МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Институт компьютерных наук
Кафедра автоматизированных систем управления

Индивидуальное домашнее задание
по дисциплине «Архитектура программных систем»

Студент АС-21-1 _____ Болдырев М.Р.

(подпись, дата)

Руководитель
Кандидат технических наук _____ Алексеев В.А.

(подпись, дата)

Липецк 2025 г.

Цель работы

Изучить подходы к документированию архитектуры программных систем, получить навыки документирования архитектуры на примере учебного проекта.

Задание

Сформулировать техническое задание на разработку программной системы, разработать документ «Архитектура программной системы» с использованием диаграмм UML, ER.

1. Техническое задание на программную систему

С каждым годом спрос на IT-специалистов стремительно растет: по данным Global Knowledge, к 2025 году мировой дефицит квалифицированных кадров в сфере программирования превысит 85 млн человек. Университеты активно внедряют цифровые инструменты для повышения качества образования, однако многие процессы — распределение курсов, проверка заданий, контроль прогресса — остаются рутинными и затратными. Онлайн-платформы становятся ключевым элементом обучения, но существующие решения часто не учитывают специфику академических программ вузов и ограничены в возможностях персонализации.

Платформа CodeSphere призвана автоматизировать обучение программированию в университетах, обеспечивая гибкое управление курсами, мгновенную проверку кода и объективную оценку успеваемости. Система позволяет оптимизировать взаимодействие студентов и преподавателей: вместо рутинной работы с документами и отчетами, преподаватели могут сосредоточиться на создании контента и индивидуальной поддержке учащихся, а студенты — на практическом освоении языков программирования в интерактивной среде.

Диаграмма вариантов использования представлена на рисунке 1.

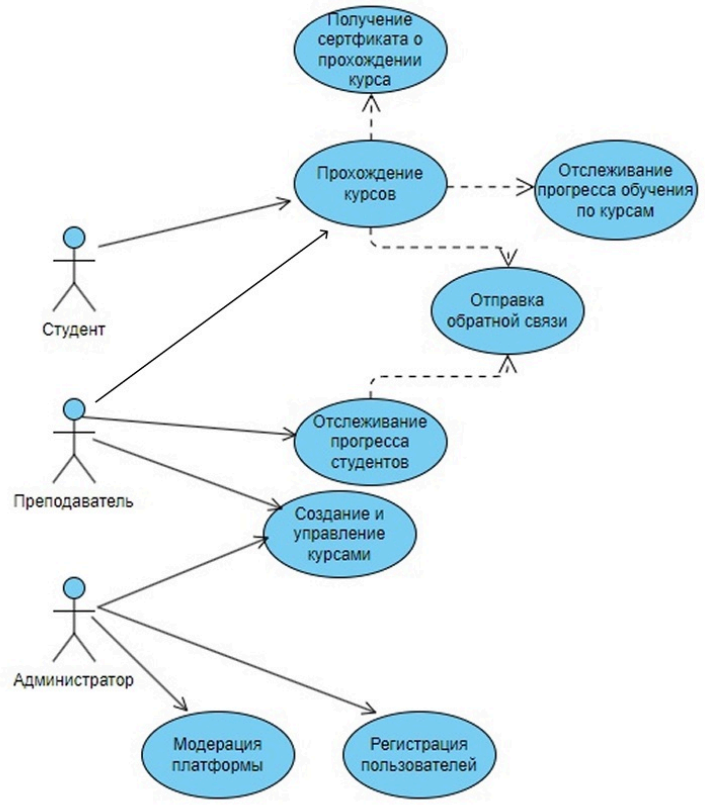

Рисунок 1 - Диаграмма вариантов использования

2. Архитектура системы

2.1. Введение

ИС «Веб-система для поддержки проведения курсов по обучению программированию CodeSphere» реализуется виде веб-приложения. Приложение будет написано с использованием фреймворка Vue Js и языком программирования Javascript.

Логика серверной части системы будет реализована с использованием с языка программирования PHP.

В качестве СУБД выбрана PostgreSQL.

2.2. Логическое представление

Исходя из планируемого функционала рассмотрим логическое представление системы в виде системы меню пользователя.

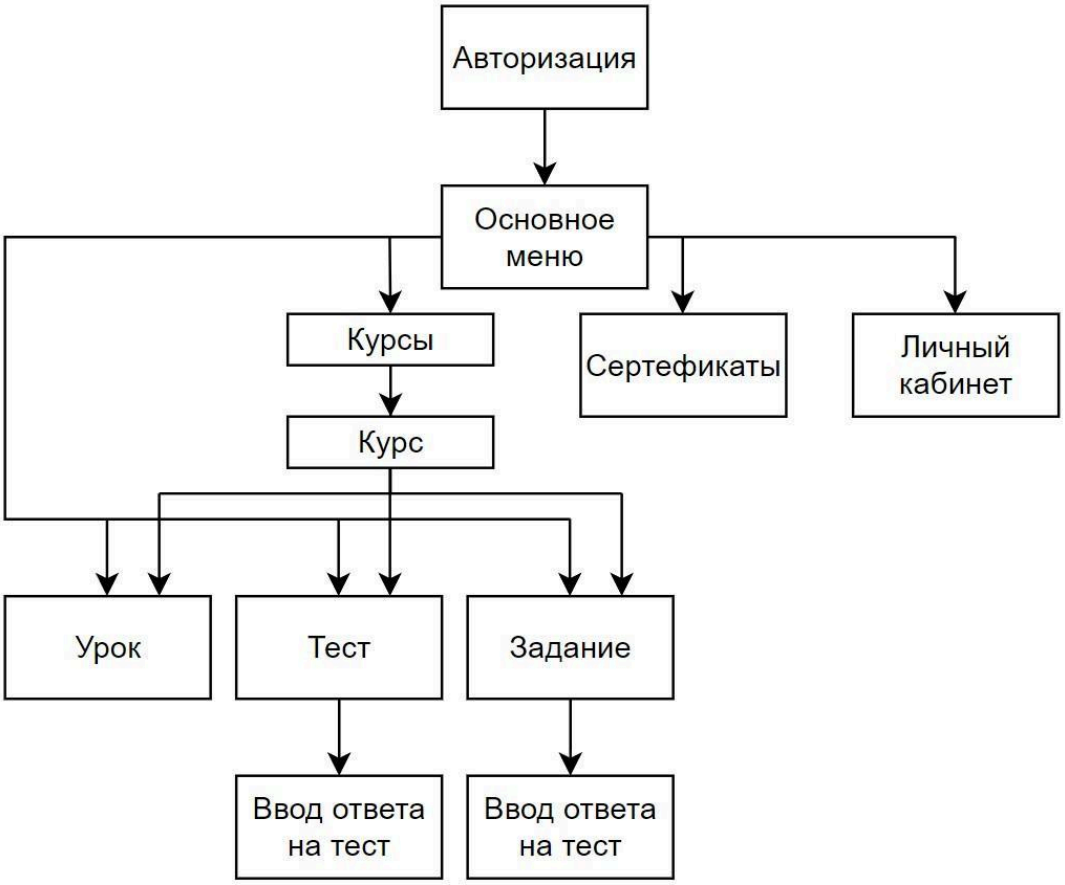

Рисунок 2 - Логическое представление

2.3. Представление разработки

Программная система CodeSphere включает Frontend-часть, состоящую из слоя пользовательского интерфейса (UI). В качестве Backend используется сервер на PHP, отвечающий за образовательную логику: автоматическую проверку заданий, генерацию сертификатов, анализ прогресса студентов с применением ML-алгоритмов. PostgreSQL реализует функции СУБД для хранения данных курсов, пользователей, прогресса обучения, а также интегрирован с сервисом авторизации. Взаимодействие компонентов организовано через REST API: фронтенд отправляет запросы к бэкенду, который обрабатывает данные, взаимодействует с внешними API (например, JDoodle для выполнения кода) и возвращает результаты. Данная декомпозиция системы на независимо разрабатываемые компоненты и взаимодействие этих компонентов представлено на рисунке 3.

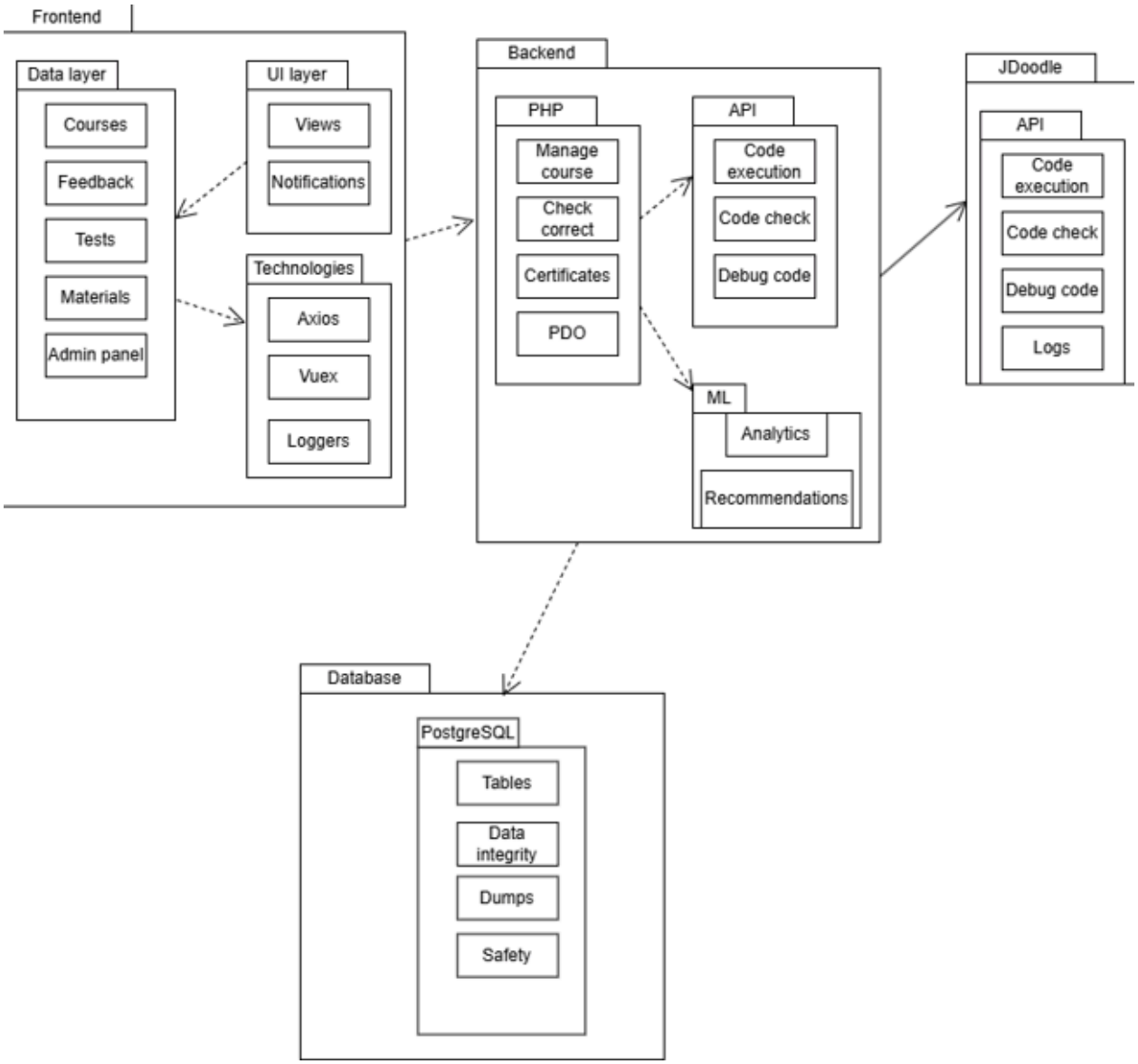

Рисунок 3 - Независимо разрабатываемые компоненты и их взаимодействие

2.4. Представление развертывания

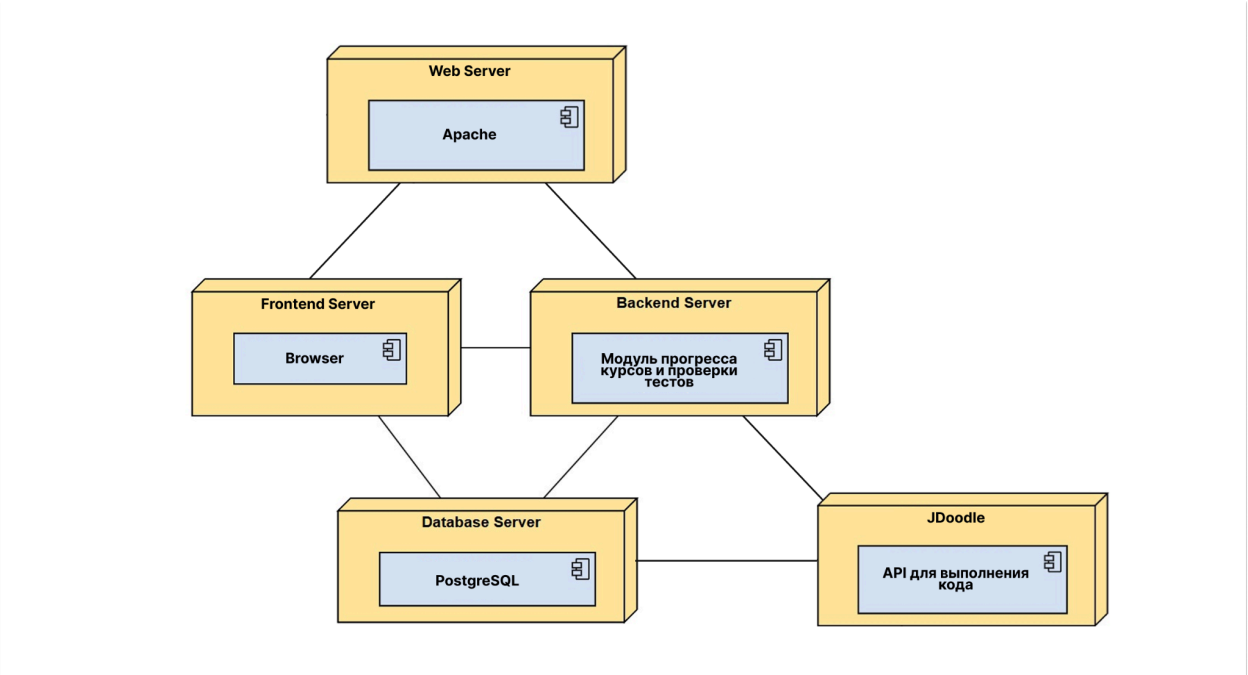

Рисунок 4 - Диаграмма развертывания

На рисунке 4 изображена диаграмма развертывания, которая предполагает, что Backend, Frontend, Apache и PostgreSQL будут располагаться на независимых серверах с целью повышения отказоустойчивости системы, а также модуль выполнения кода будет работать путем обращения к API JDoodle.

2.5. Представление данных

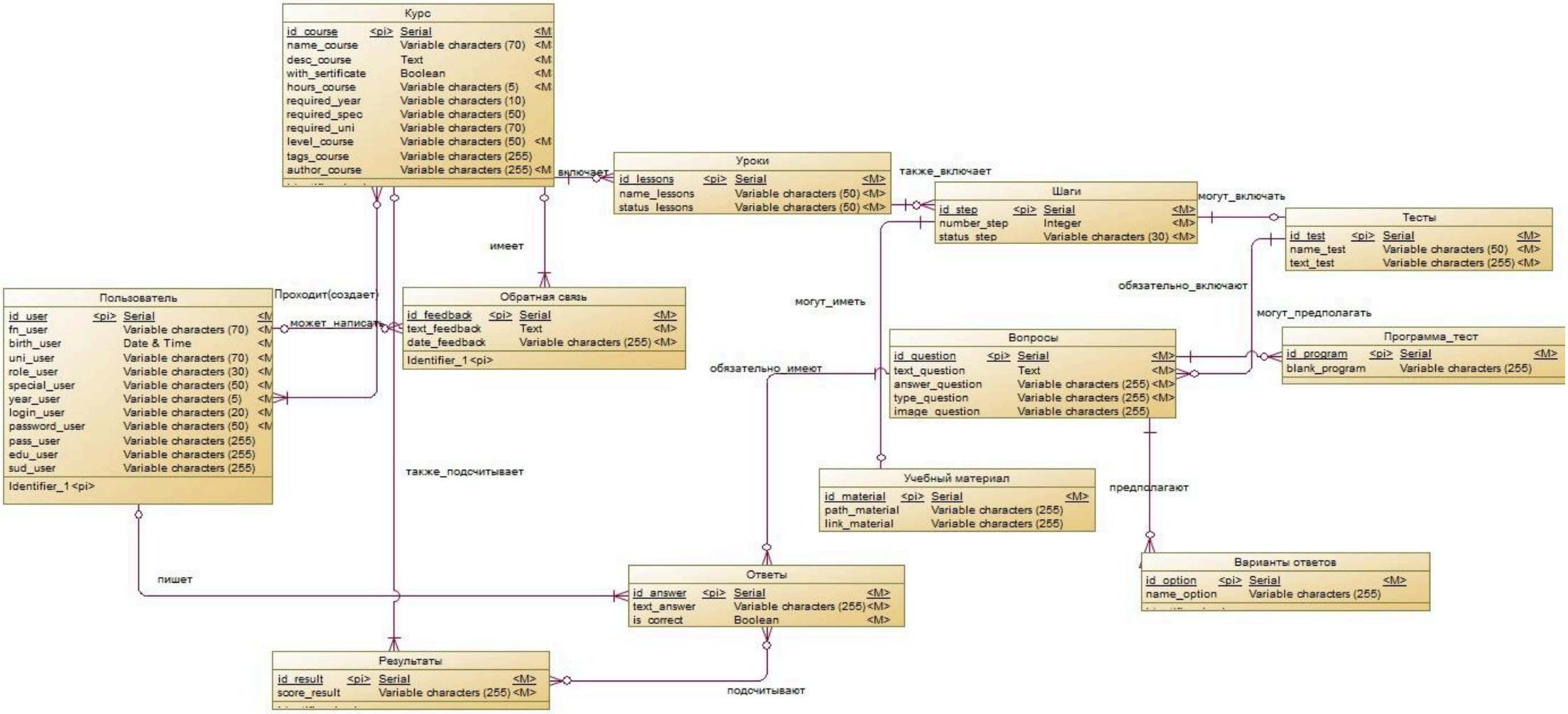
Курс
id_course <pi> Serial <M>
name_course Variable characters (70) <M>
desc_course Text <M>
with_sertificate Boolean <M>
hours_course Variable characters (5) <M>
required_year Variable characters (10)
required_spec Variable characters (50)
required_uni Variable characters (70)
level_course Variable characters (50) <M>
tags_course Variable characters (255)
author_course Variable characters (255) <M>
включает
Уроки
id_lessons <pi> Serial <M>
name_lessons Variable characters (50) <M>
status_lessons Variable characters (50) <M>
также_включает
Шаги
id_step <pi> Serial <M>
number_step Integer <M>
status_step Variable characters (30) <M>
могут_включать
Тесты
id_test <pi> Serial <M>
name_test Variable characters (50) <M>
text_test Variable characters (255) <M>
имеет
Пользователь
id_user <pi> Serial <M>
fn_user Variable characters (70) <M>
birth_user Date & Time <M>
uni_user Variable characters (70) <M>
role_user Variable characters (30) <M>
special_user Variable characters (50) <M>
year_user Variable characters (5) <M>
login_user Variable characters (20) <M>
password_user Variable characters (50) <M>
pass_user Variable characters (255)
edu_user Variable characters (255)
sud_user Variable characters (255)
Identifier_1 <pi>
Проходит(создает)
может_написать
Обратная связь
id_feedback <pi> Serial <M>
text_feedback Text <M>
date_feedback Variable characters (255) <M>
Identifier_1 <pi>
могут_иметь
обязательно_включают
могут_предполагать
Вопросы
id_question <pi> Serial <M>
text_question Text <M>
answer_question Variable characters (255) <M>
type_question Variable characters (255) <M>
image_question Variable characters (255)
Программа_тест
id_program <pi> Serial <M>
blank_program Variable characters (255)
обязательно_имеют
Учебный материал
id_material <pi> Serial <M>
path_material Variable characters (255)
link_material Variable characters (255)
предполагают
также_подсчитывает
пишет
Ответы
id_answer <pi> Serial <M>
text_answer Variable characters (255) <M>
is_correct Boolean <M>
Варианты ответов
id_option <pi> Serial <M>
name_option Variable characters (255)
Результаты
id_result <pi> Serial <M>
score_result Variable characters (255) <M>
подсчитывают

Рисунок 5 - Концептуальная модель базы данных

2.6. Ключевые сценарии

В таблице 1 представлены ключевые сценарии использования программной системы.

Таблица 1 - Ключевые сценарии

Действие	Логика обработки
Прохождение интерактивного задания по программированию	- Студент выбирает курс, урок и задание в личном кабинете. - Приложение загружает описание задания, редактор кода и примеры тестов из базы данных PostgreSQL. - Студент пишет код решения и нажимает кнопку «Отправить». - Backend отправляет код через API JDoodle на выполнение, сравнивает результат с эталонными тестами. - Система сохраняет результат проверки в PostgreSQL, обновляет прогресс студента. - Студент видит оценку (успешно/ошибка)
Автоматическая генерация сертификата	- Студент завершает последний урок курса, выполнив все обязательные задания. - Система проверяет прогресс студента в PostgreSQL (пройдено ≥ 90% материалов). - Backend генерирует PDF-сертификат с данными студента, названием курса и подписью преподавателя. - Сертификат сохраняется в личном кабинете студента. - Студент может скачать сертификат или поделиться им в социальных сетях.
Анализ прогресса обучения (алгоритмы ML)	- Преподаватель открывает раздел «Аналитика» курса, выбирает параметры (группа, период, темы). - Система запрашивает данные из PostgreSQL: результаты тестов, активность студентов, время выполнения заданий. - Backend применяет ML-модель для

	выявления слабых мест студентов, формирования heatmap ошибок. - Преподаватель получает визуализацию: графики прогресса, рекомендации по адаптации курса, список студентов для индивидуальной работы.
Интерактивная обратная связь	- Студент оставляет комментарий к уроку через форму «Задать вопрос». - Система сохраняет запрос в PostgreSQL и присваивает ему статус «Ожидает ответа». - Преподаватель получает уведомление, просматривает вопрос, публикует ответ или прикрепляет дополнительные материалы. - Студент видит ответ в личном кабинете и может продолжить обсуждение.
Регистрация студентов через куратора вуза	- Куратор вуза загружает файл с данными студентов (ФИО, email, специальность, ВУЗ) в систему. - Backend валидирует данные, генерирует уникальные логины/пароли, сохраняет их в PostgreSQL. - Система отправляет приветственные письма студентам с инструкцией по активации аккаунта. - Студент подтверждает email, получает доступ к курсам, назначенным вузом.