Федеральное государственное автономное образовательное учреждение высшего образования

«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий  
Кафедра «Информатика и информационные технологии»

Направление подготовки/ специальность: Информационные системы и технологии

ОТЧЕТ

по проектной практике

Студент: Патяка Диана Федоровна Группа: 241-335

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

Отчет принят с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики: Рябчикова Анна Валерьевна

Москва 2025

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 3](#_Toc199328367)

1. [Общая информация о проекте 3](#_Toc199328368)

2. [Описание задания по проектной практике 3](#_Toc199328369)

3. [Описание достигнутых результатов по проектной практике 5](#_Toc199328370)

[ЗАКЛЮЧЕНИЕ 12](#_Toc199328371)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 13](#_Toc199328372)

# **ВВЕДЕНИЕ**

Проект направлен на модернизацию цифровой инфраструктуры Московского политехнического университета. Основная цель — повышение эффективности взаимодействия студентов и преподавателей с проектной деятельностью через оптимизацию пользовательского интерфейса Личного кабинета (ЛК).

## ОБЩАЯ ИНФОРМАЦИЯ О ПРОЕКТЕ

**Название:** Совершенствование цифровой инфраструктуры для проектной деятельности.

**Цель:** Улучшение и модернизация цифровой инфраструктуры проектной деятельности университета; расширение функциональных возможностей цифровых сервисов.

**Задачи:**

* Анализ цифровой инфраструктуры проектной деятельности Московского Политеха.
* Поиск и реализация путей модернизации и улучшения функционала цифровых сервисов Московского Политеха.
* Разработка дизайнов сервисов проектной деятельности с добавленным функционалом для их дальнейшего внедрения.

## ОПИСАНИЕ ЗАДАНИЯ ПО ПРОЕКТНОЙ ПРАКТИКЕ

Задание состояло из двух частей. Первая часть базовая. Вторая часть вариативная. Трудоёмкость практики составляла 72 академических часа. Задание выполнялось индивидуально. Для управления версиями будет использовался Git, для написания документации — Markdown, а для создания статического веб-сайта — языки разметки HTML и CSS, но опционально использовался генератор статических сайтов Hugo. В качестве платформы для размещения репозиториев использовался как GitHub, что обеспечило гибкость в выборе инструментов.

1. Базовая часть
   1. Настройка Git и репозитория:

* Создайте личный или групповой репозиторий на GitHub или GitVerse на основе предоставленного шаблона.
* Освойте базовые команды Git: клонирование, коммит, пуш и создание веток.
* Регулярно фиксируйте изменения с осмысленными сообщениями к коммитам.
* Ожидаемое время: 5 часов.
  1. Написание документов в Markdown:
* Все материалы проекта (описание, журнал прогресса и др.) должны быть оформлены в формате Markdown.
* Изучите синтаксис Markdown и подготовьте необходимые документы.
* Ожидаемое время: 5 часов.
  1. Создание статического веб-сайта:
* Вы можете использовать только HTML и CSS для создания сайта, если освоение более сложных инструментов представляется трудным. Это делает задание доступным для студентов с базовым уровнем подготовки.
* Желательно применять генераторы статических сайтов, такие как Hugo (рекомендуется), для упрощения процесса и получения дополнительных навыков. В случае выбора Hugo можно воспользоваться инструкциями из Hugo Quick Start Guide.
* Создайте новый сайт об основном проекте по дисциплине «Проектная деятельность», выберите тему и добавьте контент. Оформление и наполнение сайта должны быть уникальными (не совпадать с работами других студентов) более, чем на 50%.
* Сайт должен включать:
* Домашнюю страницу с аннотацией проекта.
* Страницу «О проекте» с описанием проекта.
* Страницу или раздел «Участники» с описанием личного вклада каждого участника группы в проект по «Проектной деятельности».
* Страницу или раздел «Журнал» с минимум тремя постами (новостями, блоками) о прогрессе работы.
* Страницу «Ресурсы» со ссылками на полезные материалы (ссылки на организацию-партнёра, сайты и статьи, позволяющие лучше понять суть проекта).
* Оформите страницы сайта графическими материалами (фотографиями, схемами, диаграммами, иллюстрациями) и другой медиа информацией (видео).
* Ожидаемое время: изучение и настройка — 10–14 часов, дизайн и наполнение — 4–8 часов.

1. Вариативная часть (Практическая реализация технологии)

**Тема:** Создание нейронной сети на языке Python.

**Цель:** Разработка и обучение простой искусственной нейронной сети для моделирования логической операции "И".

**Задачи:**

* Подготовить обучающую выборку.
* Создать архитектуру нейросети.
* Обучить модель.
* Оценить эффективность модели.
* Вывести предсказания модели.

## ОПИСАНИЕ ДОСТИГНУТЫХ РЕЗУЛЬТАТОВ ПО ПРОЕКТНОЙ ПРАКТИКЕ

**Настройка Git и репозитория**

Был создан персональный репозиторий на платформе GitHub.

Освоены базовые команды:

* git clone
* git checkout -b feature\_branch
* git add .
* git commit -m "Сообщение"
* git push

Время на выполнение: 5 часов.

**Документация в формате Markdown**

1. Заголовки

# H1, ## H2, ### H3

1. Форматирование текста

\*\*Жирный\*\*, \*Курсив\*, ~~Зачёркнутый~~

1. Списки

* Нумерованные
* Маркированные

1. Ссылки и изображения

* [Текст](https://example.com)
* ![Alt-текст](image.jpg)

1. Таблицы

| Заголовок | Пример |

|----------|--------|

| Ячейка | Данные |

1. Блоки кода

```python

print("Hello!")

```

Время на выполнение: 5 часов.

**Создание статического веб-сайта**

Стек технологий:

* HTML
* CSS
* (опционально) Hugo — генератор статических сайтов
* Netlify

Структура сайта:

├── content/

│ └── journal/

│ ├── index.md

│ ├── analysis.md

│ ├── lk-educator.md

│ ├── lk-student.md

│ ├── pac.md

│ ├── about.md

│ ├── participants.md

│ └── resources.md

├── static/

│ └── images/

│ ├── analysis.png

│ ├── educator-qr.png

│ ├── lk-educator.png

│ ├── moskovskij\_politeh.png

│ ├── pac.png

│ └── student-qr.png

└── themes/

└── hugo-coder/

Реализованные страницы:

* Домашняя страница с аннотацией проекта.
* Страница «О проекте» с подробным описанием.
* Страница «Участники», где описан личный вклад каждого участника группы.
* Страница «Журнал» с тремя этапами разработки.
* Страница «Ресурсы» с полезными ссылками для создания сайтов.
* Ссылка на сайт: <https://tiny-brioche-4b538a.netlify.app>

Время на выполнение:

* Изучение Hugo и настройка — 12 часов
* Разработка и оформление сайта — 6 часов

**Посещение мероприятия "Создание технического задания на разработку телеграмм-ботов" от "Молодежь Москвы"**

На мероприятии я научилась структурировать ТЗ: описывать функционал, проектировать пользовательские сценарии и формулировать технические требования. Практиковалась на кейсе бота-напоминаний: прописала команды (/start, /set\_reminder), примеры ответов и требования к хранению данных. Узнала, как избежать типичных ошибок в ТЗ (размытые формулировки, недооценка нагрузки). Эти навыки помогут в разработке собственных ботов и оформлении документации.

Время на выполнение: 5 часов

**Создание нейронной сети (вариативная часть)**

Я создала простую нейросеть с одним нейроном, которая учится выполнять логическое И. Она берет два числа, умножает на веса, добавляет сдвиг, пропускает через сигмоиду и выдает число от 0 до 1. Во время обучения она сравнивает результат с правильным ответом и меняет веса, чтобы ошибаться меньше. Так сеть учится правильно говорить «1» только если оба входа равны 1, и «0» в остальных случаях.

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **A И B** |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Процесс создания:

1. Подготовка данных

* x – входные данные, все возможные комбинации 0 и 1.
* y – правильные ответы, чему должен равняться A и B.

x = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])  
y = np.array([[0], [0], [0], [1]])

1. Структура сети

Сеть состоит из одного слоя с одним нейроном, принимающим 2 входа. Это называется однослойный перцептрон.

1. Сигмоида

* Преобразует любое число в диапазон от 0 до 1.
* Используется для «плавного» принятия решения (0 или 1).
* Позволяет считать выход нейрона как вероятность.

def sigmoid(z):  
 return 1 / (1 + np.exp(-z))  
  
def sigmoid\_derivative(z):  
 return sigmoid(z) \* (1 - sigmoid(z))

1. Инициализация параметров

weights = np.random.randn(2, 1) \* 0.1 # веса (2 входа, 1 выход)  
bias = np.zeros((1,)) # смещение (bias)

1. Гиперпараметры

* **learning rate (скорость обучения):** насколько сильно мы меняем веса после каждой итерации.
* **epochs:** сколько раз нейросеть "видит" все обучающие примеры.

learning\_rate = 0.1  
epochs = 10000

1. Обучение

* Прямой проход
* Вычисляем линейную комбинацию входов и весов.
* Пропускаем через сигмоиду для получения предсказания.

z = np.dot(x, weights) + bias # линейная комбинация  
y\_pred = sigmoid(z) # активация

* Ошибка

error = y - y\_pred

* Градиенты

Чтобы сеть училась, нужно понять, в каком направлении менять веса. Градиент показывает, как сильно надо изменить каждый вес, чтобы уменьшить ошибку.

dcost\_dypred = error  
dypred\_dz = sigmoid\_derivative(z)  
  
dz = dcost\_dypred \* dypred\_dz  
  
weights\_gradient = np.dot(x.T, dz) / x.shape[0]  
bias\_gradient = np.mean(dz, axis=0)

* Обновление параметров

Меняем веса и смещение, чтобы предсказания стали точнее.

weights += learning\_rate \* weights\_gradient  
bias += learning\_rate \* bias\_gradient

1. Проверка

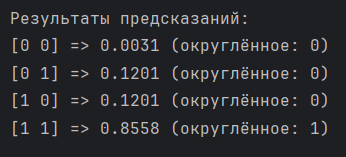
Считаем финальные предсказания и округляем их (0 или 1), чтобы сравнить с правильными ответами.

z = np.dot(x, weights) + bias  
predictions = sigmoid(z)

Код:

import numpy as np  
  
# Данные: входы и выходы  
x = np.array([[0, 0], [0, 1], [1, 0], [1, 1]]) # Входы  
y = np.array([[0], [0], [0], [1]]) # Цели (A AND B)  
  
# Сигмоида и её производная  
def sigmoid(z):  
 return 1 / (1 + np.exp(-z))  
  
def sigmoid\_derivative(z):  
 return sigmoid(z) \* (1 - sigmoid(z))  
  
# Параметры модели  
np.random.seed(42)  
weights = np.random.randn(2, 1) \* 0.1 # веса (2 входа, 1 выход)  
bias = np.zeros((1,)) # смещение (bias)  
  
# Гиперпараметры  
learning\_rate = 0.1  
epochs = 10000  
  
# Обучение (градиентный спуск)  
for epoch in range(epochs):  
 # Прямой проход  
 z = np.dot(x, weights) + bias # линейная комбинация  
 y\_pred = sigmoid(z) # активация  
  
 # Ошибка  
 error = y - y\_pred  
  
 # Градиенты  
 dcost\_dypred = error  
 dypred\_dz = sigmoid\_derivative(z)  
  
 dz = dcost\_dypred \* dypred\_dz  
  
 weights\_gradient = np.dot(x.T, dz) / x.shape[0]  
 bias\_gradient = np.mean(dz, axis=0)  
  
 # Обновление параметров  
 weights += learning\_rate \* weights\_gradient  
 bias += learning\_rate \* bias\_gradient  
  
# Проверяем результат  
print("\nРезультаты предсказаний:")  
z = np.dot(x, weights) + bias  
predictions = sigmoid(z)  
for i in range(len(x)):  
 print(f"{x[i]} => {predictions[i][0]:.4f} (округлённое: {round(predictions[i][0])})")

Результат:



Время на выполнение: 40 часов

# **ЗАКЛЮЧЕНИЕ**

В ходе практики была достигнута основная цель: приобретены навыки работы с Git, создания и оформления документации в формате Markdown, разработан уникальный статический веб-сайт, а также создана нейронная сеть. Полученные знания и умения будут использованы в дальнейшем обучении и профессиональной деятельности.

# **СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. Чакон С., Штрауб Б. Pro Git. 2-е изд. — СПб.: Питер, 2015. — 456 с.
2. Загребельный В. Git и GitHub. Полное руководство. — М.: ДМК Пресс, 2021. — 320 с.
3. Hugo. The world’s fastest framework for building websites [Электронный ресурс]. — Режим доступа: <https://gohugo.io/>.
4. Дакетт Дж. HTML и CSS. Разработка и дизайн веб-сайтов. — СПб.: Питер, 2019. — 512 с.
5. Робсон Э., Фримен Э. HTML и CSS. Разработка сайтов для начинающих. — М.: Эксмо, 2022. — 368 с.
6. Нильсон Э. CSS. Карманный справочник. — М.: Символ-Плюс, 2018. — 288 с.
7. Гудфеллоу И., Бенджио Й., Курвил А. Глубокое обучение. — М.: Диалектика, 2018. — 640 с.
8. Raschka S., Mirjalili V. Python Machine Learning. — 3rd ed. — Packt Publishing, 2019. — 770 p.
9. Нейронные сети с нуля на NumPy [Электронный ресурс] // Хабр. — Режим доступа: <https://habr.com/ru/post/312450/>.
10. Официальная документация NumPy [Электронный ресурс]. — Режим доступа: <https://numpy.org/doc/>.