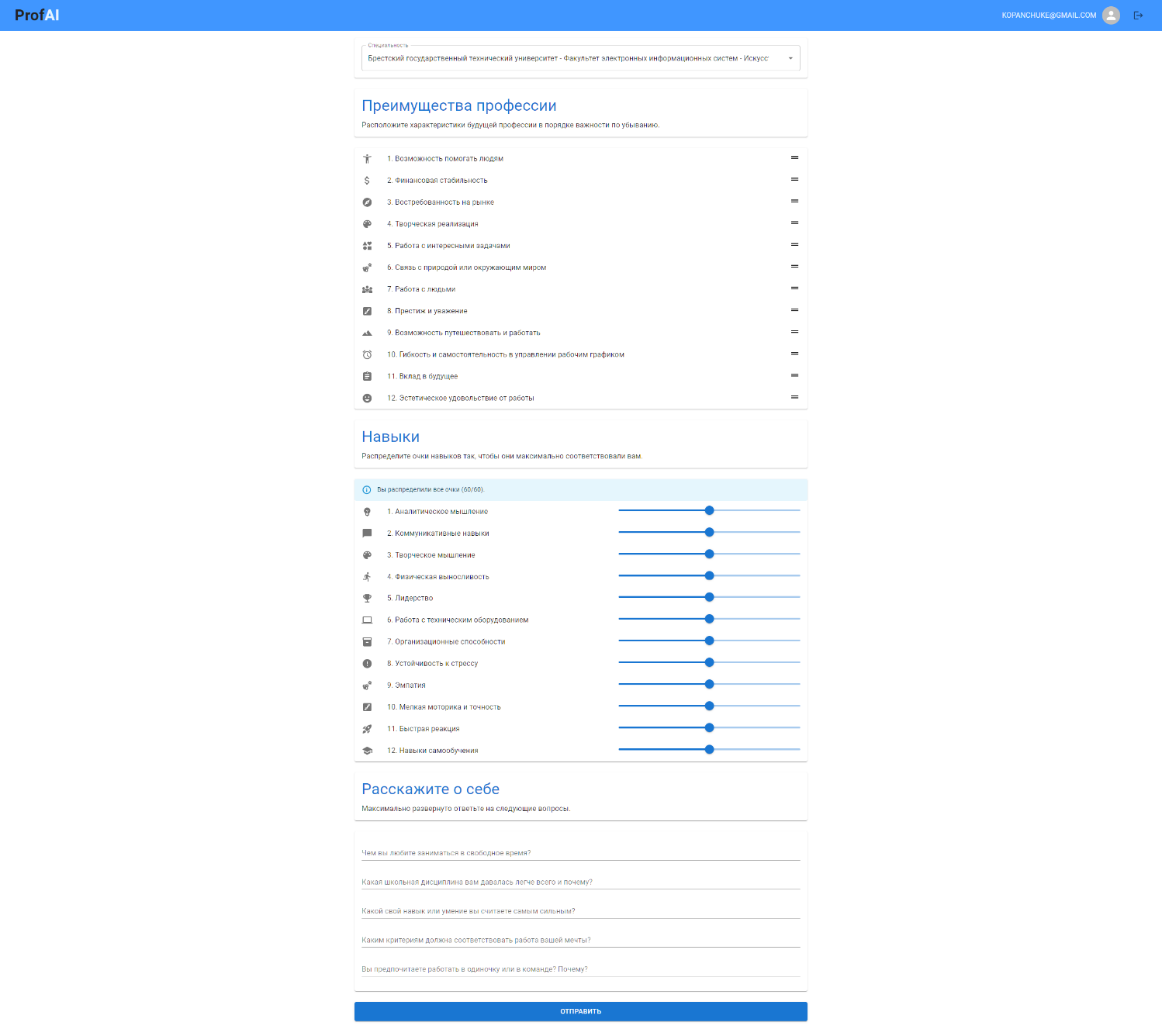


Рисунок 2.5 – Test

**API роуты:**

**api\ai**

**POST:** Этот запрос используется для отправки текста в модель ИИ и получения ответа. На вход принимает JSON с полями: text (обязательный текст для обработки), model (необязательный выбор модели, либо "r1-1776", либо "sonar") и maxTokens (необязательный лимит токенов до 500). Если данные некорректны, возвращает ошибку с кодом 400. При успешной обработке отправляет запрос к API Perplexity AI с указанными параметрами, включая авторизацию через токен, и возвращает ответ с кодом 200 и результатом работы модели. В случае ошибок на сервере или сбоя во внешнем API возвращает код 500 или 400 с соответствующими сообщениями.

**api\ai\evaluate**

**POST:** Этот запрос используется для оценки качеств пользователя на основе предоставленного описания. На вход принимает JSON с единственным обязательным полем text, содержащим текст описания пользователя. На основе этого текста формируется подробный запрос с инструкциями для модели ИИ, которая оценивает пользователя по 12 критериям (например, аналитическое мышление, лидерство, стрессоустойчивость и др.) и возвращает оценки от 0 до 10 для каждого. Ответ содержит результат работы модели в формате JSON, либо сообщение об ошибке, если запрос некорректен или произошёл сбой.

**api\auth[...nextauth]**

**GET:** Этот запрос используется для получения информации об аутентификации пользователя. Он возвращает текущую сессию, включая данные пользователя, токены доступа или состояние, если пользователь не аутентифицирован.

**POST:** Этот запрос обрабатывает вход или выход пользователя из системы. Принимает данные для авторизации (например, логин и пароль или токен OAuth). На выходе возвращает данные сессии при успешной аутентификации или ошибку при неудаче.

**api\extraFacultyInfo**

**POST:** Этот запрос используется для создания записи с дополнительной информацией о факультете (extraFacultyInfo). На вход принимает объект с данными: id записи и facultyId факультета, к которому необходимо привязать запись. На выходе возвращает созданный объект extraFacultyInfo при успешной операции или сообщение об ошибке, если запрос завершился неудачей.

**api\extraFacultyInfo[extraFacultyInfoId]**

**GET:** Этот запрос используется для получения данных об объекте extraFacultyInfo по его уникальному идентификатору extraFacultyInfoId. На вход принимает параметр extraFacultyInfoId. На выходе возвращает данные объекта, если запись существует, или сообщение об ошибке, если объект с указанным идентификатором не найден.

**POST:** Этот запрос используется для обновления данных объекта extraFacultyInfo по его уникальному идентификатору extraFacultyInfoId. На вход принимает параметр extraFacultyInfoId и объект с обновляемыми данными. На выходе возвращает обновлённый объект, если операция успешна, или сообщение об ошибке, если объект не найден или произошла ошибка обновления.

**DELETE:** Этот запрос используется для удаления объекта extraFacultyInfo по его уникальному идентификатору extraFacultyInfoId. На вход принимает параметр extraFacultyInfoId. На выходе возвращает подтверждение успешного удаления объекта и данные удалённого объекта или сообщение об ошибке, если объект с указанным идентификатором не найден.

**api\extraSpecialtyInfo**

**POST:** Этот запрос используется для создания объекта extraSpecialtyInfo в базе данных. На вход принимает объект с данными, включая id для нового объекта и specialtyId для связи с существующим объектом specialty. На выходе возвращает созданный объект extraSpecialtyInfo, если операция выполнена успешно, или сообщение об ошибке, если вводимые данные некорректны или произошла ошибка при создании.

**api\extraSpecialtyInfo[extraSpecialtyInfoId]**

**GET:** Этот запрос используется для получения данных объекта extraSpecialtyInfo по его уникальному идентификатору (extraSpecialtyInfoId). На вход принимает идентификатор. На выходе возвращает объект extraSpecialtyInfo, если он существует, или сообщение об ошибке, если объект с указанным идентификатором не найден.

**POST:** Этот запрос используется для обновления данных существующего объекта extraSpecialtyInfo. На вход принимает идентификатор объекта (extraSpecialtyInfoId) и объект с новыми данными для обновления. На выходе возвращает обновленный объект extraSpecialtyInfo или сообщение об ошибке, если идентификатор не найден или данные некорректны.

**DELETE:** Этот запрос используется для удаления объекта extraSpecialtyInfo по его уникальному идентификатору (extraSpecialtyInfoId). На вход принимает идентификатор. На выходе возвращает сообщение об успешном удалении объекта или сообщение об ошибке, если объект с указанным идентификатором не найден.

**api\extraUniversityInfo**

**POST:** Этот запрос используется для создания нового объекта extraUniversityInfo. На вход: принимает JSON с данными, включая уникальный id объекта и universityId, который указывает на связь с существующим объектом university. На выход: возвращает созданный объект extraUniversityInfo или сообщение об ошибке, если данные некорректны или произошла ошибка при создании.

**api\extraUniversityInfo[extraUniversityInfoId]**

**GET:** Этот запрос используется для получения данных об объекте extraUniversityInfo по его уникальному идентификатору extraUniversityInfoId. На вход: принимает параметр extraUniversityInfoId. На выход: возвращает объект extraUniversityInfo, если он существует, или сообщение об ошибке, если объект с указанным идентификатором не найден.

**POST:** Этот запрос используется для обновления данных объекта extraUniversityInfo по его уникальному идентификатору extraUniversityInfoId. На вход: принимает параметр extraUniversityInfoId и объект с обновляемыми данными (JSON). На выход: возвращает обновленный объект extraUniversityInfo, если операция успешна, или сообщение об ошибке, если объект не найден или данные некорректны.

**DELETE:** Этот запрос используется для удаления объекта extraUniversityInfo по его уникальному идентификатору extraUniversityInfoId. На вход: принимает параметр extraUniversityInfoId. На выход: возвращает сообщение об успешном удалении объекта и данные удалённого объекта или сообщение об ошибке, если объект с указанным идентификатором не найден.

**api\faculty**

**POST:** Этот запрос используется для создания нового объекта faculty в базе данных. На вход: принимает JSON с данными, включая: id — уникальный идентификатор объекта faculty. universityId — идентификатор связанного объекта university. На выход: возвращает созданный объект faculty при успешной операции или сообщение об ошибке, если данные некорректны или произошла ошибка при создании.

**api\faculty[facultyId]**

**GET:** Этот запрос используется для получения данных об объекте faculty по его уникальному идентификатору facultyId. На вход: принимает параметр facultyId. На выход: возвращает объект faculty, если он существует, или сообщение об ошибке, если объект с указанным идентификатором не найден.

**POST:** Этот запрос используется для обновления данных объекта faculty по его уникальному идентификатору facultyId. На вход: принимает параметр facultyId и объект с обновляемыми данными (JSON). На выход: возвращает обновленный объект faculty, если операция успешна, или сообщение об ошибке, если объект не найден или данные некорректны.

**DELETE:** Этот запрос используется для удаления объекта faculty по его уникальному идентификатору facultyId. На вход: принимает параметр facultyId. На выход: возвращает сообщение об успешном удалении объекта и данные удалённого объекта или сообщение об ошибке, если объект с указанным идентификатором не найден.

**api\faculty[facultyId]\detailed**

**GET:** Этот запрос используется для получения детализированных данных об объекте faculty, включая связанную информацию. На вход: принимает параметр facultyId. На выход: возвращает объект faculty, включающий: Связанную информацию (extraInfo). Связанные объекты specialties и их дополнительную информацию (extraInfo). Если объект с указанным идентификатором не найден, возвращает сообщение об ошибке.

**api\specialty**

**POST:** Этот запрос создаёт объект specialty в базе данных, устанавливая связь с объектом faculty через facultyId. На вход принимает JSON с данными, включая id и facultyId. На выход возвращает созданный объект specialty при успешной операции или сообщение об ошибке, если данные некорректны или произошла ошибка создания.

**api\specialty[specialtyId]**

**GET:** Этот запрос получает данные объекта specialty по его идентификатору specialtyId. На вход принимает specialtyId. Возвращает объект specialty при успешной операции или сообщение об ошибке, если объект с таким идентификаторомне найден.

**POST:** Этот запрос обновляет данные объекта specialty по его идентификатору specialtyId. На вход принимает specialtyId и JSON с новыми данными. Возвращает обновлённый объект specialty или сообщение об ошибке, если идентификатор не найден или данные некорректны.

**DELETE:** Этот запрос удаляет объект specialty по его идентификатору specialtyId. На вход принимает specialtyId. Возвращает сообщение об успешном удалении объекта или ошибку, если объект с таким идентификатором не найден.

**api\specialty[specialtyId]\detailed**

**GET:** Этот запрос возвращает подробные данные объекта specialty по идентификатору specialtyId, включая связанные сущности extraInfo и test. На вход принимает specialtyId. Возвращает объект specialty с вложенными данными при успешной операции или сообщение об ошибке, если объект с указанным идентификатором не найден.

**api\student**

**POST:** Этот запрос создаёт объект student в базе данных. На вход принимает JSON-объект с параметрами id и userId, где userId используется для связи с существующим объектом user. Возвращает созданный объект student, если операция успешна, или сообщение об ошибке в случае некорректных данных или ошибки при создании записи.

**api\student[studentId]**

**GET:** Получает данные студента по studentId. На вход принимает параметр studentId, возвращает объект студента, если он существует, или сообщение об ошибке, если запись не найдена.

**POST:** Обновляет данные студента по studentId. На вход принимает параметр studentId и JSON с обновляемыми данными. Возвращает обновлённый объект студента или сообщение об ошибке, если запись не найдена или данные некорректны.

**DELETE:** Удаляет студента по studentId. На вход принимает параметр studentId. Возвращает данные удалённого объекта или сообщение об ошибке, если запись не найдена.

**api\student[studentId]\detailed**

**GET:** Получает детализированные данные студента по studentId, включая связанные тесты. На вход принимает параметр studentId, возвращает объект студента с данными тестов, если он существует, или сообщение об ошибке, если запись не найдена.

**api\university**

**POST:** Создаёт запись университета в базе данных. На вход принимает JSON с id и userId для связи с пользователем. Возвращает созданный объект университета при успешной операции или сообщение об ошибке, если данные некорректны или произошла ошибка создания.

**api\university[universityId]**

**GET:** Получает данные университета по universityId. Если universityId отсутствует или университет не найден, возвращает ошибку 400. При успешной операции возвращает данные университета.

**POST:** Обновляет данные университета по universityId и данным, переданным в запросе. Возвращает ошибку 400, если universityId или данные неверны, и успешный ответ при обновлении.

**DELETE:** Удаляет университет по universityId. Если universityId неверный или университет не найден, возвращает ошибку 400. Успешный ответ при удалении данных.

**api\university[universityId]\detailed**

**GET:** Этот API-запрос принимает объект NextRequest и параметр universityId из маршрута, идентифицирующий университет. Если universityId отсутствует, возвращает ошибку (код 400) с сообщением "invalid universityId". В случае валидного universityId, запрашивает данные университета из базы через Prisma, включая факультеты, специализации и дополнительную информацию. Если университет не найден, возвращает ошибку (код 400) с сообщением "no university with such id". При успешном выполнении возвращает данные университета с кодом 200. В случае ошибки сервера возвращает код 500 и сообщение об ошибке.

**api\university\detailed**

**GET:** Этот API-запрос не принимает входных параметров и используется для получения списка всех университетов. Через Prisma извлекает данные университетов, включая факультеты, их специализации и дополнительную информацию. Если данные не найдены, возвращает ошибку с кодом 400 и сообщением "while getting all universities". В случае успешного выполнения возвращает код 200 с сообщением "universities were received" и данными университетов. При возникновении ошибки сервера возвращает код 500 и сообщение с описанием ошибки.

**api\user**

**POST:** Этот API-запрос принимает JSON-данные через тело запроса req, содержащие информацию о пользователе. Если данные отсутствуют или некорректны, возвращает ошибку с кодом 400 и сообщением "invalid user data". При наличии данных создает нового пользователя в базе через Prisma, используя переданные поля, включая id, если он указан. Если создание пользователя не удалось, возвращает ошибку с кодом 400 и сообщением "while creating user". В случае успеха возвращает код 200 с сообщением "user was created" и созданными данными пользователя. При ошибке сервера возвращает код 500 с описанием ошибки.

**api\user[userId]**

**GET:** Этот API-запрос принимает параметр userId из маршрута и возвращает данные пользователя с указанным id. Если userId отсутствует или пользователь не найден, возвращает ошибку с кодом 400 и соответствующим сообщением. В случае успешного выполнения возвращает код 200 с данными пользователя. При ошибке сервера возвращает код 500 и сообщение об ошибке.

**POST:** Этот API-запрос принимает параметр userId из маршрута и JSON-данные из тела запроса. Обновляет данные пользователя с указанным id. Если userId или данные некорректны, возвращает код 400. Если пользователь не найден, также возвращает код 400. При успешном обновлении возвращает код 200 с обновленными данными. Ошибки сервера обрабатываются с кодом 500 и сообщением.

**DELETE:** Этот API-запрос принимает параметр userId из маршрута и удаляет пользователя с указанным id. Если userId отсутствует или пользователь не найден, возвращает ошибку с кодом 400. При успешном удалении возвращает код 200 с подтверждением. Ошибки сервера приводят к возврату кода 500 с описанием проблемы.

**api\user[userId]\detailed**

**GET:** Этот API-запрос принимает параметр userId из маршрута и возвращает подробные данные о пользователе с указанным id. Если userId отсутствует или пользователь не найден, возвращает ошибку с кодом 400 и соответствующим сообщением. Данные включают информацию о пользователе, связанном студенте (включая тесты), университете, факультетах, специализациях и дополнительной информации. При успешном выполнении возвращает код 200 с подробными данными пользователя. В случае ошибки сервера возвращает код 500 с описанием проблемы.

**2.2 Структура базы данных**

В данной главе рассматривается структура базы данных, используемой для управления пользователями, студентами, университетами и их связанными данными в интеллектуальной системе. Основой реализации является Prisma ORM, обеспечивающая удобный интерфейс для взаимодействия с реляционной базой данных PostgreSQL. Модели, представленные в схеме, включают пользователей, их учетные записи и сессии, а также академические сущности, такие как университеты, факультеты, специальности, тесты и дополнительную информацию. Каждая модель описывает ключевые свойства и связи между объектами, что позволяет реализовать сложную логику, поддерживать целостность данных и обеспечить гибкость при разработке приложений. Эта структура обеспечивает надежную основу для работы системы, её масштабируемость и удобство использования.

**Описание полей таблиц:**

1. **User**

**id**: Уникальный идентификатор пользователя.

**name**: Имя пользователя (необязательное).

**email**: Уникальный адрес электронной почты.

**password**: Пароль пользователя.

**emailVerified**: Дата подтверждения электронной почты.

**image**: Ссылка на аватар пользователя.

**role**: Роль пользователя в системе (например, администратор, студент).

**accounts, sessions, Authenticator**: Связанные учетные записи, сессии и устройства аутентификации.

**student, university**: Связь с моделью студента или университета.

**createdAt, updatedAt**: Дата создания и последнего обновления.

1. **Student**

**id**: Уникальный идентификатор студента.

**firstName, secondName, thirdName**: Имя, фамилия и отчество студента.

**yearOfBirth**: Год рождения.

**tests**: Связанные тесты.

**userId**: Идентификатор связанного пользователя.

1. **University**

**id**: Уникальный идентификатор университета.

**latitude, longitude**: Географическое расположение.

**siteLink**: Ссылка на сайт университета.

**images**: Массив изображений.

**numberOfStudents**: Количество студентов.

**approved**: Дата одобрения университета в системе.

**name, description**: Название и описание университета.

**extraInfo, faculties**: Дополнительная информация и факультеты.

**userId**: Идентификатор связанного пользователя.

1. **Faculty**

**id**: Уникальный идентификатор факультета.

**name**: Название факультета.

**areasOfKnowledge**: Области знаний факультета.

**description**: Описание факультета.

**extraInfo, specialties**: Дополнительная информация и связанные специальности.

**universityId**: Связь с университетом.

1. **Specialty**

**id**: Уникальный идентификатор специальности.

**name**: Название специальности.

**description**: Описание специальности.

**employment**: Варианты трудоустройства.

**extraInfo**: Дополнительная информация.

**facultyId**: Связь с факультетом.

**test**: Связанные тесты.

1. **Test**

**id**: Уникальный идентификатор теста.

**benefits, skills, llmSkills**: Результаты теста: преимущества, навыки, специфические навыки.

**questions**: Вопросы теста.

**passDate**: Дата прохождения теста.

**studentId, specialtyId**: Связь со студентом и специальностью.

1. **ExtraUniversityInfo, ExtraFacultyInfo, ExtraSpecialtyInfo**

**id**: Уникальный идентификатор.

**name, value**: Название и значение дополнительной информации.

**universityId, facultyId, specialtyId**: Связь с университетом, факультетом или специальностью.

1. **Account**

**type, provider, providerAccountId**: Данные провайдера учетной записи.

**refresh\_token, access\_token**: Токены авторизации.

**expires\_at, token\_type**: Время истечения и тип токена.

**userId**: Связь с пользователем.

1. **Session**

**sessionToken**: Уникальный идентификатор сессии.

**userId**: Связь с пользователем.

**expires**: Срок действия сессии.

1. **VerificationToken**

**identifier, token**: Уникальные данные верификации.

**expires**: Время истечения токена.

1. **Authenticator**

**credentialID, credentialPublicKey**: Уникальные данные устройства аутентификации.

**userId**: Связь с пользователем.

**counter**: Счетчик использований.

**transports**: Способы передачи (например, USB, NFC).

**2.3 Что планируется доработать**

**(opened)\preview\test\[testId]**

На странице результатов тестирования будут располагаться графики основных метрик пользователя рассчитанных с помощью классического тестирования и LLM. А также ряд других тестов, которые помогут подобрать более релевантные рекомендации по специальностям. Страница несет исключительно визуальный характер, пользователю будет интересно увидеть результаты тестирования собранные в одно представление.

**(opened)\search**

На странице поиска пользователю будет представлена возможность производить поиск специальностей. Пользователю будет доступна классическая поисковая строка, при нажатии на кнопку поиска будет искаться соответствие запроса пользователя с информацией из текстовых полей таблицы университетов, факультетов, специальностей и их дополнительной информацией. Также пользователю будет доступна форма расширенного поиска, где пользователь уже сможет более точно описать то, что он хочет получить в поисковой выдаче. После завершения поиска, на странице сайта будут представлены карточки специальностей, нажатие на которую приводят к переходу на страницу превью данной специальности.

**Компонент SuggestedSpecialties**

Данный компонент будет находиться на страницах профиля студента, результата тестирования и поиска по специальностям и показывать карточки специальностей подобранных в соответствии с последним результатом тестирования пройденного пользователем, если пользователь не прошел тестирование, то в этот компоненте пользователю будет предложено сделать это.

# ЗАКЛЮЧЕНИЕ

В ходе работы была разработана часть система профориентационного тестирования, включающая веб-приложение с клиентской и серверной частями, а также базу данных для хранения и обработки информации. Реализованы ключевые функциональные компоненты:

Пользовательский интерфейс – обеспечены страницы авторизации, профилей (студента и университета), просмотра информации об учебных заведениях и прохождения тестирования.

API-интеграция – реализованы маршруты для взаимодействия с данными пользователей, университетов, факультетов, специальностей и результатов тестов, включая обработку запросов с помощью LLM.

База данных – спроектирована структура на основе PostgreSQL с использованием Prisma ORM, что обеспечивает целостность данных и гибкость при масштабировании системы.

Система позволяет:

Собирать и анализировать ответы пользователей с применением классических методов и LLM.

Предоставлять пользователям структурированную информацию о специальностях и учебных заведениях.

Перспективы доработки:

Добавление страницы с визуализацией результатов тестирования.

Реализация расширенного поиска по специальностям.

Внедрение компонента персонализированных рекомендаций на основе тестовых данных.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Официальная документация по Next.js [Электронный ресурс] – Режим доступа: <https://nextjs.org/>.

2. Официальная документация по Perplexity [Электронный ресурс] – Режим доступа: <https://www.perplexity.ai/>.

3. Официальная документация по Ant Design [Электронный ресурс] – Режим доступа: <https://ant.design/>.