Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
Петрозаводский государственный университет
Институт математики и информационных технологий

Отчет о прохождении производственной практики по получению профессиональных умений и опыта профессиональной деятельности

Выполнила студентка группы 22307
Е. А. Гасова

Направление подготовки:
09.03.04. – Программная инженерия

Место прохождения практики:
Кафедра информатики и математического обеспечения

Сроки прохождения практики:
26.05.25 – 08.06.25

Руководитель практики:
к.ф.-м.н. К. А. Кулаков

Оценка _____

Дата _____

Подпись _____

Петрозаводск 2025 г.

Содержание

Цели и задачи практики

Цель практики: применение программных инструментов и технологий для разработки программного обеспечения.

Задачи практики:

1. Изучение документации по программному каркасу FastAPI.
2. Изучение документации по программному каркасу React.
3. Реализация веб приложения с использованием программных каркасов FastAPI и React.

Тема проекта: разработка веб-приложения для проведения тестирования.

Обзор программных средств, используемых в ходе прохождения практики

FastAPI — современный высокоэффективный веб-фреймворк для Python, построенный на основе сервера Starlette. Отличается исключительной производительностью и простотой использования, автоматически генерирует интерактивную документацию Swagger/OpenAPI. В проекте выполняет роль основного инструмента для построения REST API, обеспечивая высокую скорость разработки за счет встроенной валидации данных и асинхронной обработки запросов.

Pydantic — библиотека для строгой типизации и преобразования данных в Python. Позволяет определять модели данных с автоматической валидацией. В проекте интегрирована с FastAPI для обработки входящих/исходящих данных, гарантируя их корректность и соответствие ожидаемым схемам.

SQLAlchemy — многофункциональный ORM-инструментарий для Python, предоставляющий высокоуровневый API для работы с реляционными БД. Поддерживает декларативное определение моделей и гибкие запросы. В проекте используется как прослойка между приложением и базой данных, упрощая выполнение CRUD-операций и обеспечивая безопасность данных.

MariaDB — современная реляционная СУБД, форк MySQL с улучшенной производительностью и расширенными возможностями. Отличается открытостью кода и масштабируемостью. В проекте выступает в качестве основного хранилища данных.

Docker — платформа для контейнеризации приложений, позволяющая упаковывать сервисы в изолированные среды со всеми зависимостями. В проекте используется для развертывания среды разработки.

React — JavaScript-библиотека для построения пользовательских интерфейсов на основе компонентного подхода. Использует виртуальный DOM для эффективного обновления. В проекте составляет ядро клиентской части, отвечая за отображение данных и обработку пользовательских действий.

Next.js — фреймворк для React, предоставляющий серверный рендеринг, статическую генерацию и другие оптимизации. В проекте служит основой для клиентской части, сочетая преимущества React с улучшенной SEO-оптимизацией и производительностью за счет предварительного рендеринга страниц.

TypeScript — строго типизированное надмножество JavaScript, добавляющее проверку типов на этапе компиляции. В проекте применяется для повышения надежности

кода, улучшения поддержки в IDE и предотвращения распространенных ошибок времени выполнения.

Material UI — библиотека компонентов для React, основанная на Google's Material Design. Она предоставляет готовые UI-элементы, такие как кнопки, карточки, таблицы, формы и прочее, значительно ускоряя разработку интерфейсов.

Анализ технических требований и проектные решения

1. Приложение должно быть авторским, размещено на gitlab репозитории кафедры

Требование соблюдено, ссылка на репозиторий проекта: https://dev.cs.petrsu.ru/gasova/testing-app

2. Приложение должно отображать перечень тестов

На главной странице приложения отображаются не более трёх рекомендованных для пользователя тестов, а также список всех тестов с пагинацией и возможностью сортировки и фильтрации.

3. Приложение должно давать возможность пользователю проходить тест

Пользователь, не имеющий роли Администратор, может проходить любой тест. В ходе прохождения теста пользователь отвечает на вопросы трёх видов: вопрос с выбором одного ответа из списка, вопрос с выбором нескольких ответов из списка, вопрос с вводом текста.

4. Приложение должно отображаеть результаты прохождения тестов

После прохождения теста пользователем приложение отображает результат прохождения — количество правильных ответов, количество всех вопросов в тесте, время начала прохождения теста, время окончания прохождения теста. Кроме того, пользователь может просмотреть список своих результатов прохождения тестов по всем тестам или по выбранному тесту.

5. Приложение должно группировать тесты по тематикам

Каждый тест в приложении относится к одной из категорий. Список категорий изменяется администратором приложения. При создании теста администратор выбирает одну категорию из списка.

6. Все данные должны храниться в базе данных (БД)

Данные приложения хранятся в базе данных MariaDB.

7. Приложение должно иметь интерфейс управления тестами для администратора

Приложение имеет панель администратора со следующими функциями: управление категориями (просмотр списка категорий, создание, изменение, удаление категории), управление тестами (просмотр списка тестов, просмотр результатов прохождения теста. создание, изменение, удаление теста), управление пользователями (просмотр списка пользователей, создание администратора, удаление пользователя).

8. Взаимодействие с БД должно быть реализовано через SQLAlchemy

SQLAlchemy используется для управления моделями данных и выполнения запросов к базе данных.

9. (опционально) Структура БД должна быть реализована через миграции (alembic) или через скрипты создания/обновления таблиц

Требование не было реализовано.

10. Доступ к серверной части должен быть защищен с помощью авторизации

Для доступа к функциям приложения пользователь должен пройти аутентификацию по логину и паролю, указанным при регистрации.

11. Должна присутствовать регистрация пользователей

В приложении присутствует открытая регистрация обычных пользователей по логину и паролю. Пользователь с ролью администратор имеет возможность создавать других администраторов, вводя их логин и пароль.

12. Клиентская часть приложения должна реализовывать сортировку и фильтрацию списка тестов.

На главной странице список тестов должен иметь сортировку по названию теста и категории, а также фильтр по этим двум полям. Аналогично список тестов в панели администратора должен обладать сортировкой и фильтрацией.

13. Реализовать на серверной части работу алгоритма (например, подбор рекомендованных тестов на основе результатов)

На серверной части должен быть реализован алгоритм подбора рекомендованных тестов на основе результатов пользователя. Он должен возвращать не более трёх самых популярных тестов в категориях, в которых пользователь прошёл наибольшее количество тестов, причём пользователь ещё не проходил рекомендованные тесты.

14. Приложение должно реализовывать механизм управления пользователями (блокирование)

Администратор должен иметь возможность заблокировать или разблокировать любого пользователя, кроме себя. Заблокированный пользователь не имеет доступа к приложению, данные пользователя сохраняются.

15. (опционально) Приложение должно иметь возможность отправки/отображения уведомлений/сообщений пользователю от администратора или приложения

Требование не было реализовано.

16. (опционально) Реализовать непрерывную сборку и тестирование приложения на базе Gitlab CI/CD

Требование не было реализовано.

Результаты реализации

Ссылка на репозиторий проекта: https://dev.cs.petrsu.ru/gasova/testing-app

Разработанное веб приложение для прохождения опросов полностью удовлетворяет всем обязательным техническим требованиям практики.

На главной странице системы реализовано отображение как рекомендованных тестов (не более трех), так и полного перечня доступных тестов с поддержкой пагинации, сортировки и фильтрации по названию и категориям. Пользователи могут проходить различные типы тестов, включая вопросы с выбором одного или нескольких ответов, а также вопросы с текстовым вводом.

После завершения тестирования система отображает результаты прохождения, включая количество правильных ответов, общее число вопросов в тесте, а также время начала и окончания тестирования. Пользователи имеют возможность просматривать историю своих результатов как по всем тестам, так и по конкретным тестам.

Административная часть приложения предоставляет расширенные возможности управления: создание и редактирование категорий тестов, полный контроль над тестами (добавление, изменение, удаление), а также управление пользователями, включая назначение администраторов и блокировку пользователей. Особое внимание уделено безопасности — доступ к функционалу требует обязательной авторизации.

Все данные системы хранятся в базе данных MariaDB, взаимодействие с которой организовано через SQLAlchemy. Для новых пользователей предусмотрена процедура регистрации, а администраторы могут создавать других администраторов вручную.

Система аутентификации использует JWT-токены, хранящиеся в HTTP-only cookies для безопасности. При входе в приложение генерируется токен с данными пользователя, подписанный секретным ключом. Для доступа к защищенным роутам middleware (next.js) проверяет cookies: обычных пользователей перенаправляет с путей администратора, неавторизованных — на страницу входа. Серверная часть (FastAPI) валидирует токен, проверяя подпись, срок действия и соответствие данных в БД. Реализация защищает от XSS (HTTP-only), обеспечивая безопасный контроль доступа.

Таблица 1: Метрики кода

Язык	Количество файлов	Количество строк
Python	27	1174
TypeScript	25	2741
CSS	1	26

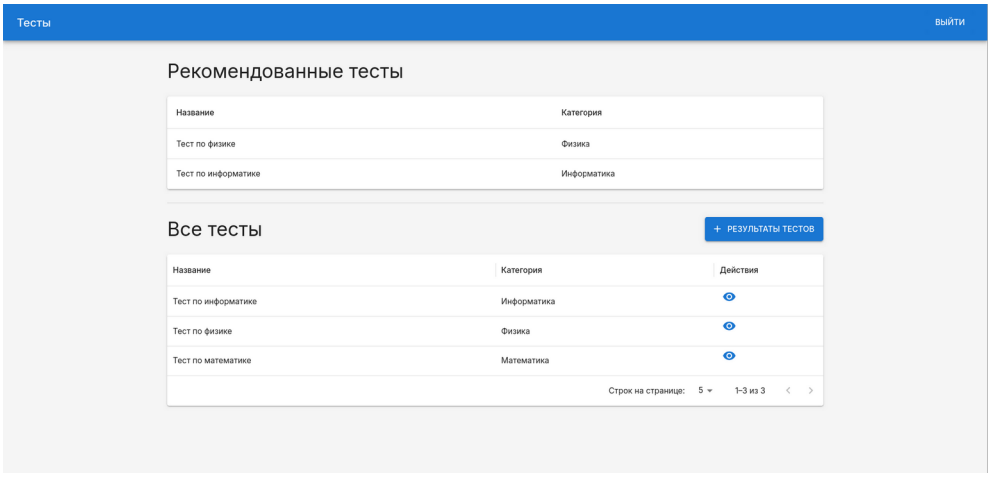

Рисунок 1: Главная страница

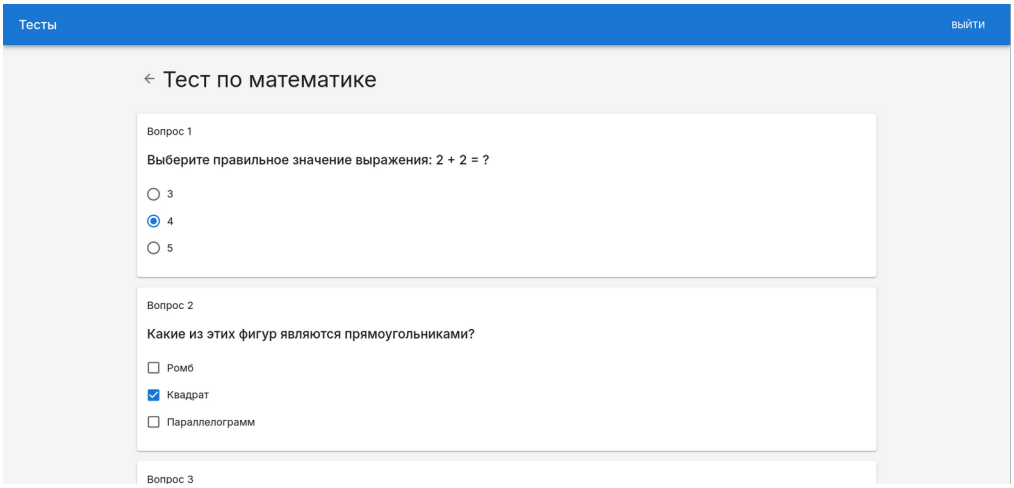

Рисунок 2: Страница прохождения теста

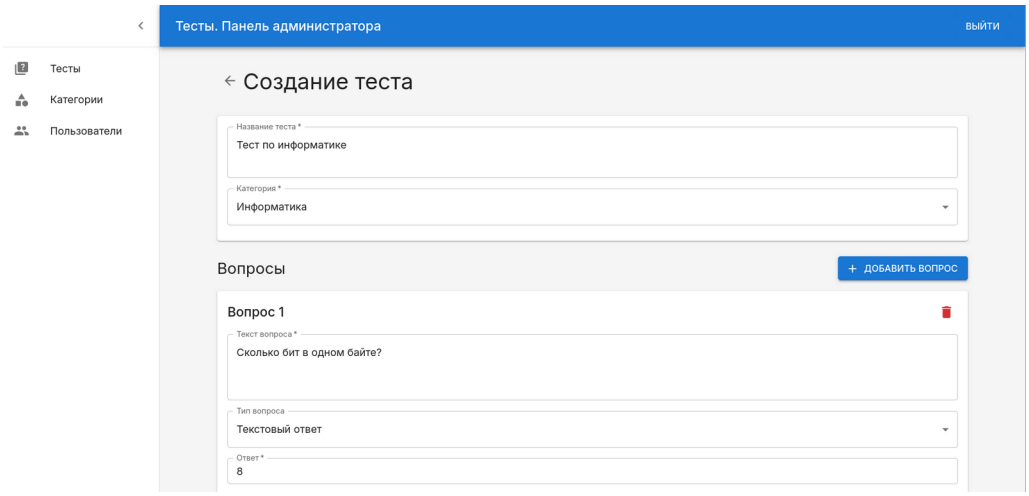

Рисунок 3: Страница панели администратора для создания теста

Дневник практики

Дата	Выполняемая работа
26.05.2025	Ознакомление с программными каркасами FastAPI и React.
27.05.2025	Анализ технических требований. Создание репозитория в GitLab. Выбор СУБД. Создание приложений и настройка Docker.
28.05.2025	Разработка серверной части веб-приложения: управление пользователями, аутентификация и авторизация пользователей с помощью JWT-токенов и HTTP-only cookies.
29.05.2025	Разработка серверной части веб-приложения: API для CRUD-операций с категориями тестов, тестами и результатами тестов.
30.05.2025	Разработка серверной части веб-приложения: реализация алгоритма для рекомендации тестов, добавление возможности пройти тест.
02.06.2025- 03.06.2025	Разработка клиентской части веб-приложения: панель администратора (управление тестами, категориями и пользователями).
04.06.2025	Разработка клиентской части веб-приложения: страницы для отображения списка тестов и результатов прохождения тестов, страница для прохождения теста.
05.06.2025	Оформление отчёта о прохождении практики.
06.06.2025	Защита практики: демонстрация разработанного веб-приложения и сдача отчёта.