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: Краснов Родион Денисович. Группа: 241-339

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

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

Руководитель практики: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва 2025

ОГЛАВЛЕНИЕ

Оглавление

[**Введение** 2](#_Toc198335625)

[**Общая информация о проекте** 3](#_Toc198335626)

[**Общая характеристика деятельности организации** 4](#_Toc198335627)

[**Описание задания по проектной практике** 5](#_Toc198335628)

[Базовая часть задания 5](#_Toc198335629)

[Практическая реализация технологии 11](#_Toc198335630)

[Описание достигнутых результатов по проектной деятельности 16](#_Toc198335631)

[Заключение 18](#_Toc198335632)

# **Введение**

В рамках проектной практики студентам группы 241-339 была поставлена задача разработать систему централизованной отчётности и предикативную модель для Московского Политеха — образовательного учреждения, активно внедряющего цифровые технологии в свою деятельность.

Мною были освоены современные инструменты разработки: Django — для создания сайта проекта, HTML/CSS — для верстки страниц, Python — для реализации технологической части (включая блокчейн), Git — для командной работы и управления версиями кода.

Данный отчет содержит описание проделанной работы, достигнутых результатов и приобретённого опыта.

# **Общая информация о проекте**

**Название проекта:** «Система централизованной отчётности и предикативная модель операционных показателей для Московского Политеха»

**Цель проекта:**

Создание комплексной системы, объединяющей возможности анализа больших данных и облачных технологий, которая позволит автоматизировать сбор и обработку информации, формирование отчетов и прогнозирование ключевых показателей деятельности университета.

**Задачи проекта:**

1. Разработка модуля анализа данных.

2. Автоматизация отчётности.

3. Интеграция с популярными CRM-системами.

4. Обеспечение безопасности данных.

5. Реализация дашбордов для визуализации данных.

# **Общая характеристика деятельности организации**

Проект реализуется совместно с Московским Политехническим университетом — одним из ведущих технических вузов страны. Университет заинтересован в повышении эффективности своей деятельности через внедрение цифровых решений.

Команда проекта регулярно взаимодействует с представителями университета для согласования требований, получения обратной связи и тестирования решения.

В ходе практики проводились встречи с представителями Московского Политеха, на которых обсуждались:

- Требования к функционалу системы

- Потенциальные точки интеграции с существующими системами университета

- Возможности использования модели прогнозирования в учебном процессе и административном управлении

Коммуникация осуществляется как очно, так и удалённо. Также запланированы демонстрационные презентации прототипа.

# **Описание задания по проектной практике**

## Базовая часть задания

Цель: создание пятистраничного сайта на фреймворке Django, который будет служить основой будущей системы. Сайт должен включать следующие разделы:

- Главная

- О проекте

- Участники

- Журнал

- Ресурсы

Этапы выполнения:

1. Планирование структуры сайта

- Определены страницы и навигация между ними.

- Создан базовый шаблон `base.html`, используемый всеми страницами.

2. Создание базового шаблона

- Реализован общий интерфейс сайта с меню навигации, футером и динамическим контентом (`{% block content %}`).

- Применены CSS-стили для анимации переходов, выделения активных ссылок и карточек новостей.

3. Реализация страниц

- Главная страница (home.txt): описание проекта, ключевые особенности, информация о команде и кнопки навигации.

- О проекте (about.txt): цель, задачи, этапы реализации, ожидаемые результаты.

- Участники (team.txt): список членов команды, их роли и краткое описание вклада.

- Журнал проекта (journal.txt): история событий, связанных с реализацией проекта, с указанием дат и этапов работ.

- Ресурсы проекта (resources.txt): ссылки на полезные материалы, документацию и рекомендации по дальнейшему изучению темы.

4. Стилизация сайта

- Реализован файл стилей `style.css` с классами для анимации, оформления карточек, меню и других элементов.

- Выполнена адаптация дизайна под мобильные устройства.

5. \*\*Завершение и тестирование\*\*

- Проверена корректность отображения всех страниц.

- Убедились в правильной работе навигации и отображения контента.

- Проведено финальное тестирование и исправление недочётов.

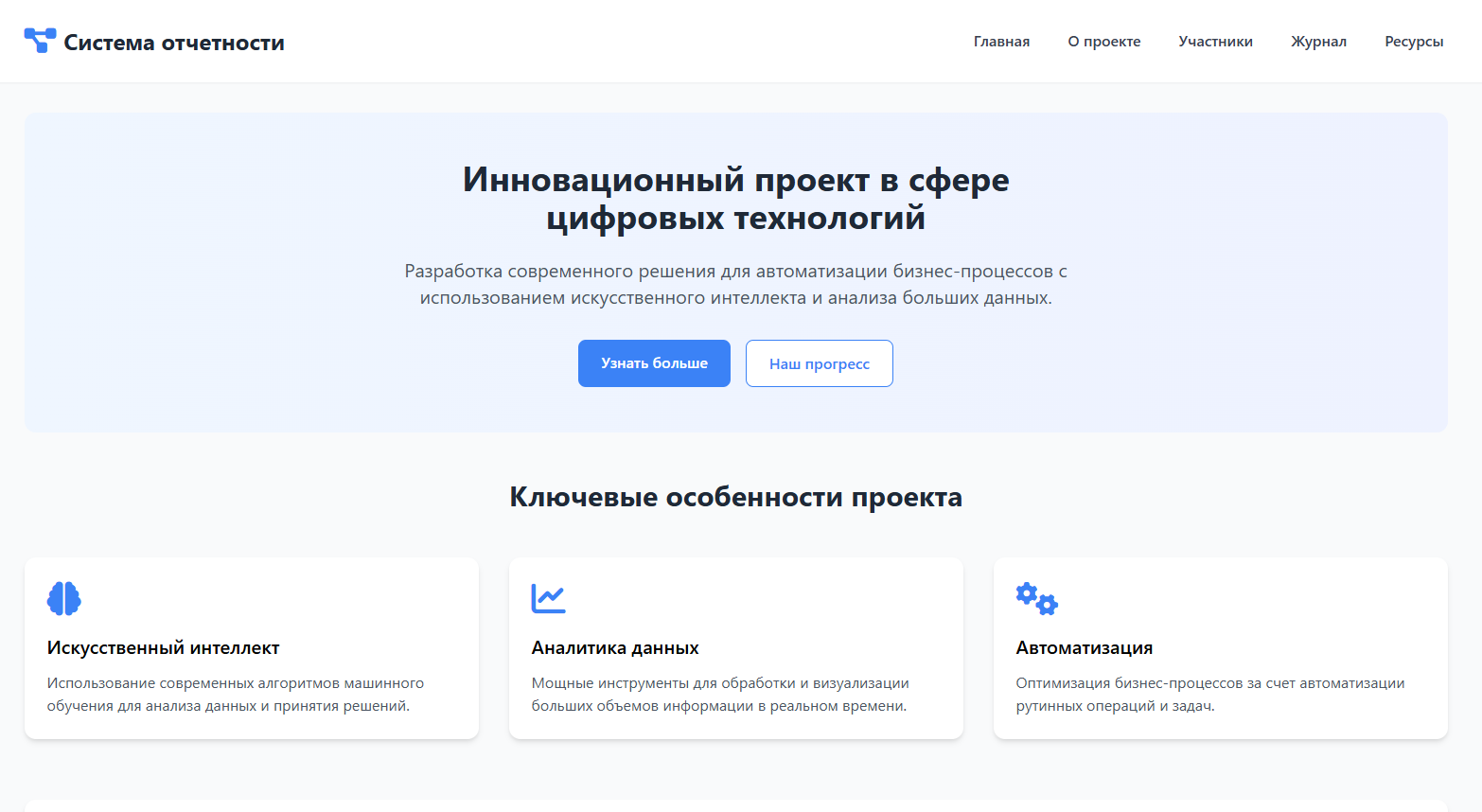


Рисунок . Главная страниц

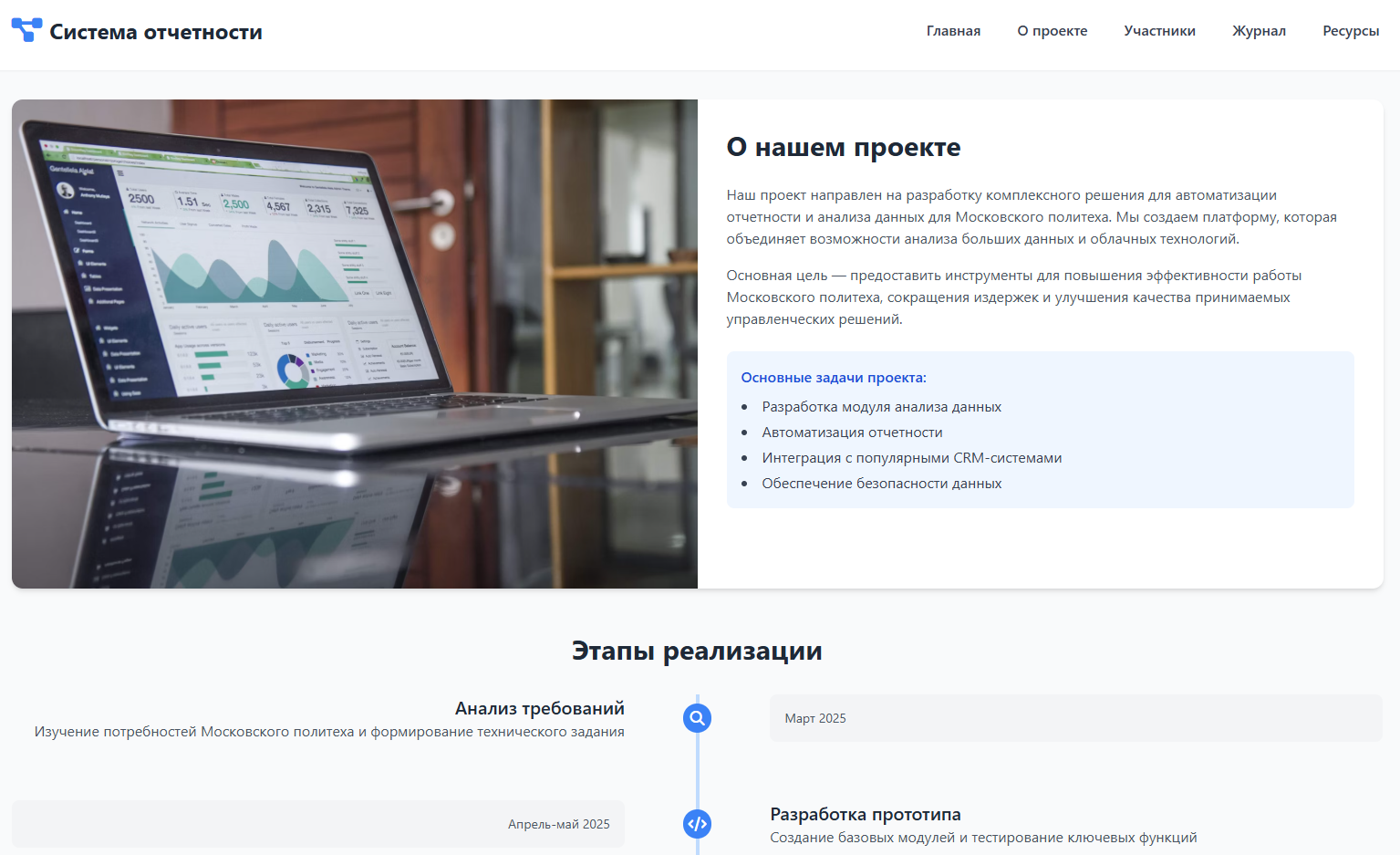


Рисунок . Страница "О проекте"

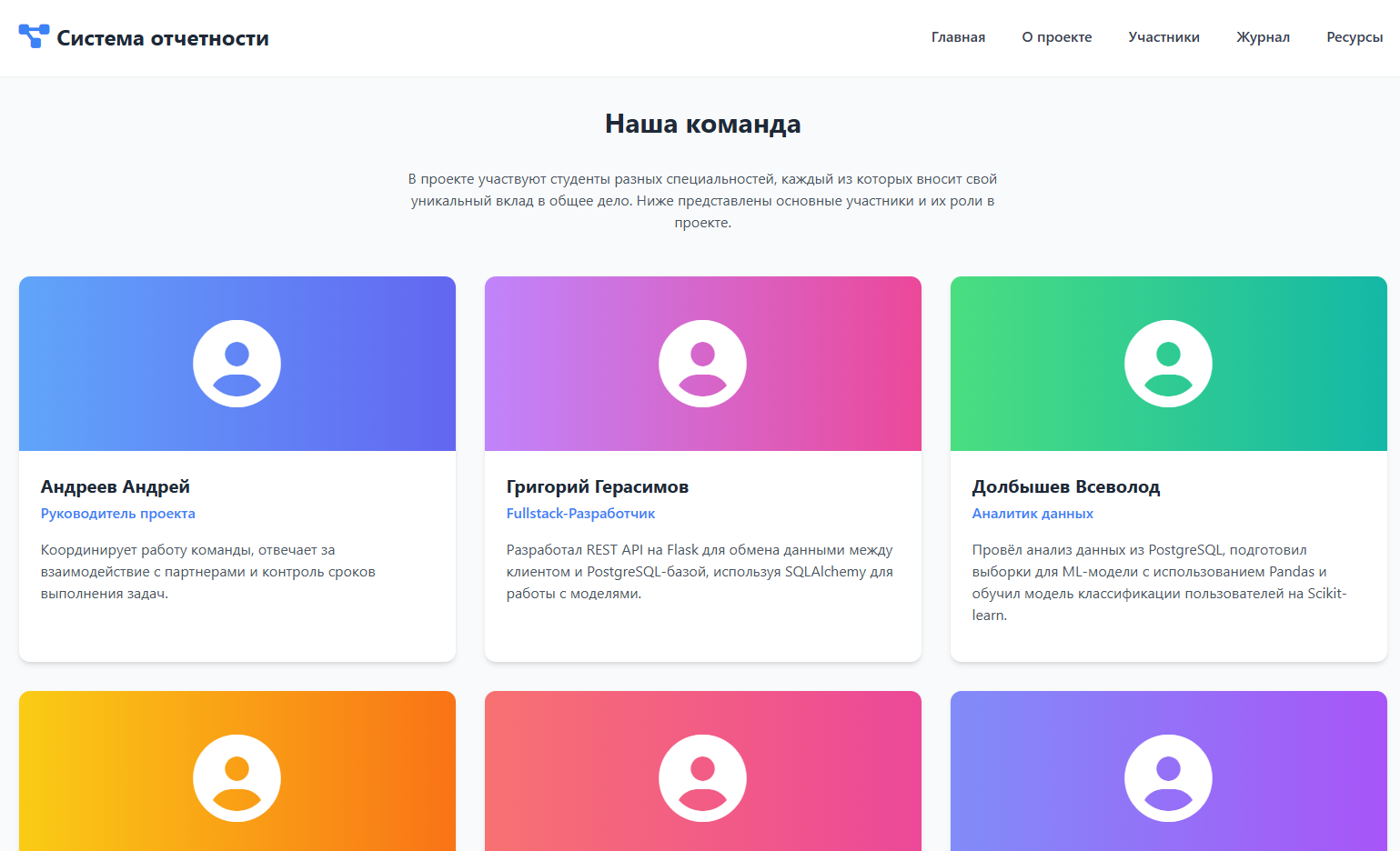


Рисунок . Страница "Участники"

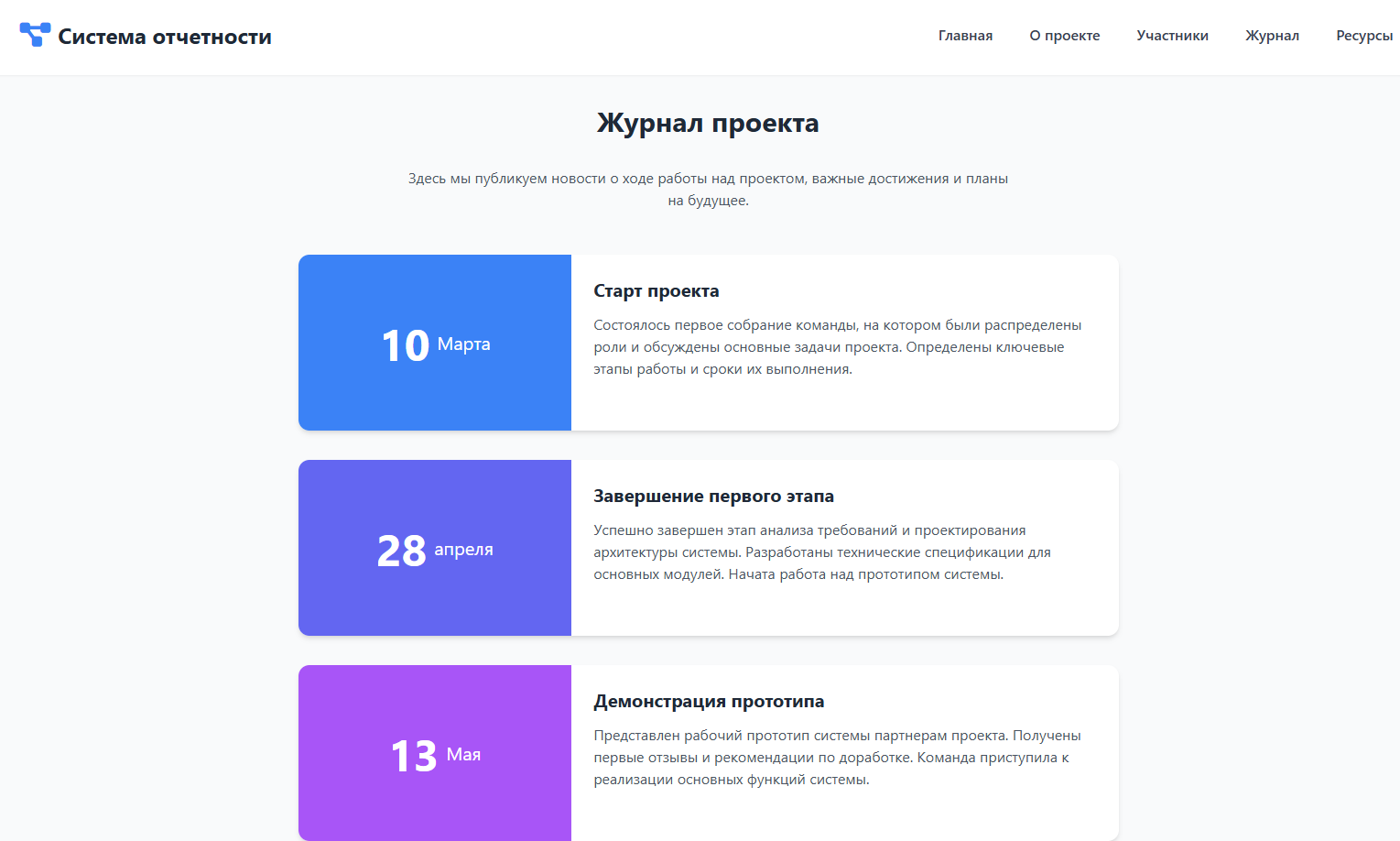


Рисунок . Страница "Журнал"

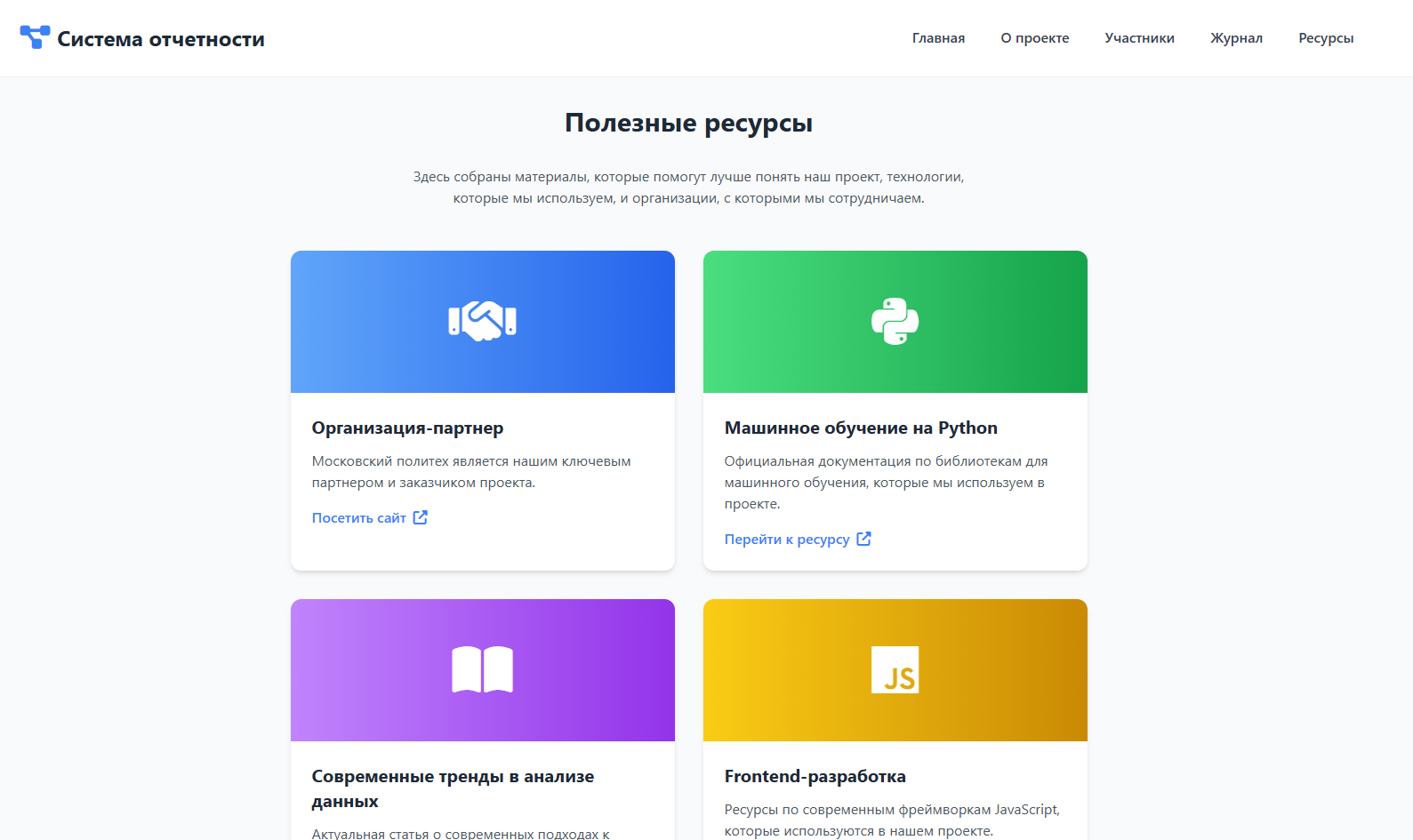


Рисунок . Страница "Ресурсы"

Результат: создан многостраничный сайт на Django, который отражает содержание и прогресс проекта. Он включает информативные разделы, визуальные элементы и работает корректно на разных устройствах.

## Практическая реализация технологии

Тема: Блокчейн на Python

Цель проекта:

Разработка простого блокчейна на языке программирования Python для понимания принципов работы децентрализованных систем, хэширования данных, проверки целостности цепочки и управления состоянием системы.

Проделанные этапы разработки:

1. Инициализация блокчейна

- Создан класс `Blockchain`, управляющий цепочкой и состоянием системы.

- Реализован метод `create\_genesis\_block()` для создания начального блока.

2. Генерация транзакций

- Реализован метод `make\_transaction()` для создания случайных транзакций между участниками.

3. Обновление состояния системы

- Метод `update\_state()` изменяет балансы участников после успешной транзакции.

4. Хэширование данных

- Реализован метод `hash\_me()` для хэширования содержимого блока с использованием SHA-256.

5. Добавление блоков

- Метод `add\_block()` добавляет новый блок в цепочку, проверяя каждую транзакцию на корректность.

6. Проверка целостности цепочки

- Метод `check\_chain\_validity()` проверяет все блоки на соответствие правилам: номер, хэш родителя, корректность транзакций.

7. Вывод информации

- Метод `print\_chain()` выводит структуру всех блоков.

- Метод `get\_current\_state()` возвращает текущее состояние системы (балансы участников).

Код:

*import* hashlib  
*import* json  
*import* random  
*import* copy  
*import* sys  
*from* pprint *import* pprint  
  
  
*class* Blockchain:  
 *def \_\_init\_\_*(*self*):  
 *self*.chain = [*self*.create\_genesis\_block()]  
 *self*.state = {}  
  
 *def* hash\_me(*self*, msg=""):  
 *"""Хэширует данные с использованием SHA-256"""  
 if not isinstance*(msg, *str*):  
 msg = json.dumps(msg, sort\_keys=*True*)  
  
 *return* hashlib.sha256(*str*(msg).encode('utf-8')).hexdigest()  
  
 *def* make\_transaction(*self*, max\_value=3):  
 *"""Создает случайную транзакцию между Alice и Bob"""* sign = *int*(random.getrandbits(1)) \* 2 - 1 *# Случайный знак: -1 или 1* amount = random.randint(1, max\_value)  
 alice\_pays = sign \* amount  
 bob\_pays = -alice\_pays  
 *return* {"Alice": alice\_pays, "Bob": bob\_pays}  
  
 *def* update\_state(*self*, txn, state):  
 *"""Обновляет состояние (балансы) на основе транзакции"""* state = copy.deepcopy(state)  
 *for* key *in* txn:  
 *if* key *in* state:  
 state[key] += txn[key]  
 *else*:  
 state[key] = txn[key]  
 *return* state  
  
 *def* is\_valid\_transaction(*self*, txn, state):  
 *"""Проверяет, корректна ли транзакция"""  
 if sum*(txn.values()) != 0:  
 *return False # Транзакция должна быть сбалансированной  
  
 for* key *in* txn:  
 balance = state.get(key, 0)  
 *if* balance + txn[key] < 0:  
 *return False # Недостаточно средств  
 return True  
  
 def* create\_genesis\_block(*self*):  
 *"""Создает генезис-блок"""* genesis\_txns = [*self*.make\_transaction() *for* \_ *in range*(10)]  
 genesis\_contents = {  
 "blockNumber": 0,  
 "parentHash": *None*,  
 "txnCount": *len*(genesis\_txns),  
 "txns": genesis\_txns  
 }  
 genesis\_hash = *self*.hash\_me(genesis\_contents)  
 *return* {"hash": genesis\_hash, "contents": genesis\_contents}  
  
 *def* add\_block(*self*, txns):  
 *"""Добавляет новый блок в цепочку"""* parent\_block = *self*.chain[-1]  
 block\_number = parent\_block["contents"]["blockNumber"] + 1  
 block\_contents = {  
 "blockNumber": block\_number,  
 "parentHash": parent\_block["hash"],  
 "txnCount": *len*(txns),  
 "txns": []  
 }  
  
 *# Проверяем и добавляем только валидные транзакции* valid\_txns = []  
 temp\_state = copy.deepcopy(*self*.state)  
 *for* txn *in* txns:  
 *if self*.is\_valid\_transaction(txn, temp\_state):  
 valid\_txns.append(txn)  
 temp\_state = *self*.update\_state(txn, temp\_state)  
  
 block\_contents["txns"] = valid\_txns  
 block\_contents["txnCount"] = *len*(valid\_txns)  
 block\_hash = *self*.hash\_me(block\_contents)  
 new\_block = {"hash": block\_hash, "contents": block\_contents}  
 *self*.chain.append(new\_block)  
 *self*.state = temp\_state  
  
 *def* check\_block\_hash(*self*, block):  
 *"""Проверяет, совпадает ли хеш с содержимым блока"""* expected\_hash = *self*.hash\_me(block["contents"])  
 *if* block["hash"] != expected\_hash:  
 *raise Exception*(f"Неверный хеш в блоке {block['contents']['blockNumber']}")  
  
 *def* check\_block\_validity(*self*, block, parent\_block, state):  
 *"""Проверяет корректность одного блока"""* block\_num = block["contents"]["blockNumber"]  
 parent\_num = parent\_block["contents"]["blockNumber"]  
  
 *if* block\_num != parent\_num + 1:  
 *raise Exception*(f"Некорректный номер блока: {block\_num}")  
  
 *if* block["contents"]["parentHash"] != parent\_block["hash"]:  
 *raise Exception*(f"Некорректный родительский хеш в блоке {block\_num}")  
  
 *self*.check\_block\_hash(block)  
  
 temp\_state = copy.deepcopy(state)  
 *for* txn *in* block["contents"]["txns"]:  
 *if not self*.is\_valid\_transaction(txn, temp\_state):  
 *raise Exception*(f"Некорректная транзакция в блоке {block\_num}: {txn}")  
 temp\_state = *self*.update\_state(txn, temp\_state)  
  
 *return* temp\_state  
  
 *def* check\_chain\_validity(*self*):  
 *"""Проверяет целостность всей цепочки"""  
 try*:  
 temp\_state = {}  
 *# Проверяем генезис-блок  
 for* txn *in self*.chain[0]["contents"]["txns"]:  
 temp\_state = *self*.update\_state(txn, temp\_state)  
 *self*.check\_block\_hash(*self*.chain[0])  
  
 parent = *self*.chain[0]  
 current\_state = temp\_state  
  
 *# Проверяем остальные блоки  
 for* block *in self*.chain[1:]:  
 current\_state = *self*.check\_block\_validity(block, parent, current\_state)  
 parent = block  
  
 *return* current\_state  
 *except Exception as* e:  
 *print*("Ошибка проверки цепочки:", e)  
 *return False  
  
 def* print\_chain(*self*):  
 *"""Выводит текущую цепочку блоков"""  
 for* i, block *in enumerate*(*self*.chain):  
 *print*(f"\nБлок {i}")  
 pprint(block)  
  
 *def* get\_current\_state(*self*):  
 *"""Возвращает текущее состояние системы (балансы)"""  
 return self*.state

Вывод:

Реализован минимальный прототип блокчейна на Python. Он позволяет:

- Создавать транзакции

- Управлять состоянием системы

- Добавлять блоки

- Проверять целостность цепочки

Проект стал отличной практикой по работе с криптографией, структурами данных и принципами децентрализованных систем. Полученные знания могут быть использованы в дальнейших разработках более сложных систем на основе блокчейн-технологий.

## Описание достигнутых результатов по проектной деятельности

Проект «Система централизованной отчётности и предикативная модель операционных показателей для Московского Политеха» представляет собой комплексное решение, направленное на автоматизацию сбора и анализа данных университета, а также прогнозирование ключевых метрик с использованием методов машинного обучения. Работа велась комплексно, с чётким распределением задач между участниками команды.

1. Отдел фронтенд-разработки

Команда отдела фронтенд-разработки сосредоточилась на создании удобного пользовательского интерфейса системы. Были реализованы:

* Страницы регистрации и авторизации;
* Дашборд с визуализацией ключевых показателей;
* Страницы просмотра и редактирования данных;
* Адаптивный дизайн, совместимый с мобильными устройствами.

Для реализации использовались HTML, CSS и шаблонизатор Django. Также были внедрены CSS-стили из файла style.txt, включая анимации переходов, эффекты наведения и выделение активных элементов меню.

2. Отдел бэкенд-разработки

В рамках бэкенд-части была разработана логика серверной части приложения, включающая:

* Обработку пользовательских запросов;
* Валидацию данных;
* Интеграцию с базой данных через ORM
* Реализацию маршрутов для работы с данными и отображения информации.

Также был реализован функционал пагинации списка записей, что позволило эффективно отображать большие объёмы данных.

3. Отдел аналитики и прогнозирования

Были начаты подготовительные работы по анализу данных университета и прогнозированию ключевых показателей:

* Проведена предобработка исторических данных;
* Подготовлены выборки для последующего обучения модели;
* Выбраны алгоритмы машинного обучения, подходящие для решения задачи прогнозирования;
* Начата интеграция аналитического модуля в общую систему.

4. Отдел управления данными

Отдел занимался проектированием и наполнением базы данных, включая:

* Создание таблиц и связей между ними;
* Настройку миграций и проверок целостности данных;

5. Отдел документации и планирования

В этом отделе велась работа по систематизации хода проекта: составлены технические задания, отчёты о прогрессе, планы работ, диаграммы Ганта и дорожные карты. Подготовлена презентационная документация для представления проекта партнёрам и руководству университета.

# Заключение

В процессе работы я получил ценный практический опыт в области веб-разработки, дизайна интерфейсов и программирования на Python.

Приняв участие в разработке сайта на Django, я освоил основы фреймворка, научился создавать динамические страницы, работать с шаблонами, формами и URL-маршрутами. Также я выполнил техническое задание по реализации блокчейна на Python, что дало мне возможность лучше понять принципы работы децентрализованных систем.

Полученные знания и опыт стали прочной основой для моего дальнейшего профессионального роста в сфере информационных технологий.