Федеральное государственное автономное образовательное учреждение
высшего образования
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий
Кафедра «Информатика и информационные технологии»

Направление подготовки/ специальность: Автоматизированные системы обработки информации и управления

ОТЧЕТ

по проектной практике

Студент: Журбей Артём Михайлович Группа: 241-339

Место прохождения практики: Московский Политех, кафедра «Информатика и информационные технологии»

Отчет принят с оценкой _____ Дата _____

Руководитель практики: Меньшикова Наталия Павловна

Москва 2025

1. Общая информация о проекте

1.1.Название проекта

В рамках учебной практики необходимо создать клиентскую часть веб-сайта с информацией по проекту в рамках предмета «Проектная деятельность», которая содержит аннотацию проекта, краткую информацию, команду и полезные ссылки, касаемые проекта.

1.2.Цели и задачи проекта

Цель проекта – реализовать веб-сайт, содержащий общую информацию о проекте.

1.3.Использованные технологии для реализации веб-сайта

Для реализации проекта была использована библиотека для разработки клиентской части веб-приложений – React.js. Данная технология позволяет быстро реализовывать масштабируемые сайты с возможностью дальнейшей поддержки. Также используется библиотека MUI. Данная библиотека позволяет использовать уже заранее заданные компоненты (более сложные компоненты, чем обычные теги HTML) с возможностью подробной настройки при помощи свойств компонента (аналог CSS в контексте).

2. Общая характеристика деятельности организации

2.1.Наименование заказчика

В рамках и практики, и проектной деятельности заказчиком проекта является ВУЗ – Федеральное государственное автономное образовательное учреждение высшего образования «Московский политехнический университет».

2.2. Организационная структура

Поскольку проект разрабатывается в рамках ВУЗа, то организационная структура соответствует организационной структуре Московского политеха. Локально структуру можно отобразить так:

- Глава и комиссия кафедры;
    - Руководитель проекта;
        - Студенты;

2.3.Описание деятельности

Проект представляет собой веб-приложение, состоящее из клиентской части, написанной на React.js и серверной части, написанной на FastAPI с использованием в качестве СУБД Postgres.

В рамках проекта было реализовано много функционала:

2.3.1. Логика проверка формул

В рамках проекта реализован функционал поиска и проверки правильности формулы (рис. 1-2).

```python
async def check_formulas(teacher_formula_str, input_variables_str, code_str) -> tuple[str, bool]:
    teacher_list = TeacherList()
    for line in teacher_formula_str.splitlines():
        line = line.rstrip()
        teacher_list.add_teacher_formula(line)

    for line in input_variables_str.splitlines():
        line = line.rstrip()
        teacher_list.input_variables.append(line)

    input_count = 0
    for line in code_str.splitlines():
        line = line.rstrip()
        if 'input()' in line:
            char = line[:line.find('=')].strip()
            teacher_list.binding_variables(char, teacher_list.input_variables[input_count])
            input_count += 1
        else:
            teacher_list.add_student_formula(line)

    res = ""
    all_formulas_correct = True
    for i in range(len(teacher_list.formulas_teacher)):
        if i in teacher_list.formulas_student:
            student_formula = "".join(teacher_list.formulas_student[i])
            teacher_formula = "".join(teacher_list.formulas_teacher[i])
            if student_formula != teacher_formula:
                all_formulas_correct = False
            res += student_formula + '\n'
        else:
            all_formulas_correct = False

    return res, all_formulas_correct
```

Рис. 1 – Первая часть логики проверки формул

```
def add_student_formula(self, line):
    normalized_input = self.normalize_formula("".join(self.formulas_teacher[0]))
    found = False

    line = line.rstrip('\n')
    if line.endswith(';'):
        line = line[:-1]

    for i in self.formulas_teacher:
        expressions = self.extract_expressions(line)
        for expr in expressions:
            normalized_line = self.normalize_formula(expr)
            if normalized_input == normalized_line:
                found = True
                break

            if found:
                break

        if found:
            self.binding_variables(line, self.formulas_teacher[i])
            self.binding_formulas(line, self.formulas_teacher[i])
```

Рис. 2 – Вторая часть логики проверки формул

2.3.2. Возможность учителем редактировать работы

Учителя могут создавать лабораторные работы для студентов, которые те могут выполнять (рис. 3-4).

```
# Редактирование лабораторной работы
@router.put("/lab", response_model=UpdateLabRequest, summary="Редактирование лабораторной работы")
async def create_lab(lab: UpdateLabRequest, authorization: str = Header(...)):

    if not authorization.startswith("Bearer "):
        return JSONResponse(status_code=HTTPStatus.UNAUTHORIZED, content={"error": "Invalid token format"})
    token = authorization[len("Bearer "):]
    decoded_token = decode_access_token(token)
    if isinstance(decoded_token, str):
        return JSONResponse(status_code=HTTPStatus.UNAUTHORIZED, content={"error": decoded_token})

    task_to_update = edit_lab(lab.task, lab.is_published)
    if task_to_update is None:
        return response_with_error(
            HTTPStatus.INTERNAL_SERVER_ERROR,
            "Ошибка редактирования лабораторной работы"
        )
    return JSONResponse(
        status_code=HTTPStatus.OK,
        content={"id": task_to_update}
    )
```

Рис. 3 – Первая часть функционала редактора

```python
def edit_lab(task: TaskWithTestCasesSchema, is_published: bool):
    """
    Редактирует неопубликованную лабораторную работу в базе данных.

    :param task: Объект TaskWithTestCasesSchema с информацией о лабораторной работе.
    :return: ID созданной лабораторной работы.
    """
    with Session() as session:
        try:
            # Обноваляем новую лабораторную работу
            updated_task = update_task_data(task, is_published)
            session.merge(updated_task)
            session.commit()

            # Удаляем старые тесткейсы из таблицы UnpublishedTestCase или TestCase
            if is_published:
                session.query(TestCase).filter_by(Task_id=task.id).delete()
            else:
                session.query(UnpublishedTestCase).filter_by(UnpublishedTask_id=task.id).delete()

            # Добавляем тесткейсы заново
            for test_case in task.testCases:
                editted_test_case = updated_task_testcase(task.id, test_case, is_published)
                session.add(editted_test_case)

            session.commit()
            return task.id
        except Exception as e:
            session.rollback()
            return None
```

Рис. 4 – Вторая часть функционала редактора

Также в рамках проекта, поскольку он достаточно долго длится (3 года), было проведено много рефакторинга и чистики кода, который был написан прошлыми студентами.

Также была проведена нормализация базы данных для адекватной работы Backend’а.

3. Описание задания по проектной практике

Задание по проектной практике представляет из себя две части, в которые входят базовая и вариативная часть.

В рамках базовой части было необходимо создать веб-сайт, который описывает саму практику, а также вклад в проект.

В рамках вариативной части было выбрано сделать собственный веб-сервер на основе микросервисной архитектуры, микросервисы которой связаны между собой с помощью технологии gRPC.

4. Описание достигнутых результатов по проектной практике

В рамках проектной практики был реализован сайт, описывающий проектную деятельность, а также форкнут и настроен репозиторий GitHub, куда и были залиты изменения.

Также в рамках вариативной части были сделаны два микросервиса – авторизации, а также просмотра и редактирования постов пользователей.

Написаны отчёты в формате Markdown и PDF.

ЗАКЛЮЧЕНИЕ

В результате, после прохождения проектной практики, были приобретены навыки и/или отработаны навыки:

- настройка репозитория GitHub с использованием Markdown;
- создания React.js клиентского приложения;
- разработки backend’а на основе языков Golang и Typescript.

В конечном итоге были получены навыки работы с Markdown, а также повторены базовые библиотеки и фреймворки для разработки серверных и клиентских приложений.