

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

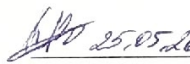
Профиль подготовки: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

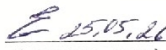
на тему

**СИСТЕМА ПРОФИОРИЕНТАЦИИ НА ОСНОВЕ
АЛГОРИТМА МАШИННОГО ОБУЧЕНИЯ**

Студент

 25.05.26 Кондратьева В.И.
(подпись, дата)

Руководитель

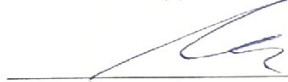
 25.05.26 Гурин Е.И.
(подпись, дата)

Нормоконтролёр

 А.С. Бычков
(подпись, дата) 25.05.2026

Работа допущена к защите (протокол заседания кафедры от 22.05.2026 г. №14)

Заведующий кафедрой

 М.А. Митрохин
(подпись)

Работа защищена с отметкой Отлично (протокол заседания ГЭК от 26.06.26 № 7)

Секретарь ГЭК

 Е.И. Гурин
(подпись)

Пенза 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

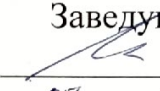
вычислительной техники

Кафедра

«Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

 М.А. Митрохин

«07» 04 2026г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Кондратьева Виктория Игоревна гр.22ВВП1 факультета вычислительной техники.

2. Тема работы: «Система профориентации на основе алгоритма машинного обучения»

Тема утверждена приказом ПГУ №347/о от "7" апреля 2026 г.

3. Руководитель бакалаврской работы д.т.н., профессор, каф. ВТ ПГУ Гурин Е.И.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать приложение «Система профориентации на основе алгоритма машинного обучения», позволяющее автоматизировать процесс профориентационного тестирования. Приложение должно обладать графическим и интуитивно понятным интерфейсом для взаимодействия с пользователем. Система должна предоставлять возможность прохождения как детального тестирования на основе экспертной системы, так и сокращенного с использованием машинного обучения. Разрабатываемое приложение также должно выполнять задачи по сбору и хранению результатов тестирования, формированию обучающей выборки для ML-модели и предоставлению администратору средств для управления базой знаний и переобучения модели.

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметных областей: изучить существующие системы профориентационного тестирования, выделить их достоинства и недостатки. Выбрать языки программирования и фреймворки для реализации серверной и клиентской частей. Определить архитектуру приложения и вспомогательные средства разработки. Изучить алгоритмы машинного обучения для прогнозирования профессиональных склонностей. Разработать экспертную систему для детального анализа личности и сбора данных для обучения модели. Реализовать веб-интерфейс для регистрации, авторизации, прохождения тестов и отображения

результатов. Разработать панель администратора для переобучения модели машинного обучения и управления сферами, вопросами.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	32	07.04.2026 – 10.04.2026	
2	Анализ требований	24	11.04.2026 – 13.04.2026	
3	Создание макета графического интерфейса	32	14.04.2026 – 17.04.2026	
4	Выбор технологий, применяемых при проектировании и разработке	32	18.04.2026 – 21.04.2026	
5	Разработка экспертной системы	64	22.04.2026 – 29.05.2026	
6	Разработка модели машинного обучения	64	30.04.2026 – 07.05.2026	
7	Разработка веб-приложения	64	08.05.2026 – 15.05.2026	
8	Оформление пояснительной записки	40	16.05.2026 – 20.05.2026	

Дата выдачи задания "7" апреля 2026 г.

Руководитель бакалаврской работы  07.04.2026 Гурин Е.И.
(подпись, дата)

Задание к исполнению принял студент  07.04.2026 Кондратьева В.И.
(подпись, дата)

Работу к защите допустить
Декан факультета  Фионова Л.Р.
(подпись, дата)
27.05.2026

Реферат

Бакалаврская работа содержит 166 листов, 43 рисунка 14 таблиц, 15 источников, 3 приложения.

СИСТЕМА ПРОФОРИЕНТАЦИИ НА ОСНОВЕ АЛГОРИТМА МАШИННОГО ОБУЧЕНИЯ

Цель работы – разработать систему профориентации на основе алгоритмов машинного обучения.

Объектом разработки является веб-приложение для профориентационного тестирования. Приложение обеспечивает два режима тестирования, отображение результатов в виде диаграммы и административную панель.

Разработка производилась на Python и JavaScript с использованием СУБД PostgreSQL. В качестве сред разработок использовались Spyder (Anaconda) и Visual Studio Code.

В результате разработано приложение с графическим интерфейсом, позволяющее производить детальную и быструю профориентационную диагностику.

Созданное приложение может быть внедрено в образовательные организации для работы со школьниками и студентами.

					ПГУ 1.090301.15.001 ПЗ				
Изм.	Лист	№ докум.	Подпись	Дата	Система профориентации на основе алгоритма машинного обучения Пояснительная записка	Лит.	Лист	Листов	
Разраб.		Кондратьева В.И.		25.05.24					
Провер.		Гурин Е.И.		25.05.24			2	165	
Н. Контр		Бычков А.С.		25.05.24		ФВТ, каф ВТ гр. 22ВВП1			
Утверд.		Митрохин М.А.		25.05.24					

Содержание

Перечень сокращений и обозначений	5
Введение	6
1 Анализ предметной области	7
1.1 Актуальность разработки	7
1.2 Анализ существующих аналогов	8
1.3 Обоснование актуальности разработки программы	13
1.4 Постановка задачи	16
2 Выбор и настройка инструментов разработки	19
2.1 Обоснование выбора типа приложения	19
2.2 Выбор языка программирования для серверной части	20
2.3 Выбор фреймворка для реализации REST API	21
2.4 Выбор языка программирования и технологий для клиентской части ...	22
2.5 Выбор СУБД	23
2.6 Выбор сред разработки	24
2.7 Настройка и развертывание среды разработки	25
3 Проектирование системы	32
3.1 Паттерн MVC	32
3.2 Архитектура приложения	33
3.3 Проектирование базы данных	35
4 Разработка системы	47
4.1 Разработка экспертной системы	47
4.2 Разработка ML-модели	53
4.2.1 Алгоритм машинного обучения Random Forest	53
4.2.2 Данные для обучения	56
4.2.3 Обучение модели	57
4.2.4 Использование модели в приложении	62
4.3 Разработка REST API	65
4.3.1 Роутер авторизации (auth_router.py)	65

4.3.2 Роутер вопросов (questions_router.py)	65
4.3.3 Роутер тестирования (test_router.py)	66
4.3.4 Роутер администрирования (admin_router.py)	67
4.4 Разработка клиентской части	68
4.4.1 Компонент LoginPage.jsx	68
4.4.2 Компонент RegisterPage.jsx	69
4.4.3 Компонент MenuPage.jsx	69
4.4.4 Компонент FullTestPage.jsx	70
4.4.5 Компонент ShortTestPage.jsx	71
4.4.6 Компонент ResultPage.jsx	72
4.4.7 Компонент HistoryPage.jsx	73
4.4.8 Компонент AdminPage.jsx	73
5 Руководство пользователя	75
6 Охрана труда	82
Заключение	84
Список использованных источников	86
Приложение А – UML диаграммы	88
Приложение Б – Листинг программы	99
Приложение В – Презентация	158

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

БД – база данных

ПЭВМ – персональная электронно-вычислительная машина

СУБД – система управления базами данных

ЭС – экспертная система

API (Application Programming Interface) – программный интерфейс приложения, набор правил для взаимодействия между программами

CSS (Cascading Style Sheets) – каскадные таблицы стилей, язык оформления внешнего вида веб-страниц

HTML (HyperText Markup Language) – язык гипертекстовой разметки, стандартный язык для создания веб-страниц

JSON (JavaScript Object Notation) – текстовый формат обмена данными, основанный на синтаксисе JavaScript

ML (Machine Learning) – машинное обучение, подполе искусственного интеллекта

MVC (Model-View-Controller) – архитектурный паттерн, разделяющий приложение на модель, представление и контроллер

REST (Representational State Transfer) – архитектурный стиль взаимодействия компонентов распределённого приложения

SQL (Structured Query Language) – язык структурированных запросов для управления базами данных

UML (Unified Modeling Language) – унифицированный язык моделирования, используемый для визуализации, проектирования и документирования программных систем

Введение

Выбор профессии является одной из самых важных задач, стоящих перед человеком. От этого выбора зависит не только материальный доход, но и возможность человека заниматься любимым делом, которое может в полной мере раскрыть индивидуальные способности. С каждым годом число новых профессий растет, поэтому остановиться на верном варианте становится все сложнее.

Один из способов решения поставленной проблемы – тесты на профориентацию. Традиционный опросник в тестах на профориентацию реализуется обычно в виде экспертной системы. Это простой и достаточно понятный подход, но и он не лишен минусов, таких как неспособность к адаптации и невозможность раскрыть весь потенциал личности. Внедрение машинного обучения, в свою очередь, способно справиться с данной проблемой и находить скрытые взаимосвязи. Таким образом, совмещение традиционного подхода в тестировании на основе экспертной системы (ЭС) и применения алгоритмов машинного обучения позволит усовершенствовать профориентационные тесты для получения наиболее точного результата [1].

Целью выпускной квалификационной работы является разработка веб-приложения для осуществления профориентационного тестирования.

В ходе выполнения данной работы необходимо:

- проанализировать существующие методы тестирования;
- разработать структуру теста и систему оценки результатов;
- реализовать веб-приложение с удобным графическим интерфейсом;
- произвести тестирование и оценить точность результатов.

Разработанное приложение позволит ускорить процесс профессионального тестирования, обеспечив персональные рекомендации.

1 Анализ предметной области

1.1 Актуальность разработки

Повышение качества профессиональной подготовки в высших учебных заведениях является актуальной задачей, которая обусловлена требованиями соответствия международным образовательным стандартам. Эффективность решения данной проблемы напрямую зависит от грамотной профориентационной работы со школьниками, которым еще предстоит выбрать место своего обучения. Это определяет профессиональную ориентацию в качестве одного из важнейших направлений работы образовательных организаций [2].

В настоящее время разработан ряд методик, направленных на профориентационное тестирование, в основе которых лежат алгоритмы машинного обучения. Изучив достоинства и недостатки нескольких систем, была разработана программа, которая объединяет в себе положительные стороны каждого из рассмотренных методов. Высокую точность системы обеспечивает детальный первоначальный анализ, реализованный с помощью классического подхода к профориентационному тестированию – экспертной системы, а скорость прохождения тестирования достигается за счёт внедрения машинного обучения.

Таким образом, разрабатываемая система позволяет решить сразу несколько задач:

- сократить время прохождения теста;
- предоставить грамотные рекомендации.

Система актуальна не только для школьников и студентов, которым только предстоит определиться с направлением будущей профессии, но и для образовательных организаций, которые заинтересованы в повышении качества профессиональной подготовки

1.2 Анализ существующих аналогов

Среди ряда методик, направленных на профориентационное тестирование можно выделить следующие решения:

- «Wizard» УрФУ [3];
- «Робот-профориентатор» Томский государственный университет;
- «AI-профориентатор»;
- «Синерджи Старт»;
- метод Тарасовой А.С. и Комаровой Э.П. [4].

Рассмотрим продукт «Wizard» УрФУ (рисунок 1). «Wizard» УрФУ – это программный продукт для профориентации и навигации абитуриентов. Работа программы состоит из нескольких этапов. На первом этапе пользователь предоставляет доступ к данным о своей учетной записи в социальной сети. На втором этапе с помощью алгоритмов машинного обучения на основе данных пользователя определяется сфера его интересов. Третий этап – сравнение интересов пользователя и информации об интересах студентов, которые уже обучаются по программам УрФУ. Далее система формирует персональные рекомендации на основе максимального совпадения признаков [3]. Анализ цифровых данных является хорошим методом для определения профессионального направления, однако у многих пользователей профиль может быть не заполнен или иметь неактуальные данные, из-за чего система не сможет сформировать достоверный результат.

«Робот-профориентатор» Томского государственного университета является альтернативой сервису «Wizard». Оба сервиса анализируют страницу испытуемого в социальной сети «ВКонтакте», однако ключевое различие сервисов заключается в конечной цели результата. Если разработка Уральского федерального университета сравнивает профиль испытуемого с данными учащихся конкретного вуза и на основе совпадения конкретных признаков подбирает подходящую программу обучения УрФУ, то «Робот-профориентатор» позволяет определить общую профессиональную сферу.

Таким образом, «Wizard» рекомендует потенциальному студенту конкретную образовательную программу в рамках одного вуза, а «Робот-профориентатор» – профессиональное направление в целом.

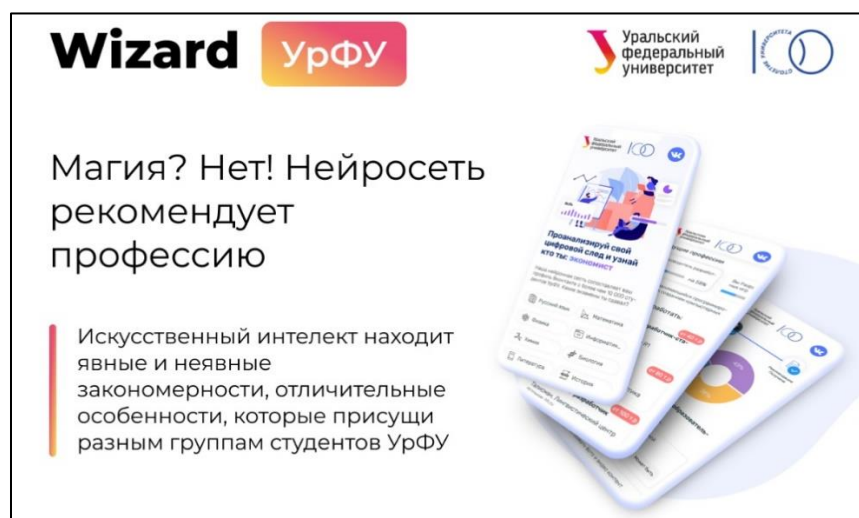


Рисунок 1 – «Wizard» УрФУ

На рисунке 2 представлена официальная группа «Робот-профориентатор»:



Рисунок 2 – «Робот-профориентатор» ТГУ

Схожим с предыдущими решениями является приложение «AI-профориентатор» (рисунок 3). Оно, как и предыдущие сервисы, формирует рекомендации на основе страницы пользователя. При нажатии одной кнопки пользователь получает следующую информацию о себе: тип личности, список персонажей с представителями этого типа, способности, ценности и личностные качества, предпочитаемые виды деятельности и профессии. Положительные и отрицательные стороны схожи с предыдущими аналогами.

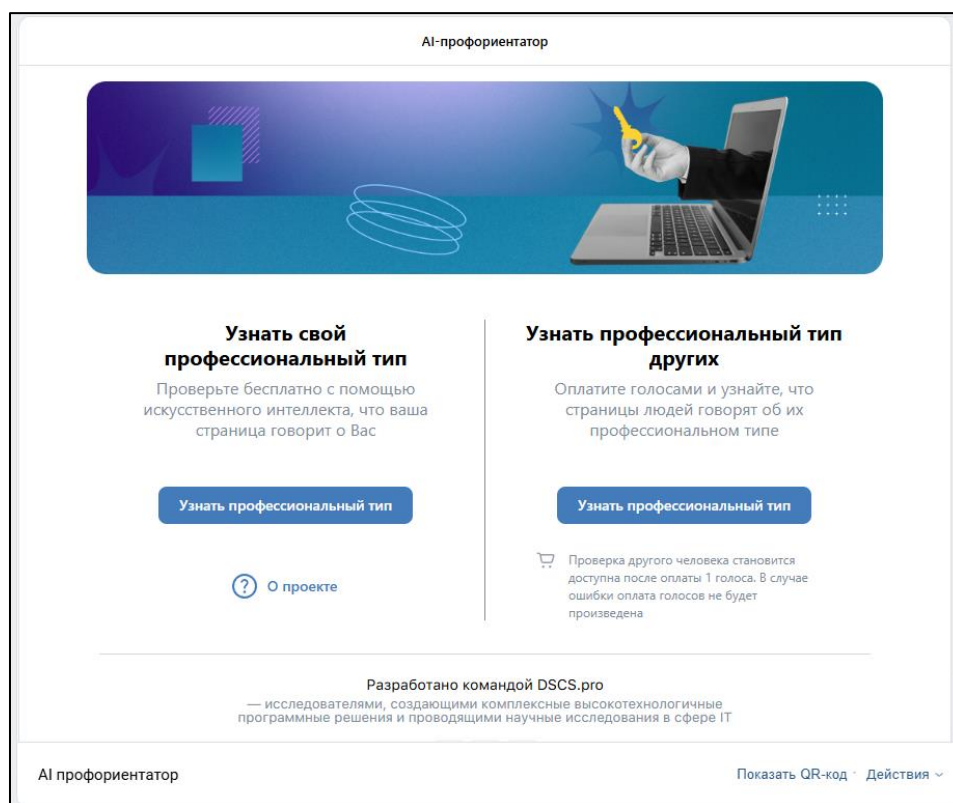


Рисунок 3 – «AI-профориентатор»

Рассмотрим платформу «Синерджи Старт». Интерфейс сайта представлен на рисунке 4. Разработка университета «Синергия» предназначена для профориентации школьников 8-11 классов. На первом этапе пользователь проходит различные онлайн-тесты: «Склонности к деятельности», «Тип интеллекта», «Опросник личностных качеств» и «Жизненные ценности». Этот этап направлен на выявление склонностей, личностных качеств и особенностей мышления. На втором этапе проводится интервью с ИИ-консультантом об

интересах и предпочтениях школьника. Затем искусственный интеллект обрабатывает данные о вакансиях и формирует подходящие профессии. Данный подход позволяет учесть не только личностные качества тестируемого, но и актуальную ситуацию на рынке труда. Однако, для получения рекомендаций необходимо пройти несколько тестов, что может занять значительное количество времени, а для получения полного перечня профессий требуется приобретение платной подписки (рисунки 5, 6).

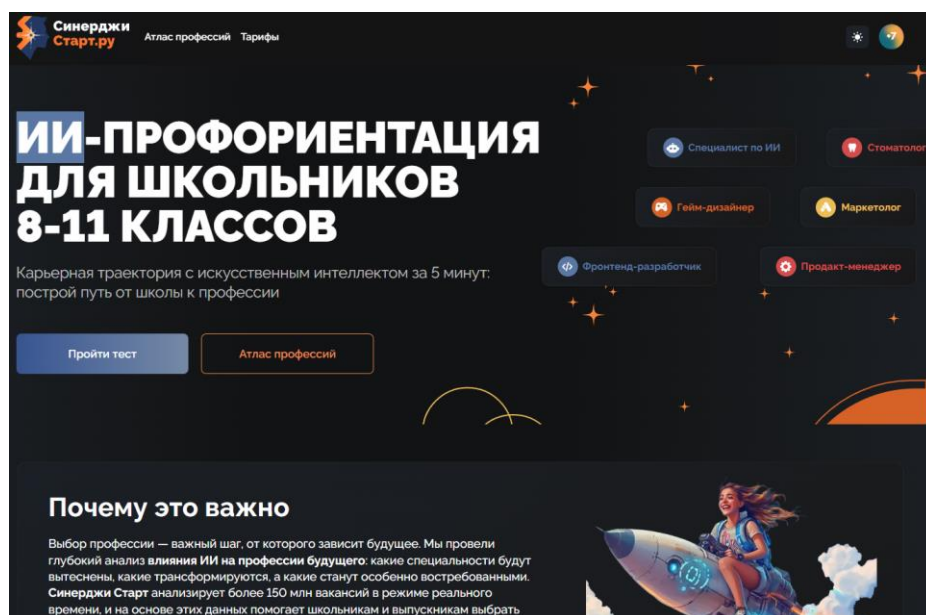


Рисунок 4 – «Синерджи Старт»

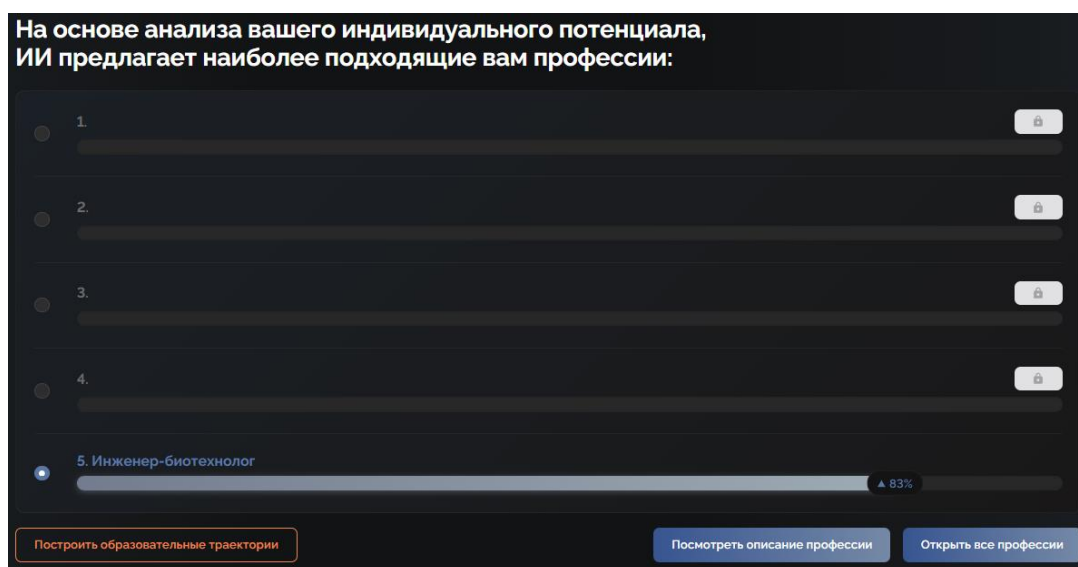


Рисунок 5 – Рекомендация только одной профессии из 5

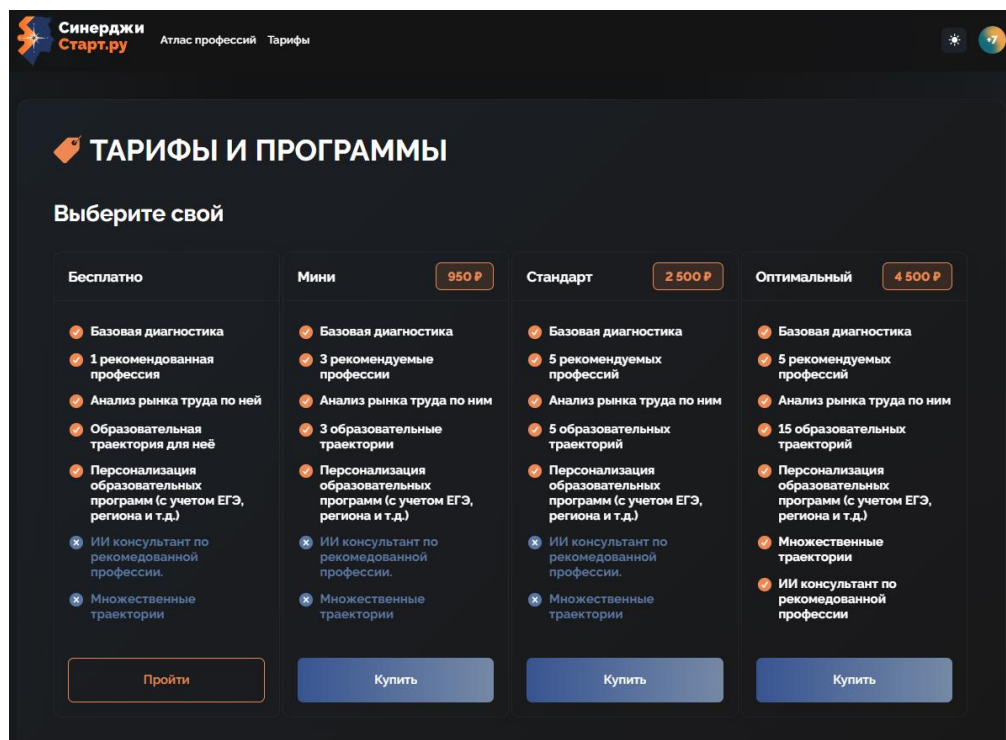


Рисунок 6 – Тарифы и программы «Синерджи Старт»

Проанализируем метод Тарасовой А.С. и Комаровой Э.П. Тарасова А.С. и Комарова Э.П. в своей работе описали созданную ими рекомендательную систему с использованием алгоритмов машинного обучения. На начальном этапе в систему загружаются профили профессий, взятые с сайта Атлас новых профессий. Составляется матрица смежности: профессии/компетенции. На втором этапе осуществляется сбор критериев оценки тестируемого. Группы критериев: результаты теста «Мой профиль» отПрофиград, школьные оценки, дополнительные достижения по школьным предметам, достижения во внеклассной деятельности и результаты теста Мегедь-Овчарова. Заключительный этап проходит в два шага: умножение профиля ученика на матрицу компетенций, а затем – результирующего вектора на матрицу профессий [4]. Данная рекомендательная система хорошо справляется с поставленной задачей и выявляет склонность даже к самым современным профессиям, однако для получения результата требуется пройти несколько

тестов. В связи с этим пользователю потребуется достаточно большое количество времени для формирования рекомендаций.

В таблице 1 представлено сравнение аналогов.

Таблица 1 – Сравнение аналогов

Название системы	Регистрация	Полное тестирование	Быстрое тестирование	Персональные рекомендации	Доступ онлайн
«Wizard» УрФУ	Через аккаунт соцсети	-	+	+	+
«Робот-профориентатор» ТГУ	Через аккаунт соцсети	-	+	+	+
«AI-профориентатор»	Через аккаунт соцсети	-	+	+	+
«Синерджи старт»	Встроенная регистрация	+	-	+ (частично платно)	+
Метод Тарасовой А.С. и Комаровой Э.П.	Встроенная регистрация	+	-	+	-
Разрабатываемая система	Встроенная регистрация	+	+	+	+

1.3 Обоснование актуальности разработки программы

Проведя анализ существующих систем в области профориентации, были выявлены некоторые ограничения. На основе таблицы 1 можно сделать следующие выводы.

Системы, использующие анализ социальных сетей («Wizard» УрФУ, «Робот-профориентатор» ТГУ и «AI-профориентатор»), позволяют быстро получить результат, однако сильно зависят от степени заполненности профиля. Если в нем недостаточно данных или он закрыт – система может выдать некорректный результат.

Подходы, реализующие более полное и детальное тестирование («Синерджи Старт» и метод Тарасовой А.С. и Комаровой Э.П.), направленные на подробный анализ личности и интересов испытуемого, предполагают значительные временные затраты.

В ходе исследования предложенных аналогов было проведено тестирование некоторых сервисов. На рисунках 7 и 8 представлены результаты приложения «AI-профориентатор», а на рисунках 5 и 9 – платформы «Синерджи Старт».

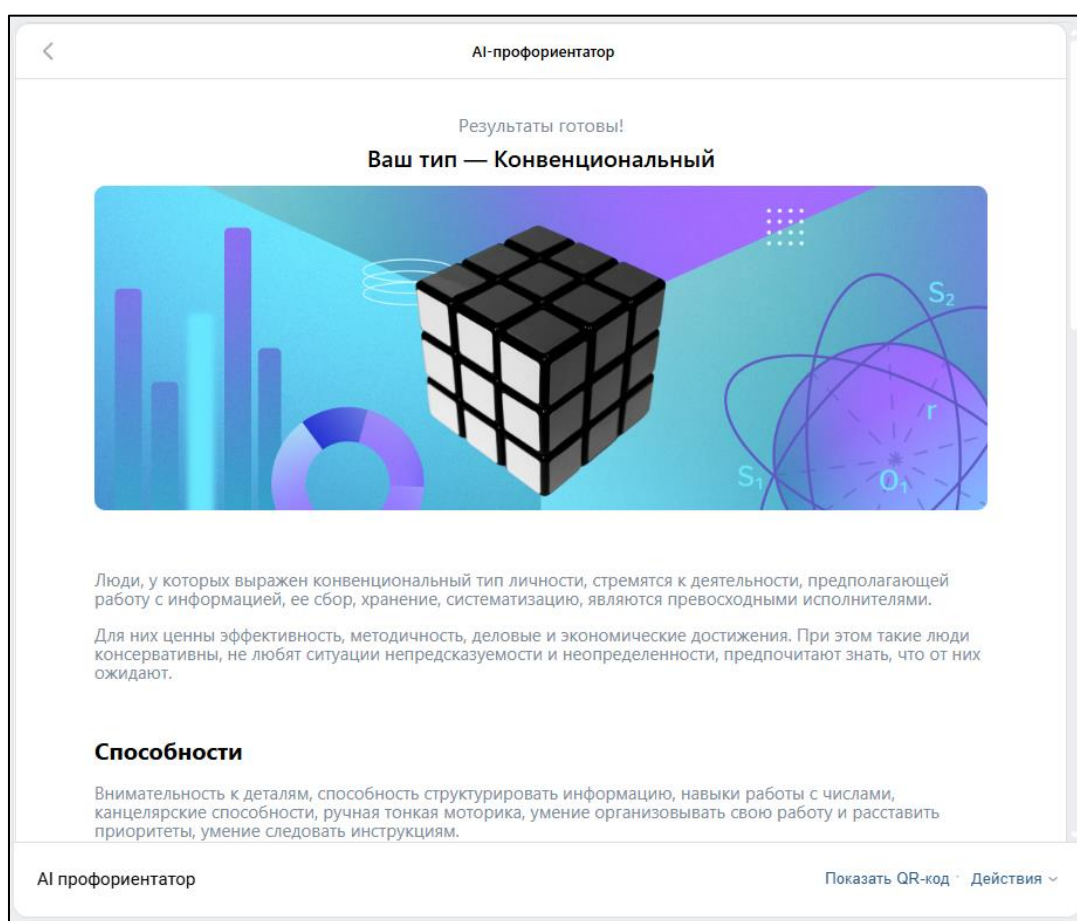


Рисунок 7 – Результаты тестирования приложения «AI-профориентатор»

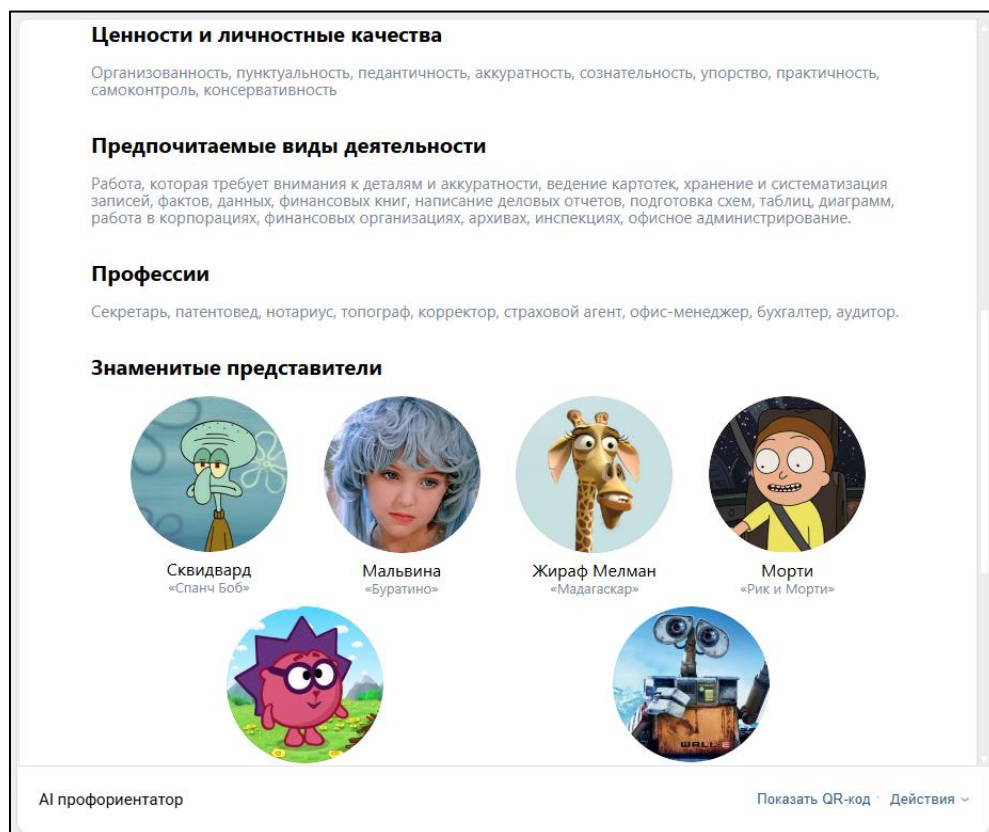


Рисунок 8 – Результаты тестирования приложения «AI-профориентатор»

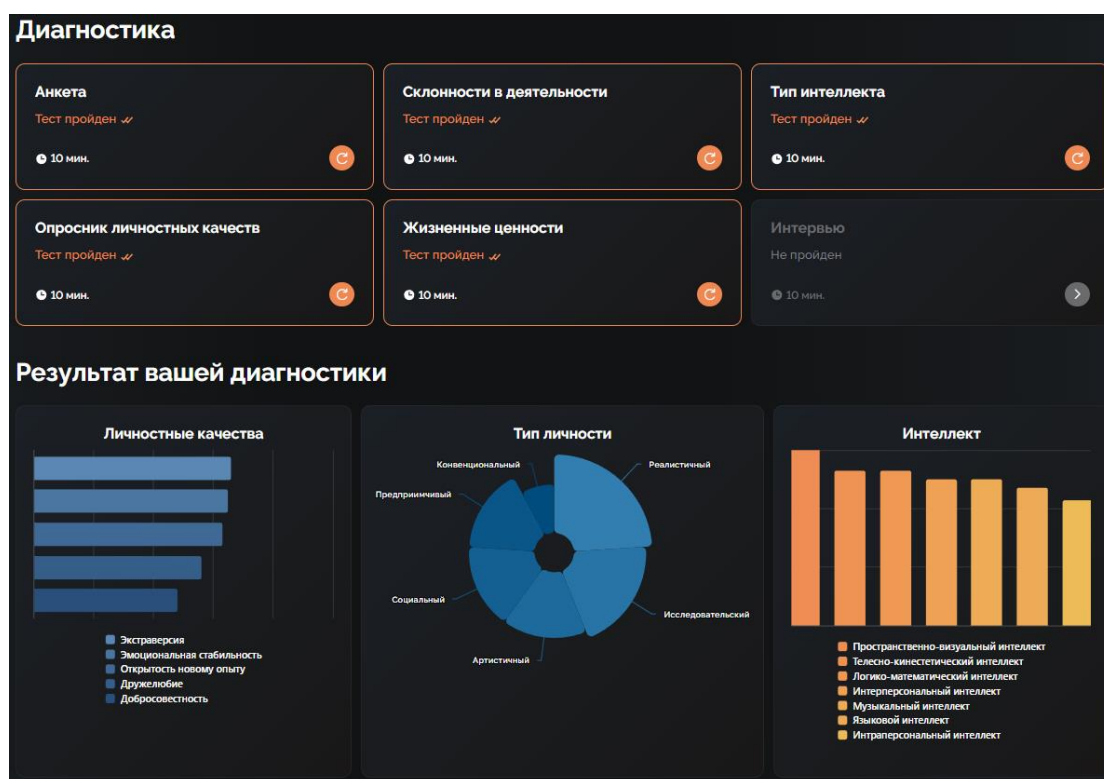


Рисунок 9 – результаты тестирования платформы «Синерджи Старт»

После сравнения полученных результатов выяснилось, что при полном 40-минутном тестировании на платформе «Синерджи Старт» тип личности был определен как «Реалистичный», а одна из рекомендуемых профессий – «Инженер-биотехнолог». Результаты приложения «AI-профориентатор» оказались совершенно противоположными. Тип личности был определен как «Конвенциональный», хотя на рисунке 9 видно, что при более глубоком тестировании конвенциональный тип занимает последнее место среди всех предложенных. В качестве предлагаемых профессий приложение предлагает: секретарь, патентовед, нотариус, топограф, корректор, страховой агент, офис-менеджер, бухгалтер, аудитор. Такое кардинальное расхождение наглядно демонстрирует, насколько неточным может быть результат, основанный на анализе одних лишь соцсетей.

Таким образом, ни одна из рассмотренных систем не способна одновременно обеспечить пользователю высокую скорость получения результата и точность диагностики.

Учитывая достоинства и недостатки изученных систем, необходимо разработать программу, которая объединяет в себе положительные стороны каждого из рассмотренных методов. Высокую точность системы обеспечивает детальный первоначальный анализ, реализованный с помощью классического подхода к профориентационному тестированию – ЭС, а скорость прохождения тестирования достигается за счёт внедрения машинного обучения.

1.4 Постановка задачи

Требуется реализовать систему для профориентационного тестирования. Система должна иметь двухэтапную структуру оценки склонностей пользователя.

Общие требования к системе:

- реализация экспертной системы для глубокого профориентационного анализа;

					ПГУ 1.090301.15.001 ПЗ	Лист
						16
Изм.	Лист	№ докум.	Подпись	Дата		

- реализация алгоритма машинного обучения для прогнозирования результатов на основе сокращенного набора общих вопросов;
- возможность прохождения как детального, так и сокращенного тестирования;
- формирование результатов тестирования с указанием вероятности в процентах;
- визуализация тестирования в виде диаграммы;
- хранение ответов пользователя в базе данных.

Требования к экспертному тестированию:

- все вопросы теста должны разделяться на явные (привязанные только к одной сфере деятельности) и неявные (связанные сразу с несколькими сферами);
- ответы на вопросы оцениваются балльной шкале;
- расчет итоговой суммы баллов по каждому из направлений;
- прохождение общих вопросов и сохранение результатов теста для последующего обучения модели.

Требования к машинному обучению:

- обучение модели производится на основе данных, полученных в результате прохождения полного тестирования;
- основа обучения – данные, полученные в результате полного тестирования;
- входные признаки – ответы на общие вопросы;
- выходные данные – вероятности предрасположенности личности каждой из сфер и номер сферы с максимальной вероятностью;
- выявление важности входных признаков.

Требования к пользовательскому интерфейсу:

- графический интерфейс;
- отображение вопроса и вариантов ответа;
- индикатор прогресса прохождения теста;

- возможность выбора типа тестирования: полное или сокращенное;
- отображение результатов в виде диаграммы.

На рисунке 10 представлена диаграмма прецедентов разрабатываемой системы.

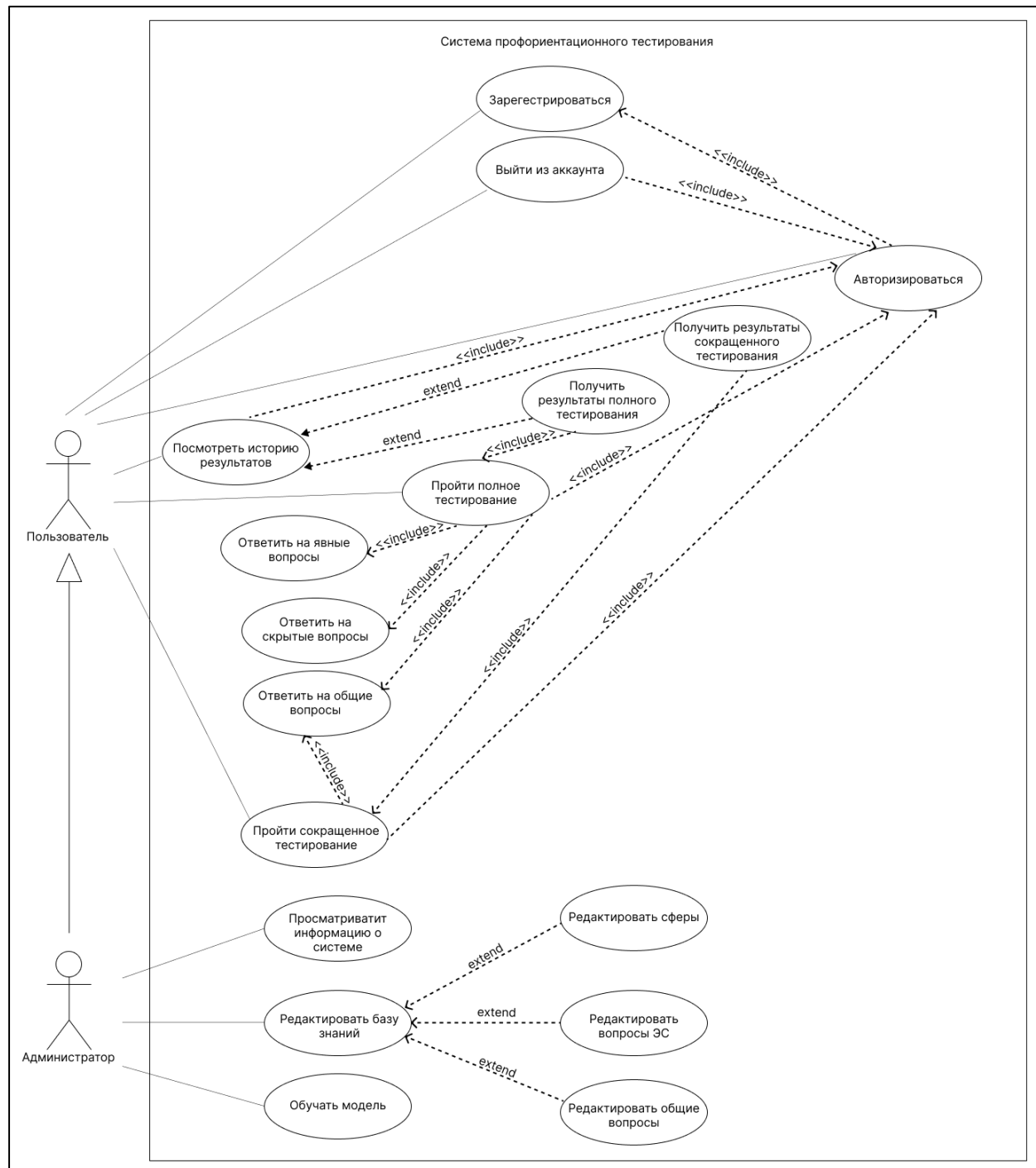


Рисунок 10 – UML-диаграмма прецедентов

2 Выбор и настройка инструментов разработки

Одной из важнейших частей при создании системы является выбор инструментов разработки. От решений, принятых на этом этапе, зависят не только сроки создания и стабильность работы системы, но и возможности её дальнейшего развития. В данном разделе рассматриваются технологии, выбранные для реализации разрабатываемой системы. Выбор инструментов осуществляется на основе сравнения альтернативных вариантов с учетом функциональных требований.

2.1 Обоснование выбора типа приложения

В настоящее время существует три основных типа приложений: десктопные, мобильные и веб-приложения. Рассмотрим подробнее каждый из них.

Десктопные приложения – это программное обеспечение, предназначенное для использования на персональных компьютерах. Оно имеет доступ к процессору, памяти и другим компьютерным ресурсам. При разработке десктопных приложений учитываются особенности операционной системы. Преимуществами данного типа являются: высокая производительность и мощность приложения, возможность использовать широкий спектр функций и продвинутый графический интерфейс. К недостаткам относят: необходимость разработки и тестирования на разных операционных системах, само приложение и его обновления требуют установки на персональный компьютер [5]. Значительным минусом также является отсутствие мобильности.

Рассмотрим мобильные приложения. Они разработаны для использования на девайсе с сенсорным экраном. Также как и десктопные, мобильные приложения имеют доступ к ресурсам устройства и учитывают специфику операционных систем. Преимуществами мобильных приложений являются возможность использования приложения независимо от местоположения

пользователя и информирование его о событиях с помощью уведомлений. Недостатки, связанные с разработкой и тестированием аналогичны десктопному типу приложений [5].

Последний тип – веб-приложения. Веб-приложение представляет собой программу, взаимодействие с которой происходит через браузер. Существенным преимуществом является независимость от операционной системы устройства. Помимо этого, нет необходимости устанавливать приложение на устройство. Недостатками данного типа являются: зависимость от интернет-соединения [5].

Разрабатываемая система направлена на уменьшение времени тестирования, поэтому человеку не рационально скачивать, а потом практически сразу удалять отдельную программу ради короткого теста. Благодаря веб-приложению пользователь переходит по ссылке, проходит тест и получает результат, не тратя дополнительное время и память устройства на установку приложения.

2.2 Выбор языка программирования для серверной части

Серверная часть должна осуществлять такие функции, как обработка запросов клиента, реализация ЭС, использование модели машинного обучения и взаимодействие с базой данных. Рассмотрим несколько вариантов для решения поставленной задачи.

Python – это высокоуровневый язык программирования с простым синтаксисом и динамической типизацией. Python является идеальным инструментом для решения задач машинного обучения, поскольку обладает высокой производительностью и богатым набором библиотек для машинного обучения, например, Pandas, NumPy, Matplotlib, PyTorch и TensorFlow [6]. Помимо этого, этот язык программирования обладает современными фреймворками для создания веб-приложений и REST API. Таким образом,

Python является мощным языком программирования для реализации поставленных задач.

Java – объектно-ориентированный язык программирования со строгой типизацией. Java по мере своего развития и развития машинного обучения начал использоваться в этой области. Этот язык также предлагает ряд инструментов и библиотек, которые используются для машинного обучения, например, WEKA, RapidMiner и Deeplearning4j [6]. Также присутствуют фреймворки для REST API и веб-приложений.

Хотя Java и обеспечивает высокую производительность, Python намного сильнее развит в сфере машинного обучения: больше библиотек, поддержка множества алгоритмов, простая разработка, что позволит реализовать нужный функционал и сократить время на разработку проекта.

2.3 Выбор фреймворка для реализации REST API

Для того, чтобы обеспечить клиент-серверное взаимодействие используется архитектурный стиль REST API, в котором данные передаются в формате JSON. Для использования на Python существует несколько популярных фреймворков: FastAPI, Flask и Django.

Django – это полнофункциональный фреймворк для веб-разработки на Python. Он включает в себя встроенную административную панель, систему аутентификации и другие функции. Этот фреймворк предлагает много способов оптимизации кода и огромное количество пакетов, работает с любым фронтендом через API. Django хорошо подходит для крупных веб-приложений, однако, поскольку он работает синхронно и включает в себя множество встроенных компонентов, с ним применяют дополнительные инструменты для выполнения длительных фоновых операций [7]. Для данной системы все его возможности избыточны.

Фреймворк Flask в свою очередь содержит только базовые возможности и является достаточно быстрым для небольших приложений и API. Он также

имеет много пакетов, однако многие из них устарели [7]. Фреймворк Flask отлично подходит для небольших проектов, но для полноценного REST API требуется подключение дополнительной библиотеки.

FastAPI в свою очередь использует библиотеки Starlette и Pydantic, которые позволяют повысить производительность и скорость обработки запросов. В отличие от синхронного Django, FastAPI построен на асинхронном протоколе, что помогает поддерживать обработку соединений без блокировки основного потока. Помимо этого, фреймворк автоматически выполняет валидацию входящих данных и генерирует документацию Swagger UI [7].

Таким образом, для реализации REST API был выбран фреймворк FastAPI, который показывает высокую производительность, не требуя подключения сторонних расширений, автоматически генерирует документацию и легко интегрируется в систему.

2.4 Выбор языка программирования и технологий для клиентской части

Клиентская часть приложения представляет собой веб-интерфейс. Для создания структуры и оформления страниц необходимо использовать язык разметки HTML и CSS, отвечающий за стили элементов. В настоящее время эти технологии не имеют альтернатив.

Для обработки действий пользователя рассмотрим JavaScript. JavaScript спроектирован как безопасный язык, который не имеет доступа к аппаратным ресурсам компьютера. Главная задача этого языка в браузере – управлять элементами веб-страницы, реагировать на действия пользователя и отправлять запросы серверу. JavaScript – технология, которая сочетает в себе три вещи: полная интеграция с HTML\CSS, встроенная поддержка всеми браузерами, простота реализации стандартных задач [8]. Эти свойства делают его самым подходящим языком программирования в рамках разрабатываемой программы.

					ПГУ 1.090301.15.001 ПЗ	Лист
						22
Изм.	Лист	№ докум.	Подпись	Дата		

Для упрощения разработки и динамического обновления страницы необходимо использовать библиотеку React. Эта библиотека позволяет использовать компонентный подход, что позволяет разбить программу на блоки – компоненты. React использует Virtual DOM для повышения производительности, что позволяет быстрее обновлять пользовательский интерфейс [9].

Таким образом, совокупность HTML, CSS, JavaScript и React позволяет сохранить баланс между простотой реализации и гибкостью компонентного подхода.

2.5 Выбор СУБД

Для хранения данных о пользователях, результатов их тестирования, вопросов теста требуется система управления базами данных (СУБД). СУБД должна быть надежной, поддерживать реляционную базу данных, обеспечить возможность работы с JSON-полями и быть совместимой с Python. Рассмотрим популярные реляционные системы управления базами данных.

SQLite – файловая БД, встраиваемая в приложение как библиотека. SQLite хорошо подходит для встроенных и мобильных приложений. Однако, предложенную СУБД не рекомендуют применять в многопользовательском режиме, а именно это и предполагает разрабатываемая программа [10].

MySQL является одной из самых популярных БД. Она имеет высокий уровень безопасности, большой функционал и подходит для веб-приложений. MySQL также имеет ряд недостатков: могут возникнуть проблемы совместимости из-за отличий от SQL-стандартов, при одновременных чтения и записи может возникнуть ряд проблем. Помимо этого, поддержка JSON имеет некоторые функциональные ограничения [10].

PostgreSQL – прогрессивная СУБД, которая соответствует стандартам SQL, в отличие от MySQL. PostgreSQL обеспечивает надежность и целостность данных, подходит для выполнения сложных процедур, предоставляет

поддержку JSON-полей. К недостаткам относят: скорость выполнения простых операций ниже, чем у MySQL [10].

Для разрабатываемой системы выбрана СУБД PostgreSQL, которая является надежной реляционной моделью с возможностью использования JSON-полей, а также хорошо совместима с Python.

2.6 Выбор сред разработки

Помимо вышеперечисленных пунктов очень важно выбрать среды разработки, которые будут использоваться для написания и отладки кода. Поскольку серверная часть будет разрабатываться на Python, а клиентская на JavaScript, для каждого из этих языков будет выбрана соответствующая среда разработки.

В качестве среды разработки для серверной части был выбран Spyder, входящий в состав дистрибутива Anaconda. Spyder – это среда, которая разрабатывалась для анализа данных и научных вычислений. Spyder предоставляет возможность запускать отдельные участки кода без запуска всего приложения, что может значительно ускорить процесс тестирования; позволяет просматривать содержимое переменных и массивов в реальном времени; имеет встроенную панель графиков. Использование Anaconda упрощает управление библиотеками и гарантирует стабильную работу проекта на других устройствах.

Visual Studio Code – среда разработки клиентской части. Visual Studio Code является универсальным редактором кода, который можно использовать для многих задач. Эта среда разработки имеет удобный интерфейс, низкое потребление системных ресурсов и высокую скорость работы. Главным ее преимуществом является система расширений, которая позволяет поддерживать любой язык. Среда Visual Studio Code была выбрана благодаря своей универсальности и наличию в ней расширений для JavaScript, React, HTML и CSS.

2.7 Настройка и развертывание среды разработки

В настройку серверной части входит установка дистрибутива Anaconda. Его можно скачать с официального сайта (рисунок 11)

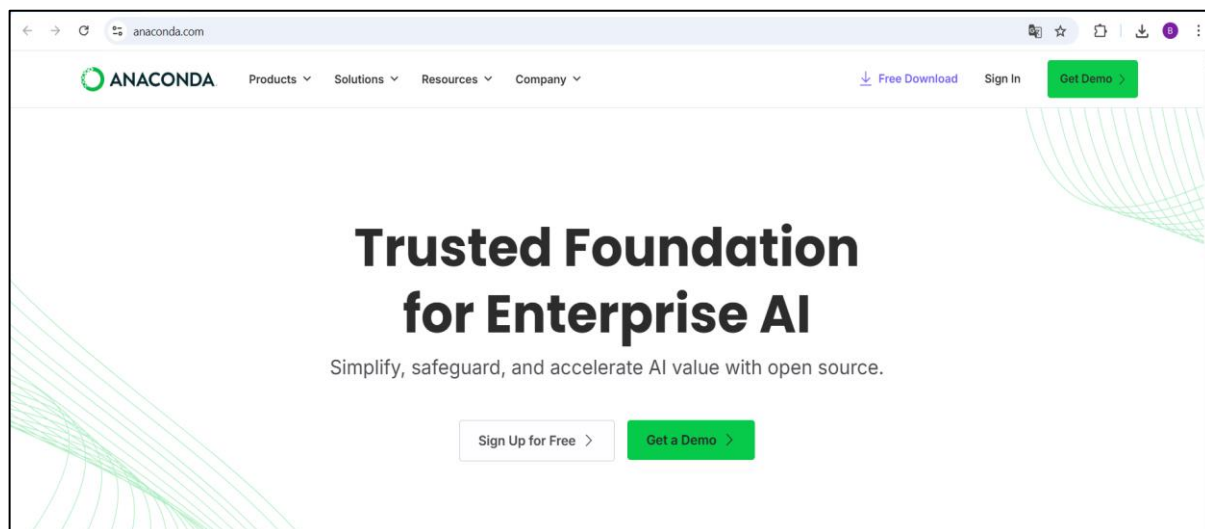


Рисунок 11 – Официальный сайт anaconda.com

В состав дистрибутива входит Anaconda Navigator, через который можно открыть Spyder (рисунок 12).

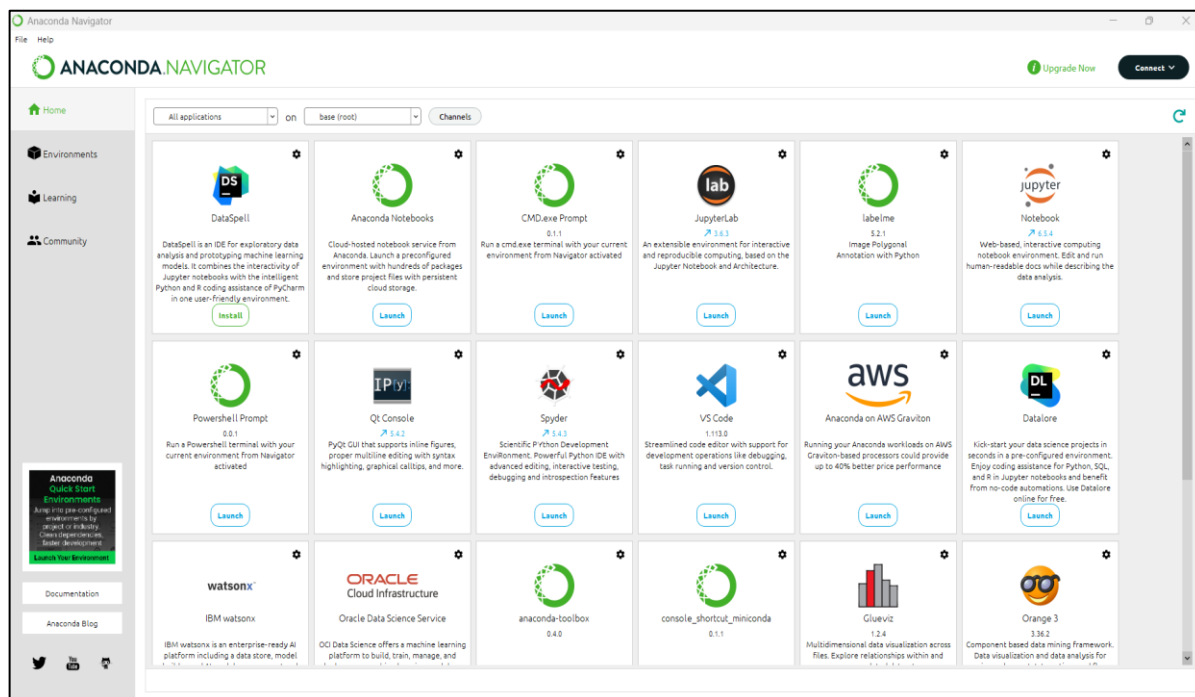
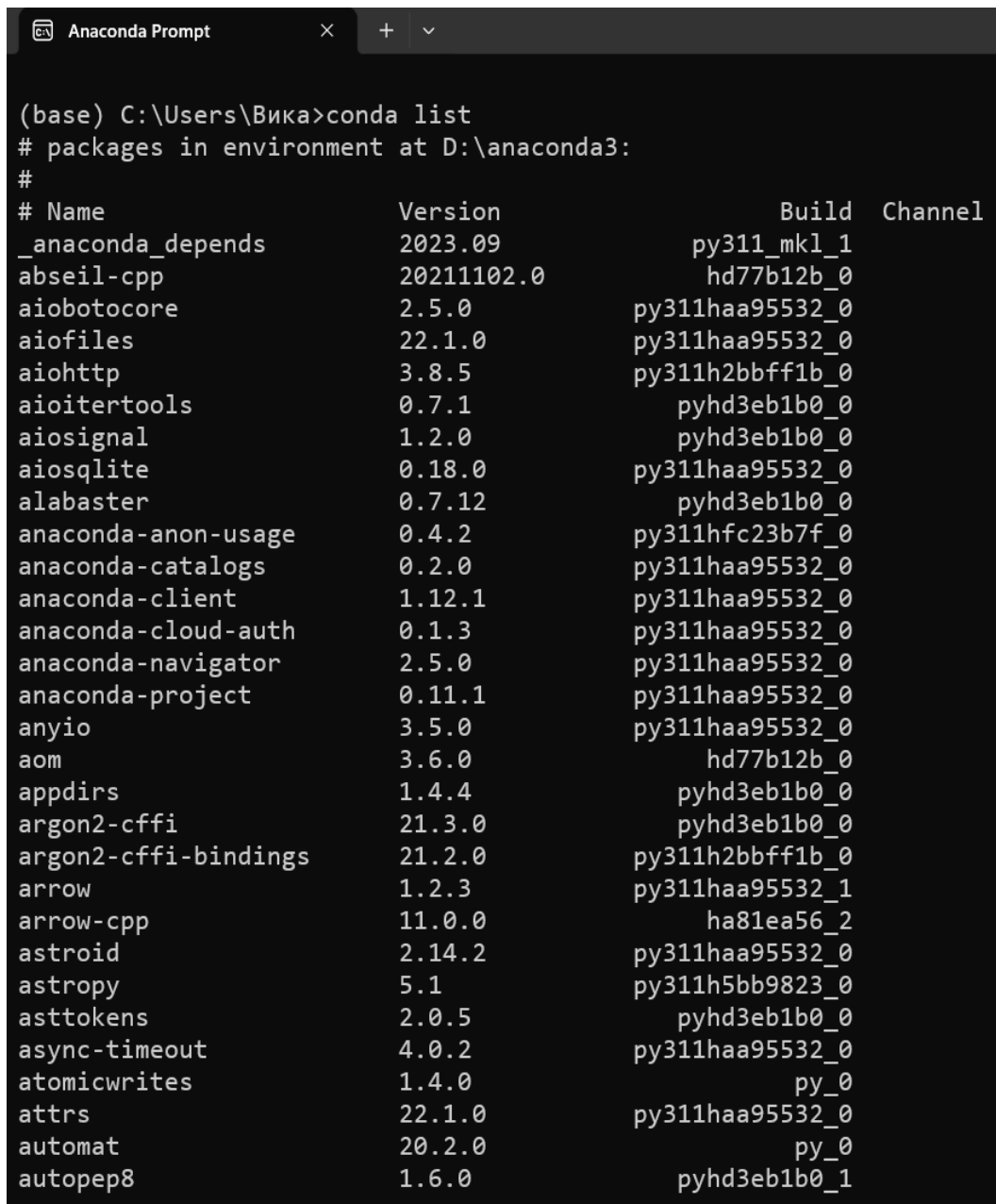


Рисунок 12 – Главное окно Anaconda Navigator

Помимо Spyder Anaconda включает в себя менеджер пакетов conda и предустановленные библиотеки NumPy, Pandas, Matplotlib и другие. Список установленных библиотек можно посмотреть через Anaconda Prompt, используя команду conda list. Пример выполнения команды показан на рисунке 13.



```
(base) C:\Users\Вика>conda list
# packages in environment at D:\anaconda3:
#
# Name                                Version                                Build                                Channel
_anaconda_depends                     2023.09                               py311_mkl_1
abseil-cpp                             20211102.0                            hd77b12b_0
aiobotocore                           2.5.0                                 py311haa95532_0
aiofiles                              22.1.0                                py311haa95532_0
aiohttp                               3.8.5                                 py311h2bbff1b_0
aioitertools                           0.7.1                                 pyhd3eb1b0_0
aiosignal                             1.2.0                                 pyhd3eb1b0_0
aiosqlite                              0.18.0                                py311haa95532_0
alabaster                              0.7.12                                pyhd3eb1b0_0
anaconda-anon-usage                    0.4.2                                 py311hfc23b7f_0
anaconda-catalogs                     0.2.0                                 py311haa95532_0
anaconda-client                        1.12.1                                py311haa95532_0
anaconda-cloud-auth                    0.1.3                                 py311haa95532_0
anaconda-navigator                     2.5.0                                 py311haa95532_0
anaconda-project                       0.11.1                                py311haa95532_0
anyio                                  3.5.0                                 py311haa95532_0
aom                                    3.6.0                                 hd77b12b_0
appdirs                               1.4.4                                 pyhd3eb1b0_0
argon2-cffi                            21.3.0                                pyhd3eb1b0_0
argon2-cffi-bindings                  21.2.0                                py311h2bbff1b_0
arrow                                  1.2.3                                 py311haa95532_1
arrow-cpp                              11.0.0                                ha81ea56_2
astroid                                2.14.2                                py311haa95532_0
astropy                                5.1                                   py311h5bb9823_0
asttokens                              2.0.5                                 pyhd3eb1b0_0
async-timeout                          4.0.2                                py311haa95532_0
atomicwrites                           1.4.0                                 py_0
attrs                                  22.1.0                                py311haa95532_0
automat                                20.2.0                                py_0
autopep8                               1.6.0                                 pyhd3eb1b0_1
```

Рисунок 13 – Выполнение команды conda list

Дополнительные пакеты устанавливаются с помощью команды pip. На рисунке 14 приведен пример установки пакета fastapi.

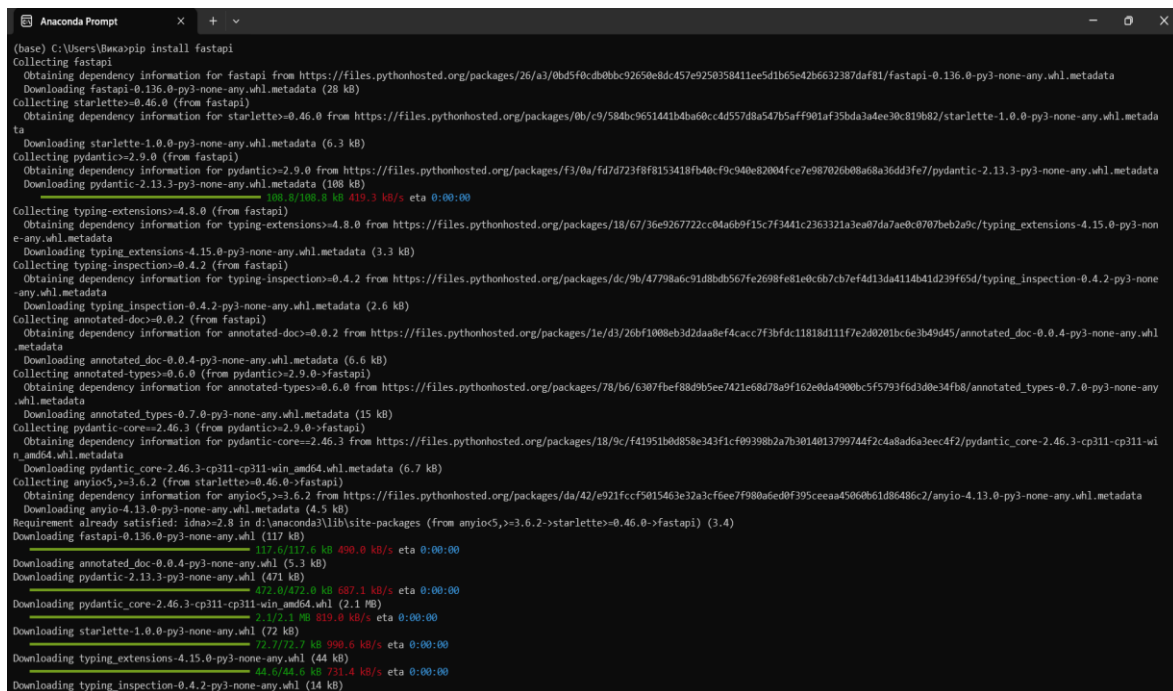


Рисунок 14 – Выполнение команды `pip install fastapi`

Для работы с данными необходимо установить СУБД PostgreSQL. Дистрибутив PostgreSQL можно скачать на официальном сайте [postgresql.org](https://www.postgresql.org) (рисунок 15)

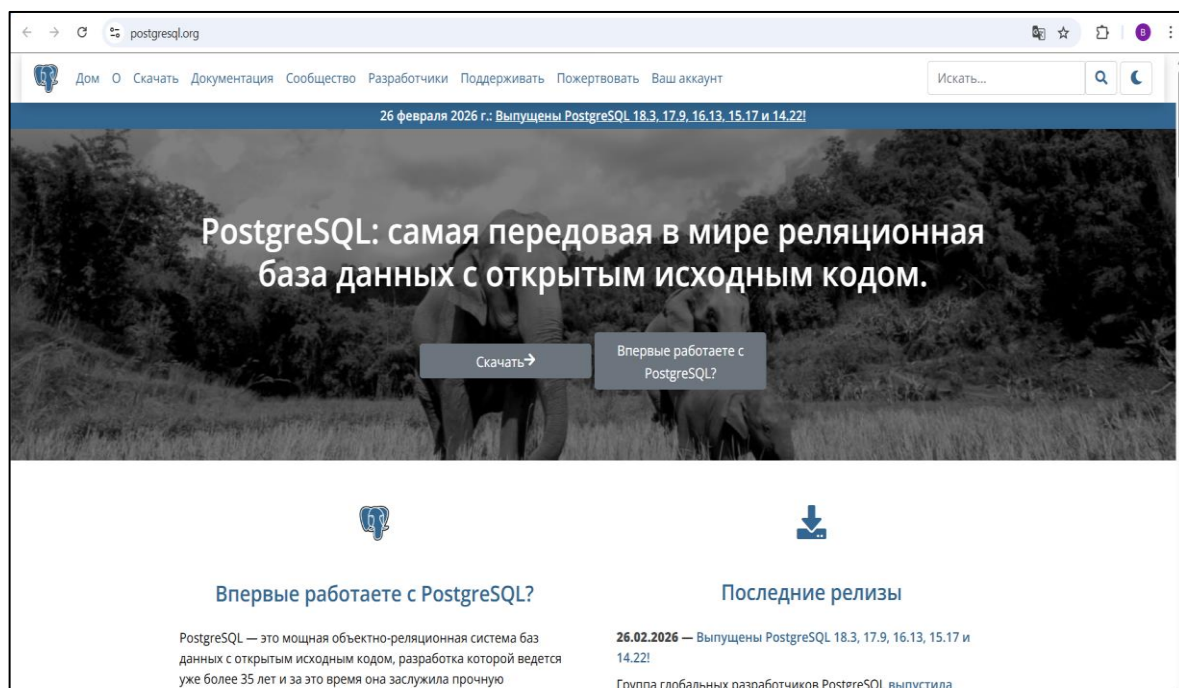


Рисунок 15 – Официальный сайт [postgresql.org](https://www.postgresql.org)

При установке автоматически будут предложены компоненты (рисунок 16):

- PostgreSQL Server – сервер для БД;
- pgAdmin 4 – графический интерфейс для управления БД;
- Stack Builder – утилита для установки дополнительных расширений;
- Command Line Tools – инструменты командной строки.

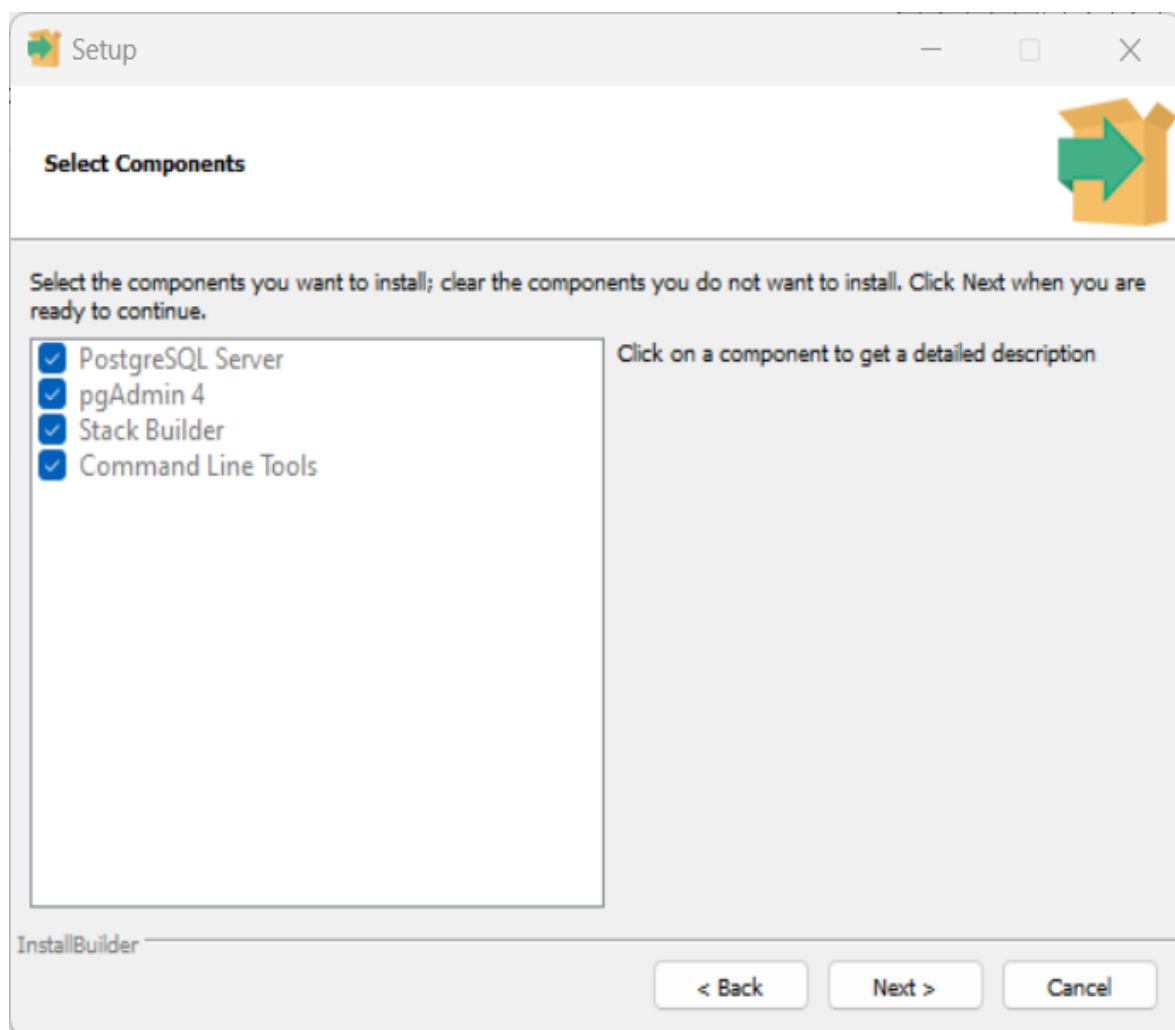


Рисунок 16 – Компоненты установки PostgreSQL

База данных и таблицы создаются автоматически при первом запуске сервера разрабатываемого приложения.

Для разработки клиентской части необходимо установить редактор Visual Studio Code на официальном сайте code.visualstudio.com (рисунок 17).

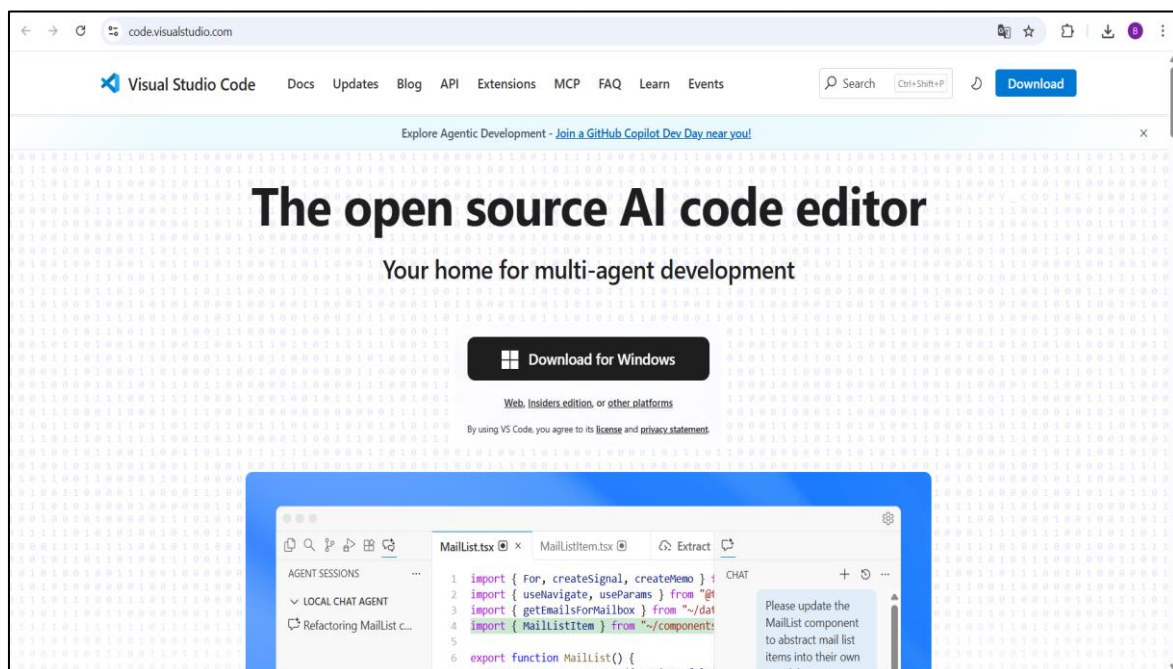


Рисунок 17 – Официальный сайт code.visualstudio.com

Дополнительные расширения можно установить через встроенный магазин расширений. Его можно открыть через горячие клавиши Ctrl+Shift+X. Магазин расширений представлен на рисунке 18.

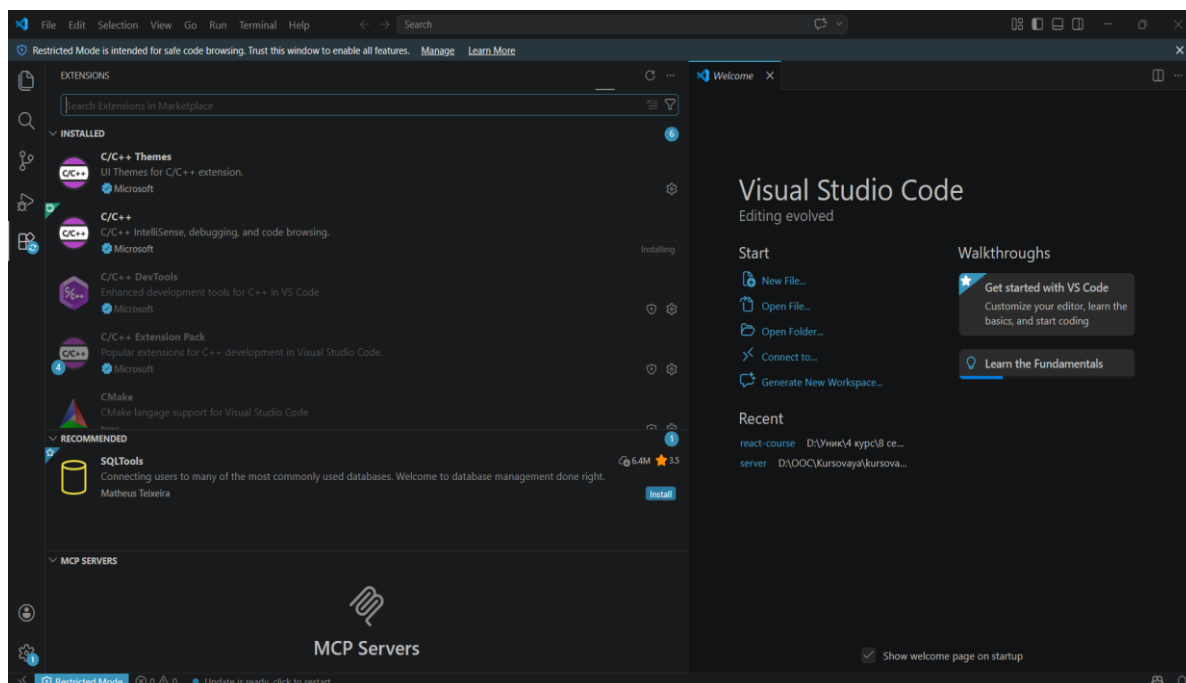


Рисунок 18 – Магазин расширений Visual Studio Code

Для работы с библиотекой React необходимо установить Node.js (рисунок 19), который включает в себя менеджер пакетов npm.

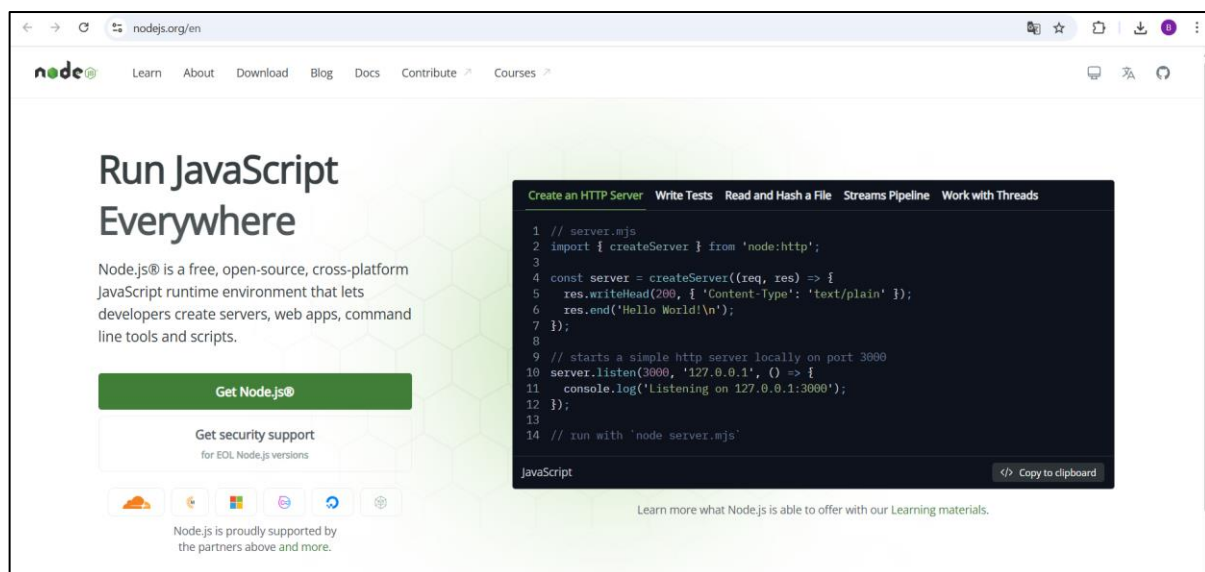


Рисунок 19 – Официальный сайт nodejs.org

Проверить установку и версии можно с помощью команд `node -v` и `npm -v` (рисунок 20)

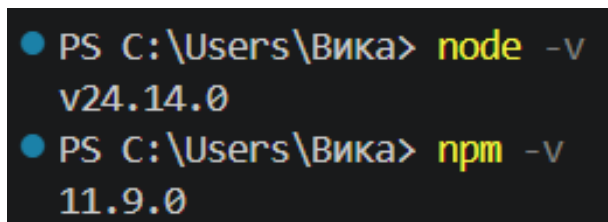


Рисунок 20 – Выполнение команд `node -v` и `npm -v`

Создание нового проекта на React выполняется с помощью команды `npm create vite@latest`. На рисунке 21 показан результат выполнения этой команды.

При этом для проверки работоспособности приложения пользователю не требуется выполнять описанные выше действия, ему достаточно просто открыть приложение в браузере.

```

PS D:\DIPLOM\DiplomProject> npm create vite@latest my-react-app -- --template react
> npx
> create-vite my-react-app --template react

◇ Install with npm and start now?
  Yes

◇ Scaffolding project in D:\DIPLOM\DiplomProject\my-react-app...

◇ Installing dependencies with npm...

added 152 packages, and audited 153 packages in 17s

36 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

◇ Starting dev server...

> my-react-app@0.0.0 dev
> vite

VITE v8.0.9 ready in 1464 ms

→ Local:   http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help

```

Рисунок 21 – Выполнение команды npm create vite@latest

					ПГУ 1.090301.15.001 ПЗ	Лист
						31
Изм.	Лист	№ докум.	Подпись	Дата		

3 Проектирование системы

3.1 Паттерн MVC

Паттерн MVC (Model-View-Controller) предполагает разделение приложения на три части, которые могут разрабатываться независимо друг от друга, но работают вместе (рисунок 22). Основная идея паттерна – независимость компонентов. Это позволяет сократить число ошибок во время разработки системы. Основные принципы паттерна:

Model – компонент, который работает с данными. Он может обращаться к базе данных, обрабатывать файлы, осуществлять вычисления, но ничего не знает о визуализации. Модель полностью абстрагирована от интерфейса.

View – компонент, который отображает данные. Он получает данные от модели и отображает их в форме, понятной пользователю. Данные могут быть представлены в различных формах: график, таблица, текст, однако в сам вид при всем этом получает один и тот же массив данных.

Controller – часть паттерна, которая управляет потоками данных между моделью и видом. Контроллер анализирует действия пользователя и определяет, какие методы необходимо использовать для обработки запроса. В веб-приложениях он обрабатывает GET, POST и другие запросы, COOKIE и другие данные, на основе которых строится логика приложения [11].

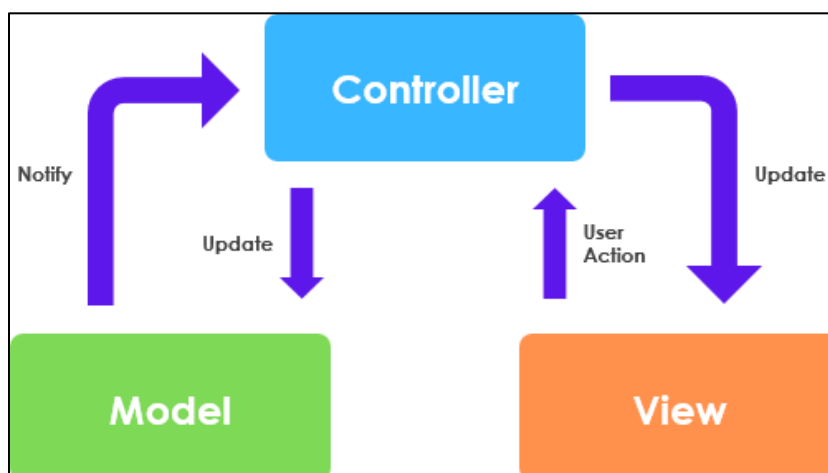


Рисунок 22 – Паттерн MVC

При разработке приложений с базой данных, графическим интерфейсом и другими сервисами очень тяжело сохранить порядок в коде. Поэтому в ходе разработки системы был использован паттерн MVC. Через визуальное отображение контроллер принимает запросы пользователя и обращается к базе данных. При этом вид не знает, как обрабатываются данные, а модель не знает, как они отображаются.

3.2 Архитектура приложения

На рисунке 23 представлена архитектура системы профориентационного тестирования.

При использовании системы пользователь взаимодействует с элементами графического интерфейса. В этот момент он использует компонент View. В разрабатываемом приложении View представляет собой веб-интерфейс, благодаря которому пользователь видит все страницы с соответствующими элементами.

Если пользователь совершает какое-либо действие, например, нажимает на кнопку, то клиент формирует и отправляет HTTP-запрос на сервер. На сервере фреймворк FastAPI на основе URL-адреса запроса направляет его к нужному контроллеру.

Controller в свою очередь получает запрос от клиента и вызывает соответствующий метод Model. Например, при прохождении полного тестирования Model вызывает систему экспертного тестирования. После выполнения необходимых действий результат возвращается обратно в контроллер, который формирует и передает HTTP-ответ клиенту. Затем View отображает результаты пользователю через графический интерфейс.

На рисунке 24 представлена диаграмма развертывания приложения. Система развернута на двух узлах: устройство пользователя и сервер.

На устройстве пользователя находится веб-браузер, через который используется приложение.

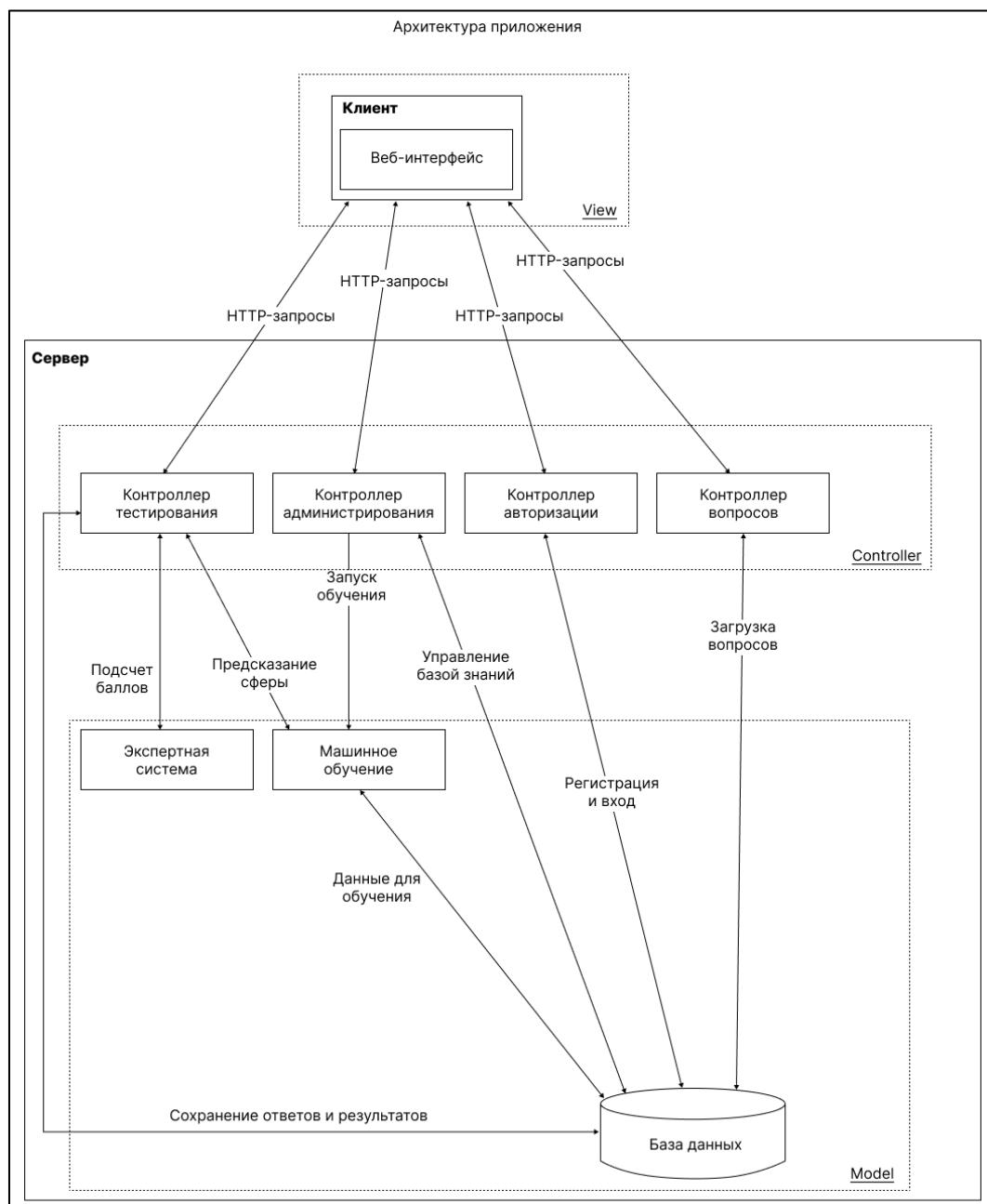


Рисунок 23 – Архитектура системы профориентационного тестирования

На сервере располагаются следующие компоненты:

- FastAPI – фреймворк, на котором реализована серверная часть,
- база данных (PostgreSQL),
- random_forest_model.pkl – файл с обученной ML-моделью.

Взаимодействие между компонентами происходит следующим образом: браузер отправляет HTTP-запросы к FastAPI, FastAPI выполняет запросы к БД для получения и сохранения данных. При необходимости сервер загружает ML-

модель из файла random_forest_model.pkl. Ответ через FastAPI возвращается обратно в браузер.

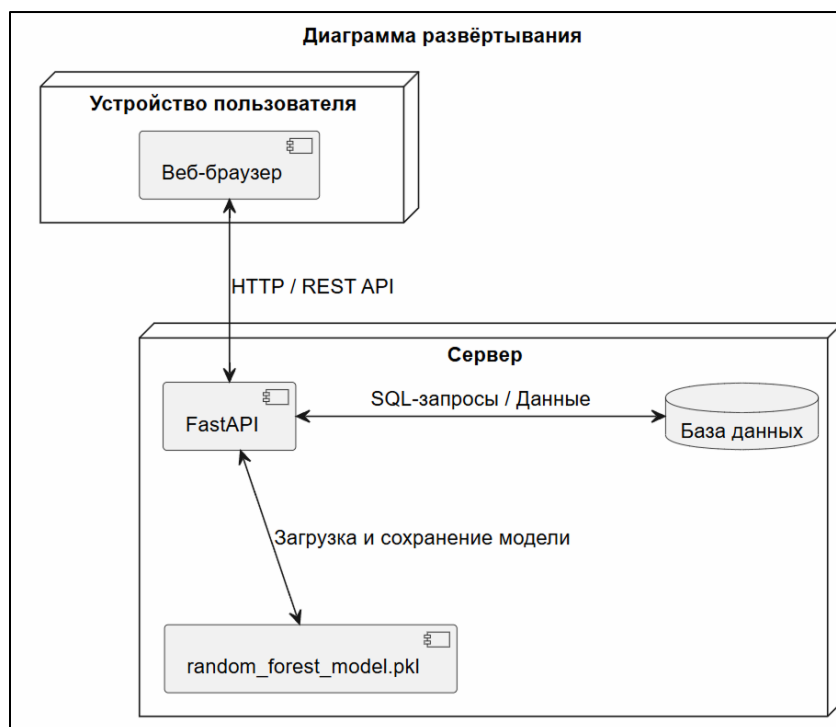


Рисунок 24 – UML-диаграмма развёртывания

В приложении А.2 – А.8 представлены диаграммы последовательностей, показывающие взаимодействие компонентов системы при реализации сценариев: авторизация, регистрация, полный тест, короткий тест, история тестов, администрирование, выход.

В приложении А.9 представлены UML-диаграммы деятельности по сценариям.

В приложении А.10 представлена диаграмма классов приложения.

3.3 Проектирование базы данных

Для хранения информации о пользователях, вопросах и ответах теста и других данных необходимо спроектировать базу данных. Рассмотрим сущности, из которых состоит система профориентационного тестирования:

- users – таблица, содержащая данные учетных записей пользователей и администраторов (таблица 2);
- spheres – таблица, содержащая информацию о профессиональных сферах деятельности, по которым оцениваются результаты тестирования (таблица 3);
- questions – таблица, содержащая вопросы ЭС для полного теста с разделением на явные и скрытые (таблица 4);
- question_sphere – связующая таблица, реализующая связь многие ко многим между вопросами и сферами (таблица 5);
- general_questions – таблица, содержащая общие вопросы для сокращенного тестирования с вариантами ответов (таблица 6);
- test_sessions – таблица, хранящая информацию о сессиях прохождения тестов пользователями (таблица 7);
- answers – таблица, содержащая ответы пользователей на вопросы ЭС (таблица 8);
- general_answers – таблица, содержащая ответы на вопросы для ML-модели (таблица 9);
- full_test_results – таблица, хранящая результаты полного тестирования по каждой сфере (таблица 10);
- short_test_results – таблица, хранящая результаты короткого тестирования по каждой сфере (таблица 11);
- ml_training_data – таблица, содержащая выборку для машинного обучения: ответы на общие вопросы и целевую сферу (таблица 12);
- ml_models – таблица, хранящая информацию об обученных версиях ML-модели, их точности и важности признаков (таблица 13).

Таблица 2 – Атрибуты таблицы users

Поле	Тип	Роль / Ограничения	Описание
id_users	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор пользователя
login	VARCHAR (50)	NOT NULL, UNIQUE	Логин для входа в систему
password_hash	VARCHAR (255)	NOT NULL	Хэш пароля
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Дата и время регистрации
role	VARCHAR (20)	NOT NULL, DEFAULT 'user'	Роль: user или admin

Таблица 3 – Атрибуты таблицы spheres

Поле	Тип	Роль / Ограничения	Описание
id_spheres	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор сферы
name	VARCHAR (100)	NOT NULL, UNIQUE	Название профессиональной сферы
description	TEXT		Краткое описание сферы
max_score	INT	NOT NULL	Максимально возможный балл по сфере

Таблица 4 – Атрибуты таблицы questions

Поле	Тип	Роль / Ограничения	Описание
id_questions	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор вопроса
text	TEXT	NOT NULL	Текст вопроса
type	VARCHAR(10)	NOT NULL	Тип вопроса: explicit или hidden
is_active	BOOLEAN	DEFAULT TRUE	Активен ли вопрос

Таблица 5 – Атрибуты таблицы question_sphere

Поле	Тип	Роль / Ограничения	Описание
id_question_sphere	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор связи
question_id	INT	FOREIGN KEY (questions.id_questions), NOT NULL	ID вопроса
sphere_id	INT	FOREIGN KEY (spheres.id_spheres), NOT NULL	ID сферы

Таблица 6 – Атрибуты таблицы general_questions

Поле	Тип	Роль / Ограничения	Описание
id_general_questions	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор вопроса
text	TEXT	NOT NULL	Текст вопроса
options	JSON	NOT NULL	Варианты ответов (массив строк)
is_active	BOOLEAN	DEFAULT TRUE	Активен ли вопрос

Таблица 7 – Атрибуты таблицы test_sessions

Поле	Тип	Роль / Ограничения	Описание
id_test_sessions	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор сессии
user_id	INT	FOREIGN KEY (users.id_users), NOT NULL	ID пользователя
test_type	VARCHAR(10)	NOT NULL	Тип теста: full или short
started_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Время начала тестирования
completed_at	TIMESTAMP		Время завершения тестирования

Таблица 8 – Атрибуты таблицы answers

Поле	Тип	Роль / Ограничения	Описание
id_answers	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор ответа
session_id	INT	FOREIGN KEY (test_sessions.id_test_sessions), NOT NULL	ID сессии
question_id	INT	FOREIGN KEY (questions.id_questions), NOT NULL	ID вопроса
answer_value	INT	NOT NULL, CHECK (1..5)	Ответ по шкале от 1 до 5

Таблица 9 – Атрибуты таблицы general_answers

Поле	Тип	Роль / Ограничения	Описание
id_general_answers	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор ответа
session_id	INT	FOREIGN KEY (test_sessions.id_test_sessions), NOT NULL	ID сессии
question_id	INT	FOREIGN KEY (general_questions.id_general_questions), NOT NULL	ID общего вопроса
answer_value	VARCHAR (255)	NOT NULL	Текст выбранного варианта ответа
answer_index	INT		Числовой индекс ответа (для ML)

Таблица 10 – Атрибуты таблицы full_test_results

Поле	Тип	Роль / Ограничения	Описание
id_full_test_results	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор результата
session_id	INT	FOREIGN KEY (test_sessions.id_test_sessions), NOT NULL	ID сессии
sphere_id	INT	FOREIGN KEY (spheres.id_spheres), NOT NULL	ID сферы
score	INT	NOT NULL	Итоговый балл

Поле	Тип	Роль / Ограничения	Описание
inclination_level	VARCHAR (20)	NOT NULL	Уровень склонности

Таблица 11 – Атрибуты таблицы short_test_results

Поле	Тип	Роль / Ограничения	Описание
id_short_test_results	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор результата
session_id	INT	FOREIGN KEY (test_sessions.id_test_sessions), NOT NULL	ID сессии
sphere_id	INT	FOREIGN KEY (spheres.id_spheres), NOT NULL	ID сферы
probability	FLOAT	NOT NULL	Вероятность в процентах

Таблица 12 – Атрибуты таблицы ml_training_data

Поле	Тип	Роль / Ограничения	Описание
id_ml_training_data	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор записи
session_id	INT	FOREIGN KEY (test_sessions.id_test_sessions), NOT NULL	ID сессии полного теста
features	JSON	NOT NULL	Ответы на общие вопросы (признаки)
target_sphere_id	INT	FOREIGN KEY (spheres.id_spheres), NOT NULL	ID сферы с максимальным баллом

Таблица 13 – Атрибуты таблицы ml_models

Поле	Тип	Роль / Ограничения	Описание
id_ml_models	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор модели
version	VARCHAR (20)	NOT NULL, UNIQUE	Версия модели
model_file	VARCHAR (255)	NOT NULL	Путь к файлу обученной модели
accuracy	FLOAT	NOT NULL	Точность модели (0..1)
trained_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Дата и время обучения
is_active	BOOLEAN	DEFAULT FALSE	Используется ли модель в данный момент
features_importance	JSON		Важность признаков

В таблице 14 представлены связи между таблицами системы.

Таблица 14 – Связи между таблицами

№	Родительская таблица	Дочерняя таблица	Тип связи	Внешний ключ	Описание
1	users	test_sessions	1:M	user_id – id_users	Один пользователь может пройти несколько тестов

№	Родительская таблица	Дочерняя таблица	Тип связи	Внешний ключ	Описание
2	test_sessions	answers	1:M	session_id – id_test_sessions	Одна сессия содержит много ответов на вопросы ЭС
3	test_sessions	general_answers	1:M	session_id – id_test_sessions	Одна сессия содержит много ответов на общие вопросы
4	test_sessions	full_test_results	1:M	session_id – id_test_sessions	Одна сессия – запись результатов (по 1 на сферу)
5	test_sessions	short_test_results	1:M	session_id – id_test_sessions	Одна сессия – запись результатов (по 1 на сферу)
6	test_sessions	ml_training_data	1:M	session_id – id_test_sessions	Одна сессия может дать несколько записей для обучения (при равных баллах)
7	questions	question_sphere	1:M	question_id – id_questions	Один вопрос может быть связан с несколькими сферами

№	Родительская таблица	Дочерняя таблица	Тип связи	Внешний ключ	Описание
8	spheres	question_sphere	1:M	sphere_id – id_spheres	Одна сфера может быть связана с несколькими вопросами
9	questions	answers	1:M	question_id – id_questions	Один вопрос может иметь много ответов от разных пользователей
10	general _questions	general_answers	1:M	question_id – id_general_questions	Один общий вопрос может иметь много ответов
11	spheres	full_test_results	1:M	sphere_id – id_spheres	Одна сфера может фигурировать во многих результатах полных тестов
12	spheres	short_test_results	1:M	sphere_id – id_spheres	Одна сфера может фигурировать во многих результатах коротких тестов

№	Родительская таблица	Дочерняя таблица	Тип связи	Внешний ключ	Описание
13	spheres	ml_training_data	1:M	target_sphere_id – id_spheres	Одна сфера может быть целевой для многих записей обучающей выборки

Физическая модель базы данных представлена на рисунке 25.

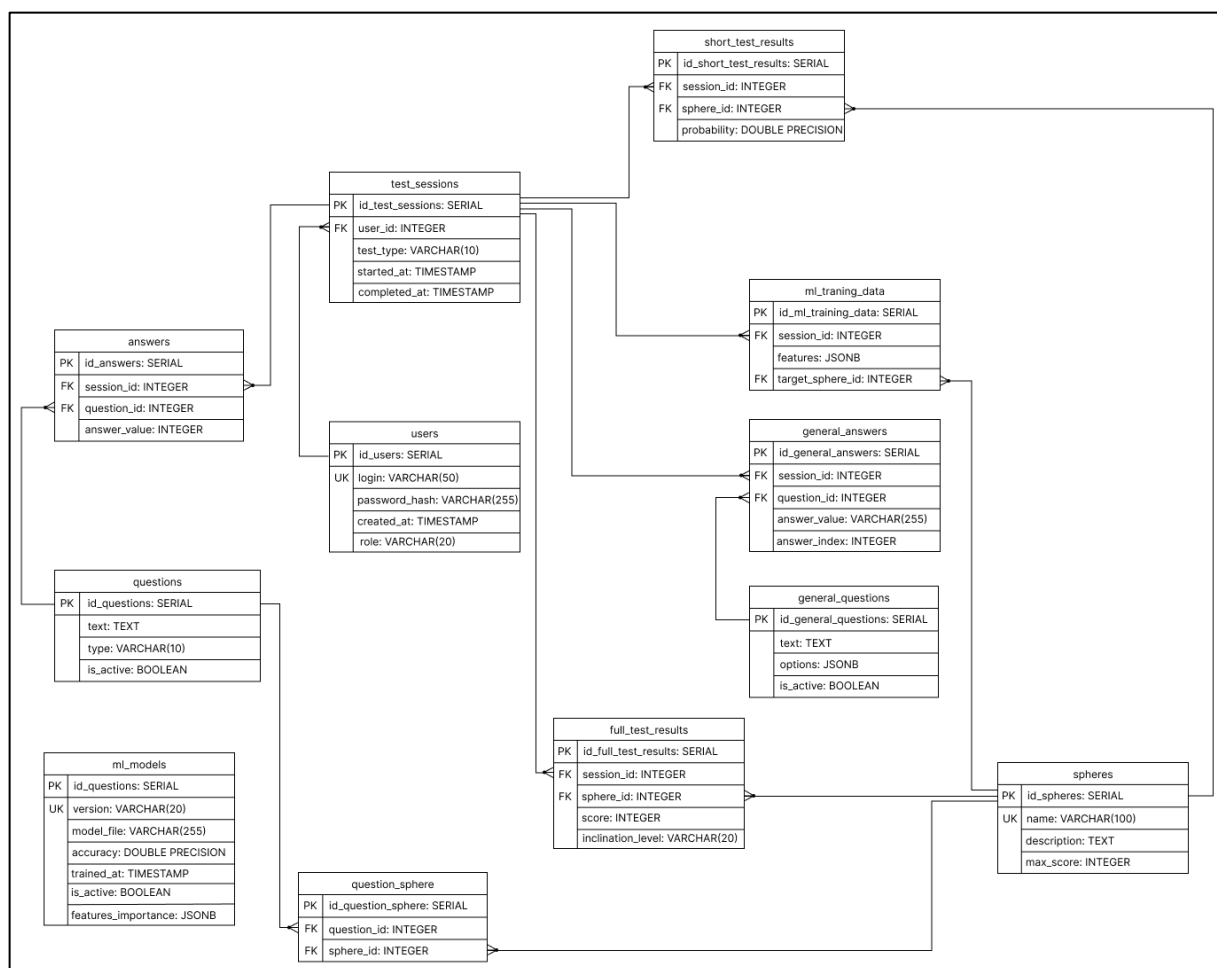


Рисунок 25 – Физическая модель базы данных

Логическая модель базы данных представлена на рисунке 26.

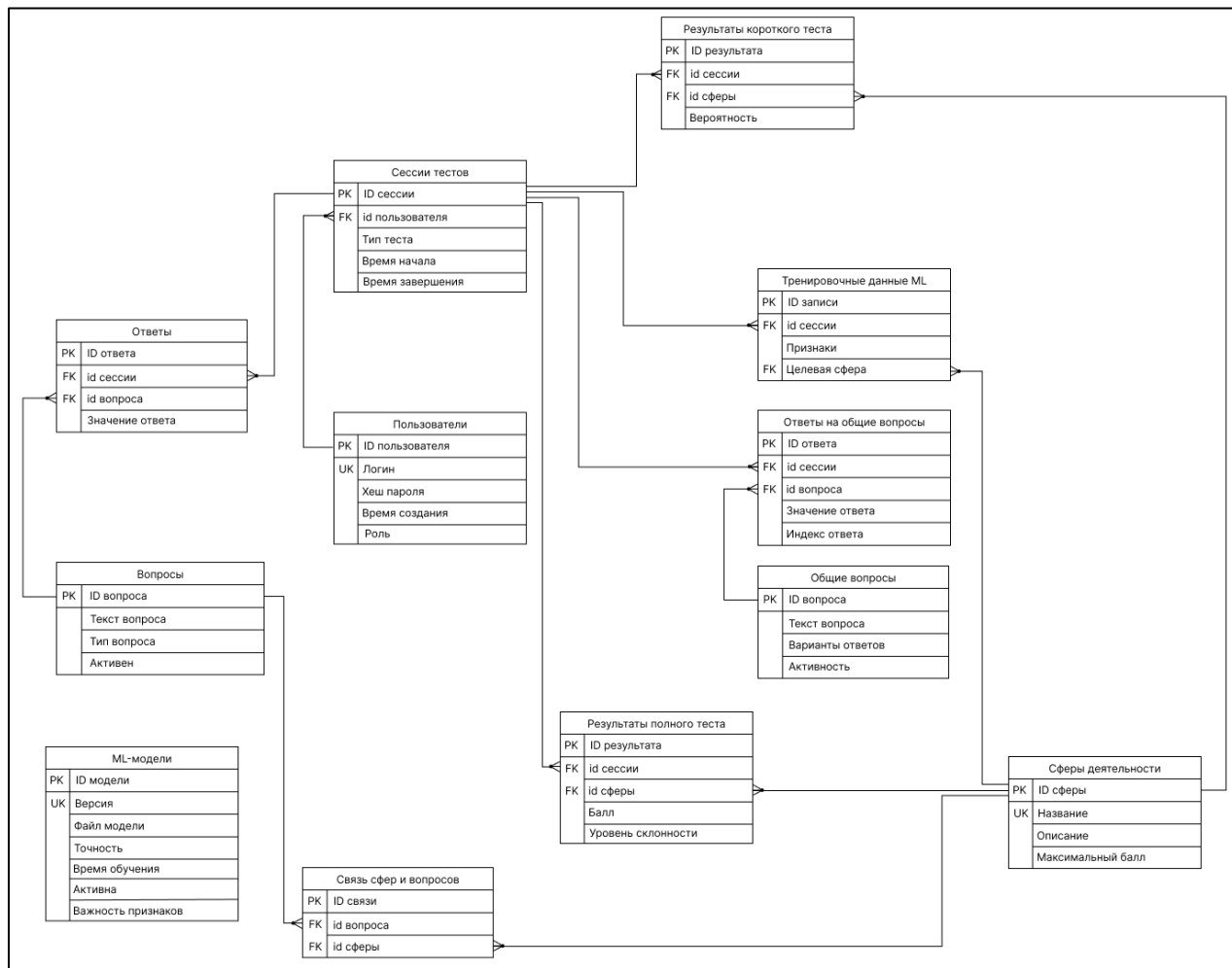


Рисунок 26 – Логическая модель базы данных

4 Разработка системы

Разрабатываемая система осуществляет профориентационное тестирование в два этапа:

- сбор данных о профессиональных склонностях пользователя с помощью подробного тестирования (ЭС);
- обучение модели для прогнозирования на основе результатов первого этапа.

Вопросы, сферы деятельности и другие данные могут быть адаптированы под нужды заказчика. В рамках данной работы рассмотрение системы будет производиться на конкретном примере.

4.1 Разработка экспертной системы

Важной частью ЭС является база знаний, которая включает в себя вопросы и профессиональные сферы деятельности.

Система оценивает склонности по пятнадцати сферам деятельности: «Информационные технологии», «Медицина и здравоохранение», «Наука и образование», «Инженерия и технологии», «Бизнес и финансы», «Творчество и искусство», «Право и государственное управление», «Сельское хозяйство и экология», «Социальная сфера», «Военное дело и безопасность», «Спорт и фитнес», «Мода и красота», «Транспорт и логистика», «Туризм и гостиничный бизнес», «Развлечения и игровая индустрия».

Помимо сфер ЭС включает в себя вопросы, которые в свою очередь делятся на два типа: явные и скрытые. Эти вопросы оцениваются по 5-балльной шкале и хранятся в таблице questions.

Явные вопросы (105 штук) связаны лишь с одной сферой. Пример вопроса: «Мне нравится разбираться в программных алгоритмах» («Информационные технологии – IT»).

Скрытые вопросы (46 шт.). Эти вопросы связаны одновременно с несколькими сферами. Данные вопросы предназначены для выявления скрытых

предрасположенностей человека. Пример вопроса: «Я ценю баланс между инновациями и надежностью» («Инженерия и технологии», «Медицина и здравоохранение», «Военное дело и безопасность», «Сельское хозяйство и экология»).

За получение вопросов для прохождения полного тестирования отвечает функция `get_expert_questions` из файла `questions_router.py`. Для начала формируется запрос к таблице `questions`. Из этой таблицы берутся только записи, у которых поле `is_active` имеет значение `True`. Это необходимо для того, чтобы пользователь получил только активные вопросы.

```
@router.get("/expert", response_model=list[QuestionResponse])
def get_expert_questions(db: Session = Depends(get_db)):
    questions = (
        db.query(Question)
        .filter(Question.is_active == True)
        .all()
    )
```

Для каждого полученного вопроса формируется список названий профессиональных сфер, соответствующих этому вопросу. Для этого используем связующую таблицу `question_spheres`. Она реализует связь многие ко многим между таблицами `questions` и `spheres`.

```
result = []
for q in questions:
    sphere_names = [
        qs.sphere.name
        for qs in q.question_spheres
    ]
```

Затем формируется объект `QuestionResponse`, который будет отправлен клиенту.

```
result.append(QuestionResponse(
    id=q.id_questions,
    text=q.text,
    type=q.type,
```

```

        spheres=sphere_names
    ))
return result

```

За работу ЭС отвечает файл expert_system.py.

Логика оценивания для явных вопросов: ответ «5» приносит сфере 4 балла (максимум за один вопрос), ответ «4» – 2 балла, ответ «3» – 1 балл, ответ «2» – 0.5 балла, ответ «1» – не приносит баллов.

Логика оценивания для скрытых вопросов: ответ «5» – каждой сфере начисляется по 2 балла, ответ «4» – по 1 баллу, ответ «3» – по 0.5 балла, ответ «2» – по 0.25 балла. При этом каждая сфера не может получить более шести баллов за все скрытые вопросы.

Баллы за явные вопросы в зависимости от ответа (1-5):

```

EXPLICIT_SCORES = {
    5: 4,
    4: 2,
    3: 1,
    2: 0.5,
    1: 0
}

```

Баллы за скрытые вопросы в зависимости от ответа (1-5):

```

HIDDEN_SCORES = {
    5: 2,
    4: 1,
    3: 0.5,
    2: 0.25,
    1: 0
}

```

Максимальный суммарный балл, который одна сфера может получить со всех скрытых вопросов:

```

MAX_HIDDEN_SCORE_PER_SPHERE = 6

```

Функция `calculate_scores` вычисляет баллы по сферам на основе ответов пользователя. Эта функция принимает аргумент `answers_with_spheres` –

список словарей, в каждом из которых указано значение ответа, тип вопроса и список сфер, которые относятся к этому вопросу. Вначале создаются два словаря: `scores` – итоговые баллы по каждой сфере и `implicit_scores` – сумма баллов, полученных от скрытых вопросов.

```
def calculate_scores(answers_with_spheres):
```

```
    scores = {}          # итоговые баллы по каждой сфере
    implicit_scores = {} # сумма баллов, полученных от скрытых
    вопросов
```

Далее функция обрабатывает каждый ответ пользователя.

```
    for item in answers_with_spheres:
        answer = item['answer_value']
        q_type = item['question_type']
        spheres = item['spheres']
```

Если вопрос явный – из словаря `EXPLICIT_SCORES` получаем баллы в зависимости от ответа и начисляем их сфере, которая соответствует вопросу.

```
        if q_type == 'explicit':
            points = EXPLICIT_SCORES.get(answer, 0)
            for sphere_id in spheres:
                scores[sphere_id] = scores.get(sphere_id, 0) +
points
```

Если вопрос скрытый – из словаря `HIDDEN_SCORES` получаем баллы в зависимости от ответа и начисляем их соответствующим сферам. При этом для каждой сферы проверяется лимит баллов.

```
        elif q_type == 'hidden':
            points = HIDDEN_SCORES.get(answer, 0)
            for sphere_id in spheres:
                current_implicit = implicit_scores.get(sphere_id,
0)

                if current_implicit < MAX_HIDDEN_SCORE_PER_SPHERE:
                    available = MAX_HIDDEN_SCORE_PER_SPHERE -
current_implicit

                    actual_points = min(points, available)
```

```

        scores[sphere_id] = scores.get(sphere_id, 0) +
actual_points
        implicit_scores[sphere_id] = current_implicit
+ actual_points

```

После начисления баллов за все ответы, баллы округляются.

```

for sphere_id in scores:
    scores[sphere_id] = int(round(scores[sphere_id]))
return scores

```

Функция `get_inclination_level` определяет уровень склонности на основе процента от максимального балла.

```

def get_inclination_level(score, max_possible_score):
    percent = (score / max_possible_score) * 100 if
max_possible_score > 0 else 0
    if percent >= 76:
        return "максимальная"
    elif percent >= 56:
        return "умеренная"
    elif percent >= 23:
        return "слабая"
    else:
        return "минимальная"

```

Функция `get_top_spheres` возвращает список всех сфер с максимальным баллом (целевых сфер). Целевые сферы впоследствии будут использоваться для обучающей выборки ML-модели.

```

def get_top_spheres(scores):
    if not scores:
        return []
    max_score = max(scores.values())
    return [sphere_id for sphere_id, score in scores.items() if
score == max_score]

```

Помимо явных и неявных вопросов, которые влияют на результат, пользователю необходимо ответить на общие вопросы, которые впоследствии будут использоваться для обучения модели. В отличие от вопросов ЭС, у

каждого общего вопроса есть свои варианты ответа. Пример вопроса: «Ваш возраст?» («до 14 лет»; «15-17 лет»; «18-22 года»; «23-30 лет»; «31-45 лет»; «старше 46 лет»)

Общие вопросы хранятся в таблице `general_questions`.

За получение вопросов для прохождения общих вопросов отвечает функция `get_general_questions` из файла `questions_router.py`.

```
# Возвращает список всех активных общих вопросов
@router.get("/general",
response_model=list[GeneralQuestionResponse])
def get_general_questions(db: Session = Depends(get_db)):
    questions = (
        db.query(GeneralQuestion)
        .filter(GeneralQuestion.is_active == True)
        .all()
    )
    return [
        GeneralQuestionResponse(
            id=q.id_general_questions,
            text=q.text,
            options=q.options
        )
        for q in questions
    ]
```

После того, как пользователь ответил на все общие вопросы, ответы на них сохраняются в таблицу `general_answers`.

Формирование обучающей выборки происходит путем сохранения в таблицу `ml_training_data` ответов на все общие вопросы в виде индексов и целевой сферы (сферы с наивысшим результатом).

Код формирования обучающей выборки:

```
# Получаем все активные общие вопросы из БД
general_questions = db.query(GeneralQuestion).filter(
    GeneralQuestion.is_active == True
```

```

).order_by(GeneralQuestion.id_general_questions).all()
# Формируем список индексов ответов
features_list = []
for gq in general_questions:
    answer_text =
general_answers_dict.get(gq.id_general_questions, "")
    idx = _get_answer_index(gq.id_general_questions, answer_text,
db)
    features_list.append(idx)
# Получаем список сфер с максимальным баллом
top_spheres = get_top_spheres(scores) if scores else [1]
# Сохраняем в обучающую выборку (для каждой сферы-лидера)
for sphere_id in top_spheres:
    db.add(MLTrainingData(
        session_id=session.id_test_sessions,
        features={"answers": features_list},
        target_sphere_id=sphere_id
    ))

```

Таким образом ЭС производит глубокий анализ личности человека, выявляя его профессиональные предрасположенности, опираясь на явные и скрытые вопросы, а также собирает данные для обучения модели благодаря общим вопросам.

4.2 Разработка ML-модели

4.2.1 Алгоритм машинного обучения Random Forest

В качестве алгоритма машинного обучения был выбран метод случайного леса (Random Forest [12, 13, 14]), относящийся к ансамблевым методам [13].

Ансамблевые подходы основаны на объединении нескольких моделей с целью получения комбинированного прогноза, что позволяет создать более мощную и устойчивую модель. Это обеспечивает превосходство их предсказания по точности над одиночными классификаторами благодаря компенсированию индивидуальных погрешностей отдельных моделей.

					ПГУ 1.090301.15.001 ПЗ	Лист
						53
Изм.	Лист	№ докум.	Подпись	Дата		

Применяемый метод Random Forest реализует эту стратегию путем построения ансамбля решающих деревьев. Данный алгоритм предназначен для задач классификации. Деревья обучаются независимо и параллельно на случайной части данных и относят классифицируемый объект к одному из возможных классов. Итоговый результат определяется большинством голосов деревьев, что повышает предсказательную способность и точность модели (рисунок 27) [1].

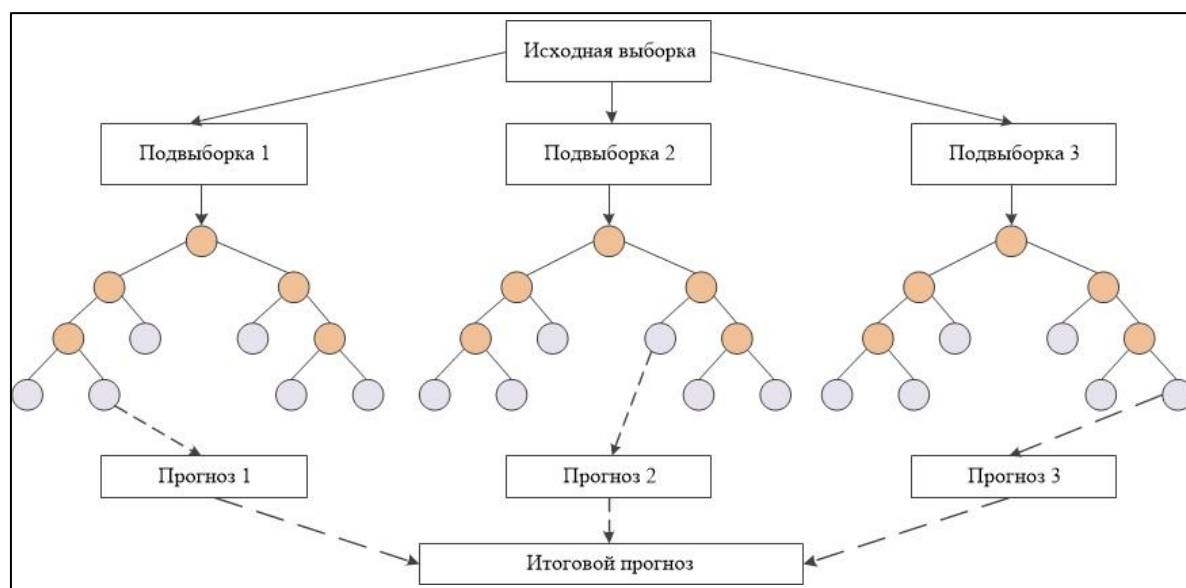


Рисунок 27 – Случайный лес

Метод Random Forest обладает рядом преимуществ.

Во-первых, этот метод не требует масштабирования признаков. Random Forest не нуждается в нормализации данных, потому что деревья решений принимают решения на основе пороговых значений, а не расстояний между точками. Логистическая регрессия и метод опорных векторов (SVM) наоборот требуют предварительной нормализации. Это важно для разрабатываемой системы, потому что ответы на общие вопросы представляют собой индексы вариантов ответов. При этом не у всех вопросов количество этих индексов одинаковое. Отсутствие необходимости масштабирования снижает риск ошибок.

Во-вторых, Random Forest устойчив к переобучению благодаря использованию ансамбля деревьев. Каждое дерево обучается на случайной выборке данных и случайном наборе признаков. Градиентный бустинг сильнее склонен к переобучению на небольшом количестве данных. Устойчивость к обучению очень важна для системы, которая будет постоянно переобучаться на новых данных.

Одним из ключевых преимуществ метода Random Forest является оценка важности признаков. Для разрабатываемой системы важно отслеживать, какие общие вопросы сильнее всего влияют на результат предсказания сферы, что позволит улучшить качество тестирования в будущем. В отличие от Random Forest, SVM и градиентный бустинг в этом плане являются «черными ящиками».

В методе Random Forest каждое дерево обучается на собственной bootstrap sample, то есть каждое дерево берет случайный набор данных из исходного набора путем многократного выбора случайного элемента, возвращая его обратно. При этом каждый элемент может попасться как несколько раз, так и ни разу. Это позволяет Random Forest работать с несбалансированными данными. В контексте данной работы это важно, потому что мы не знаем, сколько пользователей, имеющих направленность к конкретной сфере, будут проходить тест. Логистическая регрессия и SVM чувствительны к несбалансированным данным. То есть при значительном преобладании одного класса эти алгоритмы могут игнорировать более редкие классы.

Random Forest предсказывает результат с высокой скоростью, так как деревья работают параллельно. Время зависит от количества и глубины деревьев. Даже если использовать 100 деревьев, результат будет вычисляться за миллисекунды. Это значительно быстрее градиентного бустинга, где деревья строятся последовательно, и SVM. В контексте веб-приложения скорость

предсказания сильно важна, потому что пользователь не будет ждать больше нескольких секунд.

Таким образом, метод Random Forest сочетает в себе ряд преимуществ, которые делают его наиболее подходящим среди других методов.

4.2.2 Данные для обучения

Данные для обучения модели машинного обучения были собраны в результате прохождения полного тестирования реальными пользователями, а также сгенерированы искусственным путем. Формирование обучающей выборки происходило следующим образом.

Изначально 21 человек прошел полное тестирование: 151 вопрос ЭС и тринадцать общих вопросов. Профессиональная сфера каждого человека определялась экспертной системой с помощью расчета баллов. Таким образом, была сформирована выборка из 21 набора (общих вопросов и целевая сфера).

Для увеличения объема обучающей выборки остальные записи были сформированы вручную. Для каждой сферы ответы на общие вопросы давались так, как мог бы на них ответить человек, склонный к данной сфере. Например, если взять сферу «Медицина и здравоохранение», то ответы как правило показывали интерес к химии или биологии, но отражали индивидуальность: медицинский работник может любить рисовать или играть в шахматы.

В результате была сформирована обучающая выборка из 273 записей и сохранена в таблицу ml_training_data.

Определение оптимальной архитектуры модели:

- вход: тринадцать признаков (ответы на вопросы теста);
- организация алгоритма: 100 параллельных деревьев решений. Каждое дерево имеет полный доступ ко всем признакам, но использует разные подмножества при разбиениях. При каждом разбиении дерева рассматривается $\sqrt{13} \approx 3-4$ случайных признака;

– выход: вероятность каждого класса (профессиональные сферы) и номер класса с максимальной вероятностью.

4.2.3 Обучение модели

Обучение модели реализовано в файле `train_model.py`. Его можно запустить как через командную строку, так и через приложение, используя панель администратора. Для работы с моделью использовалась библиотека Scikit-Learn.

Загрузка данных из таблицы `ml_training_data` и проверка количества записей осуществляется следующим образом:

```
records = db.query(MLTrainingData).all()
print(f"Загружено записей из БД: {len(records)}")

if len(records) < 10:
    print(f"Недостаточно данных для обучения. Нужно минимум 10, сейчас {len(records)}.")
    return None
```

Из каждой записи извлекаются признаки и целевая сфера и добавляются в списки `X` и `y` соответственно. После этого списки преобразуются в массивы NumPy с помощью `np.array()`.

```
X = []
y = []
for record in records:
    features = record.features.get("answers", [])
    X.append([int(f) for f in features])
    y.append(record.target_sphere_id)
X = np.array(X)
y = np.array(y)
```

Далее данные делятся в соотношении 70% на обучающую выборку и 30% на тестовую. Для этого используется функция `train_test_split`.

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
```

)

Создается экземпляр RandomForestClassifier с параметрами: `n_estimators=100` – количество деревьев в лесу и `random_state=42` – функция случайного состояния. Параметр `max_features` берется как значение по умолчанию, что соответствует оптимальной архитектуре.

```
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

На тестовой выборке оценивается точность модели с помощью `score`, в консоли выводятся результаты: точность, количество записей, размер обучающей и тестовой выборки. Затем модель сохраняется в файл.

```
accuracy = model.score(X_test, y_test)
print(f"    Точность на тестовых данных: {accuracy:.2%}")
print(f"    Всего записей: {len(records)}")
print(f"    Обучающая выборка: {len(X_train)}")
print(f"    Тестовая выборка: {len(X_test)}")

model_dir =
os.path.join(os.path.dirname(os.path.abspath(__file__)))
model_path = os.path.join(model_dir,
"random_forest_model.pkl")

with open(model_path, "wb") as f:
    pickle.dump(model, f)
```

Для выявления важности каждого признака используется `feature_importances_`. После этого информация о модели загружается в таблицу `ml_models`.

```
importances = model.feature_importances_.tolist()

print(f"    Важность признаков: {[round(i, 4) for i in
importances]}")

# Определяем все существующие версии моделей, чтобы найти
следующий свободный номер

existing_versions = db.query(MLModel.version).all()
```

```

existing_numbers = set()

for v in existing_versions:

    try:

        # Извлекаем число из строки "vX.0"

        num = int(v[0].replace("v", "").split(".")[0])

        existing_numbers.add(num)

    except (ValueError, IndexError):

        # Если версия имеет нестандартный формат, пропускаем её

        continue

    # Находим минимальное положительное целое число, которого
нет в existing_numbers

    version_num = 1

    while version_num in existing_numbers:

        version_num += 1

    version = f"v{version_num}.0"

    print(f"    Следующая свободная версия: {version}")

    # Деактивируем все предыдущие модели

    db.query(MLModel).update({"is_active": False})

    # Сохраняем запись о модели в БД

    model_record = MLModel(

        version=version,

        model_file=model_path,

        accuracy=float(accuracy),

        is_active=True,

```

```

features_importance=importances

)

db.add(model_record)

db.commit()

```

Результаты обучения модели представлены на рисунке 28.

```

Загружено записей из БД: 273
Размер матрицы признаков: (273, 13)
Количество классов: 15

Модель обучена!
Точность на тестовых данных: 58.54%
Всего записей: 273
Обучающая выборка: 191
Тестовая выборка: 82
Модель сохранена: D:\DIPLOM\DiplomProject\backend\ml\random_forest_model.pkl
Важность признаков: [0.0924, 0.0471, 0.0764, 0.1187, 0.2023, 0.1009, 0.0692,
0.049, 0.0363, 0.071, 0.0516, 0.0445, 0.0404]
Следующая свободная версия: v10.0
Запись о модели сохранена в БД (version=v10.0)

```

Рисунок 28 – Результаты обучения модели

Выявили 3 самых значимых вопроса, которые больше всего влияли на результат: «Какой школьный предмет вам всегда давался легче других?», «По какому предмету у вас были стабильно высокие оценки?», «Чем вы предпочитаете заниматься в свободное время?». Данная информация позволяет в дальнейшем оптимизировать опросник, исключив наименее значимые вопросы без существенной потери точности.

Были получены результаты прохождения полного и короткого теста от одного и того же человека. Сравнив результат ЭС (рисунок 29) с тестированием с помощью модели машинного обучения (рисунок 30), можно заметить, что целевая сфера определена верно. Повысить точность можно за счет корректировки вопросов с точки зрения психологии и использования более разнообразных реальных данных для обучения модели.

Таким образом, система показывает хороший результат и перспективу для дальнейшего развития.

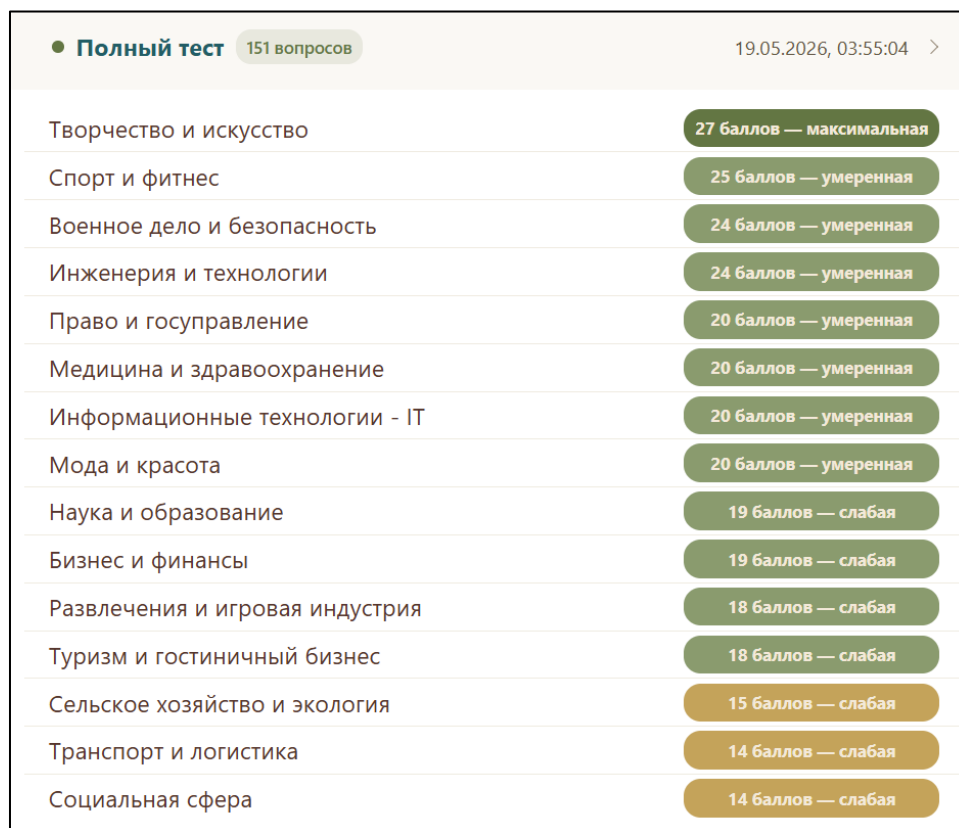


Рисунок 29 – Результат ЭС



Рисунок 30 – Результат ML-модели

4.2.4 Использование модели в приложении

Работа ML-модели описана в файле `ml_service.py`. Функция `load_model()` загружает модель из файла в формате `pickle`.

```
MODEL_PATH = os.path.join(os.path.dirname(os.path.abspath(__file__)), "..", "ml", "random_forest_model.pkl")

def load_model():
    if not os.path.exists(MODEL_PATH):
        print(f"Модель не найдена: {MODEL_PATH}")
        return None
    with open(MODEL_PATH, "rb") as f:
        return pickle.load(f)
```

Функция `_get_answer_index()` преобразует текстовый ответ пользователя в числовой формат для удобства и устранения ошибок при опечатке.

```
def _get_answer_index(question_id: int, answer_value: str) -> int:
    db = SessionLocal()
    try:
        question = db.query(GeneralQuestion).filter(
            GeneralQuestion.id_general_questions == question_id
        ).first()
        if question and question.options:
            try:
                return question.options.index(answer_value) + 1
            except ValueError:
                return 0
        return 0
    finally:
        db.close()
```

Функция `predict` предсказывает вероятности по сферам на основе ответов на общие вопросы. Сначала открывается соединение с БД с помощью `SessionLocal()`, загружаются все активные вопросы и сферы, соединение закрывается через `close()`. Затем осуществляется проверка корректности

данных: список ответов не пуст и количество ответов соответствует количеству вопросов.

```
def predict(general_answers: list):
    db = SessionLocal()
    try:
        # Получаем активные общие вопросы в порядке возрастания ID
        questions = db.query(GeneralQuestion).filter(
            GeneralQuestion.is_active == True
        ).order_by(GeneralQuestion.id_general_questions).all()

        questions_count = len(questions)

        # Получаем все сферы
        spheres =
db.query(Sphere).order_by(Sphere.id_spheres).all()
        sphere_ids = [s.id_spheres for s in spheres]
        sphere_count = len(sphere_ids)
    finally:
        db.close()
    if questions_count == 0:
        print("Нет активных общих вопросов в БД")
        return []

    if sphere_count == 0:
        print("Нет сфер в БД")
        return []

    # Проверяем количество полученных ответов
    if len(general_answers) != questions_count:
        print(f"Предупреждение: получено {len(general_answers)}
ответов, ожидалось {questions_count}")
    # Обрезаем или дополняем до нужного количества
    if len(general_answers) > questions_count:
        general_answers = general_answers[:questions_count]
    else:
```

```

        general_answers.extend([""] * (questions_count -
len(general_answers)))

```

Модель загружается с помощью функции `load_model()`. После чего происходит преобразование вариантов ответов в соответствующие индексы.

```

# Загружаем модель
model = load_model()

# Если модель не загружена, возвращаем равномерное
распределение
if model is None:
    equal_prob = round(100 / sphere_count, 2)
    return [
        {"sphere_id": sphere_id, "probability": equal_prob}
        for sphere_id in sphere_ids
    ]
# Преобразуем ответы в индексы
numeric = []
for i, answer in enumerate(general_answers):
    question = questions[i]
    idx = _get_answer_index(question.id_general_questions,
answer)
    numeric.append(idx)

```

После преобразования ответов в список индексов выполняется новое преобразование в массив NumPy с помощью функции `np.array([numeric])`. Вызывается метод `predict_proba(X)` обученной модели и возвращает массив вероятностей для каждого класса, соответствующего одной из сфер. Результат переводится в проценты и сортируется по убыванию.

```

# Выполняем предсказание
X = np.array([numeric])
probabilities = model.predict_proba(X)[0]

# Сопоставляем классы модели с ID сфер
result = []

```

```

for cls, prob in zip(model.classes_, probabilities):
    result.append({
        "sphere_id": int(cls),
        "probability": round(prob * 100, 2)
    })

# Сортируем по убыванию вероятности
result.sort(key=lambda x: x["probability"], reverse=True)
return result

```

Таким образом, файл `ml_service.py` реализует работу с моделью Random Forest.

4.3 Разработка REST API

Для обеспечения взаимодействия между клиентской и серверной частями используются REST API-запросы. Все запросы описаны в четырех файлах: `auth_router.py`, `questions_router.py`, `test_router.py` и `admin_router.py`.

4.3.1 Роутер авторизации (`auth_router.py`)

В файле `auth_router.py` содержится два маршрута, связанных с авторизацией и регистрацией пользователей.

POST `/auth/register` (функция `register`). Запрос осуществляется при регистрации нового пользователя. Функция проверяет, существует ли пользователь с таким логином в базе данных. Если существует – возвращает ошибку, если нет – создает нового пользователя и сохраняет в базу данных.

POST `/auth/login` (функция `login`). Запрос осуществляется при входе пользователя в систему. Функция ищет логин пользователя в базе данных по логину. Если пользователь не найден или пароль не совпадает, возвращается ошибка. При совпадении функция возвращает `id` пользователя, логин и роль.

4.3.2 Роутер вопросов (`questions_router.py`)

Файл `questions_router.py` содержит маршруты для получения вопросов тестирования.

GET /questions/expert (функция get_expert_questions). Запрос возвращает список всех активных вопросов ЭС. Из базы данных для каждого вопроса возвращаются его id, текст, тип и список связанных сфер.

GET /questions/general (функция get_general_questions). Запрос возвращает список всех активных общих вопросов. Для каждого вопроса возвращаются его id, текст и массив ответов в формате JSON.

GET /questions/info (функция get_expert_info). Запрос возвращает информацию о системе: количество активных вопросов, максимально возможный балл за сферу.

4.3.3 Роутер тестирования (test_router.py)

Файл test_router.py содержит маршруты для приема ответов пользователей и возвратов результатов тестирования.

POST /test/full (функция submit_full_test). Запрос обрабатывает прохождение полного тестирования. Функция принимает ответы на вопросы ЭС и на общие вопросы, сохраняет их в БД. С помощью функции calculate_scores() вычисляются баллы по каждой сфере, а с помощью функции get_inclination_results() определяется уровень склонности для каждой. Результаты теста сохраняются в таблицу full_test_results. Затем из ответов на общие вопросы и целевой сферы (определяется функцией get_top_spheres()) формируется обучающая выборка и сохраняется в таблицу ml_training_data. Функции calculate_scores(), get_inclination_results() и get_top_spheres() находятся в файле expert_system.py. Клиенту возвращается список сфер с баллами и склонностями.

POST /test/short (функция submit_short_test). Запрос обрабатывает прохождение короткого тестирования. Функция принимает ответы на общие вопросы и передает их в функцию predict() из файла ml_service.py, которая загружает обученную модель Random Forest и возвращает вероятность принадлежности к каждой сфере. Результаты сохраняются в таблицу short_test_results и возвращаются клиенту.

GET /test/history (функция get_test_history). Запрос возвращает историю всех тестов, которые проходил текущий пользователь. Для каждого теста возвращается его тип (длинный или короткий), дата и время прохождения и результаты.

4.3.4 Роутер администрирования (admin_router.py)

Файл admin_router.py содержит маршруты управления системой. Они доступны только пользователям с ролью admin.

GET /admin/stats (функция get_stats). Запрос возвращает общую статистику: суммарное количество тестов, количество полных и коротких тестов, количество пользователей и список последних пяти пройденных тестов.

POST /admin/train_model (функция train_model_endpoint). Запрос запускает переобучение ML-модели.

Помимо этих запросов в роутере прописаны запросы для управления сферами, вопросами ЭС и общими вопросами.

Запросы, управляющие профессиональными сферами.

GET /admin/spheres – получение списка сфер.

POST /admin/spheres – создание новой сферы.

PUT /admin/spheres/{id} – редактирование сферы.

DELETE /admin/spheres/{id} – удаление сферы. Если к сфере привязаны вопросы – удаление запрещено.

Запросы, управляющие вопросами ЭС.

GET /admin/questions – получение списка вопросов.

POST /admin/questions – создание нового вопроса.

PUT /admin/questions /{id} – редактирование вопроса.

DELETE /admin/questions /{id} – удаление вопроса.

PATCH /admin/ questions/{id}/toggle – изменение флага активности вопроса.

Запросы, управляющие общими вопросами.

GET /admin/general-questions – получение списка вопросов.

POST /admin/ general-questions – создание нового вопроса.

PUT /admin/ general-questions/{id} – редактирование вопроса.

DELETE /admin/ general-questions/{id} – удаление вопроса.

Таким образом, роутеры обеспечивают возможность взаимодействия клиента с сервером.

4.4 Разработка клиентской части

Клиентская часть включает в себя несколько ключевых элементов. Index.html – главный HTML-файл. Файл main.jsx – точка входа: инициализирует и монтирует корневой компонент. Файл App.jsx – реализует маршрутизацию между страницами. Файл api.js – HTTP-клиент для отправки запросов серверу, который автоматически добавляет к запросу ID пользователя. В файл global.css вынесены глобальные стили. Компоненты страниц расположены в папке pages.

4.4.1 Компонент LoginPage.jsx

Страница LoginPage.jsx содержит форму входа в приложение с полями для ввода логина и пароля и кнопкой. При отправке формы посылается POST-запрос /auth/login. При успехе ID и роль пользователя сохраняются в localStorage и открывается окно меню. При ошибке выводится соответствующее сообщение.

```
const handleSubmit = async (e) => {
    e.preventDefault();
    setError('');
    try {
        const response = await api.post('/auth/login', {
login, password });
        localStorage.setItem('role', response.data.user?.role
|| 'user');
        localStorage.setItem('userId',
response.data.user?.id);
        navigate('/menu');
```

```

    } catch (err) {
        setError(err.response?.data?.detail || 'Ошибка
входа');
    }
};

```

4.4.2 Компонент RegisterPage.jsx

Страница RegisterPage.jsx содержит форму регистрации в приложении с полями для ввода логина и пароля и кнопкой. При отправке формы посылается POST-запрос /auth/register. При успехе пользователя перенаправляют на страницу входа. При ошибке выводится соответствующее сообщение.

```

const handleSubmit = async (e) => {
    e.preventDefault();
    setError('');
    setSuccess('');
    try {
        await api.post('/auth/register', { login, password });
        setSuccess('Регистрация успешна! Перенаправляем на
вход...');
        setTimeout(() => navigate('/login'), 1500);
    } catch (err) {
        setError(err.response?.data?.detail || 'Ошибка
регистрации');
    }
};

```

4.4.3 Компонент MenuPage.jsx

Главное меню содержит две карточки для выбора тестирования (полный и короткий) и кнопки «История тестов», «Админ-панель» и «Выйти». При этом кнопка «Админ-панель» отображается только администраторам.

При загрузке страницы выполняется функция getInfo() из файла api.js, которая выполняет GET-запрос /questions/info для получения количества вопросов и максимального балла. Это необходимо для динамического

отображения информации. Полученные данные сохраняются с помощью функции `setInfo`.

```
useEffect(() => {
  const loadInfo = async () => {
    try {
      const data = await getInfo();
      setInfo({
        ...data,
        loading: false
      });
    } catch (err) {
      console.error('Ошибка загрузки info:', err);
      setInfo(prev => ({ ...prev, loading: false }));
    }
  };
  loadInfo();
}, []);
```

4.4.4 Компонент `FullTestPage.jsx`

При прохождении полного теста пользователь сначала отвечает на вопросы ЭС, а затем на общие вопросы. Для ответов на вопросы экспертного тестирования предусмотрены кнопки от 1 до 5, а также кнопки, позволяющие перемещаться между вопросами. Для ответов на общие вопросы предоставлен список из вариантов ответа.

Вопросы загружаются через GET-запрос `/questions/expert` и `/questions/general`. Ответы сохраняются в состоянии компонента. После завершения всех вопросов отправляется POST-запрос `/test/full` и пользователь перенаправляется на страницу результатов.

Для удобства была реализована поддержка клавиш 1-5 для выбора варианта ответа и стрелок влево и вправо для перемещения между вопросами. При этом переход к следующему ответу возможен только после ответа на предыдущий.

```

useEffect(() => {
  const handleKeyDown = (e) => {
    const key = e.key;
    if (key >= '1' && key <= '5') {
      const num = parseInt(key);
      if (stage === 'expert' && questions[current]) {
        handleSelect(questions[current].id, num);
        if (current < questions.length - 1) {
          setTimeout(() => setCurrent(current + 1),
150);
        }
      }
    }
    if (key === 'ArrowLeft' && current > 0)
setCurrent(current - 1);
    if (key === 'ArrowRight') {
      if (stage === 'expert' && current <
questions.length - 1 && answers[questions[current]?.id]) {
        setCurrent(current + 1);
      } else if (stage === 'general' && current <
generalQuestions.length - 1 &&
generalAnswers[generalQuestions[current]?.id]) {
        setCurrent(current + 1);
      }
    }
  };
  window.addEventListener('keydown', handleKeyDown);
  return () => window.removeEventListener('keydown',
handleKeyDown);
}, [current, stage, questions, generalQuestions, answers,
generalAnswers]);

```

4.4.5 Компонент ShortTestPage.jsx

Эта страница отвечает за прохождение короткого тестирования. Вопросы загружаются через GET-запрос /questions/general.

					ПГУ 1.090301.15.001 ПЗ	Лист
						71
Изм.	Лист	№ докум.	Подпись	Дата		

После ответов на все вопросы функция `handleSubmit` формирует массив из ответов пользователя и отправляет POST-запрос `/test/short`. Пользователь перенаправляется на страницу результатов.

```
const handleSubmit = async () => {
  setSubmitting(true);
  try {
    const general_answers =
Object.entries(answers).map(([question_id, answer_value]) => ({
  question_id: parseInt(question_id),
  answer_value,
}));
    const response = await api.post('/test/short', {
general_answers });
    navigate('/result/short', { state: { results:
response.data.results } });
  } catch (err) {
    setError('Ошибка отправки ответов');
  } finally {
    setSubmitting(false);
  }
}
```

4.4.6 Компонент ResultPage.jsx

`ResultPage.jsx` отображает результаты тестирования в виде диаграммы. Для полного тестирования результаты отображаются в баллах, а для короткого – в процентах вероятности. Результаты сортируются по убыванию и делаются нужными цветами. Цвет зависит от процента набранного результата. При этом для полного теста процент вычисляется относительно максимального балла для каждой сферы. А для короткого теста процент вычисляется относительно максимальной вероятности среди всех сфер, что позволяет показать, насколько каждая сфера близка к лидеру.

```
// Цвет для полного теста (по проценту баллов)
const getScoreColor = (score, maxPossibleScore) => {
```

```

    const percent = (score / maxPossibleScore) * 100;
    if (percent <= 25) return '#c47a5a';
    if (percent <= 50) return '#c4a35a';
    if (percent <= 75) return '#8a9b6e';
    return '#637643';
};

// Цвет для короткого теста (по доле относительно максимальной
вероятности)

const getProbabilityColor = (probability, maxProbability) => {
    const percent = (probability / maxProbability) * 100;
    if (percent <= 25) return '#c47a5a';
    if (percent <= 50) return '#c4a35a';
    if (percent <= 75) return '#8a9b6e';
    return '#637643';
};

```

4.4.7 Компонент HistoryPage.jsx

Страница HistoryPage.jsx отображает список всех тестов, пройденных пользователем. Данные загружаются через GET-запрос /test/history. При нажатии на конкретный тест показывается полная информация.

```

const loadHistory = async () => {
    try {
        const response = await api.get('/test/history');
        setHistory(response.data);
    } catch (err) {
        setError('Ошибка загрузки истории');
    } finally {
        setLoading(false);
    }
};

```

4.4.8 Компонент AdminPage.jsx

На страницу администрации может зайти только пользователь с ролью admin. Она содержит вкладки «Статистика», «Сферы», «Вопросы» и «Общие вопросы».

Во вкладке «Статистика» отображается общая статистика (количество пройденных тестов и пользователей), последние пройденные тесты и кнопка «Обучить модель». Нажатие этой кнопки запускает переобучение модели машинного обучения через POST-запрос /admin/train-model.

```
const handleTrainModel = async () => {
    if (!confirm('Запустить обучение модели на текущих
данных?')) return;
    setTraining(true);
    setError('');
    try {
        const response = await api.post('/admin/train-model');
        showSuccess(`Модель обучена! Точность:
${(response.data.accuracy * 100).toFixed(1)}%`);
    } catch (err) {
        setError(err.response?.data?.detail || 'Ошибка
обучения');
    } finally {
        setTraining(false);
    }
};
```

Во вкладках «Сферы», «Вопросы» и «Общие вопросы» содержатся таблицы, в которых можно редактировать соответствующую информацию. Строки таблицы можно добавлять, изменять и удалять.

5 Руководство пользователя

Для начала работы в программе необходимо авторизоваться в системе. Для этого необходимо ввести данные в поля «Логин» и «Пароль», а затем нажать кнопку «Войти». Окно авторизации представлено на рисунке 31.

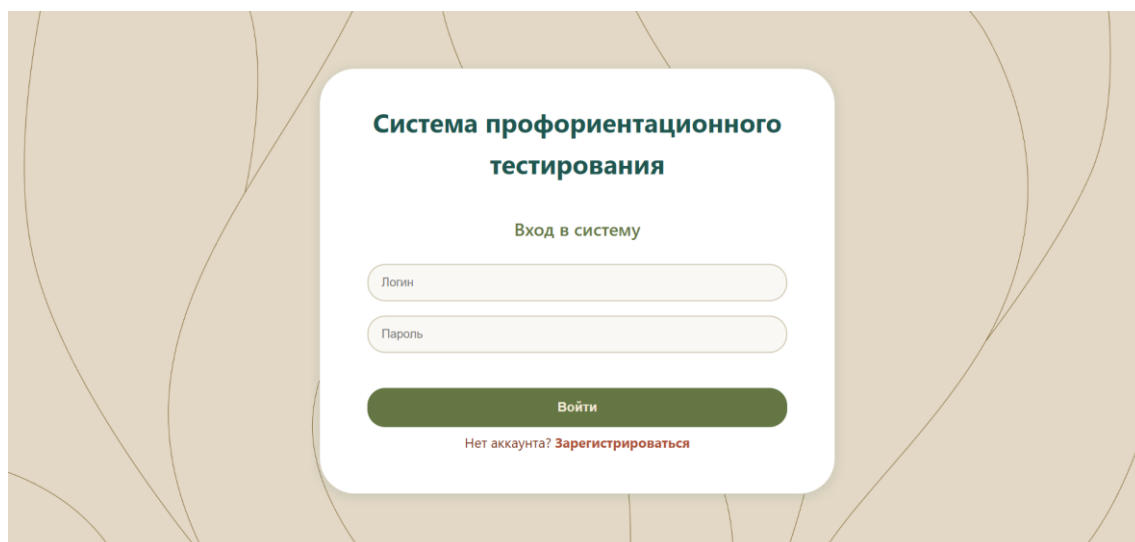


Рисунок 31 – Окно авторизации

Если при нажатии кнопки «Вход» хотя бы одно поле не заполнено – появится сообщение об ошибке. Если логин или пароль введены неверно, то также выводится сообщение об ошибке.

При успешной авторизации пользователь перенаправляется в главное меню.

Если у пользователя нет учетной записи, ему необходимо зарегистрироваться. Для этого необходимо нажать на ссылку «Зарегистрироваться». Откроется форма регистрации (рисунок 32), в которой также нужно заполнить поля «Логин» и «Пароль».

Логин должен быть уникальным, а пароль должен состоять более, чем из 6 символов. После заполнения всех полей необходимо нажать кнопку «Зарегистрироваться». При успешной регистрации пользователь перенаправляется на страницу входа, иначе выведется сообщение об ошибке.

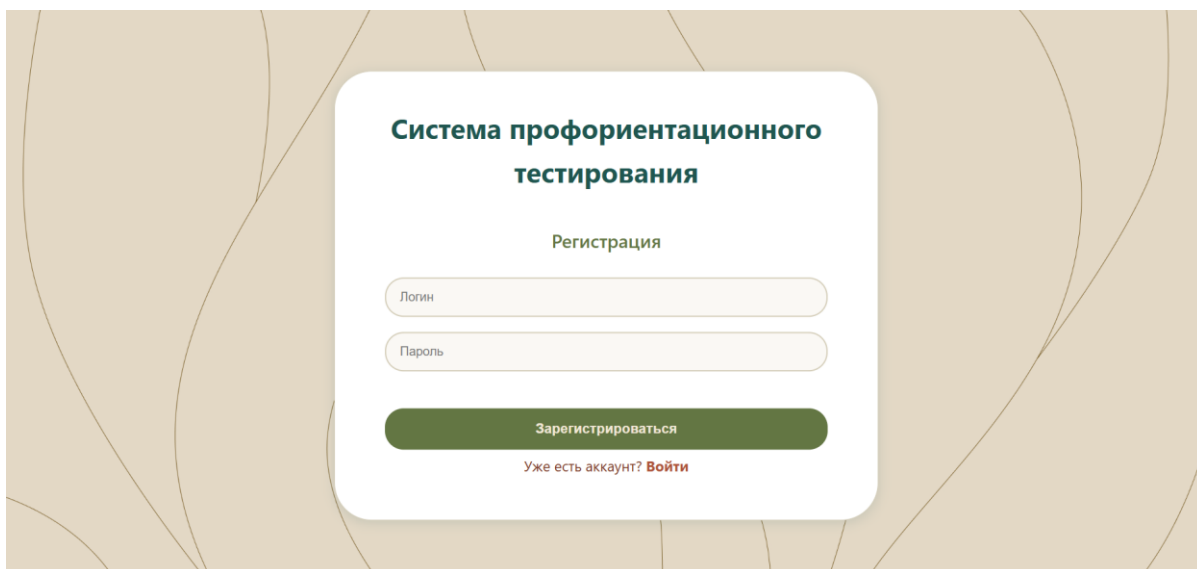


Рисунок 32 – Окно регистрации

После успешного входа в систему открывается главное меню программы. Если роль пользователя определена как admin, то в окне меню есть дополнительная кнопка (рисунок 33), которая позволяет зайти в панель администратора. Иначе кнопка «Админ-панель» не отображается (рисунок 34). Также в меню предусмотрена кнопка выхода из системы.

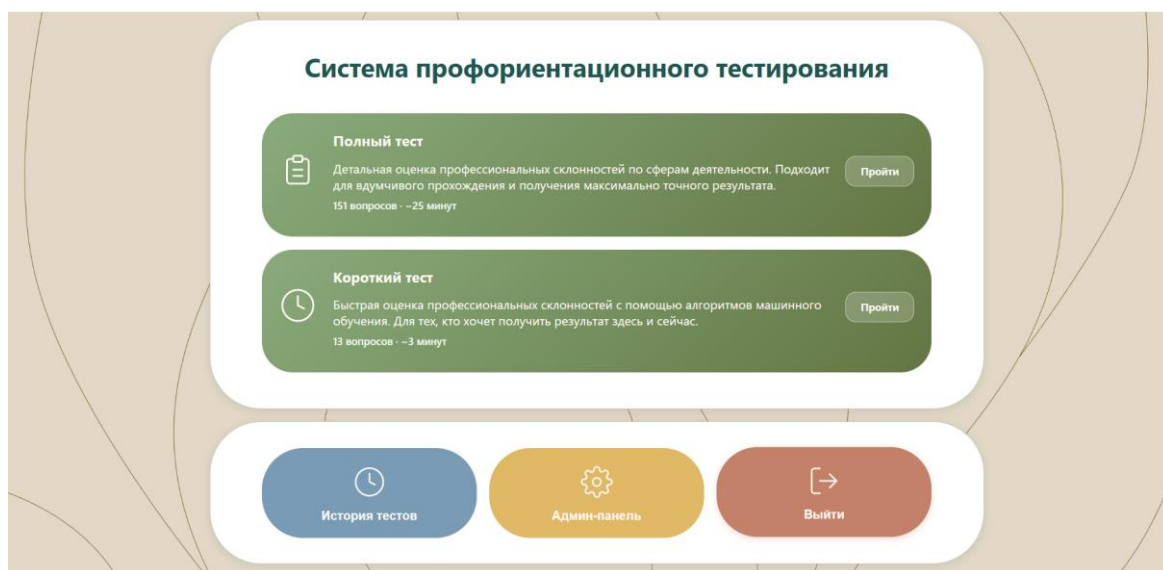


Рисунок 33 – Окно меню

При выборе полного тестирования пользователь последовательно отвечает на вопросы экспертной системы (рисунок 35).

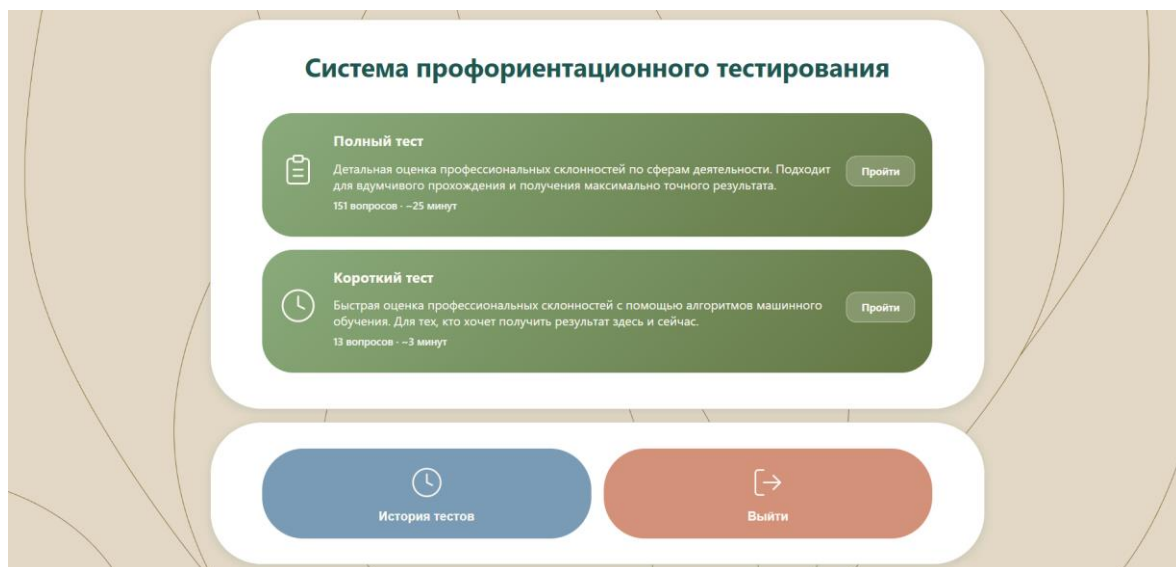


Рисунок 34 – Окно меню

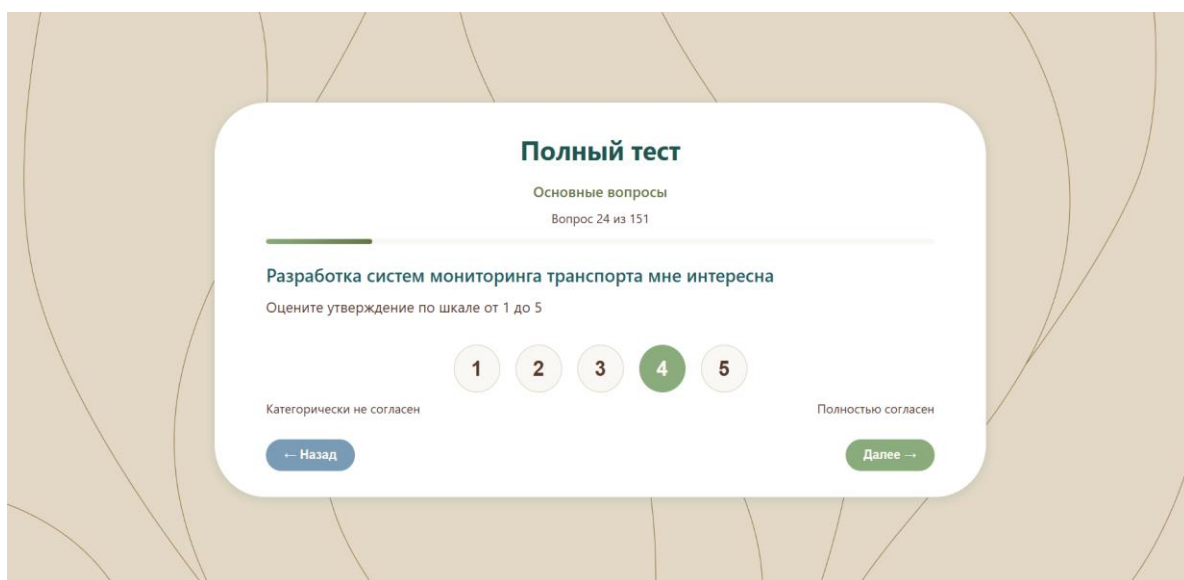


Рисунок 35 – Окно вопросов ЭС

Каждый вопрос оценивается по шкале от 1 до 5, где 1 – категорически не согласен, а 5 – полностью согласен. По мере прохождения теста заполняется шкала прогресса.

После ответа на вопросы ЭС пользователю предлагается ответить на общие вопросы (рисунок 36). Здесь пользователю необходимо выбрать один из вариантов ответа.

Рисунок 36 – Окно общих вопросов

После ответов на все вопросы и нажатия кнопки «Завершить тест» система обрабатывает ответы и перенаправляет пользователя на страницу с результатами (рисунок 37). Результаты отображаются в виде диаграммы с указанием уровня склонности.



Рисунок 37 – Окно результатов полного теста

При выборе короткого тестирования пользователь отвечает только на общие вопросы (рисунок 36) и нажимает «Завершить тест». Система с

помощью модели машинного обучения предсказывает наиболее подходящую сферу для пользователя на основе его ответов (рисунок 38).

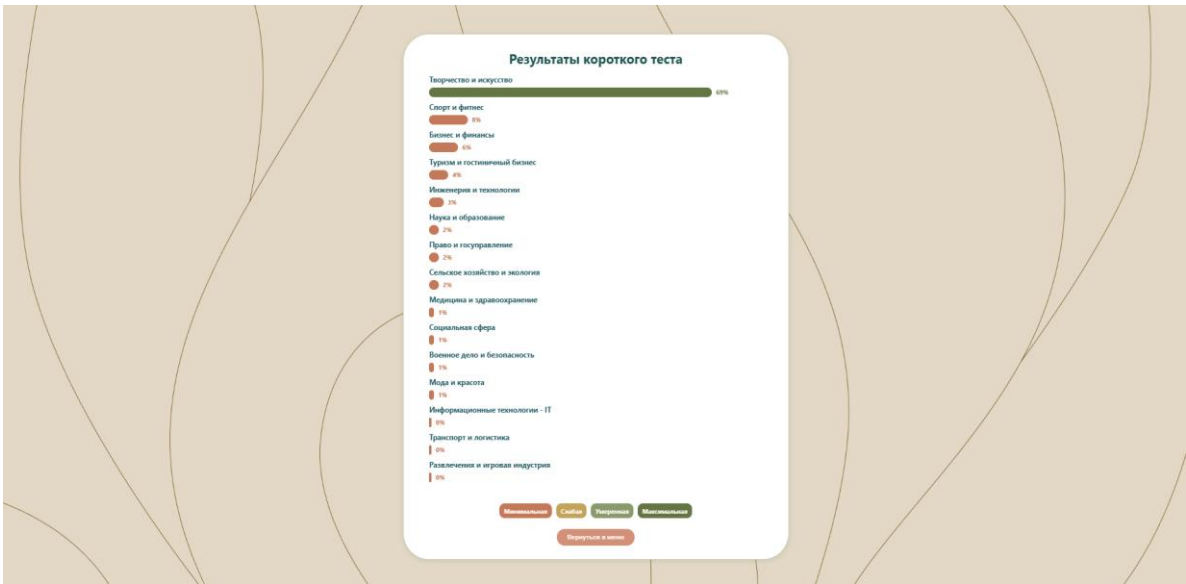


Рисунок 38 – Окно результатов короткого теста

В разделе «История тестов» (рисунок 39) пользователь может посмотреть список пройденных им тестов. При нажатии на конкретный тест можно получить информацию о его результатах.

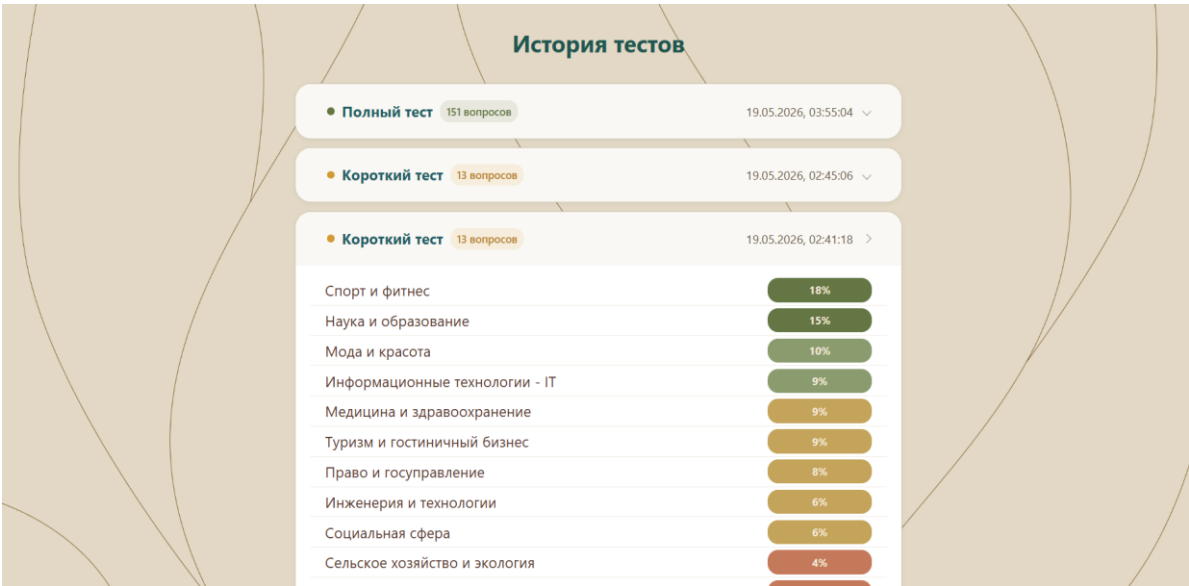


Рисунок 39 – Окно истории тестов

Панель администратора содержит несколько вкладок: «Статистика» (рисунок 40), «Сферы» (рисунок 41), «Вопросы» (рисунок 42), «Общие вопросы» (рисунок 43).

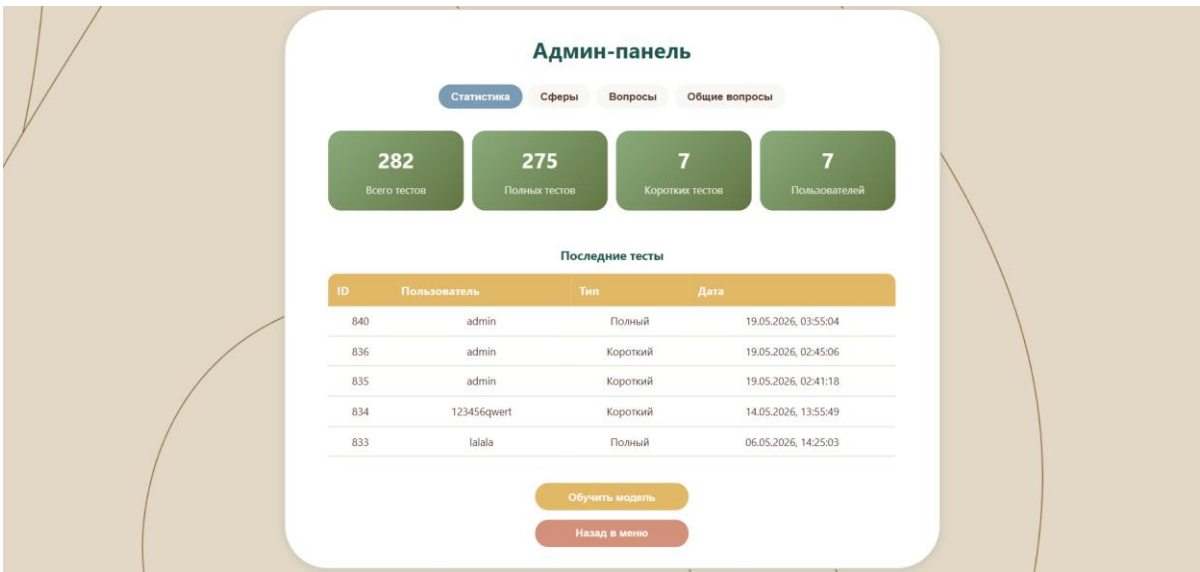


Рисунок 40 – Вкладка «Статистика»

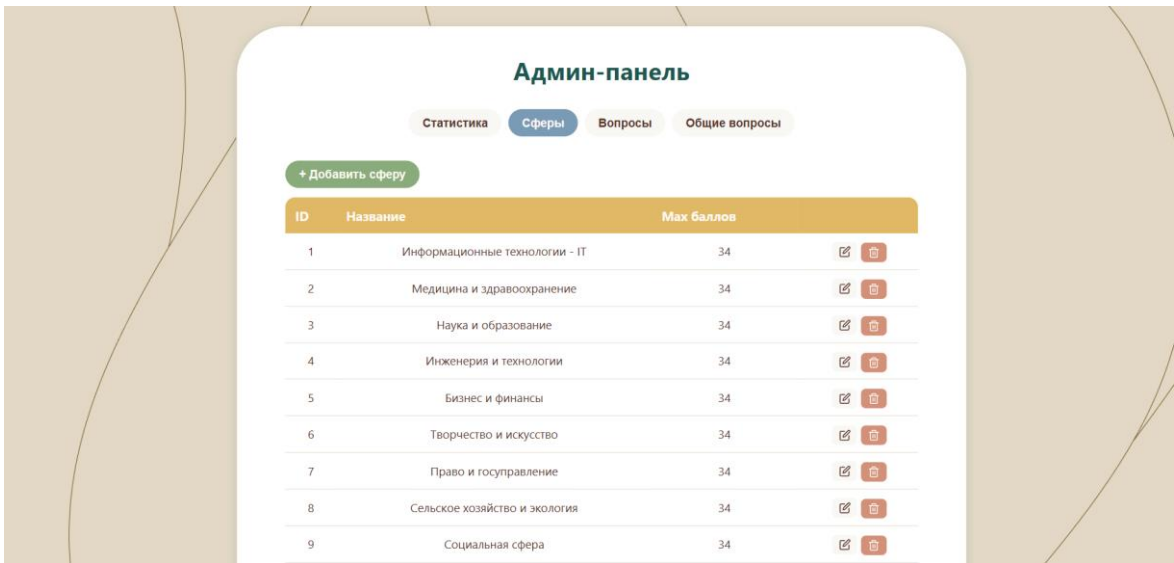


Рисунок 41 – Вкладка «Сферы»

Во вкладке «Статистика» отображается количество пройденных тестов, количество зарегистрированных пользователей, список последних пройденных тестов и кнопка «Обучить модель».

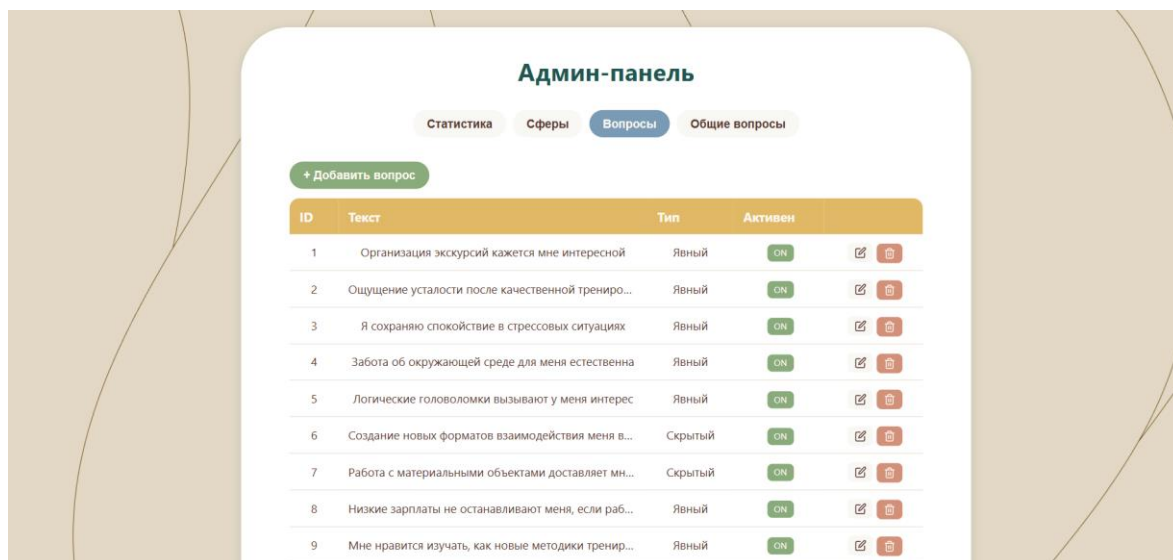


Рисунок 42 – Вкладка «Вопросы»

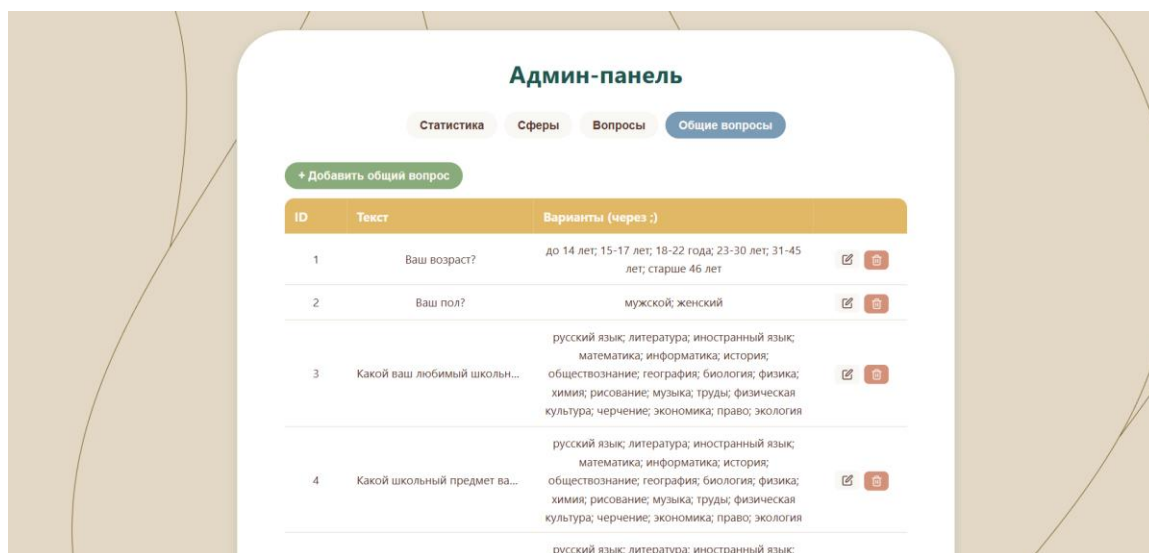


Рисунок 43 – Вкладка «Общие вопросы»

Во вкладке «Сферы» представлен список профориентационных сфер. Каждую сферу можно отредактировать, удалить или добавить новую. Если к сфере привязан хотя бы один вопрос, то система запрещает удаление.

Во вкладке «Вопросы» отображается список всех вопросов ЭС. Каждый из них можно добавить, изменить или временно отключить. Также можно добавить новый вопрос.

Во вкладке «Общие вопросы» содержится список вопросов для машинного обучения. Из также можно изменять, удалять или создавать новые.

6 Охрана труда

Любая деятельность, связанная с эксплуатацией персональных компьютеров, должна осуществляться с соблюдением санитарно-гигиенических и эргономических норм [15]. Ниже представлены основные требования, предъявляемые к организации рабочего места и режиму труда.

При расстановке рабочих мест с ПЭВМ необходимо соблюдать расстояние между рабочими столами:

- между тыльными сторонами соседних мониторов – 2,0 м;
- между их боковыми поверхностями – 1,2 м.

Экран монитора рекомендуется располагать на расстоянии 60-70 см от глаз пользователя. Клавиатура следует размещать на расстоянии 10-30 см от края столешницы.

Высота рабочей поверхности стола рекомендуется 725 мм, пространство для ног должно иметь ширину – не менее 500 мм. При необходимости используется подставка для ног шириной от 300 мм и глубиной от 400 мм. Рабочее кресло должно быть подъемно-поворотным, с регулировкой высоты сиденья (400–550 мм) и угла наклона спинки. Сиденье и спинка изготавливаются из полумягкого, воздухопроницаемого материала.

Освещённость на поверхности стола должна составлять 300–500 люкс, при этом не допускается образование бликов на экране. Яркость белого поля монитора должна быть не менее 35 кд/м², контрастность изображения – не менее 3:1. Частота обновления экрана для жидкокристаллических дисплеев должна составлять не менее 60 Гц.

В процессе работы оборудования не должны превышать следующие уровни: напряжённость электрического поля (5 Гц – 2 кГц) – 25 В/м, плотность магнитного потока – 250 нТл, напряжённость электростатического поля – 15 кВ/м, эквивалентный уровень звука – 50 дБА.

					ПГУ 1.090301.15.001 ПЗ	Лист
						82
Изм.	Лист	№ докум.	Подпись	Дата		

В зависимости от вида деятельности (считывание информации, ввод данных, творческая работа) и нагрузки за смену устанавливаются категории тяжести работ I, II или III, для которых предусмотрены регламентированные перерывы. При 8-часовой смене их продолжительность составляет от 50 до 90 минут, при 12-часовой смене – от 80 до 140 минут.

При обнаружении неисправностей (повреждённая изоляция, запах гари, посторонние шумы) необходимо немедленно отключить питание и сообщить руководителю. В случае возгорания следует обесточить оборудование, вызвать пожарную охрану и приступить к тушению имеющимися средствами. При поражении электрическим током – отключить питание и оказать первую помощь пострадавшему.

После окончания работы оператор обязан корректно закрыть все программы, отключить питание в установленной последовательности и привести рабочее место в порядок.

Заключение

В ходе выполнения выпускной квалификационной работы была разработана система профориентационного тестирования, которая объединяет в себе подход на основе экспертной системы и метод машинного обучения.

Был проведен анализ существующих аналогов, выявлены достоинства и недостатки изученных систем. Опираясь на это, были сформулированы требования к разрабатываемой системе, где самые основные: обеспечение высокой точности результата за счет детального тестирования и высокой скорости получения результата за счет применения модели машинного обучения.

Были выполнены следующие задачи:

- разработана ЭС, оценивающая склонности по сферам деятельности с помощью балльной шкалы;
- на основе данных, полученных в результате прохождения полного теста, была обучена модель машинного обучения Random Forest, способная предсказывать наиболее подходящую сферу с помощью общих вопросов;
- реализовано веб-приложение, обеспечивающее регистрацию, авторизацию, прохождение полного и короткого тестов, отображение результатов в виде диаграммы, историю тестирования и административную панель.

В настоящее время модель обучается только на сфере, имеющий максимальный балл, поэтому результатом является только одна профессиональная сфера. Это позволяет системе определять основное направление склонностей пользователя.

Перспективой для дальнейшего развития системы является обучение отдельных моделей для каждой позиции в рейтинге сфер, то есть модель для определения первого места в рейтинге, модель для второго места, для третьего и так далее. Такой подход позволит получить не только основную направленность человека, но и полную градацию предпочтений человека.

Помимо этого, полное тестирование можно использовать для удобного сбора данных от реальных пользователей, в следствии чего повысится точность модели. Дополнительным улучшением является корректировка вопросов с точки зрения психологии.

Разработанная система полностью готова к эксплуатации и может быть использована для проведения профориентационных тестов. Система позволяет проводить как детальную диагностику через полный тест, так и получать быстрый результат через короткий тест.

					ПГУ 1.090301.15.001 ПЗ	Лист
						85
Изм.	Лист	№ докум.	Подпись	Дата		

Список использованных источников

1. Машинное обучение при создании профориентационных цифровых платформ на основе экспертного опроса, когнитивных стилей познавательной деятельности и метода «Random Forest» / Н.С. Карамышева, А.А. Зубриянова, В.И. Кондратьева, С.А. Зинкин // Известия Юго-Западного государственного университета. Т. 30. №2. 2026.

2. Афанасьев В.В., Куницына С.М., Нечаев М.П., Фролова С.Л. Система профориентации: проблемы, тенденции, опыт реализации и перспективы развития // Обучение и воспитание: теория и методика. 2019. С. 6-16.

3. Забокрицкая Л.Д., Орешкина Т.А., Обабков И.Н., Чепуров Е.Г. Применение алгоритма машинного обучения для профориентации абитуриентов высшего учебного заведения // Вестник Томского государственного университета. 2022. № 485. С. 217–225. doi: 10.17223/15617793/485/24.

4. Тарасова А. С., Комарова Э. П. Проектирование рекомендательной системы профориентации школьников средствами машинного обучения // Гуманитарные исследования. Педагогика и психология. 2024. № 17. С. 7-14.

5. Милько М.А. Преимущества выбора веб-приложений в разработке // Научный аспект. — 2024. — № 3. — Т. 1: Информационные технологии. — С. 98–103.

6. Давыдов А.В., Жусупова А.К., Салыкова О.С. Сравнение различных языков программирования, применяемых в машинном обучении // Вестник науки. 2023. № 2 (59). Т. 3. С. 155-164.

7. Худницкий А.В., Шауро Е.А. Сравнительный анализ FastAPI, Django и Flask: производительность, удобство разработки и выбор оптимального фреймворка для современных веб-приложений // Компьютерные системы и сети : материалы 61-й научной конференции аспирантов, магистрантов и студентов, Минск, 22–26 апреля 2025 г. / Белорусский государственный университет информатики и радиоэлектроники. — Минск, 2025. — С. 412–415.

					ПГУ 1.090301.15.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		86

8. Кантор И. Современный учебник JavaScript [Электронный ресурс] – URL: <https://learn.javascript.ru>. (дата обращения: 15.04.2026)

9. Кулдасевич Д. Р., Кондратёнок Е. В. Современные технологии в веб-разработке на примере «React» // Информационные технологии в образовании, науке и производстве : материалы XI межд. науч.-техн. конф. – Минск: БНТУ, 2024. – С. 188-194.

10. Васильева К. Н., Хусаинова Г. Я. Реляционные базы данных // Colloquium-Journal. — 2019. — № 28-2(52). — С. 23-24. DOI: 10.24411/2520-6990-2019-11217.

11. Бахтин И.В. Главные принципы MVC и смысл использования в разработке программных продуктов // Форум молодых учёных. – 2020. – № 1(41) – С. 63-64.

12. Демиденко А. Случайный лес: Как приручить одну из самых мощных ML-моделей. – СелфПаб, 2025. – 80 с.

13. Митяков, Е. С. Искусственный интеллект и машинное обучение : учебное пособие для вузов / Е. С. Митяков, А. Г. Шмелева, А. И. Ладынин. — 2-е изд., стер. — Санкт-Петербург : Лань, 2026. — 252с.

14. Breiman L. Random Forests // Machine Learning. 2001. Vol. 45. No. 1. P. 5–32.

15. ИОТ № 146. Инструкция по охране труда при работе на компьютере [Электронный ресурс]. – Режим доступа: [ohrana-truda_16554551881584617552.pdf](#). (дата обращения: 19.05.2026).

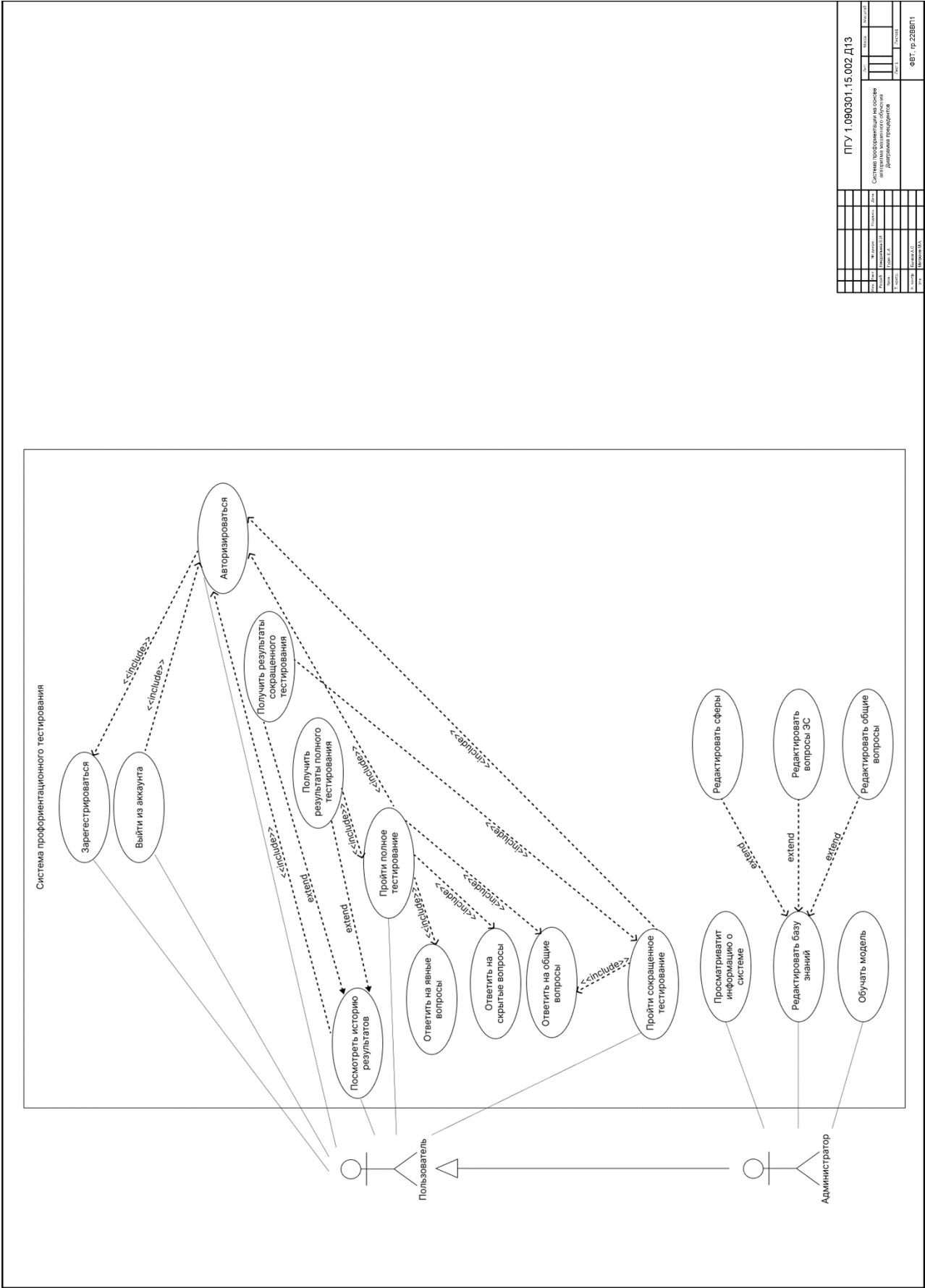
Приложение А
(обязательное)

Система профориентации на основе алгоритма машинного обучения

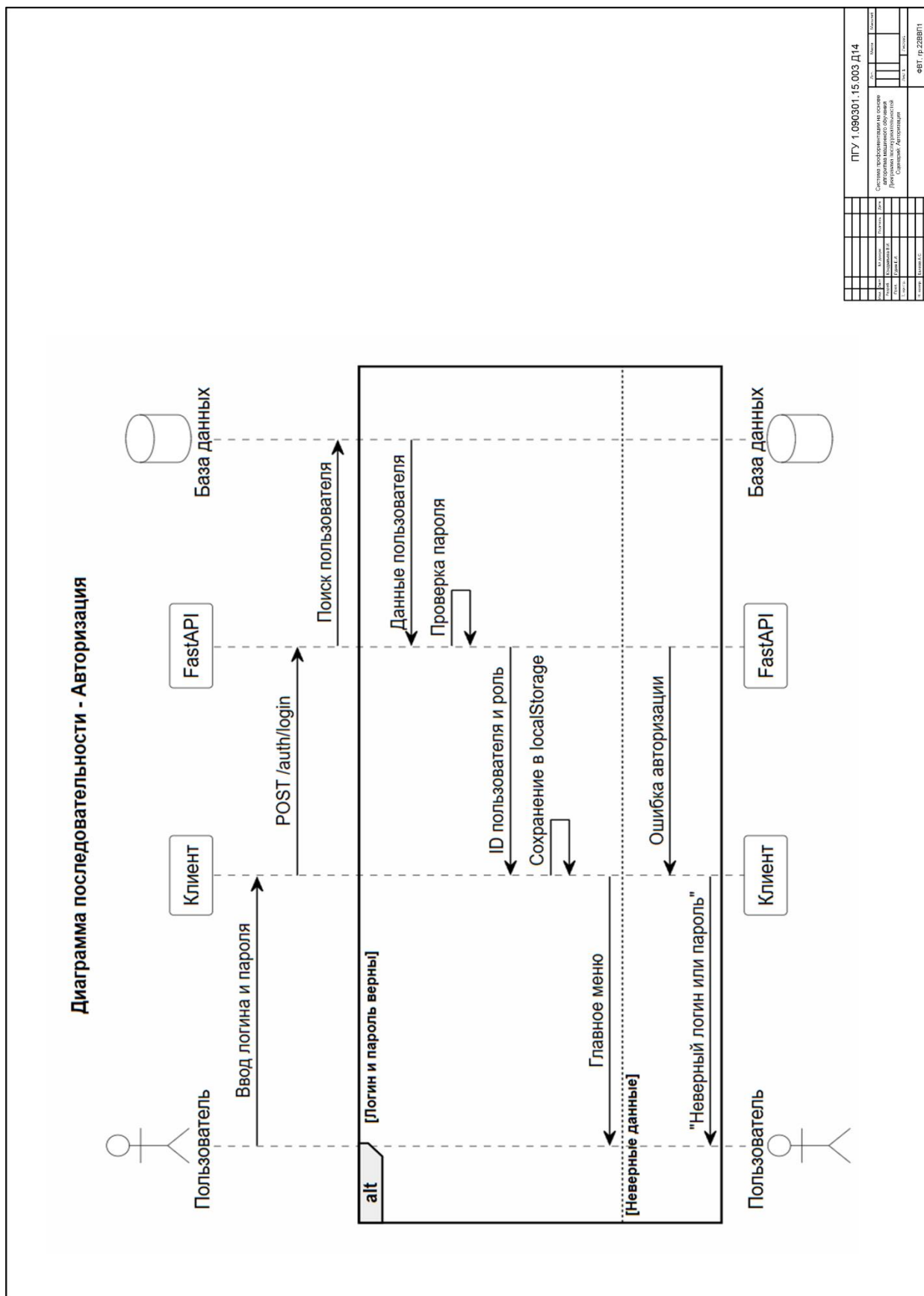
UML диаграммы

					ПГУ 1.090301.15.001 ПЗ	Лист
						88
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.1 – UML – диаграмма прецедентов

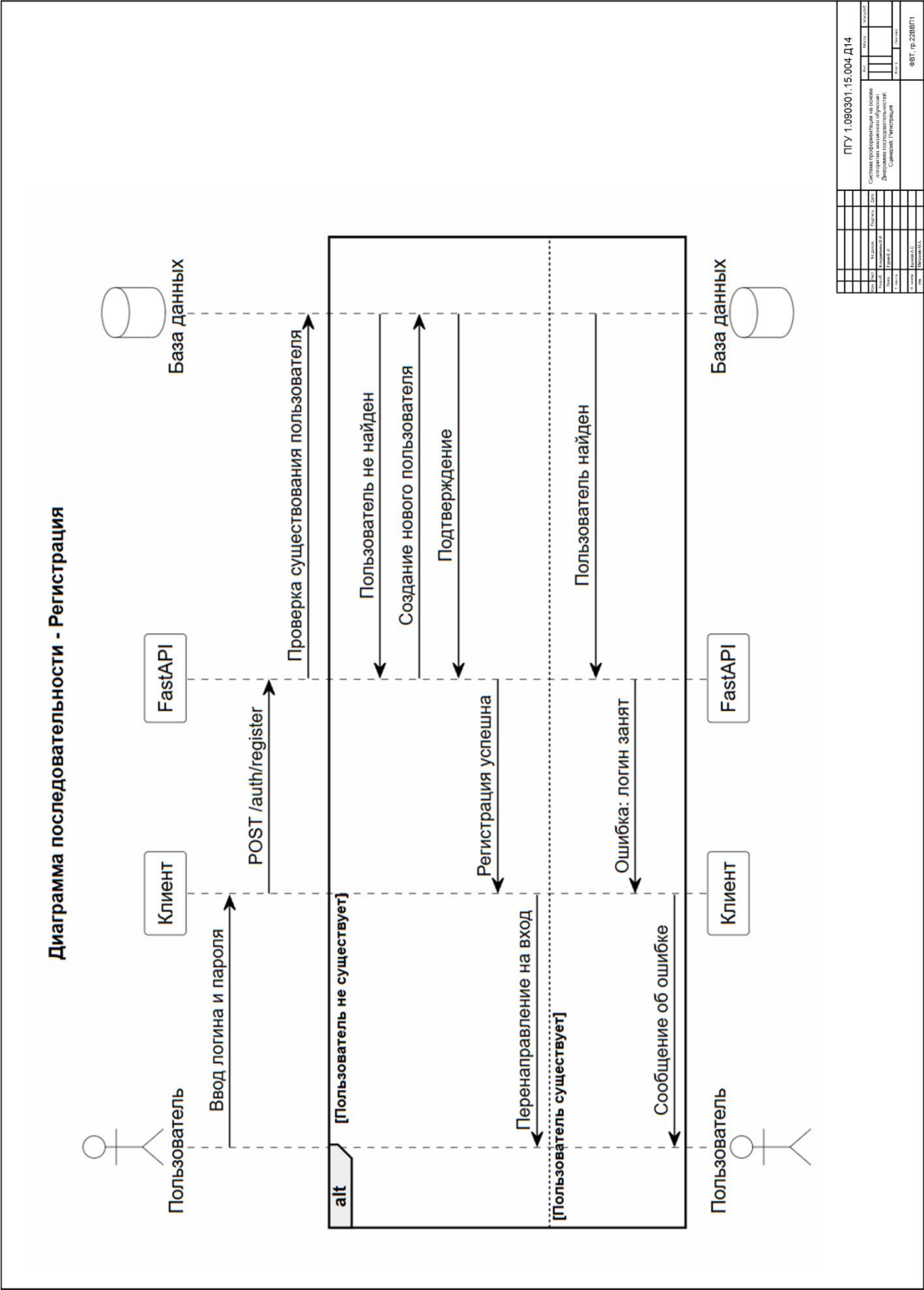


Сценарий: Авторизация



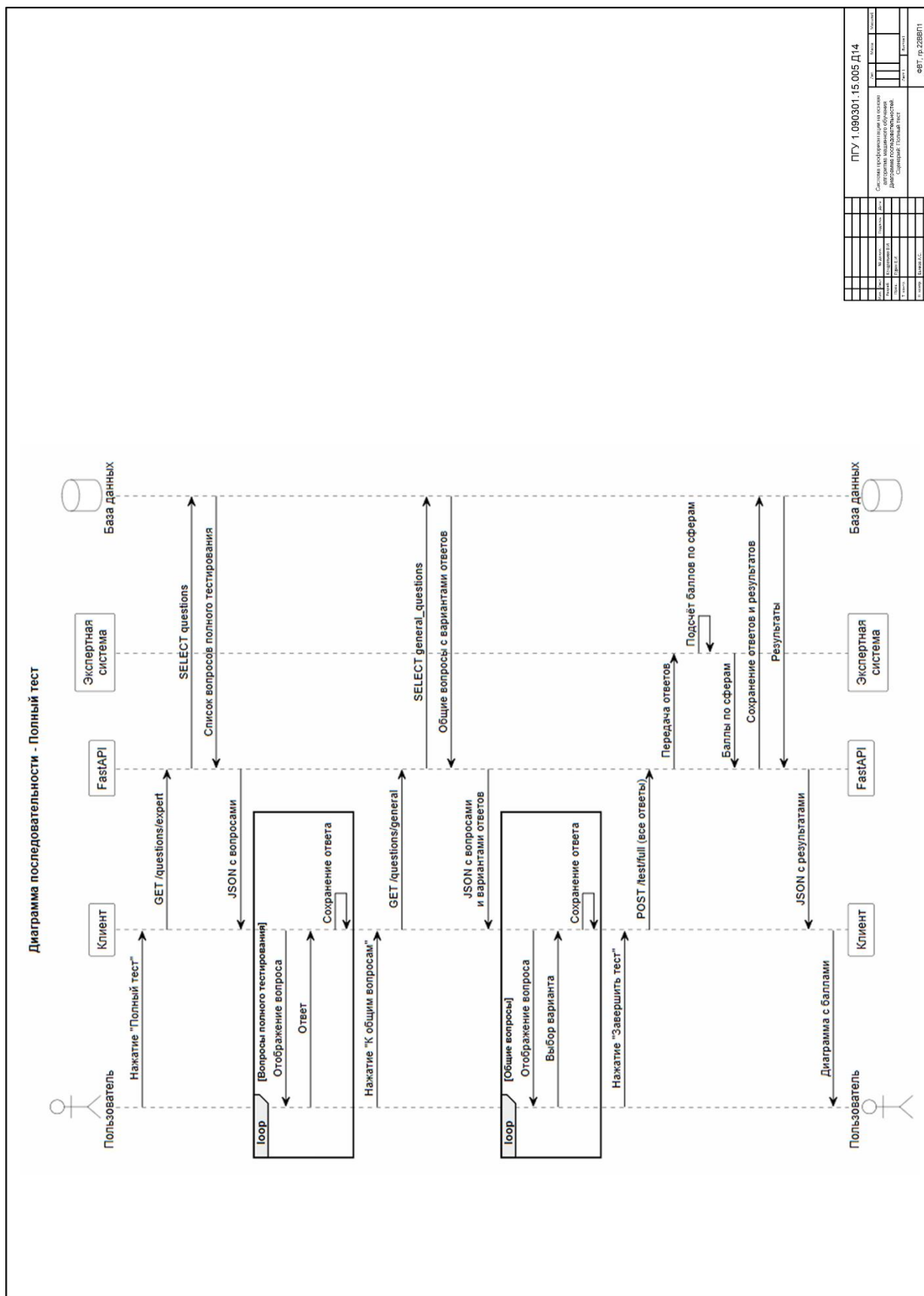
Приложение А.3 – UML – диаграмма последовательностей.

Сценарий: Регистрация



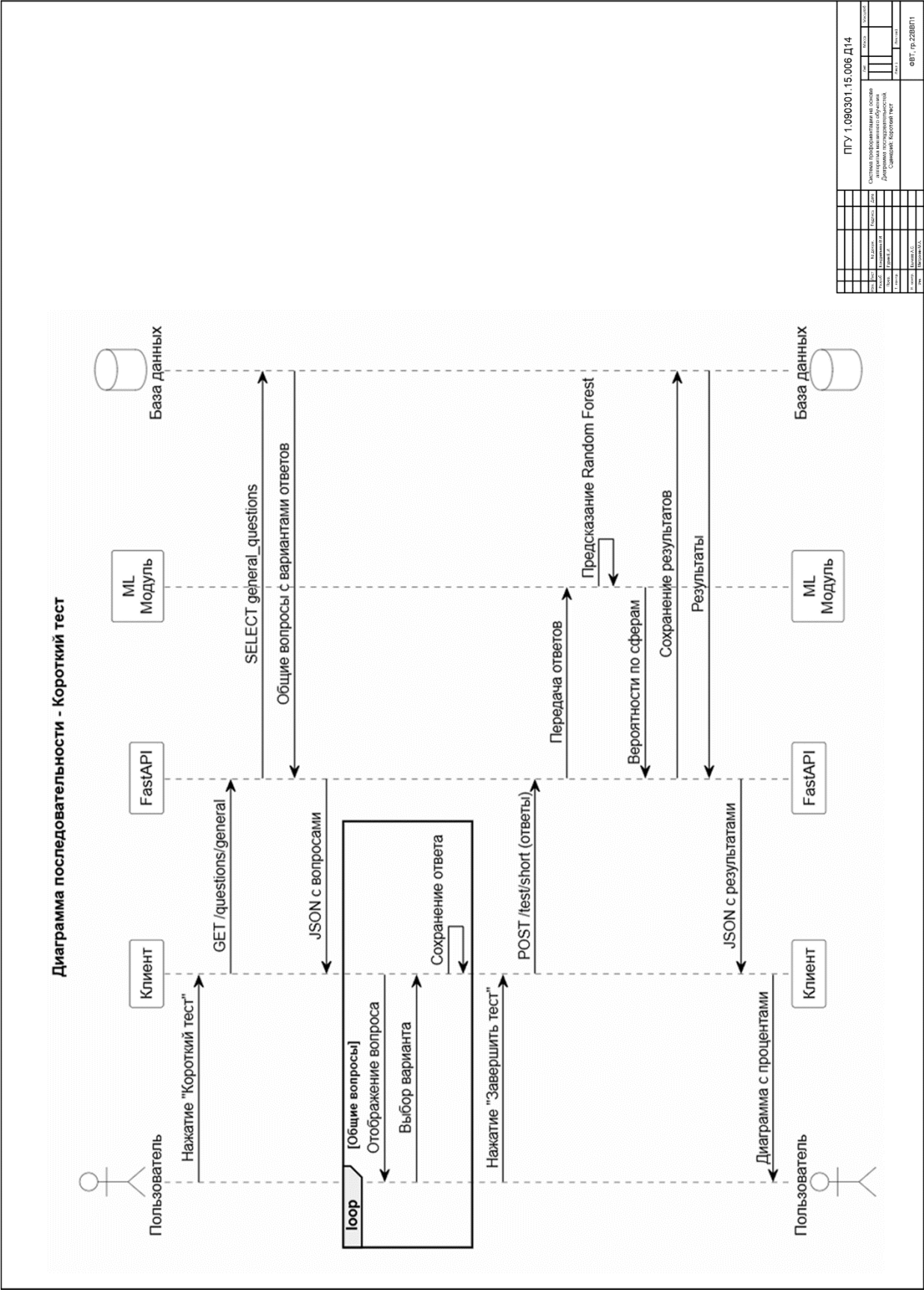
ПГУ 1.090301.15.004 Д14									
Система розробленої як засіб застереження від витоку даних з інформаційних систем Самсоніт Інтернел									
№ 01.1									
№ 01.2									
№ 01.3									
№ 01.4									
№ 01.5									
№ 01.6									
№ 01.7									
№ 01.8									
№ 01.9									
№ 01.10									
№ 01.11									
№ 01.12									
№ 01.13									
№ 01.14									
№ 01.15									
№ 01.16									
№ 01.17									
№ 01.18									
№ 01.19									
№ 01.20									
№ 01.21									
№ 01.22									
№ 01.23									
№ 01.24									
№ 01.25									
№ 01.26									
№ 01.27									
№ 01.28									
№ 01.29									
№ 01.30									
№ 01.31									
№ 01.32									
№ 01.33									
№ 01.34									
№ 01.35									
№ 01.36									
№ 01.37									
№ 01.38									
№ 01.39									
№ 01.40									
№ 01.41									
№ 01.42									
№ 01.43									
№ 01.44									
№ 01.45									
№ 01.46									
№ 01.47									
№ 01.48									
№ 01.49									
№ 01.50									
№ 01.51									
№ 01.52									
№ 01.53									
№ 01.54									
№ 01.55									
№ 01.56									
№ 01.57									
№ 01.58									
№ 01.59									
№ 01.60									
№ 01.61									
№ 01.62									
№ 01.63									
№ 01.64									
№ 01.65									
№ 01.66									
№ 01.67									
№ 01.68									
№ 01.69									
№ 01.70									
№ 01.71									
№ 01.72									
№ 01.73									
№ 01.74									
№ 01.75									
№ 01.76									
№ 01.77									
№ 01.78									
№ 01.79									
№ 01.80									
№ 01.81									
№ 01.82									
№ 01.83									
№ 01.84									
№ 01.85									
№ 01.86									
№ 01.87									
№ 01.88									
№ 01.89									
№ 01.90									
№ 01.91									
№ 01.92									
№ 01.93									
№ 01.94									
№ 01.95									
№ 01.96									
№ 01.97									
№ 01.98									
№ 01.99									
№ 02.00									
№ 02.01									
№ 02.02									
№ 02.03									
№ 02.04									
№ 02.05									
№ 02.06									
№ 02.07									
№ 02.08									
№ 02.09									
№ 02.10									
№ 02.11									
№ 02.12									
№ 02.13									
№ 02.14									
№ 02.15									
№ 02.16									
№ 02.17									
№ 02.18									
№ 02.19									
№ 02.20									
№ 02.21									
№ 02.22									
№ 02.23									
№ 02.24									
№ 02.25									
№ 02.26									
№ 02.27									
№ 02.28									
№ 02.29									
№ 02.30									
№ 02.31									
№ 02.32									
№ 02.33									
№ 02.34									
№ 02.35									
№ 02.36									
№ 02.37									
№ 02.38									
№ 02.39									
№ 02.40									
№ 02.41									
№ 02.42									
№ 02.43									
№ 02.44									
№ 02.45									
№ 02.46									
№ 02.47									
№ 02.48									
№ 02.49									
№ 02.50									
№ 02.51									
№ 02.52									
№ 02.53									
№ 02.54									
№ 02.55									
№ 02.56									
№ 02.57									
№ 02.58									
№ 02.59									
№ 02.60									
№ 02.61									
№ 02.62									
№ 02.63									
№ 02.64									
№ 02.65									
№ 02.66									
№ 02.67									
№ 02.68									
№ 02.69									
№ 02.70									
№ 02.71									
№ 02.72									
№ 02.73									
№ 02.74									
№ 02.75									
№ 02.76									
№ 02.77									
№ 02.78									
№ 02.79									
№ 02.80									
№ 02.81									
№ 02.82									
№ 02.83									
№ 02.84									
№ 02.85									
№ 02.86									
№ 02.87									
№ 02.88									
№ 02.89									
№ 02.90									
№ 02.91									
№ 02.92									
№ 02.93									
№ 02.94									
№ 02.95									
№ 02.96									
№ 02.97									
№ 02.98									
№ 02.99									
№ 03.00									
№ 03.01									
№ 03.02									
№ 03.03									
№ 03.04									
№ 03.05									
№ 03.06									
№ 03.07									
№ 03.08									
№ 03.09									
№ 03.10									
№ 03.11									
№ 03.12									
№ 03.13									
№ 03.14									
№ 03.15									
№ 03.16									
№ 03.17									
№ 03.18									
№ 03.19									
№ 03.20									
№ 03.21									
№ 03.22									
№ 03.23									
№ 03.24									
№ 03.25									
№ 03.26									
№ 03.27									
№ 03.28									
№ 03.29									
№ 03.30									
№ 03.31									
№ 03.32									
№ 03.33									
№ 03.34									
№ 03.35									
№ 03.36									
№ 03.37									
№ 03.38									
№ 03.39									
№ 03.40									
№ 03.41									
№ 03.42									
№ 03.43									
№ 03.44									
№ 03.45									
№ 03.46									
№ 03.47									
№ 03.48									
№ 03.49									
№ 03.50									
№ 03.51									
№ 03.52									
№ 03.53									
№ 03.54									
№ 03.55									
№ 03.56									
№ 03.57									
№ 03.58									
№ 03.59									
№ 03.60									
№ 03.61									
№ 03.62									
№ 03.63									
№ 03.64									
№ 03.65									
№ 03.66									
№ 03.67									
№ 03.68									
№ 03.69									
№ 03.70									
№ 03.71									
№ 03.72									
№ 03.73									
№ 03.74									
№ 03.75									
№ 03.76									
№ 03.77									
№ 03.78									
№ 03.79									
№ 03.80									
№ 03.81									
№ 03.82									
№ 03.83									
№ 03.84									
№ 03.85									
№ 03.86									
№ 03.87									
№ 03.88									
№ 03.89									
№ 03.90									
№ 03.91									
№ 03.92									
№ 03.93									
№ 03.94									
№ 03.95									
№ 03.96									
№ 03.97									
№ 03.98									
№ 03.99									
№ 04.00									
№ 04.01									
№ 04.02									
№ 04.03									
№ 04.04									
№ 04.05									
№ 04.06									
№ 04.07									
№ 04.08									
№ 04.09									
№ 04.10									
№ 04.11									
№ 04.12									
№ 04.13									
№ 04.14									
№ 04.15									
№ 04.16									
№ 04.17									
№ 04.18									
№ 04.19									
№ 04.20									
№ 04.21									
№ 04.22									
№ 04.23									
№ 04.24									
№ 04.25									
№ 04.26									
№ 04.27									
№ 04.28									
№ 04.29									
№ 04.30									
№ 04.31									
№ 04.32									
№ 04.33									
№ 04.34									
№ 04.35									
№ 04.36									
№ 04.37									
№ 04.38									
№ 04.39									
№ 04.40									
№ 04.41									
№ 04.42									
№ 04.43									
№ 04.44									
№ 04.45									
№ 04.46									
№ 04.47									
№ 04.48									
№ 04.49									
№ 04.50									
№ 04.51									
№ 04.52									
№ 04.53									
№ 04.54									
№ 04.55									
№ 04.56									
№ 04.57									
№ 04.58									
№ 04.59									
№ 04.60									
№ 04.61									
№ 04.62									
№ 04.63									
№ 04.64									
№ 04.65									
№ 04.66									
№ 04.67									
№ 04.68									
№ 04.69									
№ 04.70									
№ 04.71									
№ 04.72									
№ 04.73									
№ 04.74									
№ 04.75									
№ 04.76									
№ 04.77									
№ 04.78									
№ 04.79									
№ 04.80									
№ 04.81									
№ 04.82									
№ 04.83									
№ 04.84									
№ 04.85									
№ 04.86									
№ 04.87									
№ 04.88									
№ 04.89									
№ 04.90									
№ 04.91									
№ 04.92									
№ 04.93									
№ 04.94									
№ 04.95									
№ 04.96									
№ 04.97									
№ 04.98									
№ 04.99									
№ 05.00									
№ 05.01									
№ 05.02									
№ 05.03									
№ 05.04									
№ 05.05									
№ 05.06									
№ 05.07									
№ 05.08									
№ 05.09									
№ 05.10									
№ 05.11									
№ 05.12									
№ 05.13									
№ 05.14									
№ 05.15									
№ 05.16									
№ 05.17									
№ 05.18									
№ 05.19									
№ 05.20									
№ 05.21									
№ 05.22									
№ 05.23									
№ 05.24									
№ 05.25									
№ 05.26									
№ 05.27									
№ 05.28									
№ 05.29									
№ 05.30									
№ 05.31									
№ 05.32									
№ 05.33									
№ 05.34									
№ 05.35									
№ 05.36									
№ 05.37									
№ 05.38									
№ 05.39									
№ 05.40									
№ 05.41									
№ 05.42									
№ 05.43									
№ 05.44									
№ 05.45									
№ 05.46									
№ 05.47									
№ 05.48									
№ 05.49									
№ 05.50									
№ 05.51									
№ 05.52									
№ 05.53									
№ 05.54									
№ 05.55									
№ 05.56									
№ 05.57									
№ 05.58									
№ 05.59									
№ 05.60									
№ 05.61									
№ 05.62									
№ 05.63									
№ 05.64									
№ 05.65									
№ 05.66									
№ 05.67									
№ 05.68									
№ 05.69									
№ 05.70									
№ 05.71									
№ 05.72									
№ 05.73									
№ 05.74									
№ 05.75									
№ 05.76									
№ 05.77									
№ 05.78									
№ 05.79									
№ 05.80									
№ 05.81									
№ 05.82									
№ 05.83									
№ 05.84									
№ 05.85									
№ 05.86									
№ 05.87									
№ 05.88									
№ 05.89									
№ 05.90									
№ 05.91									
№ 05.92									
№ 05.93									
№ 05.94									
№ 05.95									
№ 05.96									
№ 05.97									
№ 05.98									
№ 05.99									
№ 06.00									
№ 06.01									
№ 06.02									
№ 06.03									
№ 06.04									
№ 06.05									
№ 06.06									
№ 06.07									
№ 06.08									
№ 06.09									
№ 06.10									
№ 06.11									
№ 06.12									
№ 06.13									
№ 06.14									
№ 06.15									
№ 06.16									
№ 06.17									
№ 06.18									
№ 06.19									
№ 06.20									
№ 06.21									
№ 06.22									
№ 06.23									
№ 06.24									
№ 06.25									
№ 06.26									
№ 06.27									
№ 06.28									
№ 06.29									
№ 06.30									
№ 06.31									
№ 06.32									
№ 06.33									
№ 06.34									
№ 06.35									
№ 06.36									
№ 06.37									
№ 06.38									
№ 06.39									
№ 06.40									
№ 06.41									
№ 06.42									
№ 06.43									
№ 06.44									
№ 06.45									
№ 06.46									
№ 06.47									
№ 06.48									
№ 06.49									
№ 06.50									
№ 06.51									
№ 06.52									
№ 06.53									
№ 06.54									
№ 06.55									
№ 06.56									
№ 06.57									
№ 06.58									
№ 06.59									
№ 06.60									
№ 06.61									
№ 06.62									
№ 06.63									
№ 06.64									
№ 06.65									
№ 06.66									
№ 06.67									
№ 06.68									
№ 06.69									
№ 06.70									
№ 06.71									
№ 06.72									
№ 06.73									
№ 06.74									
№ 06.75									
№ 06.76									
№ 06.77									
№ 06.78									
№ 06.79									
№ 06.80									
№ 06.81									
№ 06.82									
№ 06.83									
№ 06.84									
№ 06.85									
№ 06.86									
№ 06.87									
№ 06.88									
№ 06.89									
№ 06.90									
№ 06.91									
№ 06.92									
№ 06.93									
№ 06.94									
№ 06.95									
№ 06.96									
№ 06.97									
№ 06.98									
№ 06.99									
№ 07.00									
№ 07.01									
№ 07.02									
№ 07.03									
№ 07.04									
№ 07.05									
№ 07.06									
№ 07.07									
№ 07.08									
№ 07.09									
№ 07.10									
№ 07.11									
№ 07.12									
№ 07.13									
№ 07.14									
№ 07.15									

Сценарий: Полный тест



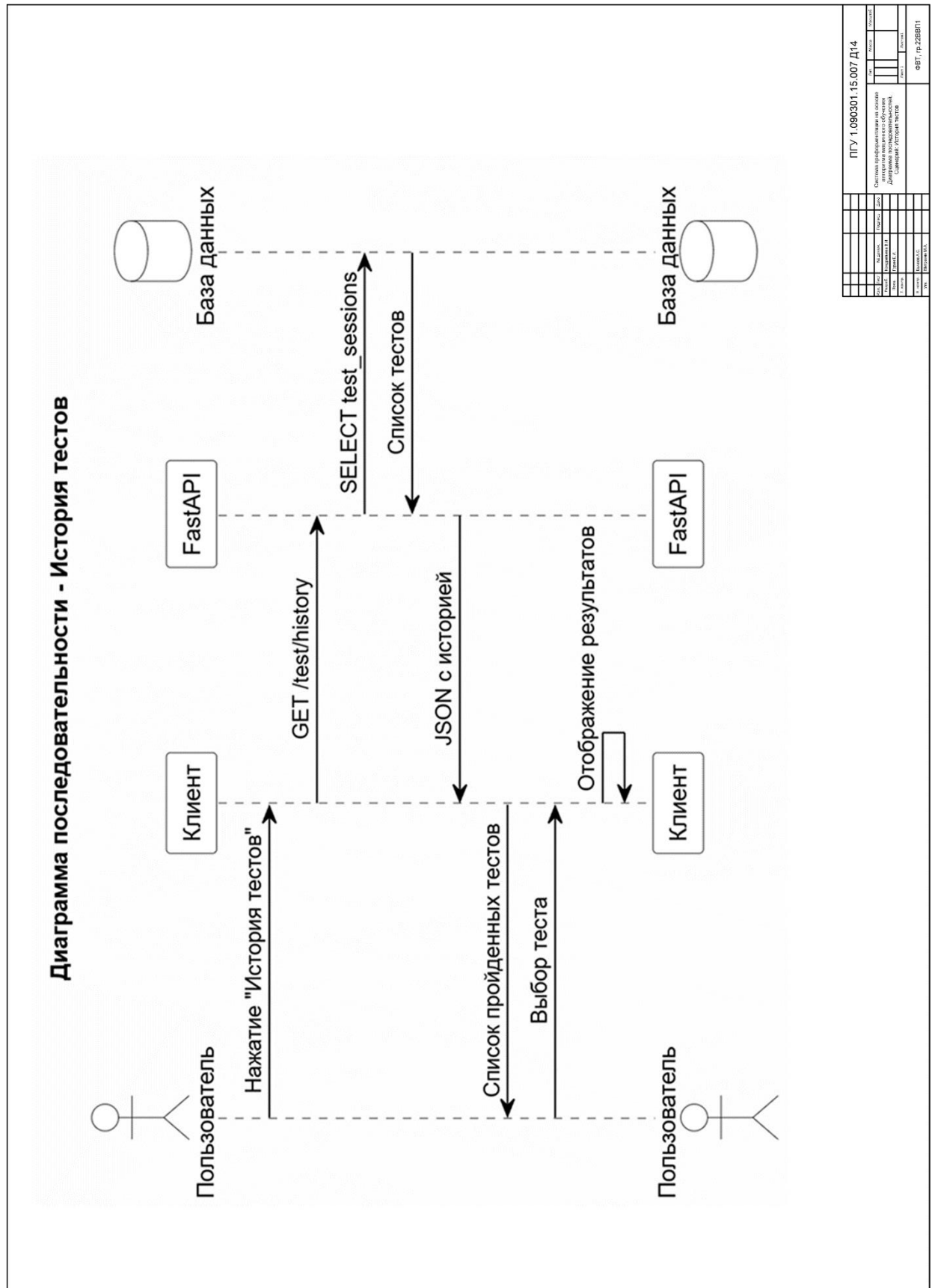
Приложение А.5 – UML – диаграмма последовательностей.

Сценарий: Короткий тест



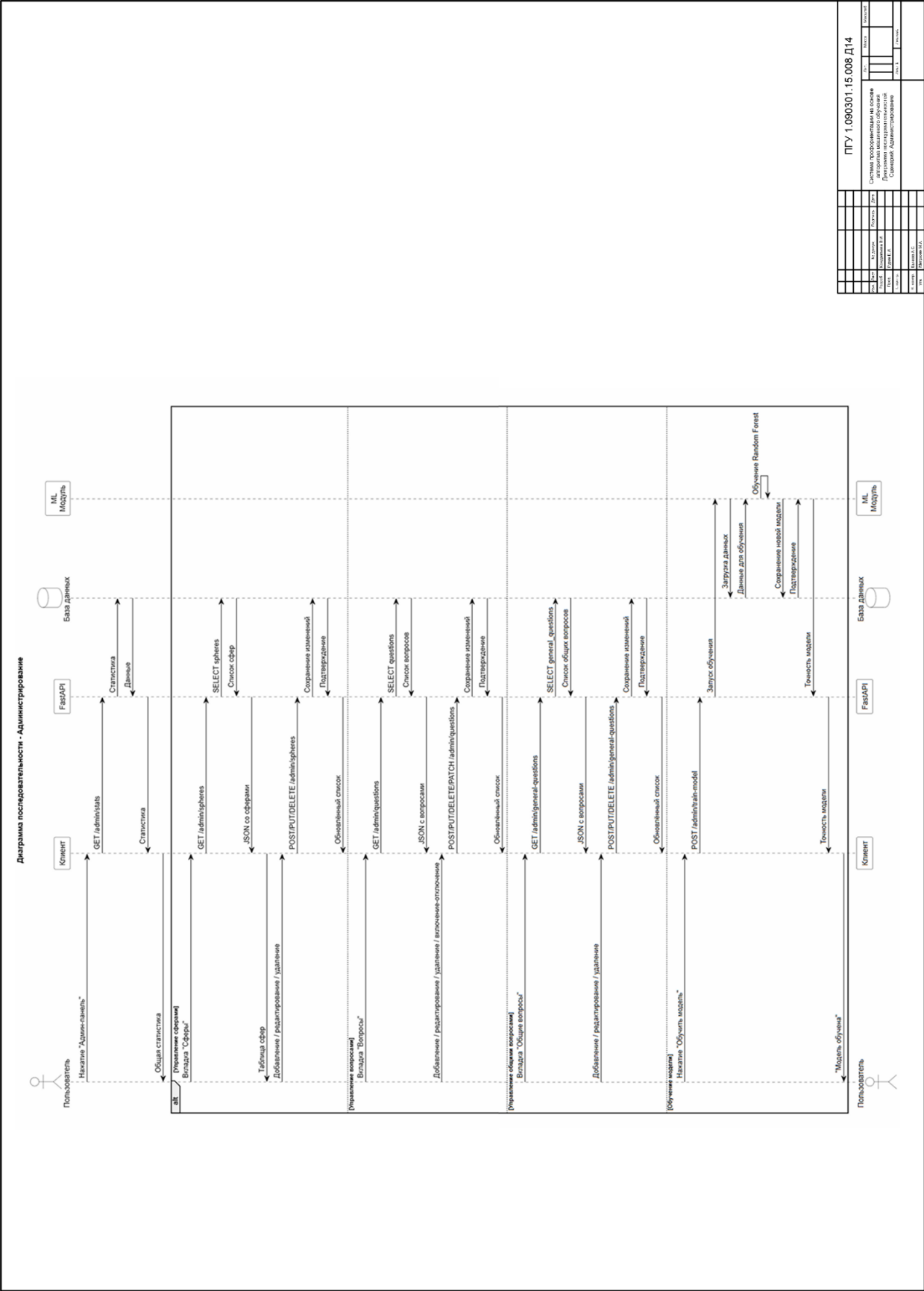
																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					</
--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	----

Сценарий: История тестов



Приложение А.7 – UML – диаграмма последовательностей.

Сценарий: Администрирование



ПГУ 1.090301.15.008 Д14									
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия

Приложение А.9 – UML – диаграмма деятельности по сценариям

Диаграмма деятельности - Регистрация и вход

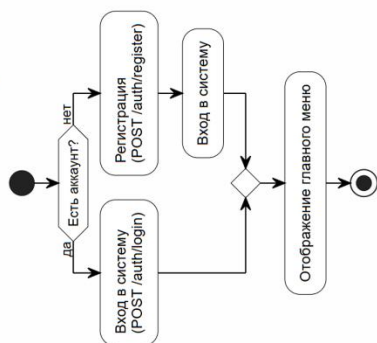


Диаграмма деятельности - История тестов

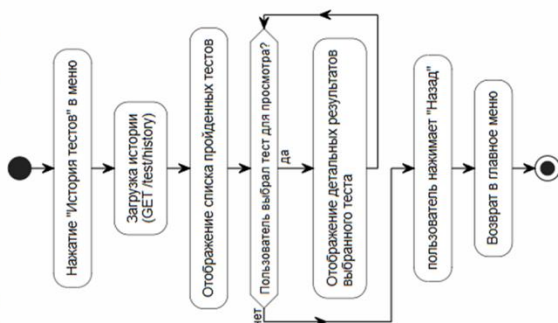


Диаграмма деятельности - Полный тест

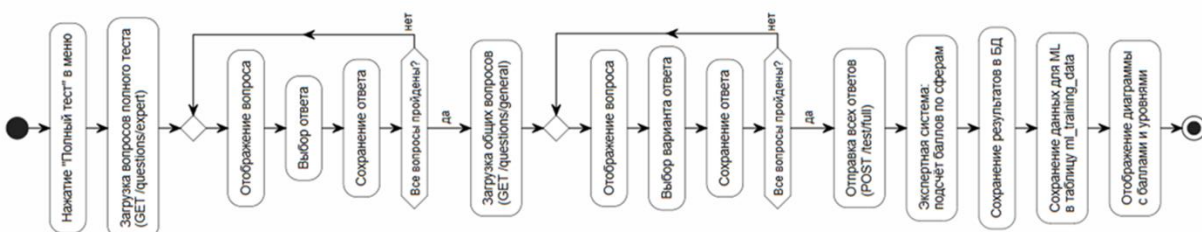


Диаграмма деятельности - Админ-панель

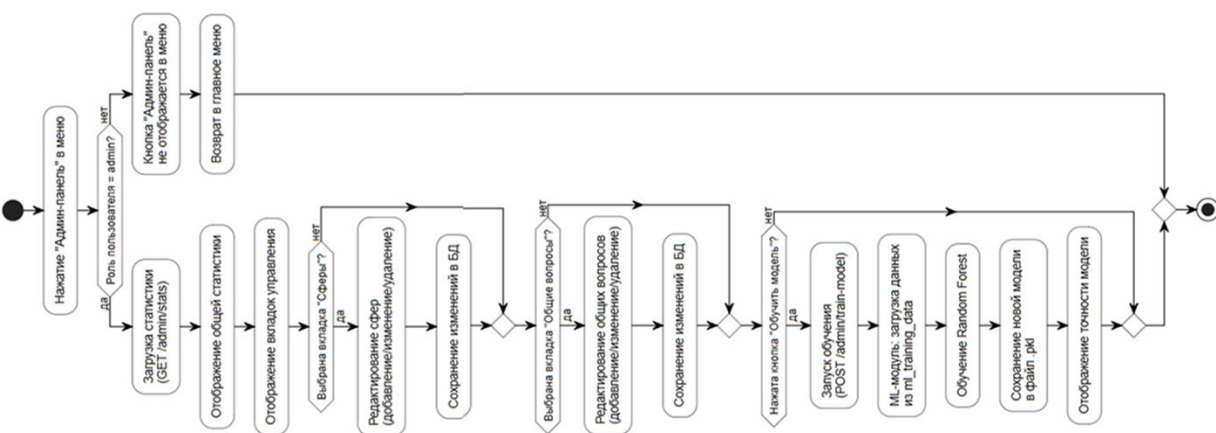


Диаграмма деятельности - Короткий тест

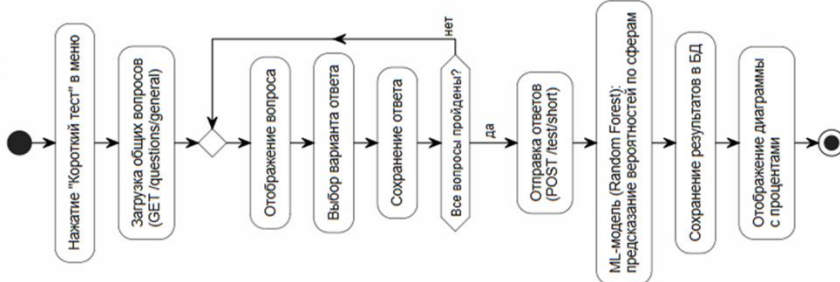
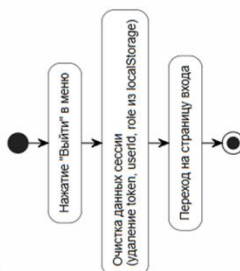


Диаграмма деятельности - Выход из системы



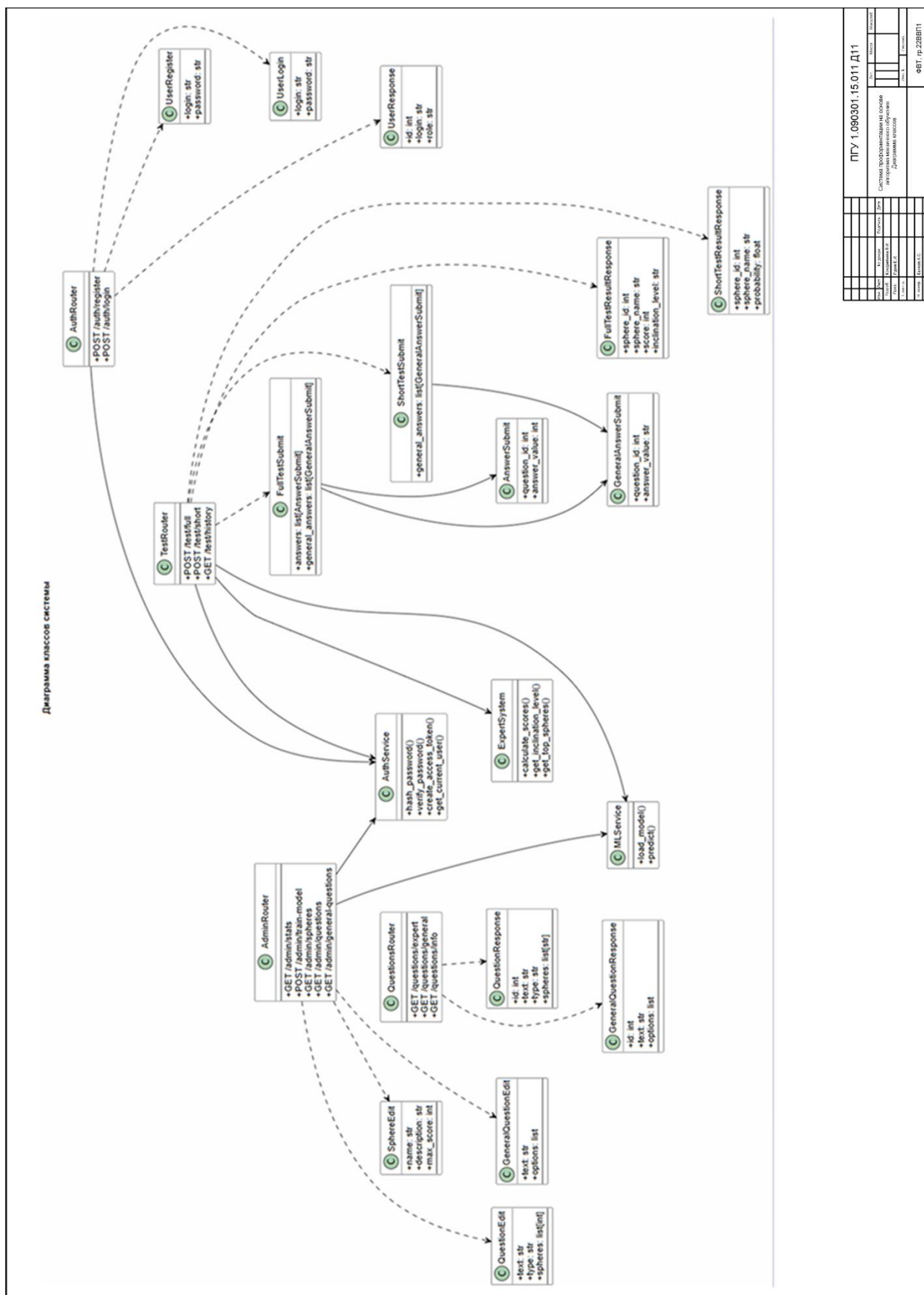
ПГУ 1.090301.15.010 Д17

Имя	Фамилия	Дата	Подпись	Подпись
Иванов	Иван	2023.10.10	Иванов И.И.	Иванов И.И.
Петров	Петр	2023.10.10	Петров П.П.	Петров П.П.
Сидоров	Сидор	2023.10.10	Сидоров С.С.	Сидоров С.С.
Тихонов	Тихон	2023.10.10	Тихонов Т.Т.	Тихонов Т.Т.
Васильев	Василий	2023.10.10	Васильев В.В.	Васильев В.В.
Кузнецов	Кузнец	2023.10.10	Кузнецов К.К.	Кузнецов К.К.
Левченко	Левченко	2023.10.10	Левченко Л.Л.	Левченко Л.Л.
Новиков	Новик	2023.10.10	Новиков Н.Н.	Новиков Н.Н.
Попов	Попов	2023.10.10	Попов П.П.	Попов П.П.
Рябенко	Рябенко	2023.10.10	Рябенко Р.Р.	Рябенко Р.Р.
Смирнов	Смирнов	2023.10.10	Смирнов С.С.	Смирнов С.С.
Толкачев	Толкачев	2023.10.10	Толкачев Т.Т.	Толкачев Т.Т.
Федотов	Федотов	2023.10.10	Федотов Ф.Ф.	Федотов Ф.Ф.
Харченко	Харченко	2023.10.10	Харченко Х.Х.	Харченко Х.Х.
Цыганов	Цыганов	2023.10.10	Цыганов Ц.Ц.	Цыганов Ц.Ц.
Чайков	Чайков	2023.10.10	Чайков Ч.Ч.	Чайков Ч.Ч.
Шевченко	Шевченко	2023.10.10	Шевченко Ш.Ш.	Шевченко Ш.Ш.
Щербина	Щербина	2023.10.10	Щербина Щ.Щ.	Щербина Щ.Щ.
Юрьев	Юрьев	2023.10.10	Юрьев Ю.Ю.	Юрьев Ю.Ю.
Яковлев	Яковлев	2023.10.10	Яковлев Я.Я.	Яковлев Я.Я.

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.15.001 ПЗ

Приложение А.10 – UML – диаграмма классов



ПГУ 1.090301.15.011 Д11									
№	Имя	Фамилия	Пол	Дата	Время	Место	Страна	Город	Улица
1	Иванов	Иван	М	15.01.2011	10:00	Москва	Россия	Москва	ул. Мясницкая
2	Петров	Петр	М	15.01.2011	10:05	Москва	Россия	Москва	ул. Мясницкая
3	Сидоров	Сидор	М	15.01.2011	10:10	Москва	Россия	Москва	ул. Мясницкая
4	Климов	Климов	М	15.01.2011	10:15	Москва	Россия	Москва	ул. Мясницкая
5	Васильев	Василий	М	15.01.2011	10:20	Москва	Россия	Москва	ул. Мясницкая
6	Попов	Попов	М	15.01.2011	10:25	Москва	Россия	Москва	ул. Мясницкая
7	Смирнов	Смирнов	М	15.01.2011	10:30	Москва	Россия	Москва	ул. Мясницкая
8	Михайлов	Михайлов	М	15.01.2011	10:35	Москва	Россия	Москва	ул. Мясницкая
9	Кузнецов	Кузнецов	М	15.01.2011	10:40	Москва	Россия	Москва	ул. Мясницкая
10	Лебедев	Лебедев	М	15.01.2011	10:45	Москва	Россия	Москва	ул. Мясницкая

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.15.001 ПЗ

Приложение Б
(обязательное)

Система профориентации на основе алгоритма машинного обучения

Листинг программы

					ПГУ 1.090301.15.001 ПЗ	Лист
						99
Изм.	Лист	№ докум.	Подпись	Дата		

Файл «train_model.py»

```
import os
import sys
import pickle
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

# Добавляем родительскую папку в путь для импорта
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))

from database import SessionLocal
from models import MLTrainingData, MLModel
from sqlalchemy import func

# Обучает модель на данных из ml_training_data и сохраняет в файл
def train_model():
    db = SessionLocal()

    try:
        # Загружаем все записи из ml_training_data
        records = db.query(MLTrainingData).all()
        print(f"Загружено записей из БД: {len(records)}")

        if len(records) < 10:
            print(f"Недостаточно данных для обучения. Нужно минимум 10, сейчас {len(records)}.")
            return None

        # Преобразуем данные в массивы X (признаки) и y (целевая переменная)
        X = []
        y = []

        for record in records:
            features = record.features.get("answers", [])
            # Преобразуем строки в числа, если нужно
            numeric_features = []
            for f in features:
                if isinstance(f, str):
                    # Берём индекс варианта ответа из хэша (для воспроизводимости)
                    numeric_features.append(hash(f) % 1000)
                else:
                    numeric_features.append(int(f))
            X.append(numeric_features)
            y.append(record.target_sphere_id)

        X = np.array(X)
        y = np.array(y)
```

```

print(f"Размер матрицы признаков: {X.shape}")
print(f"Количество классов: {len(set(y))}")

# Разделяем на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# Обучаем модель
model = RandomForestClassifier(n_estimators=100,
random_state=42)
model.fit(X_train, y_train)

# Оцениваем точность
accuracy = model.score(X_test, y_test)
print(f"\nМодель обучена!")
print(f"    Точность на тестовых данных: {accuracy:.2%}")
print(f"    Всего записей: {len(records)}")
print(f"    Обучающая выборка: {len(X_train)}")
print(f"    Тестовая выборка: {len(X_test)}")

# Сохраняем модель в файл
model_dir =
os.path.join(os.path.dirname(os.path.abspath(__file__)))
model_path = os.path.join(model_dir,
"random_forest_model.pkl")

with open(model_path, "wb") as f:
    pickle.dump(model, f)

print(f"    Модель сохранена: {model_path}")

# Важность признаков
importances = model.feature_importances_.tolist()
print(f"    Важность признаков: {[round(i, 4) for i in
importances]}")

# Определяем все существующие версии моделей, чтобы найти
следующий свободный номер
existing_versions = db.query(MLModel.version).all()
existing_numbers = set()
for v in existing_versions:
    try:
        # Извлекаем число из строки "vX.0"
        num = int(v[0].replace("v", "").split(".")[0])
        existing_numbers.add(num)
    except (ValueError, IndexError):
        # Если версия имеет нестандартный формат,
пропускаем её
        continue

```

```

        # Находим минимальное положительное целое число, которого
нет в existing_numbers
        version_num = 1
        while version_num in existing_numbers:
            version_num += 1

        version = f"v{version_num}.0"
        print(f"    Следующая свободная версия: {version}")

        # Деактивируем все предыдущие модели
        db.query(MLModel).update({"is_active": False})

        # Сохраняем запись о модели в БД
        model_record = MLModel(
            version=version,
            model_file=model_path,
            accuracy=float(accuracy),
            is_active=True,
            features_importance=importances
        )
        db.add(model_record)
        db.commit()

        print(f"    Запись о модели сохранена в БД
(version={version})")

        return accuracy, importances

    finally:
        db.close()

if __name__ == "__main__":
    print("=" * 55)
    print("ОБУЧЕНИЕ МОДЕЛИ RANDOM FOREST")
    print("=" * 55)
    train_model()

```

Файл «admin_router.py»

```

from fastapi import APIRouter, Depends, HTTPException, Header,
status
from sqlalchemy.orm import Session
from sqlalchemy import func
from database import get_db
from models import User, Sphere, Question, QuestionSphere,
GeneralQuestion, TestSession, FullTestResult, MLTrainingData

router = APIRouter(prefix="/admin", tags=["Администрирование"])

def _check_admin(x_user_id: int, db: Session):

```

```

        user = db.query(User).filter(User.id_users ==
x_user_id).first()
        if not user or user.role != 'admin':
            raise HTTPException(
                status_code=status.HTTP_403_FORBIDDEN,
                detail="Доступ запрещён. Требуются права
администратора."
            )
        return user

#Возвращает статистику по тестам
@router.get("/stats")
def get_stats(db: Session = Depends(get_db)):

    total_full =
db.query(TestSession).filter(TestSession.test_type ==
"full").count()
    total_short =
db.query(TestSession).filter(TestSession.test_type ==
"short").count()
    total_users = db.query(User).filter(User.role ==
"user").count()

    recent_tests = (
        db.query(TestSession, User.login)
        .join(User, TestSession.user_id == User.id_users)
        .order_by(TestSession.started_at.desc())
        .limit(5)
        .all()
    )

    return {
        "total_tests": total_full + total_short,
        "total_full": total_full,
        "total_short": total_short,
        "total_users": total_users,
        "top_spheres": [],
        "avg_scores": [],
        "recent_tests": [
            {
                "id": ts.id_test_sessions,
                "login": login,
                "type": ts.test_type,
                "date": ts.started_at.isoformat() if ts.started_at
else None
            }
            for ts, login in recent_tests
        ]
    }

# Запускает обучение модели

```

```

@router.post("/train-model")
def train_model_endpoint(db: Session = Depends(get_db), x_user_id:
int = Header(default=3)):

    _check_admin(x_user_id, db)

    try:
        import sys
        import os

sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__
file__))))
        from ml.train_model import train_model

        result = train_model()
        if result is None:
            raise HTTPException(status_code=400,
detail="Недостаточно данных для обучения (нужно минимум 10
записей) ")

        accuracy, importances = result
        return {
            "message": "Модель успешно обучена",
            "accuracy": round(accuracy, 4),
            "records_used": db.query(MLTrainingData).count()
        }
    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Ошибка
обучения: {str(e)}")

@router.get("/spheres")
def list_spheres(db: Session = Depends(get_db)):
    return db.query(Sphere).order_by(Sphere.id_spheres).all()

@router.put("/spheres/{sphere_id}")
def update_sphere(sphere_id: int, data, db: Session =
Depends(get_db), x_user_id: int = Header(default=3)):
    from schemas import SphereEdit
    _check_admin(x_user_id, db)
    sphere = db.query(Sphere).filter(Sphere.id_spheres ==
sphere_id).first()
    if not sphere:
        raise HTTPException(status_code=404, detail="Сфера не
найдена")
    sphere.name = data.name
    sphere.description = data.description
    sphere.max_score = data.max_score
    db.commit()
    return {"message": "Сфера обновлена"}

```

```

@router.post("/spheres", status_code=201)
def create_sphere(data, db: Session = Depends(get_db), x_user_id:
int = Header(default=3)):
    from schemas import SphereEdit
    _check_admin(x_user_id, db)
    existing = db.query(Sphere).filter(Sphere.name ==
data.name).first()
    if existing:
        raise HTTPException(status_code=400, detail="Сфера с таким
названием уже существует")
    sphere = Sphere(name=data.name, description=data.description,
max_score=data.max_score)
    db.add(sphere)
    db.commit()
    return {"message": "Сфера создана", "id": sphere.id_spheres}

@router.delete("/spheres/{sphere_id}")
def delete_sphere(sphere_id: int, db: Session = Depends(get_db),
x_user_id: int = Header(default=3)):
    _check_admin(x_user_id, db)
    sphere = db.query(Sphere).filter(Sphere.id_spheres ==
sphere_id).first()
    if not sphere:
        raise HTTPException(status_code=404, detail="Сфера не
найдена")
    linked =
db.query(QuestionSphere).filter(QuestionSphere.sphere_id ==
sphere_id).count()
    if linked > 0:
        raise HTTPException(status_code=400, detail=f"Нельзя
удалить: к сфере привязано {linked} вопросов")
    db.delete(sphere)
    db.commit()
    return {"message": "Сфера удалена"}

@router.get("/questions")
def list_questions(db: Session = Depends(get_db)):
    questions =
db.query(Question).order_by(Question.id_questions).all()
    result = []
    for q in questions:
        sphere_ids = [qs.sphere_id for qs in q.question_spheres]
        result.append({
            "id": q.id_questions,
            "text": q.text,
            "type": q.type,
            "is_active": q.is_active,
            "spheres": sphere_ids
        })

```

```

    return result

@router.put("/questions/{question_id}")
def update_question(question_id: int, data, db: Session = Depends(get_db), x_user_id: int = Header(default=3)):
    from schemas import QuestionEdit
    _check_admin(x_user_id, db)
    q = db.query(Question).filter(Question.id_questions == question_id).first()
    if not q:
        raise HTTPException(status_code=404, detail="Вопрос не найден")
    q.text = data.text
    q.type = data.type
    db.query(QuestionSphere).filter(QuestionSphere.question_id == question_id).delete()
    for sphere_id in data.spheres:
        db.add(QuestionSphere(question_id=question_id, sphere_id=sphere_id))
    db.commit()
    return {"message": "Вопрос обновлён"}

@router.post("/questions", status_code=201)
def create_question(data, db: Session = Depends(get_db), x_user_id: int = Header(default=3)):
    from schemas import QuestionEdit
    _check_admin(x_user_id, db)
    q = Question(text=data.text, type=data.type, is_active=True)
    db.add(q)
    db.flush()
    for sphere_id in data.spheres:
        db.add(QuestionSphere(question_id=q.id_questions, sphere_id=sphere_id))
    db.commit()
    return {"message": "Вопрос создан", "id": q.id_questions}

@router.delete("/questions/{question_id}")
def delete_question(question_id: int, db: Session = Depends(get_db), x_user_id: int = Header(default=3)):
    _check_admin(x_user_id, db)
    q = db.query(Question).filter(Question.id_questions == question_id).first()
    if not q:
        raise HTTPException(status_code=404, detail="Вопрос не найден")
    db.delete(q)
    db.commit()
    return {"message": "Вопрос удалён"}

@router.patch("/questions/{question_id}/toggle")

```

```

def toggle_question(question_id: int, db: Session =
Depends(get_db), x_user_id: int = Header(default=3)):
    _check_admin(x_user_id, db)
    q = db.query(Question).filter(Question.id_questions ==
question_id).first()
    if not q:
        raise HTTPException(status_code=404, detail="Вопрос не
найден")
    q.is_active = not q.is_active
    db.commit()
    return {"message": f"Вопрос {'активирован' if q.is_active else
'скрыт'}", "is_active": q.is_active}

@router.get("/general-questions")
def list_general_questions(db: Session = Depends(get_db)):
    return
db.query(GeneralQuestion).order_by(GeneralQuestion.id_general_ques
tions).all()

@router.put("/general-questions/{question_id}")
def update_general_question(question_id: int, data, db: Session =
Depends(get_db), x_user_id: int = Header(default=3)):
    from schemas import GeneralQuestionEdit
    _check_admin(x_user_id, db)
    q =
db.query(GeneralQuestion).filter(GeneralQuestion.id_general_questi
ons == question_id).first()
    if not q:
        raise HTTPException(status_code=404, detail="Вопрос не
найден")
    q.text = data.text
    q.options = data.options
    db.commit()
    return {"message": "Общий вопрос обновлён"}

@router.post("/general-questions", status_code=201)
def create_general_question(data, db: Session = Depends(get_db),
x_user_id: int = Header(default=3)):
    from schemas import GeneralQuestionEdit
    _check_admin(x_user_id, db)
    q = GeneralQuestion(text=data.text, options=data.options,
is_active=True)
    db.add(q)
    db.commit()
    return {"message": "Общий вопрос создан", "id":
q.id_general_questions}

@router.delete("/general-questions/{question_id}")

```

```

def delete_general_question(question_id: int, db: Session =
Depends(get_db), x_user_id: int = Header(default=3)):
    _check_admin(x_user_id, db)
    q =
db.query(GeneralQuestion).filter(GeneralQuestion.id_general_questi
ons == question_id).first()
    if not q:
        raise HTTPException(status_code=404, detail="Вопрос не
найден")
    db.delete(q)
    db.commit()
    return {"message": "Общий вопрос удалён"}

```

Файл «auth_router.py»

```

from datetime import datetime
from fastapi import APIRouter, Depends, HTTPException, status
from sqlalchemy.orm import Session
from database import get_db
from models import User
from schemas import UserRegister, UserLogin, UserResponse
from auth import hash_password, verify_password

router = APIRouter(prefix="/auth", tags=["Авторизация"])

@router.post("/register", status_code=status.HTTP_201_CREATED)
def register(user_data: UserRegister, db: Session =
Depends(get_db)):
    existing_user = db.query(User).filter(User.login ==
user_data.login).first()
    if existing_user:
        raise HTTPException(
            status_code=status.HTTP_400_BAD_REQUEST,
            detail="Пользователь с таким логином уже существует"
        )

    new_user = User(
        login=user_data.login,
        password_hash=hash_password(user_data.password),
        role="user",
        created_at=datetime.now()
    )
    db.add(new_user)
    db.commit()
    db.refresh(new_user)

    return UserResponse.from_orm_with_mapping(new_user)

@router.post("/login")
def login(user_data: UserLogin, db: Session = Depends(get_db)):
    user = db.query(User).filter(User.login ==
user_data.login).first()

```

```

        if not user or not verify_password(user_data.password,
user.password_hash):
            raise HTTPException(
                status_code=status.HTTP_401_UNAUTHORIZED,
                detail="Неверный логин или пароль"
            )

    return {
        "user": {"id": user.id_users, "login": user.login, "role":
user.role}
    }

```

Файл «questions_router.py»

```

from fastapi import APIRouter, Depends
from sqlalchemy.orm import Session
from database import get_db
from models import Question, GeneralQuestion, Sphere
from schemas import QuestionResponse, GeneralQuestionResponse

router = APIRouter(prefix="/questions", tags=["Вопросы теста"])

@router.get("/expert", response_model=list[QuestionResponse])
def get_expert_questions(db: Session = Depends(get_db)):
    questions = (
        db.query(Question)
        .filter(Question.is_active == True)
        .all()
    )

    result = []
    for q in questions:
        sphere_names = [
            qs.sphere.name
            for qs in q.question_spheres
        ]
        result.append(QuestionResponse(
            id=q.id_questions,
            text=q.text,
            type=q.type,
            spheres=sphere_names
        ))

    return result

@router.get("/general",
response_model=list[GeneralQuestionResponse])
def get_general_questions(db: Session = Depends(get_db)):
    questions = (
        db.query(GeneralQuestion)
        .filter(GeneralQuestion.is_active == True)
        .all()
    )

```

```

    return [
        GeneralQuestionResponse(
            id=q.id_general_questions,
            text=q.text,
            options=q.options
        )
        for q in questions
    ]

# Возвращает информацию о системе (количество вопросов, макс.
балл)
@router.get("/info")
def get_test_info(db: Session = Depends(get_db)):
    expert_count = db.query(Question).filter(Question.is_active ==
True).count()
    general_count =
db.query(GeneralQuestion).filter(GeneralQuestion.is_active ==
True).count()

    first_sphere = db.query(Sphere).first()
    max_score = first_sphere.max_score if first_sphere else 34

    return {
        "expert_questions_count": expert_count,
        "general_questions_count": general_count,
        "max_possible_score": max_score
    }

```

Файл «test_router.py»

```

from datetime import datetime
from fastapi import APIRouter, Depends, Header
from sqlalchemy.orm import Session
from database import get_db
from models import (
    TestSession, Answer, GeneralAnswer, FullTestResult,
    ShortTestResult, MLTrainingData, Question, Sphere,
    GeneralQuestion, User
)
from schemas import (
    FullTestSubmit, FullTestResultsResponse,
    FullTestResultResponse,
    ShortTestSubmit, ShortTestResultsResponse,
    ShortTestResultResponse
)
from services.expert_system import calculate_scores,
get_inclination_level, get_top_spheres
from services.ml_service import predict

router = APIRouter(prefix="/test", tags=["Тестирование"])

```

```

def _get_answer_index(question_id: int, answer_value: str, db:
Session) -> int:
    question = db.query(GeneralQuestion).filter(
        GeneralQuestion.id_general_questions == question_id
    ).first()
    if question and question.options:
        try:
            return question.options.index(answer_value) + 1
        except ValueError:
            return 0
    return 0

def _get_or_create_user(user_id: int, db: Session):
    user = db.query(User).filter(User.id_users == user_id).first()
    if not user:
        user = User(
            id_users=user_id,
            login=f"user_{user_id}",
            password_hash="auto",
            role="user",
            created_at=datetime.now()
        )
        db.add(user)
        db.flush()
    return user

@router.post("/full", response_model=FullTestResultsResponse)
def submit_full_test(
    test_data: FullTestSubmit,
    db: Session = Depends(get_db),
    x_user_id: int = Header(default=3)
):
    user = _get_or_create_user(x_user_id, db)

    session = TestSession(
        user_id=user.id_users,
        test_type="full",
        started_at=datetime.now()
    )
    db.add(session)
    db.flush()

    for answer in test_data.answers:
        db.add(Answer(
            session_id=session.id_test_sessions,
            question_id=answer.question_id,
            answer_value=answer.answer_value
        ))

    general_answers_dict = {}
    for ga in test_data.general_answers:

```

```

        idx = _get_answer_index(ga.question_id, ga.answer_value,
db)
        db.add(GeneralAnswer(
            session_id=session.id_test_sessions,
            question_id=ga.question_id,
            answer_value=ga.answer_value,
            answer_index=idx
        ))
        general_answers_dict[ga.question_id] = ga.answer_value

db.flush()

question_ids = [a.question_id for a in test_data.answers]
questions = (
    db.query(Question)
    .filter(Question.id_questions.in_(question_ids))
    .all()
)

question_spheres = {}
for q in questions:
    sphere_ids = [qs.sphere_id for qs in q.question_spheres]
    question_spheres[q.id_questions] = {
        'type': q.type,
        'spheres': sphere_ids
    }

answers_with_spheres = []
for answer in test_data.answers:
    q_info = question_spheres.get(answer.question_id)
    if q_info:
        answers_with_spheres.append({
            'answer_value': answer.answer_value,
            'question_type': q_info['type'],
            'spheres': q_info['spheres']
        })

scores = calculate_scores(answers_with_spheres)

all_spheres = db.query(Sphere).all()
results = []

for sphere in all_spheres:
    score = scores.get(sphere.id_spheres, 0)
    level = get_inclination_level(score, sphere.max_score)

    db.add(FullTestResult(
        session_id=session.id_test_sessions,
        sphere_id=sphere.id_spheres,
        score=score,
        inclination_level=level
    ))

```

```

    ))

    results.append(FullTestResultResponse(
        sphere_id=sphere.id_spheres,
        sphere_name=sphere.name,
        score=score,
        inclination_level=level
    ))

    # Получаем список общих вопросов из бд
    general_questions = db.query(GeneralQuestion).filter(
        GeneralQuestion.is_active == True
    ).order_by(GeneralQuestion.id_general_questions).all()

    features_list = []
    for gq in general_questions:
        answer_text =
general_answers_dict.get(gq.id_general_questions, "")
        idx = _get_answer_index(gq.id_general_questions,
answer_text, db)
        features_list.append(idx)

    top_spheres = get_top_spheres(scores) if scores else [1]

    for sphere_id in top_spheres:
        db.add(MLTrainingData(
            session_id=session.id_test_sessions,
            features={"answers": features_list},
            target_sphere_id=sphere_id
        ))

    session.completed_at = datetime.now()
    db.commit()

    return FullTestResultsResponse(
        session_id=session.id_test_sessions,
        results=results
    )

@router.post("/short", response_model=ShortTestResultsResponse)
def submit_short_test(
    test_data: ShortTestSubmit,
    db: Session = Depends(get_db),
    x_user_id: int = Header(default=3)
):
    user = _get_or_create_user(x_user_id, db)

    session = TestSession(
        user_id=user.id_users,
        test_type="short",
        started_at=datetime.now()

```

```

    )
    db.add(session)
    db.flush()

    general_answers_list = []
    for ga in test_data.general_answers:
        idx = _get_answer_index(ga.question_id, ga.answer_value,
db)
        db.add(GeneralAnswer(
            session_id=session.id_test_sessions,
            question_id=ga.question_id,
            answer_value=ga.answer_value,
            answer_index=idx
        ))
        general_answers_list.append(ga.answer_value)

    db.flush()

    predictions = predict(general_answers_list)

    results = []
    for pred in predictions:
        sphere = db.query(Sphere).filter(Sphere.id_spheres ==
pred["sphere_id"]).first()
        db.add(ShortTestResult(
            session_id=session.id_test_sessions,
            sphere_id=pred["sphere_id"],
            probability=pred["probability"]
        ))
        results.append(ShortTestResultResponse(
            sphere_id=pred["sphere_id"],
            sphere_name=sphere.name if sphere else "",
            probability=pred["probability"]
        ))

    session.completed_at = datetime.now()
    db.commit()

    return ShortTestResultsResponse(
        session_id=session.id_test_sessions,
        results=results
    )

@router.get("/history")
def get_test_history(
    db: Session = Depends(get_db),
    x_user_id: int = Header(default=3)
):
    sessions = (
        db.query(TestSession)
        .filter(TestSession.user_id == x_user_id)

```

```

        .order_by(TestSession.started_at.desc())
        .all()
    )

    history = []
    for session in sessions:
        item = {
            "session_id": session.id_test_sessions,
            "test_type": session.test_type,
            "started_at": session.started_at.isoformat() if
session.started_at else None,
            "completed_at": session.completed_at.isoformat() if
session.completed_at else None,
        }

        if session.test_type == "full":
            results = (
                db.query(FullTestResult, Sphere.name)
                .join(Sphere, FullTestResult.sphere_id ==
Sphere.id_spheres)
                .filter(FullTestResult.session_id ==
session.id_test_sessions)
                .order_by(FullTestResult.score.desc())
                .all()
            )
            item["results"] = [
                {
                    "sphere_name": name,
                    "score": result.score,
                    "inclination_level": result.inclination_level
                }
                for result, name in results
            ]
        else:
            results = (
                db.query(ShortTestResult, Sphere.name)
                .join(Sphere, ShortTestResult.sphere_id ==
Sphere.id_spheres)
                .filter(ShortTestResult.session_id ==
session.id_test_sessions)
                .order_by(ShortTestResult.probability.desc())
                .all()
            )
            item["results"] = [
                {
                    "sphere_name": name,
                    "probability": result.probability
                }
                for result, name in results
            ]

```

```

        history.append(item)

    return history

```

Файл «expert_system.py»

```

# Баллы за явные вопросы в зависимости от ответа (1-5)
EXPLICIT_SCORES = {
    5: 4,
    4: 2,
    3: 1,
    2: 0.5,
    1: 0
}

# Баллы за скрытые вопросы в зависимости от ответа (1-5)
HIDDEN_SCORES = {
    5: 2,
    4: 1,
    3: 0.5,
    2: 0.25,
    1: 0
}

# Максимальный суммарный балл, который одна сфера может получить
# со всех скрытых вопросов
MAX_HIDDEN_SCORE_PER_SPHERE = 6

#Вычисляет баллы по сферам на основе ответов пользователя.
def calculate_scores(answers_with_spheres):

    """
    Аргументы:
        answers_with_spheres: список словарей вида:
        {
            'answer_value': 4,                # ответ
            'question_type': 'explicit',      # тип вопроса
            'spheres': [1, 5, 12]            # ID связанных
        }
        пользователя (1-5)
        сфер

    Возвращает:
        словарь {sphere_id: total_score}
    """
    scores = {} # итоговые баллы по каждой сфере
    implicit_scores = {} # сумма баллов, полученных от скрытых
    вопросов

    for item in answers_with_spheres:
        answer = item['answer_value']
        q_type = item['question_type']
        spheres = item['spheres']

```

```

        if q_type == 'explicit':
            # Явный вопрос: начисляем баллы одной сфере
            points = EXPLICIT_SCORES.get(answer, 0)
            for sphere_id in spheres:
                scores[sphere_id] = scores.get(sphere_id, 0) +
points

        elif q_type == 'hidden':
            # Скрытый вопрос: начисляем баллы нескольким сферам с
ограничением
            points = HIDDEN_SCORES.get(answer, 0)
            for sphere_id in spheres:
                # Проверяем лимит 6 баллов за скрытые вопросы
                current_implicit = implicit_scores.get(sphere_id,
0)

                if current_implicit < MAX_HIDDEN_SCORE_PER_SPHERE:
                    # Сколько ещё можно начислить до лимита
                    available = MAX_HIDDEN_SCORE_PER_SPHERE -
current_implicit

                    actual_points = min(points, available)
                    scores[sphere_id] = scores.get(sphere_id, 0) +
actual_points

                    implicit_scores[sphere_id] = current_implicit
+ actual_points

            # Округляем до целых
            for sphere_id in scores:
                scores[sphere_id] = int(round(scores[sphere_id]))

        return scores

#Определяет уровень склонности на основе процента от
максимального балла
def get_inclination_level(score, max_possible_score):

    percent = (score / max_possible_score) * 100 if
max_possible_score > 0 else 0

    if percent >= 76:
        return "максимальная"
    elif percent >= 56:
        return "умеренная"
    elif percent >= 23:
        return "слабая"
    else:
        return "минимальная"

#Возвращает список всех сфер с максимальным баллом.
def get_top_spheres(scores):
    """
    Аргументы:

```

```

        scores: словарь {sphere_id: total_score} из
calculate_scores()
    Возвращает:
        список [sphere_id, sphere_id, ...]

Пример:
    scores = {1: 42, 2: 42, 3: 35...} возвращает [1, 2]
"""
if not scores:
    return []
max_score = max(scores.values())
return [sphere_id for sphere_id, score in scores.items() if
score == max_score]

```

Файл «ml_service.py»

```

import pickle
import numpy as np
import os
import sys

# Добавляем родительскую папку для импорта
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__
file__))))

from database import SessionLocal
from models import GeneralQuestion, Sphere

MODEL_PATH =
os.path.join(os.path.dirname(os.path.abspath(__file__)), "..",
"ml", "random_forest_model.pkl")

# Загружает обученную модель из файла
def load_model():
    if not os.path.exists(MODEL_PATH):
        print(f"Модель не найдена: {MODEL_PATH}")
        return None
    with open(MODEL_PATH, "rb") as f:
        return pickle.load(f)

# Вычисляет индекс ответа через БД
def _get_answer_index(question_id: int, answer_value: str) -> int:
    db = SessionLocal()
    try:
        question = db.query(GeneralQuestion).filter(
            GeneralQuestion.id_general_questions == question_id
        ).first()
        if question and question.options:
            try:
                return question.options.index(answer_value) + 1
            except ValueError:
                return 0
    
```

```

        return 0
    finally:
        db.close()

# Предсказывает вероятности по сферам на основе ответов на общие
вопросы
def predict(general_answers: list):
    db = SessionLocal()
    try:
        # Получаем активные общие вопросы в порядке возрастания ID
        questions = db.query(GeneralQuestion).filter(
            GeneralQuestion.is_active == True
        ).order_by(GeneralQuestion.id_general_questions).all()

        questions_count = len(questions)

        # Получаем все сферы
        spheres =
db.query(Sphere).order_by(Sphere.id_spheres).all()
        sphere_ids = [s.id_spheres for s in spheres]
        sphere_count = len(sphere_ids)
    finally:
        db.close()

    # Если нет вопросов или сфер в БД – возвращаем пустой
результат
    if questions_count == 0:
        print("Нет активных общих вопросов в БД")
        return []

    if sphere_count == 0:
        print("Нет сфер в БД")
        return []

    # Проверяем количество полученных ответов
    if len(general_answers) != questions_count:
        print(f"Предупреждение: получено {len(general_answers)}
ответов, ожидалось {questions_count}")
        # Обрезаем или дополняем до нужного количества
        if len(general_answers) > questions_count:
            general_answers = general_answers[:questions_count]
        else:
            general_answers.extend([""] * (questions_count -
len(general_answers)))

    # Загружаем модель
    model = load_model()

    # Если модель не загружена, возвращаем равномерное
распределение
    if model is None:

```

```

        equal_prob = round(100 / sphere_count, 2)
        return [
            {"sphere_id": sphere_id, "probability": equal_prob}
            for sphere_id in sphere_ids
        ]

# Преобразуем ответы в индексы
numeric = []
for i, answer in enumerate(general_answers):
    question = questions[i]
    idx = _get_answer_index(question.id_general_questions,
answer)
    numeric.append(idx)

# Выполняем предсказание
X = np.array([numeric])
probabilities = model.predict_proba(X)[0]

# Сопоставляем классы модели с ID сфер
result = []
for cls, prob in zip(model.classes_, probabilities):
    result.append({
        "sphere_id": int(cls),
        "probability": round(prob * 100, 2)
    })

# Сортируем по убыванию вероятности
result.sort(key=lambda x: x["probability"], reverse=True)
return result

```

Файл «auth.py»

```

from datetime import datetime
from passlib.context import CryptContext
import os
from dotenv import load_dotenv

load_dotenv()

pwd_context = CryptContext(schemes=["bcrypt"], deprecated="auto")

def hash_password(password: str) -> str:
    return pwd_context.hash(password)

def verify_password(plain_password: str, hashed_password: str) -> bool:
    return pwd_context.verify(plain_password, hashed_password)

```

Файл «database.py»

```

from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

```

```

import os
from dotenv import load_dotenv

load_dotenv()

DATABASE_URL = os.getenv("DATABASE_URL")

engine = create_engine(DATABASE_URL)
SessionLocal = sessionmaker(autocommit=False, autoflush=False,
bind=engine)
Base = declarative_base()

def get_db():
    """Возвращает сессию базы данных."""
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()

```

Файл «main.py»

```

from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from database import engine
from models import Base
from routers.auth_router import router as auth_router
from routers.questions_router import router as questions_router
from routers.test_router import router as test_router
from routers.admin_router import router as admin_router

Base.metadata.create_all(bind=engine)

app = FastAPI(
    title="Профориентационный сервис",
    description="API для профориентационного тестирования",
    version="1.0.0"
)

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_methods=["*"],
    allow_headers=["*"],
)

app.include_router(auth_router)
app.include_router(questions_router)
app.include_router(test_router)
app.include_router(admin_router)

@app.get("/")
def root():

```

```
return {"message": "Сервер профориентационного тестирования  
работает"}
```

Файл «models.py»

```
from sqlalchemy import Column, Integer, String, Text, Boolean,  
Float, DateTime, JSON, ForeignKey, func  
from sqlalchemy.orm import relationship  
from database import Base  
  
class User(Base):  
    __tablename__ = "users"  
  
    id_users = Column(Integer, primary_key=True,  
autoincrement=True)  
    login = Column(String(50), unique=True, nullable=False)  
    password_hash = Column(String(255), nullable=False)  
    created_at = Column(DateTime, server_default=func.now())  
    role = Column(String(20), nullable=False, default="user")  
  
    test_sessions = relationship("TestSession",  
back_populates="user")  
  
class Sphere(Base):  
    __tablename__ = "spheres"  
  
    id_spheres = Column(Integer, primary_key=True,  
autoincrement=True)  
    name = Column(String(100), unique=True, nullable=False)  
    description = Column(Text)  
    max_score = Column(Integer, nullable=False)  
  
    question_spheres = relationship("QuestionSphere",  
back_populates="sphere")  
    full_test_results = relationship("FullTestResult",  
back_populates="sphere")  
    short_test_results = relationship("ShortTestResult",  
back_populates="sphere")  
  
class Question(Base):  
    __tablename__ = "questions"  
  
    id_questions = Column(Integer, primary_key=True,  
autoincrement=True)  
    text = Column(Text, nullable=False)  
    type = Column(String(10), nullable=False)  
    is_active = Column(Boolean, default=True)  
  
    question_spheres = relationship("QuestionSphere",  
back_populates="question")  
    answers = relationship("Answer", back_populates="question")  
  
class QuestionSphere(Base):
```

```

__tablename__ = "question_sphere"

id_question_sphere = Column(Integer, primary_key=True,
autoincrement=True)
question_id = Column(Integer,
ForeignKey("questions.id_questions", ondelete="CASCADE"),
nullable=False)
sphere_id = Column(Integer, ForeignKey("spheres.id_spheres",
ondelete="CASCADE"), nullable=False)

question = relationship("Question",
back_populates="question_spheres")
sphere = relationship("Sphere",
back_populates="question_spheres")

class GeneralQuestion(Base):
__tablename__ = "general_questions"

id_general_questions = Column(Integer, primary_key=True,
autoincrement=True)
text = Column(Text, nullable=False)
options = Column(JSON, nullable=False)
is_active = Column(Boolean, default=True)

general_answers = relationship("GeneralAnswer",
back_populates="question")

class TestSession(Base):
__tablename__ = "test_sessions"

id_test_sessions = Column(Integer, primary_key=True,
autoincrement=True)
user_id = Column(Integer, ForeignKey("users.id_users",
ondelete="CASCADE"), nullable=False)
test_type = Column(String(10), nullable=False)
started_at = Column(DateTime, server_default=func.now())
completed_at = Column(DateTime)

user = relationship("User", back_populates="test_sessions")
answers = relationship("Answer", back_populates="session")
general_answers = relationship("GeneralAnswer",
back_populates="session")
full_test_results = relationship("FullTestResult",
back_populates="session")
short_test_results = relationship("ShortTestResult",
back_populates="session")
ml_training_data = relationship("MLTrainingData",
back_populates="session") # ← убрали , uselist=False

class Answer(Base):
__tablename__ = "answers"

```

```

        id_answers = Column(Integer, primary_key=True,
autoincrement=True)
        session_id = Column(Integer,
ForeignKey("test_sessions.id_test_sessions", ondelete="CASCADE"),
nullable=False)
        question_id = Column(Integer,
ForeignKey("questions.id_questions", ondelete="CASCADE"),
nullable=False)
        answer_value = Column(Integer, nullable=False)

        session = relationship("TestSession",
back_populates="answers")
        question = relationship("Question", back_populates="answers")

class GeneralAnswer(Base):
    __tablename__ = "general_answers"

    id_general_answers = Column(Integer, primary_key=True,
autoincrement=True)
    session_id = Column(Integer,
ForeignKey("test_sessions.id_test_sessions", ondelete="CASCADE"),
nullable=False)
    question_id = Column(Integer,
ForeignKey("general_questions.id_general_questions",
ondelete="CASCADE"), nullable=False)
    answer_value = Column(String(255), nullable=False)
    answer_index = Column(Integer, nullable=True)

    session = relationship("TestSession",
back_populates="general_answers")
    question = relationship("GeneralQuestion",
back_populates="general_answers")

class FullTestResult(Base):
    __tablename__ = "full_test_results"

    id_full_test_results = Column(Integer, primary_key=True,
autoincrement=True)
    session_id = Column(Integer,
ForeignKey("test_sessions.id_test_sessions", ondelete="CASCADE"),
nullable=False)
    sphere_id = Column(Integer, ForeignKey("spheres.id_spheres",
ondelete="CASCADE"), nullable=False)
    score = Column(Integer, nullable=False)
    inclination_level = Column(String(20), nullable=False)

    session = relationship("TestSession",
back_populates="full_test_results")

```

```

    sphere = relationship("Sphere",
back_populates="full_test_results")

class ShortTestResult(Base):
    __tablename__ = "short_test_results"

    id_short_test_results = Column(Integer, primary_key=True,
autoincrement=True)
    session_id = Column(Integer,
ForeignKey("test_sessions.id_test_sessions", ondelete="CASCADE"),
nullable=False)
    sphere_id = Column(Integer, ForeignKey("spheres.id_spheres",
ondelete="CASCADE"), nullable=False)
    probability = Column(Float, nullable=False)

    session = relationship("TestSession",
back_populates="short_test_results")
    sphere = relationship("Sphere",
back_populates="short_test_results")

class MLTrainingData(Base):
    __tablename__ = "ml_training_data"

    id_ml_training_data = Column(Integer, primary_key=True,
autoincrement=True)
    session_id = Column(Integer,
ForeignKey("test_sessions.id_test_sessions", ondelete="CASCADE"),
nullable=False) # ← убрали unique=True
    features = Column(JSON, nullable=False)
    target_sphere_id = Column(Integer,
ForeignKey("spheres.id_spheres", ondelete="CASCADE"),
nullable=False)

    session = relationship("TestSession",
back_populates="ml_training_data")
    target_sphere = relationship("Sphere")

class MLModel(Base):
    __tablename__ = "ml_models"

    id_ml_models = Column(Integer, primary_key=True,
autoincrement=True)
    version = Column(String(20), unique=True, nullable=False)
    model_file = Column(String(255), nullable=False)
    accuracy = Column(Float, nullable=False)
    trained_at = Column(DateTime, server_default=func.now())
    is_active = Column(Boolean, default=False)
    features_importance = Column(JSON)

```

Файл «schemas.py»

```
from pydantic import BaseModel, Field
```

					ПГУ 1.090301.15.001 ПЗ	Лист
						125
Изм.	Лист	№ докум.	Подпись	Дата		

```

from typing import Optional

# Схемы для авторизации
class UserRegister(BaseModel):
    login: str = Field(..., min_length=3, max_length=50,
examples=["ivanov"])
    password: str = Field(..., min_length=6, max_length=100,
examples=["password123"])

class UserLogin(BaseModel):
    login: str = Field(..., examples=["ivanov"])
    password: str = Field(..., examples=["password123"])

class UserResponse(BaseModel):
    id: int
    login: str
    role: str

    class Config:
        from_attributes = True

    @classmethod
    def from_orm_with_mapping(cls, obj):
        return cls(
            id=obj.id_users,
            login=obj.login,
            role=obj.role
        )

# Схемы для вопросов
class QuestionResponse(BaseModel):
    id: int
    text: str
    type: str
    spheres: list[str]

    class Config:
        from_attributes = True

class GeneralQuestionResponse(BaseModel):
    id: int
    text: str
    options: list

# Схемы для тестирования
class AnswerSubmit(BaseModel):
    question_id: int
    answer_value: int = Field(..., ge=1, le=5)

class GeneralAnswerSubmit(BaseModel):

```

```

    question_id: int
    answer_value: str

class FullTestSubmit(BaseModel):
    answers: list[AnswerSubmit]
    general_answers: list[GeneralAnswerSubmit]

class FullTestResultResponse(BaseModel):
    sphere_id: int
    sphere_name: str
    score: int
    inclination_level: str

class FullTestResultsResponse(BaseModel):
    session_id: int
    results: list[FullTestResultResponse]

class ShortTestSubmit(BaseModel):
    general_answers: list[GeneralAnswerSubmit]

class ShortTestResultResponse(BaseModel):
    sphere_id: int
    sphere_name: str
    probability: float

class ShortTestResultsResponse(BaseModel):
    session_id: int
    results: list[ShortTestResultResponse]

# Схемы для админ-панели
class SphereEdit(BaseModel):
    name: str = Field(..., min_length=3, max_length=100)
    description: Optional[str] = None
    max_score: int = Field(..., gt=0)

class QuestionEdit(BaseModel):
    text: str = Field(..., min_length=5)
    type: str = Field(..., pattern="^(explicit|hidden)$")
    spheres: list[int] = Field(..., min_length=1)

class GeneralQuestionEdit(BaseModel):
    text: str = Field(..., min_length=5)
    options: list = Field(..., min_length=2)

```

Файл «AdminPage.jsx»

```

import { useState, useEffect } from 'react';
import { useNavigate } from 'react-router-dom';
import api from '../services/api';
import '../styles/global.css';

```

					ПГУ 1.090301.15.001 ПЗ	Лист
						127
Изм.	Лист	№ докум.	Подпись	Дата		

```

function AdminPage() {
  const [tab, setTab] = useState('stats');
  const [spheres, setSpheres] = useState([]);
  const [questions, setQuestions] = useState([]);
  const [generalQuestions, setGeneralQuestions] = useState([]);
  const [stats, setStats] = useState({});
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState('');
  const [success, setSuccess] = useState('');
  const [training, setTraining] = useState(false);
  const navigate = useNavigate();

  const [editSphere, setEditSphere] = useState(null);
  const [editQuestion, setEditQuestion] = useState(null);
  const [editGeneral, setEditGeneral] = useState(null);

  useEffect(() => {
    loadAll();
  }, []);

  const loadAll = async () => {
    setLoading(true);
    try {
      const [sRes, qRes, gRes, statsRes] = await
Promise.all([
        api.get('/admin/spheres'),
        api.get('/admin/questions'),
        api.get('/admin/general-questions'),
        api.get('/admin/stats'),
      ]);
      setSpheres(sRes.data);
      setQuestions(qRes.data);
      setGeneralQuestions(gRes.data);
      setStats(statsRes.data);
    } catch (err) {
      setError('Ошибка загрузки данных');
    } finally {
      setLoading(false);
    }
  };

  const showSuccess = (msg) => {
    setSuccess(msg);
    setTimeout(() => setSuccess(''), 3000);
  };

  const handleTrainModel = async () => {
    if (!confirm('Запустить обучение модели на текущих
данных?')) return;
    setTraining(true);
    setError('');
  };

```

```

        try {
            const response = await api.post('/admin/train-model');
            showSuccess(`Модель обучена! Точность:
${(response.data.accuracy * 100).toFixed(1)}%`);
        } catch (err) {
            setError(err.response?.data?.detail || 'Ошибка
обучения');
        } finally {
            setTraining(false);
        }
    };

    // ===== SPHERES =====
    const saveSphere = async () => {
        try {
            if (editSphere.id) await
api.put(`/admin/spheres/${editSphere.id}`, editSphere);
            else await api.post('/admin/spheres', editSphere);
            setEditSphere(null);
            showSuccess('Сфера сохранена');
            loadAll();
        } catch (err) {
            setError(err.response?.data?.detail || 'Ошибка
сохранения');
        }
    };

    const deleteSphere = async (id) => {
        if (!confirm('Удалить сферу?')) return;
        try {
            await api.delete(`/admin/spheres/${id}`);
            showSuccess('Сфера удалена');
            loadAll();
        } catch (err) {
            setError(err.response?.data?.detail || 'Ошибка
удаления');
        }
    };

    // ===== QUESTIONS =====
    const saveQuestion = async () => {
        try {
            if (editQuestion.id) await
api.put(`/admin/questions/${editQuestion.id}`, editQuestion);
            else await api.post('/admin/questions', editQuestion);
            setEditQuestion(null);
            showSuccess('Вопрос сохранён');
            loadAll();
        } catch (err) {
            setError(err.response?.data?.detail || 'Ошибка
сохранения');
        }
    };

```

```

    }
  };

  const deleteQuestion = async (id) => {
    if (!confirm('Удалить вопрос?')) return;
    try {
      await api.delete(`/admin/questions/${id}`);
      showSuccess('Вопрос удалён');
      loadAll();
    } catch (err) {
      setError(err.response?.data?.detail || 'Ошибка
удаления');
    }
  };

  const toggleQuestion = async (id) => {
    try {
      await api.patch(`/admin/questions/${id}/toggle`);
      showSuccess('Статус изменён');
      loadAll();
    } catch (err) {
      setError(err.response?.data?.detail || 'Ошибка');
    }
  };

  // ===== GENERAL QUESTIONS =====
  const saveGeneral = async () => {
    try {
      const data = {
        text: editGeneral.text,
        options: editGeneral.options.split(';').map(s =>
s.trim()).filter(s => s),
      };
      if (editGeneral.id) await api.put(`/admin/general-
questions/${editGeneral.id}`, data);
      else await api.post('/admin/general-questions', data);
      setEditGeneral(null);
      showSuccess('Общий вопрос сохранён');
      loadAll();
    } catch (err) {
      setError(err.response?.data?.detail || 'Ошибка
сохранения');
    }
  };

  const deleteGeneral = async (id) => {
    if (!confirm('Удалить общий вопрос?')) return;
    try {
      await api.delete(`/admin/general-questions/${id}`);
      showSuccess('Общий вопрос удалён');
      loadAll();
    }
  };

```

```

    } catch (err) {
      setError(err.response?.data?.detail || 'Ошибка
удаления');
    }
  };

const AnimatedNumber = ({ target, duration = 1000 }) => {
  const [current, setCurrent] = useState(0);
  useEffect(() => {
    if (target === 0) { setCurrent(0); return; }
    const steps = 60;
    const interval = duration / steps;
    const increment = target / steps;
    let step = 0;
    const timer = setInterval(() => {
      step++;
      if (step >= steps) { setCurrent(target);
clearInterval(timer); }
      else setCurrent(Math.round(increment * step));
    }, interval);
    return () => clearInterval(timer);
  }, [target]);
  return <>{current}</>;
};

const EditIcon = () => (
  <svg width="18" height="18" viewBox="0 0 24 24"
fill="none" stroke="currentColor" strokeWidth="2"
strokeLinecap="round" strokeLinejoin="round">
    <path d="M11 4H4a2 2 0 0 0-2 2v14a2 2 0 0 0 2 2h14a2 2
0 0 0 2-2v-7"/>
    <path d="M18.5 2.5a2.121 2.121 0 0 1 3 3L12 15l-4 1 1-
4 9.5-9.5z"/>
  </svg>
);

const DeleteIcon = () => (
  <svg width="18" height="18" viewBox="0 0 24 24"
fill="none" stroke="currentColor" strokeWidth="2"
strokeLinecap="round" strokeLinejoin="round">
    <polyline points="3 6 5 6 21 6"/>
    <path d="M19 6v14a2 2 0 0 1-2 2H7a2 2 0 0 1-2 2V6m3
0V4a2 2 0 0 1 2-2h4a2 2 0 0 1 2 2v2"/>
    <line x1="10" y1="11" x2="10" y2="17"/>
    <line x1="14" y1="11" x2="14" y2="17"/>
  </svg>
);

if (loading) return <div className="admin-wrapper"><div
className="admin-container">Загрузка...</div></div>;

```

```

return (
  <div className="admin-wrapper">
    <div className="admin-container">
      <h1>Админ-панель</h1>
      {success && <div className="admin-
success">{success}</div>}
      {error && <div className="admin-
error">{error}</div>}

      <div className="admin-tabs">
        <button onClick={() => setTab('stats')}
className={`admin-tab ${tab === 'stats' ? 'admin-tab-active' :
''}`}>Статистика</button>
        <button onClick={() => setTab('spheres')}
className={`admin-tab ${tab === 'spheres' ? 'admin-tab-active' :
''}`}>Сферы</button>
        <button onClick={() => setTab('questions')}
className={`admin-tab ${tab === 'questions' ? 'admin-tab-active' :
''}`}>Вопросы</button>
        <button onClick={() => setTab('general')}
className={`admin-tab ${tab === 'general' ? 'admin-tab-active' :
''}`}>Общие вопросы</button>
      </div>

      {/* ===== STATS ===== */}
      {tab === 'stats' && (
        <>
          <div className="admin-stats-grid">
            <div className="admin-stat-card">
              <div className="admin-stat-
number"><AnimatedNumber target={stats.total_tests || 0} /></div>
              <div className="admin-stat-
label">Всего тестов</div>
            </div>
            <div className="admin-stat-card">
              <div className="admin-stat-
number"><AnimatedNumber target={stats.total_full || 0} /></div>
              <div className="admin-stat-
label">Полных тестов</div>
            </div>
            <div className="admin-stat-card">
              <div className="admin-stat-
number"><AnimatedNumber target={stats.total_short || 0} /></div>
              <div className="admin-stat-
label">Коротких тестов</div>
            </div>
            <div className="admin-stat-card">
              <div className="admin-stat-
number"><AnimatedNumber target={stats.total_users || 0} /></div>
              <div className="admin-stat-
label">Пользователей</div>
            </div>
          </div>
        </>
      )}
    </div>
  </div>
)

```

```

        </div>
    </div>

    <h3 style={{ marginTop: 40, marginBottom:
20, color: '#1f5650', fontSize: 28, textAlign: 'center'
}}>Последние тесты</h3>

    <table className="admin-table">

    <thead><tr><th>ID</th><th>Пользователь</th><th>Тип</th><th>Дата</t
h></tr></thead>

        <tbody>
            {stats.recent_tests?.map((t, i) =>
(
                <tr key={i}>
                    <td>{t.id}</td>
                    <td>{t.login}</td>
                    <td>{t.type === 'full' ?
'Полный' : 'Короткий'}</td>
                    <td>{new
Date(t.date).toLocaleString('ru-RU')}</td>
                </tr>
            )
        )}
        </tbody>
    </table>

    <button onClick={handleTrainModel}
className="admin-train-btn" disabled={training}>
        {training ? 'Обучение...' : 'Обучить
модель'}
    </button>
</>
)}

{ /* ===== SPHERES ===== */
{tab === 'spheres' && (
    <>
        <button onClick={() => setEditSphere({
name: '', description: '', max_score: 34 })} className="admin-add-
btn">+ Добавить сферу</button>
        <table className="admin-table">

        <thead><tr><th>ID</th><th>Название</th><th>Max
баллов</th><th></th></tr></thead>
            <tbody>
                {spheres.map(s => (
                    <tr key={s.id_spheres}>
                        <td>{s.id_spheres}</td>
                        <td>{s.name}</td>
                        <td>{s.max_score}</td>
                        <td>

```

```

                                <button onClick={() =>
setEditSphere({ id: s.id_spheres, name: s.name, description:
s.description, max_score: s.max_score })} className="admin-small-
btn" title="Редактировать"><EditIcon /></button>
                                <button onClick={() =>
deleteSphere(s.id_spheres)} className="admin-danger-btn"
title="Удалить"><DeleteIcon /></button>
                                </td>
                            </tr>
                        }}}
                    </tbody>
                </table>
            </>
        )}

        { /* ===== QUESTIONS ===== */
        { tab === 'questions' && (
            <>
                <button onClick={() => setEditQuestion({
text: '', type: 'explicit', spheres: [] })} className="admin-add-
btn">+ Добавить вопрос</button>
                <table className="admin-table">

<thead><tr><th>ID</th><th>Текст</th><th>Тип</th><th>Активен</th><t
h></th></tr></thead>

                <tbody>
                    {questions.map(q => (
                        <tr key={q.id} style={{
opacity: q.is_active ? 1 : 0.5 }}>
                            <td>{q.id}</td>
                            <td style={{ maxWidth:
500, overflow: 'hidden', textOverflow: 'ellipsis', whiteSpace:
'nowrap', fontSize: 24 }}>{q.text}</td>
                            <td>{q.type === 'explicit'
? 'ЯВНЫЙ' : 'Скрытый'}</td>
                            <td>
                                <button onClick={() =>
toggleQuestion(q.id)} className="admin-small-btn"
                                style={{
backgroundColor: q.is_active ? '#8aab7c' : '#d4917a', color:
'#fdf9f3' }}>
                                    {q.is_active ?
'ON' : 'OFF'}
                                </button>
                            </td>
                        </tr>
                    )}
                    <tr>
                        <td>
                            <button onClick={() =>
setEditQuestion({ id: q.id, text: q.text, type: q.type, spheres:
q.spheres })} className="admin-small-btn"
                            title="Редактировать"><EditIcon /></button>

```

```

deleteQuestion(q.id)} className="admin-danger-btn"
title="Удалить"><DeleteIcon /></button>
</td>
</tr>
)}}
</tbody>
</table>
</>
)}

{/* ===== GENERAL QUESTIONS ===== */}
{tab === 'general' && (
  <>
    <button onClick={() => setEditGeneral({
text: '', options: '' })} className="admin-add-btn">+ Добавить
общий вопрос</button>
    <table className="admin-table">
      <thead>
        <tr><th style={{ width: '10%'
}}>ID</th><th style={{ width: '30%' }}>Текст</th><th style={{
width: '45%' }}>Варианты (через ;)</th><th style={{ width: '15%'
}}></th></tr>
        </thead>
        <tbody>
          {generalQuestions.map(gq => (
            <tr
key={gq.id_general_questions}>
            <td>{gq.id_general_questions}</td>
            <td style={{ maxWidth:
200, overflow: 'hidden', textOverflow: 'ellipsis', whiteSpace:
'nowrap', fontSize: 24 }}>{gq.text}</td>
            <td style={{ fontSize: 24,
wordBreak: 'break-word' }}>{gq.options?.join('; ')}</td>
            <td>
              <button onClick={() =>
setEditGeneral({ id: gq.id_general_questions, text: gq.text,
options: gq.options?.join('; ') })} className="admin-small-btn"
title="Редактировать"><EditIcon /></button>
              <button onClick={() =>
deleteGeneral(gq.id_general_questions)} className="admin-danger-
btn" title="Удалить"><DeleteIcon /></button>
            </td>
          </tr>
          )}}
        </tbody>
      </table>
    </>
  )}
}

```

```

        <button className="admin-back-btn" onClick={() =>
navigate('/menu')}>Назад в меню</button>
    </div>

    {/* ===== EDIT MODALS ===== */}
    {editSphere && (
        <div className="admin-modal">
            <h3>{editSphere.id ? 'Редактировать сферу' :
'Новая сфера'}</h3>
                <input value={editSphere.name} onChange={e =>
setEditSphere({ ...editSphere, name: e.target.value })}
placeholder="Название" className="admin-input" />
                <input value={editSphere.description || ''}
onChange={e => setEditSphere({ ...editSphere, description:
e.target.value })} placeholder="Описание" className="admin-input"
/>
                <input type="number"
value={editSphere.max_score} onChange={e => setEditSphere({
...editSphere, max_score: parseInt(e.target.value) })}
placeholder="Макс. баллов" className="admin-input" />
                <button onClick={saveSphere} className="admin-
save-btn">Сохранить</button>
                <button onClick={() => setEditSphere(null)}
className="admin-cancel-btn">Отмена</button>
            </div>
        )}

    {editQuestion && (
        <div className="admin-modal">
            <h3>{editQuestion.id ? 'Редактировать вопрос'
: 'Новый вопрос'}</h3>
                <textarea value={editQuestion.text}
onChange={e => setEditQuestion({ ...editQuestion, text:
e.target.value })} placeholder="Текст вопроса" className="admin-
textarea" />
                <select value={editQuestion.type} onChange={e
=> setEditQuestion({ ...editQuestion, type: e.target.value })}
className="admin-input">
                    <option value="explicit">ЯВНЫЙ</option>
                    <option value="hidden">Скрытый</option>
                </select>
                <div style={{ marginTop: 10 }}>
                    <strong>Сферы (ID через запятую):</strong>
                    <input
value={editQuestion.spheres?.join(', ')}
onChange={e => setEditQuestion({
...editQuestion, spheres: e.target.value.split(',').map(s =>
parseInt(s.trim())) .filter(n => !isNaN(n)) }}
placeholder="Например: 1, 4, 5"
className="admin-input" />
                </div>
            </div>
        )}

```

```

        <button onClick={saveQuestion}
className="admin-save-btn">Сохранить</button>
        <button onClick={() => setEditQuestion(null)}
className="admin-cancel-btn">Отмена</button>
      </div>
    )}

    {editGeneral && (
      <div className="admin-modal">
        <h3>{editGeneral.id ? 'Редактировать общий
вопрос' : 'Новый общий вопрос'}</h3>
        <textarea value={editGeneral.text} onChange={e
=> setEditGeneral({ ...editGeneral, text: e.target.value })}
placeholder="Текст вопроса" className="admin-textarea" />
        <input value={editGeneral.options} onChange={e
=> setEditGeneral({ ...editGeneral, options: e.target.value })}
placeholder="Варианты через точку с запятой" className="admin-
input" />
        <button onClick={saveGeneral}
className="admin-save-btn">Сохранить</button>
        <button onClick={() => setEditGeneral(null)}
className="admin-cancel-btn">Отмена</button>
      </div>
    )}
  </div>
);
}

export default AdminPage;

```

Файл «FullTestPage.jsx»

```

import { useState, useEffect } from 'react';
import { useNavigate } from 'react-router-dom';
import api from '../services/api';
import '../styles/global.css';

function FullTestPage() {
  const [questions, setQuestions] = useState([]);
  const [generalQuestions, setGeneralQuestions] = useState([]);
  const [current, setCurrent] = useState(0);
  const [answers, setAnswers] = useState({});
  const [generalAnswers, setGeneralAnswers] = useState({});
  const [loading, setLoading] = useState(true);
  const [stage, setStage] = useState('expert');
  const [submitting, setSubmitting] = useState(false);
  const [error, setError] = useState('');
  const navigate = useNavigate();

  useEffect(() => {
    loadQuestions();
  }, []);

```

```

useEffect(() => {
  const handleKeyDown = (e) => {
    const key = e.key;
    if (key >= '1' && key <= '5') {
      const num = parseInt(key);
      if (stage === 'expert' && questions[current]) {
        handleSelect(questions[current].id, num);
        if (current < questions.length - 1) {
          setTimeout(() => setCurrent(current + 1),
150);
        }
      }
    }
    if (key === 'ArrowLeft' && current > 0)
setCurrent(current - 1);
    if (key === 'ArrowRight') {
      if (stage === 'expert' && current <
questions.length - 1 && answers[questions[current]?.id]) {
        setCurrent(current + 1);
      } else if (stage === 'general' && current <
generalQuestions.length - 1 &&
generalAnswers[generalQuestions[current]?.id]) {
        setCurrent(current + 1);
      }
    }
  };
  window.addEventListener('keydown', handleKeyDown);
  return () => window.removeEventListener('keydown',
handleKeyDown);
}, [current, stage, questions, generalQuestions, answers,
generalAnswers]);

const loadQuestions = async () => {
  try {
    const [expertRes, generalRes] = await Promise.all([
      api.get('/questions/expert'),
      api.get('/questions/general'),
    ]);
    setQuestions(expertRes.data);
    setGeneralQuestions(generalRes.data);
  } catch (err) {
    setError('Ошибка загрузки вопросов');
  } finally {
    setLoading(false);
  }
};

const handleSelect = (questionId, value) => {
  setAnswers({ ...answers, [questionId]: value });
};

```

```

const handleGeneralSelect = (questionId, value) => {
  setGeneralAnswers({ ...generalAnswers, [questionId]: value
});
};

const handleNext = () => {
  if (current < questions.length - 1) setCurrent(current +
1);
};

const handlePrev = () => {
  if (current > 0) setCurrent(current - 1);
};

const handleFinishExpert = () => {
  setCurrent(0);
  setStage('general');
};

const handleSubmit = async () => {
  setSubmitting(true);
  setError('');
  try {
    const answersList =
Object.entries(answers).map(([question_id, answer_value]) => ({
  question_id: parseInt(question_id),
  answer_value,
})));
    const generalList =
Object.entries(generalAnswers).map(([question_id, answer_value])
=> ({
  question_id: parseInt(question_id),
  answer_value,
})));
    const response = await api.post('/test/full', {
      answers: answersList,
      general_answers: generalList,
    });
    navigate('/result/full', { state: { results:
response.data.results } });
  } catch (err) {
    setError('Ошибка отправки ответов: ' +
(err.response?.data?.detail || ''));
  } finally {
    setSubmitting(false);
  }
};

if (loading) return <div className="full-test-wrapper"><div
className="full-test-container">Загрузка...</div></div>;

```

```

    if (stage === 'expert') {
      const q = questions[current];
      const isLast = current === questions.length - 1;
      const isAnswered = answers[q?.id] !== undefined;
      const allAnswered = questions.every(q => answers[q.id] !==
undefined);

      return (
        <div className="full-test-wrapper">
          <div className="full-test-container">
            <h2>Полный тест</h2>
            <div className="full-test-stage">Основные
вопросы</div>
            <div className="full-test-progress">Вопрос
{current + 1} из {questions.length}</div>
            <div className="full-test-progress-bar">
              <div className="full-test-progress-fill"
style={{ width: `${((current + 1) / questions.length) * 100}%` }}
/>
            </div>
            {error && <p className="full-test-
error">{error}</p>}

            <div className="full-test-question">
              <h3>{q?.text}</h3>
              <p className="full-test-hint">Оцените
утверждение по шкале от 1 до 5</p>
              <div className="full-test-scale">
                {[1, 2, 3, 4, 5].map(num => (
                  <button
                    key={num}
                    onClick={() =>
handleSelect(q.id, num)}
                    className={`full-test-scale-
button ${answers[q.id] === num ? 'full-test-scale-selected' :
''}`}
                  >
                    {num}
                  </button>
                ))}
              </div>
              <div className="full-test-scale-labels">
                <span>Категорически не согласен</span>
                <span>Полностью согласен</span>
              </div>
            </div>

            <div className="full-test-nav">
              <button onClick={handlePrev}
disabled={current === 0}

```

```

        className="full-test-nav-button full-
test-nav-back">← Назад</button>
        {isLast ? (
            <button onClick={handleFinishExpert}
disabled={!allAnswered}
            className="full-test-nav-button
full-test-nav-general">
                К общим вопросам →
            </button>
        ) : (
            <button onClick={handleNext}
disabled={!isAnswered}
            className="full-test-nav-button
full-test-nav-next">Далее →</button>
        )}
    </div>
</div>
</div>
);
}

if (stage === 'general') {
    const gq = generalQuestions[current];
    const isLast = current === generalQuestions.length - 1;
    const isAnswered = generalAnswers[gq?.id] !== undefined;
    const allAnswered = generalQuestions.every(q =>
generalAnswers[q.id] !== undefined);

    return (
        <div className="full-test-wrapper">
            <div className="full-test-container">
                <h2>Полный тест</h2>
                <div className="full-test-stage">Общие
вопросы</div>
                <div className="full-test-progress">Вопрос
{current + 1} из {generalQuestions.length}</div>
                {error && <p className="full-test-
error">{error}</p>}

                <div className="full-test-question">
                    <h3>{gq?.text}</h3>
                    <div className="full-test-options">
                        {gq?.options?.map((option, idx) => (
                            <div
                                key={idx}
                                className={`full-test-option
${generalAnswers[gq.id] === option ? 'full-test-option-selected' :
''}`}
                                onClick={() =>
handleGeneralSelect(gq.id, option)}
                            >

```

```

                                {option}
                                </div>
                            }}}
                        </div>
                    </div>

                    <div className="full-test-nav">
                        <button onClick={handlePrev}
disabled={current === 0}
                                className="full-test-nav-button full-
test-nav-back">← Назад</button>
                            {isLast ? (
                                <button onClick={handleSubmit}
disabled={!allAnswered || submitting}
                                className="full-test-nav-button
full-test-nav-submit">
                                    {submitting ? 'Отправка...' :
'Завершить тест'}
                                </button>
                            ) : (
                                <button onClick={handleNext}
disabled={!isAnswered}
                                className="full-test-nav-button
full-test-nav-next">Далее →</button>
                            )}
                        </div>
                    </div>
                </div>
            );
        }
    }

export default FullTestPage;

```

Файл «HistoryPage.jsx»

```

import { useState, useEffect } from 'react';
import { useNavigate, useLocation } from 'react-router-dom';
import api from '../services/api';
import '../styles/global.css';

function HistoryPage() {
    const [history, setHistory] = useState([]);
    const [loading, setLoading] = useState(true);
    const [error, setError] = useState('');
    const [expanded, setExpanded] = useState(null);
    const navigate = useNavigate();
    const location = useLocation();
    const info = location.state?.info || {};

    useEffect(() => {

```

```

        loadHistory();
    }, []);

const loadHistory = async () => {
    try {
        const response = await api.get('/test/history');
        setHistory(response.data);
    } catch (err) {
        setError('Ошибка загрузки истории');
    } finally {
        setLoading(false);
    }
};

const formatDate = (dateStr) => {
    if (!dateStr) return '-';
    return new Date(dateStr).toLocaleString('ru-RU');
};

// Цвет для полного теста (по проценту баллов)
const getScoreColor = (score, maxPossibleScore) => {
    const percent = (score / maxPossibleScore) * 100;
    if (percent <= 25) return '#c47a5a';
    if (percent <= 50) return '#c4a35a';
    if (percent <= 75) return '#8a9b6e';
    return '#637643';
};

// Цвет для короткого теста (по доле относительно максимальной
вероятности)
const getProbabilityColor = (probability, maxProbability) => {
    const percent = (probability / maxProbability) * 100;
    if (percent <= 25) return '#c47a5a';
    if (percent <= 50) return '#c4a35a';
    if (percent <= 75) return '#8a9b6e';
    return '#637643';
};

const expertCount = info.expert_questions_count || 151;
const generalCount = info.general_questions_count || 13;
const maxPossibleScore = info.max_possible_score || 34;

if (loading) return <div className="history-
container">Загрузка...</div>;

return (
    <div className="history-container">
        <h1>История тестов</h1>
        {error && <p className="auth-error">{error}</p>}

        {history.length === 0 && !loading && (

```

```

        <p style={{ textAlign: 'center', color: '#8a7e6b',
fontSize: '20px' }}>Вы ещё не проходили тесты.</p>
    )}

```

```

    {history.map((test, idx) => {
      // Для короткого теста вычисляем максимальную
вероятность
      let maxProbability = 1;
      if (test.test_type === 'short' &&
test.results?.length > 0) {
        maxProbability =
Math.max(...test.results.map(r => r.probability));
      }

      return (
        <div key={idx} className="history-card">
          <div className="history-card-header"
            onClick={() => setExpanded(expanded
=== idx ? null : idx)}>
            <div className="history-card-left">
              <span className={`history-
indicator ${test.test_type === 'full' ? 'indicator-full' :
'indicator-short'}}` />
              <span className="history-card-
title">
                {test.test_type === 'full' ?
'Полный тест' : 'Короткий тест'}
              </span>
              <span className={`history-type-
badge ${test.test_type === 'full' ? 'badge-full' : 'badge-
short'}}`>
                {test.test_type === 'full' ?
`${expertCount} вопросов` : `${generalCount} вопросов`}
              </span>
            </div>
            <div className="history-card-right">
              <span className="history-card-
date">{formatDate(test.started_at)}</span>
              <span className={`history-arrow
${expanded === idx ? 'arrow-up' : 'arrow-down'}}` />
            </div>
          </div>
          {expanded === idx && (
            <div className="history-card-body">
              {test.test_type === 'full' ? (
                test.results?.map((r, i) => (
                  <div key={i}
className="history-result-row">
                    <span
className="history-sphere-name">{r.sphere_name}</span>

```

```

                                <span
className="history-badge"
                                style={{
backgroundColor: getScoreColor(r.score, maxPossibleScore) }}>
                                {r.score} баллов –
{r.inclination_level}
                                </span>
                                </div>
                                ))
                                ) : (
                                test.results?.map((r, i) => (
                                <div key={i}
className="history-result-row">
                                <span
className="history-sphere-name">{r.sphere_name}</span>
                                <span
className="history-probability"
                                style={{
backgroundColor: getProbabilityColor(r.probability,
maxProbability) }}>
                                {r.probability.toFixed(0)}%
                                </span>
                                </div>
                                ))
                                )}
                                </div>
                                )}
                                </div>
                                );
                                }}}

                                <button className="history-back-btn" onClick={() =>
navigate('/menu')}>
                                Вернуться в меню
                                </button>
                                </div>
                                );
                                }

```

```
export default HistoryPage;
```

Файл «LoginPage.jsx»

```

import { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import api from '../services/api';
import '../styles/global.css';

function LoginPage() {
  const [login, setLogin] = useState('');
  const [password, setPassword] = useState('');

```

```

const [error, setError] = useState('');
const navigate = useNavigate();

const handleSubmit = async (e) => {
  e.preventDefault();
  setError('');
  try {
    const response = await api.post('/auth/login', {
login, password });
    localStorage.setItem('role', response.data.user?.role
|| 'user');
    localStorage.setItem('userId',
response.data.user?.id);
    navigate('/menu');
  } catch (err) {
    setError(err.response?.data?.detail || 'Ошибка
входа');
  }
};

return (
  <div className="auth-container">
    <div className="auth-title">Система
профориентационного тестирования</div>
    <h1>Вход в систему</h1>
    {error && <p className="auth-error">{error}</p>}
    <form onSubmit={handleSubmit} className="auth-form">
      <input type="text" placeholder="Логин"
value={login}
      onChange={(e) => setLogin(e.target.value)}
className="auth-input" required />
      <input type="password" placeholder="Пароль"
value={password}
      onChange={(e) => setPassword(e.target.value)}
className="auth-input" required />
      <button type="submit" className="auth-
button">Войти</button>
    </form>
    <p className="auth-bottom-text">Нет аккаунта? <a
href="/register" className="auth-link">Зарегистрироваться</a></p>
  </div>
);
}

export default LoginPage;

```

Файл «MenuPage.jsx»

```

import { useState, useEffect } from 'react';
import { useNavigate } from 'react-router-dom';
import api, { getInfo } from '../services/api';
import '../styles/global.css';

```

```

function MenuPage() {
  const navigate = useNavigate();
  const role = localStorage.getItem('role') || 'user';

  const [info, setInfo] = useState({
    expert_questions_count: 151,
    general_questions_count: 13,
    max_possible_score: 34,
    loading: true
  });

  useEffect(() => {
    const loadInfo = async () => {
      try {
        const data = await getInfo();
        setInfo({
          ...data,
          loading: false
        });
      } catch (err) {
        console.error('Ошибка загрузки info:', err);
        setInfo(prev => ({ ...prev, loading: false }));
      }
    };
    loadInfo();
  }, []);

  const handleLogout = () => {
    localStorage.removeItem('token');
    localStorage.removeItem('role');
    localStorage.removeItem('userId');
    navigate('/login');
  };

  const handleFullTest = () => {
    navigate('/test/full', { state: { info } });
  };

  const handleShortTest = () => {
    navigate('/test/short', { state: { info } });
  };

  const handleHistory = () => {
    navigate('/history', { state: { info } });
  };

  const handleAdmin = () => {
    navigate('/admin', { state: { info } });
  };
}

```

```

const expertCount = info.expert_questions_count;
const generalCount = info.general_questions_count;

return (
  <div className="menu-wrapper">
    <div className="menu-container">
      <div className="auth-title">Система
профориентационного тестирования</div>

      <div className="menu-cards">
        <div className="menu-card"
onClick={handleFullTest}>
          <div className="menu-card-icon">
            <svg width="72" height="72" viewBox="0
0 24 24" fill="none" stroke="#fdf9f3" strokeWidth="1.2"
strokeLinecap="round" strokeLinejoin="round">
              <path d="M9 5H7a2 2 0 0 0 0-2 2v12a2
2 0 0 0 2 2h10a2 2 0 0 0 2-2V7a2 2 0 0 0 0-2-2h-2"/>
              <rect x="9" y="3" width="6"
height="4" rx="1"/>
              <path d="M9 12h6M9 16h6"/>
            </svg>
          </div>
          <div className="menu-card-content">
            <h3>Полный тест</h3>
            <p className="menu-card-desc">
Детальная оценка профессиональных
склонностей по сферам деятельности. Подходит для вдумчивого
прохождения и получения максимально точного результата.
            </p>
            <p className="menu-card-
time">{expertCount} вопросов · ~{Math.round(expertCount / 6)}
минут</p>
            <span className="menu-card-
btn">Пройти</span>
          </div>

          <div className="menu-card"
onClick={handleShortTest}>
            <div className="menu-card-icon">
              <svg width="72" height="72" viewBox="0
0 24 24" fill="none" stroke="#fdf9f3" strokeWidth="1.2"
strokeLinecap="round" strokeLinejoin="round">
                <circle cx="12" cy="12" r="10"/>
                <polyline points="12 6 12 12 16
14"/>
              </svg>
            </div>
            <div className="menu-card-content">
              <h3>Короткий тест</h3>

```

```

        <p className="menu-card-desc">
            Быстрая оценка профессиональных
            склонностей с помощью алгоритмов машинного обучения. Для тех, кто
            хочет получить результат здесь и сейчас.
        </p>
        <p className="menu-card-
time">{generalCount} вопросов · ~{Math.round(generalCount / 4)}
минут</p>
        </div>
        <span className="menu-card-
btn">Пройти</span>
    </div>
</div>
</div>

<div className="menu-bottom-container">
    <button onClick={handleHistory} className="menu-
button menu-button-purple">
        <svg className="menu-button-icon" viewBox="0 0
24 24" fill="none" stroke="#fdf9f3" strokeWidth="1.2"
strokeLinecap="round" strokeLinejoin="round">
            <circle cx="12" cy="12" r="10"/>
            <polyline points="12 6 12 12 16 14"/>
        </svg>
        История тестов
    </button>

    {role === 'admin' && (
        <button onClick={handleAdmin} className="menu-
button menu-button-orange">
            <svg className="menu-button-icon"
viewBox="0 0 24 24" fill="none" stroke="#fdf9f3" strokeWidth="1.2"
strokeLinecap="round" strokeLinejoin="round">
                <circle cx="12" cy="12" r="3"/>
                <path d="M19.4 15a1.65 1.65 0 0 0 -1.82-.33
1.821.06.06a2 2 0 0 1-2.83 2.831-.06-.06a1.65 1.65 0 0 0-1.82-.33
1.65 1.65 0 0 0-1 1.51V21a2 2 0 0 1-4 0v-.09A1.65 1.65 0 0 0 9
19.4a1.65 1.65 0 0 0-1.82.331-.06.06a2 2 0 0 1-2.83-2.831.06-
.06A1.65 1.65 0 0 0 4.68 15a1.65 1.65 0 0 0-1.51-1H3a2 2 0 0 1 0-
4h.09A1.65 1.65 0 0 0 4.6 9a1.65 1.65 0 0 0-.33-1.821-.06-.06a2 2
0 0 1 2.83-2.831.06.06A1.65 1.65 0 0 0 9 4.68a1.65 1.65 0 0 0 1-
1.51V3a2 2 0 0 1 4 0v.09a1.65 1.65 0 0 0 1 1.51 1.65 1.65 0 0 0
1.82-.331.06-.06a2 2 0 0 1 2.83 2.831-.06.06A1.65 1.65 0 0 0 19.4
9a1.65 1.65 0 0 0 1.51 1H21a2 2 0 0 1 0 4h-.09a1.65 1.65 0 0 0-
1.51 1z"/>
            </svg>
            Админ-панель
        </button>
    ) }

```

```

        <button onClick={handleLogout} className="menu-
button menu-button-red">
            <svg className="menu-button-icon" viewBox="0 0
24 24" fill="none" stroke="#fdf9f3" strokeWidth="1.2"
strokeLinecap="round" strokeLinejoin="round">
                <path d="M9 21H5a2 2 0 0 1-2-2V5a2 2 0 0 1
2-2h4"/>
                <polyline points="16 17 21 12 16 7"/>
                <line x1="21" y1="12" x2="9" y2="12"/>
            </svg>
            Выйти
        </button>
    </div>
</div>
);
}

export default MenuPage;

```

Файл «RegisterPage.jsx»

```

import { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import api from '../services/api';
import '../styles/global.css';

function RegisterPage() {
    const [login, setLogin] = useState('');
    const [password, setPassword] = useState('');
    const [error, setError] = useState('');
    const [success, setSuccess] = useState('');
    const navigate = useNavigate();

    const handleSubmit = async (e) => {
        e.preventDefault();
        setError('');
        setSuccess('');
        try {
            await api.post('/auth/register', { login, password });
            setSuccess('Регистрация успешна! Перенаправляем на
вход...');
            setTimeout(() => navigate('/login'), 1500);
        } catch (err) {
            setError(err.response?.data?.detail || 'Ошибка
регистрации');
        }
    };

    return (
        <div className="auth-container">
            <div className="auth-title">Система
профориентационного тестирования</div>

```

```

    <h1>Регистрация</h1>
    {error && <p className="auth-error">{error}</p>}
    {success && <p className="auth-success">{success}</p>}
    <form onSubmit={handleSubmit} className="auth-form">
      <input type="text" placeholder="Логин"
value={login}
      onChange={(e) => setLogin(e.target.value)}
      className="auth-input" required />
      <input type="password" placeholder="Пароль"
value={password}
      onChange={(e) => setPassword(e.target.value)}
      className="auth-input" required />
      <button type="submit" className="auth-
button">Зарегистрироваться</button>
    </form>
    <p className="auth-bottom-text">Уже есть аккаунт? <a
href="/login" className="auth-link">Войти</a></p>
  </div>
);
}

export default RegisterPage;

```

Файл «ResultPage.jsx»

```

import { useState, useEffect } from 'react';
import { useLocation, useNavigate } from 'react-router-dom';
import '../styles/global.css';

function ResultPage() {
  const location = useLocation();
  const navigate = useNavigate();
  const results = location.state?.results || [];
  const info = location.state?.info || {};
  const isFullTest = results.length > 0 && 'score' in
results[0];
  const [animated, setAnimated] = useState(false);

  useEffect(() => {
    setTimeout(() => setAnimated(true), 150);
  }, []);

  const sorted = [...results].sort((a, b) => {
    if (isFullTest) return b.score - a.score;
    return b.probability - a.probability;
  });

  const maxPossibleScore = info.max_possible_score || 34;
  const maxScore = isFullTest ? maxPossibleScore :
Math.max(...sorted.map(r => r.probability), 1);

  const getBarColor = (percent) => {

```

```

        if (percent <= 25) return '#c47a5a';
        if (percent <= 50) return '#c4a35a';
        if (percent <= 75) return '#8a9b6e';
        return '#637643';
    };

    const highThreshold = Math.round(maxPossibleScore * 0.76);
    const mediumThreshold = Math.round(maxPossibleScore * 0.56);
    const lowThreshold = Math.round(maxPossibleScore * 0.23);

    return (
        <div className="result-wrapper">
            <div className="result-container">
                <h1>{isFullTest ? 'Результаты полного теста' :
'Результаты короткого теста'}</h1>

                {isFullTest && (
                    <p className="result-hint">
                        Максимум {maxPossibleScore} баллов.
Склонность: ≥{highThreshold} – максимальная, ≥{mediumThreshold} –
умеренная, ≥{lowThreshold} – слабая, &lt;{lowThreshold} –
МИНИМАЛЬНАЯ
                    </p>
                )}

                <div className="result-chart">
                    {sorted.map((r, idx) => {
                        const targetWidth = isFullTest
                            ? (r.score / maxPossibleScore) * 100
                            : (r.probability / maxScore) * 100;
                        const width = animated ? `${targetWidth}%`
: '0%';

                        return (
                            <div key={idx} className="result-bar-
container">
                                <div className="result-bar-
label">{r.sphere_name}</div>
                                <div className="result-bar-row">
                                    <div className="result-bar"
style={{
                                        width,
                                        backgroundColor:
getBarColor(targetWidth),
                                        transition: 'width 0.8s
ease-out, background-color 0.8s',
                                    }} />
                                    {isFullTest ? (
                                        <span className="result-
value" style={{ color: getBarColor(targetWidth) }}>

```

```

{r.inclination_level}
{r.score} баллов –
    </span>
    ) : (
        <span className="result-
value" style={{ color: getBarColor(targetWidth) }}>
{r.probability.toFixed(0)}%
        </span>
    )}
    </div>
</div>
);
    )}
</div>

<div className="result-legend">
    <span className="result-legend-item" style={{
backgroundColor: '#c47a5a' }}>Минимальная</span>
    <span className="result-legend-item" style={{
backgroundColor: '#c4a35a' }}>Слабая</span>
    <span className="result-legend-item" style={{
backgroundColor: '#8a9b6e' }}>Умеренная</span>
    <span className="result-legend-item" style={{
backgroundColor: '#637643' }}>Максимальная</span>
</div>

    <button className="result-back-btn" onClick={() =>
navigate('/menu')}>
        Вернуться в меню
    </button>
</div>
</div>
);
}

export default ResultPage;

```

Файл «ShortTestPage.jsx»

```

import { useState, useEffect } from 'react';
import { useNavigate } from 'react-router-dom';
import api from '../services/api';
import '../styles/global.css';

function ShortTestPage() {
    const [questions, setQuestions] = useState([]);
    const [current, setCurrent] = useState(0);
    const [answers, setAnswers] = useState({});
    const [loading, setLoading] = useState(true);
    const [submitting, setSubmitting] = useState(false);
    const [error, setError] = useState('');

```

```

const navigate = useNavigate();

useEffect(() => {
  loadQuestions();
}, []);

useEffect(() => {
  const handleKeyDown = (e) => {
    if (e.key === 'ArrowLeft' && current > 0)
setCurrent(current - 1);
    if (e.key === 'ArrowRight' && current <
questions.length - 1 && answers[questions[current]?.id]) {
      setCurrent(current + 1);
    }
  };
  window.addEventListener('keydown', handleKeyDown);
  return () => window.removeEventListener('keydown',
handleKeyDown);
}, [current, questions, answers]);

const loadQuestions = async () => {
  try {
    const response = await api.get('/questions/general');
    setQuestions(response.data);
  } catch (err) {
    setError('Ошибка загрузки вопросов');
  } finally {
    setLoading(false);
  }
};

const handleSelect = (questionId, value) => {
  setAnswers({ ...answers, [questionId]: value });
};

const handleNext = () => {
  if (current < questions.length - 1) setCurrent(current +
1);
};

const handlePrev = () => {
  if (current > 0) setCurrent(current - 1);
};

const handleSubmit = async () => {
  setSubmitting(true);
  try {
    const general_answers =
Object.entries(answers).map(([question_id, answer_value]) => ({
      question_id: parseInt(question_id),
      answer_value,

```

```

    ));
    const response = await api.post('/test/short', {
general_answers });
    navigate('/result/short', { state: { results:
response.data.results } });
    } catch (err) {
        setError('Ошибка отправки ответов');
    } finally {
        setSubmitting(false);
    }
};

if (loading) return <div className="short-test-wrapper"><div
className="short-test-container">Загрузка...</div></div>;

const q = questions[current];
const isLast = current === questions.length - 1;
const isAnswered = answers[q?.id] !== undefined;
const allAnswered = questions.every(q => answers[q.id] !==
undefined);

return (
    <div className="short-test-wrapper">
        <div className="short-test-container">
            <h2>Короткий тест</h2>
            <div className="short-test-progress">Вопрос
{current + 1} из {questions.length}</div>
            {error && <p className="short-test-
error">{error}</p>}

            <div className="short-test-question">
                <h3>{q?.text}</h3>
                <div className="short-test-options">
                    {q?.options?.map((option, idx) => (
                        <div
                            key={idx}
                            className={`short-test-option
${answers[q.id] === option ? 'short-test-option-selected' : ''}`}
                            onClick={() => handleSelect(q.id,
option)}
                        >
                            {option}
                        </div>
                    ))}
                </div>
            </div>

            <div className="short-test-nav">
                <button onClick={handlePrev} disabled={current
=== 0}

```

```

        className="short-test-nav-button short-
test-nav-back">← Назад</button>
        {isLast ? (
            <button onClick={handleSubmit}
disabled={!allAnswered || submitting}
            className="short-test-nav-button
short-test-nav-submit">
                {submitting ? 'Отправка...' :
'Завершить тест'}
            </button>
        ) : (
            <button onClick={handleNext}
disabled={!isAnswered}
            className="short-test-nav-button
short-test-nav-next">Далее →</button>
        )}
    </div>
</div>
</div>
);
}

```

```
export default ShortTestPage;
```

Файл «api.js»

```

import axios from 'axios';

const API_URL = 'http://localhost:8000';

const api = axios.create({
  baseURL: API_URL,
  headers: {
    'Content-Type': 'application/json',
  },
});

// Добавляем ID пользователя ко всем запросам
api.interceptors.request.use((config) => {
  const userId = localStorage.getItem('userId');
  if (userId) {
    config.headers['X-User-Id'] = userId;
  }
  return config;
});

export default api;

// Функция для получения информации о системе
export const getInfo = async () => {
  const response = await api.get('/questions/info');
  return response.data;
}

```

```
};
```

Файл «App.jsx»

```
import { BrowserRouter, Routes, Route, Navigate } from 'react-router-dom';
import LoginPage from './pages/LoginPage';
import RegisterPage from './pages/RegisterPage';
import MenuPage from './pages/MenuPage';
import ShortTestPage from './pages/ShortTestPage';
import FullTestPage from './pages/FullTestPage';
import ResultPage from './pages/ResultPage';
import HistoryPage from './pages/HistoryPage';
import AdminPage from './pages/AdminPage';

function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/login" element={<LoginPage />} />
        <Route path="/register" element={<RegisterPage />} />
      </Routes>
    </BrowserRouter>
  );
}

export default App;
```

Файл «main.jsx»

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.css'
import App from './App.jsx'

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <App />
  </StrictMode>
);
```

```
</StrictMode>,  
)
```

					ПГУ 1.090301.15.001 ПЗ	Лист
						158
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение В
(обязательное)

Система профориентации на основе алгоритма машинного обучения

Презентация

					ПГУ 1.090301.15.001 ПЗ	Лист
						159
Изм.	Лист	№ докум.	Подпись	Дата		

СИСТЕМА ПРОФОРИЕНТАЦИИ НА ОСНОВЕ АЛГОРИТМА МАШИННОГО ОБУЧЕНИЯ

Автор работы:
студентка группы 22ВВП1
Кондратьева В.И.

Научный руководитель:
д.т.н, профессор каф. ВТ
Гурин Е.И.

Пенза 2026

Цель и актуальность работы

Цель работы:

Разработка системы профориентации на основе алгоритмов машинного обучения.

Актуальность:

Разработанная система востребована как среди учащихся, которым предстоит определиться с будущей профессией, так и среди образовательных учреждений, стремящихся повысить эффективность профориентационной деятельности.

Слайд 2

					ПГУ 1.090301.15.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		160

Сравнение аналогов

Название системы	Регистрация	Полное тестирование	Быстрое тестирование	Персональные рекомендации	Доступ онлайн
«Wizard» УрФУ	Через аккаунт соцсети	-	+	+	+
«Робот-профориентатор» ТГУ	Через аккаунт соцсети	-	+	+	+
«AI-профориентатор»	Через аккаунт соцсети	-	+	+	+
«Синерджи старт»	Встроенная регистрация	+	-	+ (частично платно)	+
Метод Тарасовой А.С. и Комаровой Э.П.	Встроенная регистрация	+	-	+	-
Разрабатываемая система	Встроенная регистрация	+	+	+	+

Слайд 3

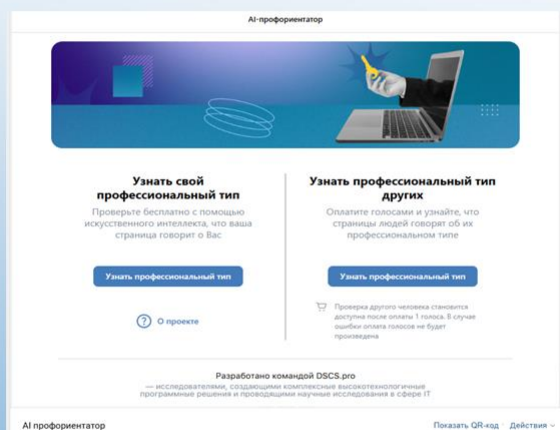


«Wizard» УрФУ

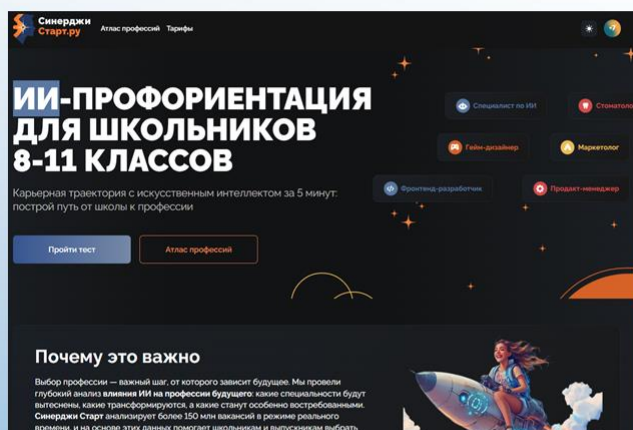


«Робот-профориентатор» ТГУ

Слайд 4



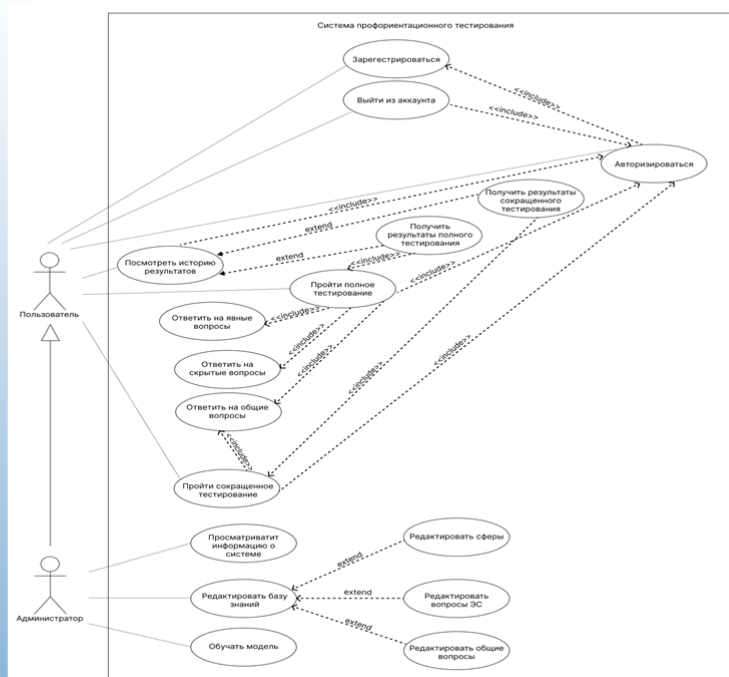
«AI-профориентатор»



«Синерджи Старт»

Слайд 5

UML-диаграмма вариантов использования



Слайд 6

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.15.001 ПЗ

Лист
162

Технологический стек



JavaScript

Visual Studio Code



PostgreSQL



spyder



HTML



CSS

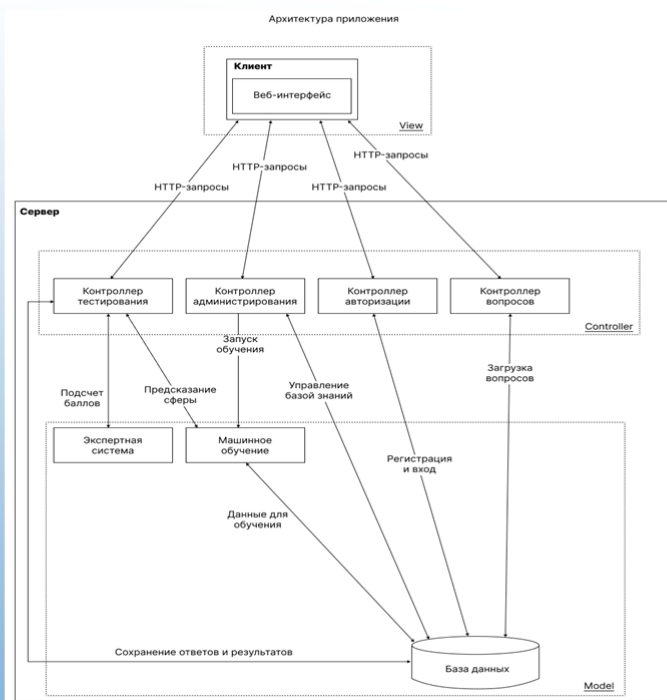


FastAPI



Слайд 7

Архитектура приложения



Слайд 8

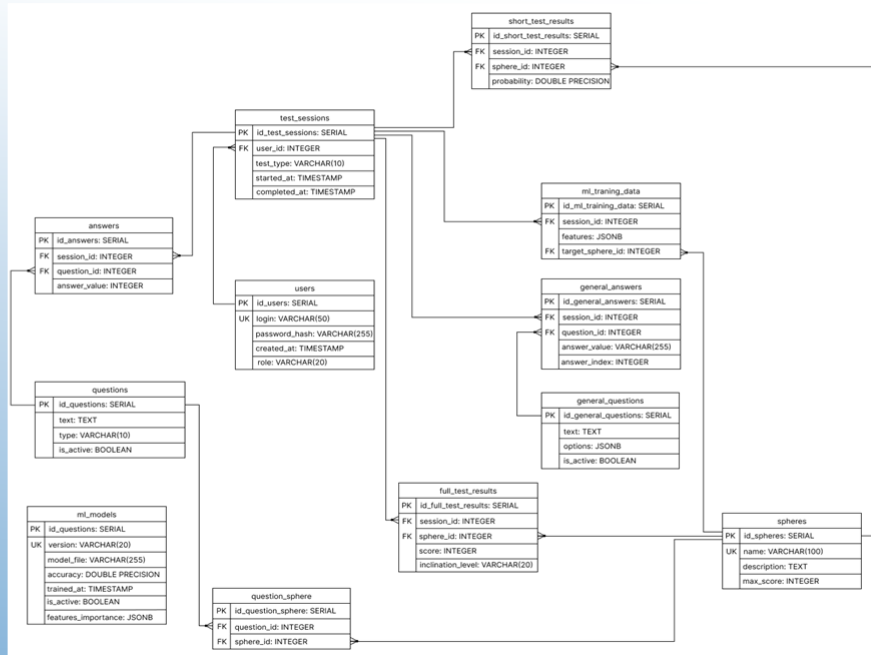
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.15.001 ПЗ

Лист

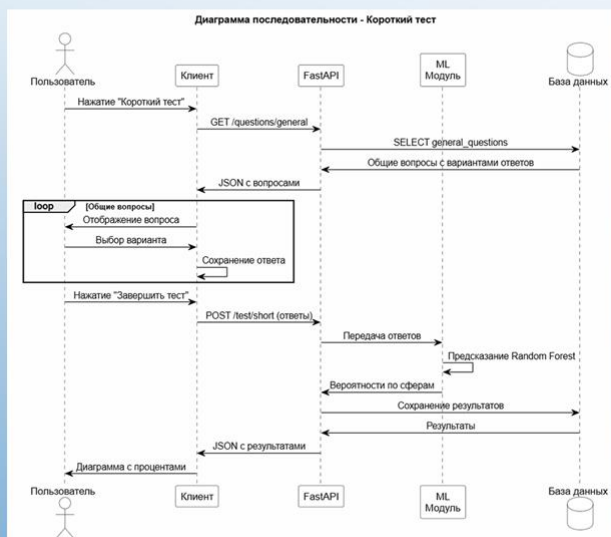
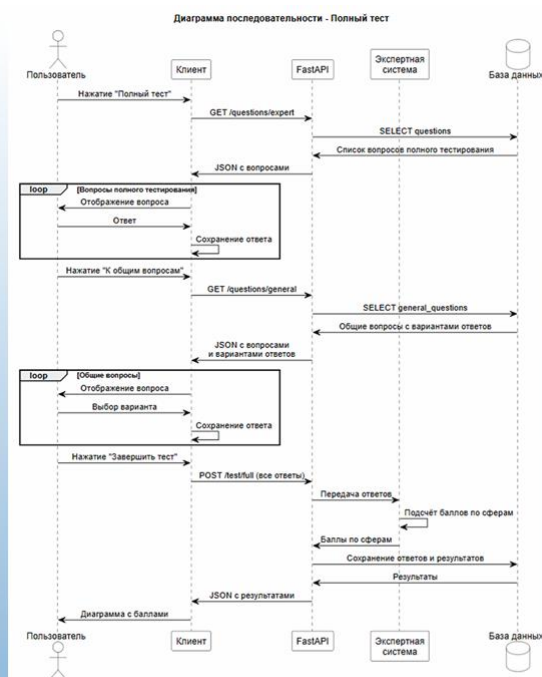
163

Физическая модель базы данных



Слайд 9

Сравнение короткого и полного тестов



Слайд 10

Сравнение результатов

● Короткий тест 13 вопросов 21.05.2026, 03:47:01 >		● Полный тест 151 вопросов 19.05.2026, 03:55:04 >	
Творчество и искусство	69%	Творчество и искусство	27 баллов — максимальная
Спорт и фитнес	8%	Спорт и фитнес	25 баллов — умеренная
Бизнес и финансы	6%	Военное дело и безопасность	24 баллов — умеренная
Туризм и гостиничный бизнес	4%	Инженерия и технологии	24 баллов — умеренная
Инженерия и технологии	3%	Право и госуправление	20 баллов — умеренная
Наука и образование	2%	Медицина и здравоохранение	20 баллов — умеренная
Право и госуправление	2%	Информационные технологии - IT	20 баллов — умеренная
Сельское хозяйство и экология	2%	Мода и красота	20 баллов — умеренная
Медицина и здравоохранение	1%	Наука и образование	19 баллов — слабая
Социальная сфера	1%	Бизнес и финансы	19 баллов — слабая
Военное дело и безопасность	1%	Развлечения и игровая индустрия	18 баллов — слабая
Мода и красота	1%	Туризм и гостиничный бизнес	18 баллов — слабая
Информационные технологии - IT	0%	Сельское хозяйство и экология	15 баллов — слабая
Транспорт и логистика	0%	Транспорт и логистика	14 баллов — слабая
Развлечения и игровая индустрия	0%	Социальная сфера	14 баллов — слабая

Слайд 11

Результат ML-модели

Результат экспертной системы

Заключение

Результаты работы:

- ▶ Разработана экспертная система
- ▶ Обучена ML-модель Random Forest
- ▶ Реализовано веб-приложение

Результаты работы:

- ▶ Обучение отдельных моделей для каждой позиции рейтинга сфер
- ▶ Сбор реальных данных для повышения точности
- ▶ Корректировка вопросов с точки зрения психологии

Слайд 12

СПАСИБО ЗА ВНИМАНИЕ

					ПГУ 1.090301.15.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		166