**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**Расчётно-графическое задание**

По дисциплине: Технологии Web-программирования

Тема: **«Сервис для облачного машинного обучения**»

|  |  |
| --- | --- |
|  | Выполнил:  студент группы ПВ-192  Кальченко Владислав  Проверил:  Картамышев С.В. |

Белгород 2022

Оглавление

[HTML. Разработка макетов и верстка шаблонов web-приложения с помощью языков HTML и CSS 3](#_Toc119319629)

[Главная страница 3](#_Toc119319630)

[Контентная страница 4](#_Toc119319631)

[Страница авторизации 4](#_Toc119319632)

[Клиентское программирование 5](#_Toc119319633)

[Главная страница 5](#_Toc119319634)

[Контентная страница 5](#_Toc119319635)

[Страница авторизации 8](#_Toc119319636)

[Серверное программирование 10](#_Toc119319637)

[docker-compose.yaml 10](#_Toc119319638)

[Dockerfile программы сервера 11](#_Toc119319639)

[Запрос в Postman 11](#_Toc119319640)

[Разработка и проектирование базы данных 11](#_Toc119319641)

[Схема базы данных в нотации IDEF1X 12](#_Toc119319642)

[Код модели Операции 12](#_Toc119319643)

[REST API 12](#_Toc119319644)

[Схема API (в Postman) 13](#_Toc119319645)

[Работа с HTTP запросами 17](#_Toc119319646)

[Код запроса 17](#_Toc119319647)

[Результат работы запроса 19](#_Toc119319648)

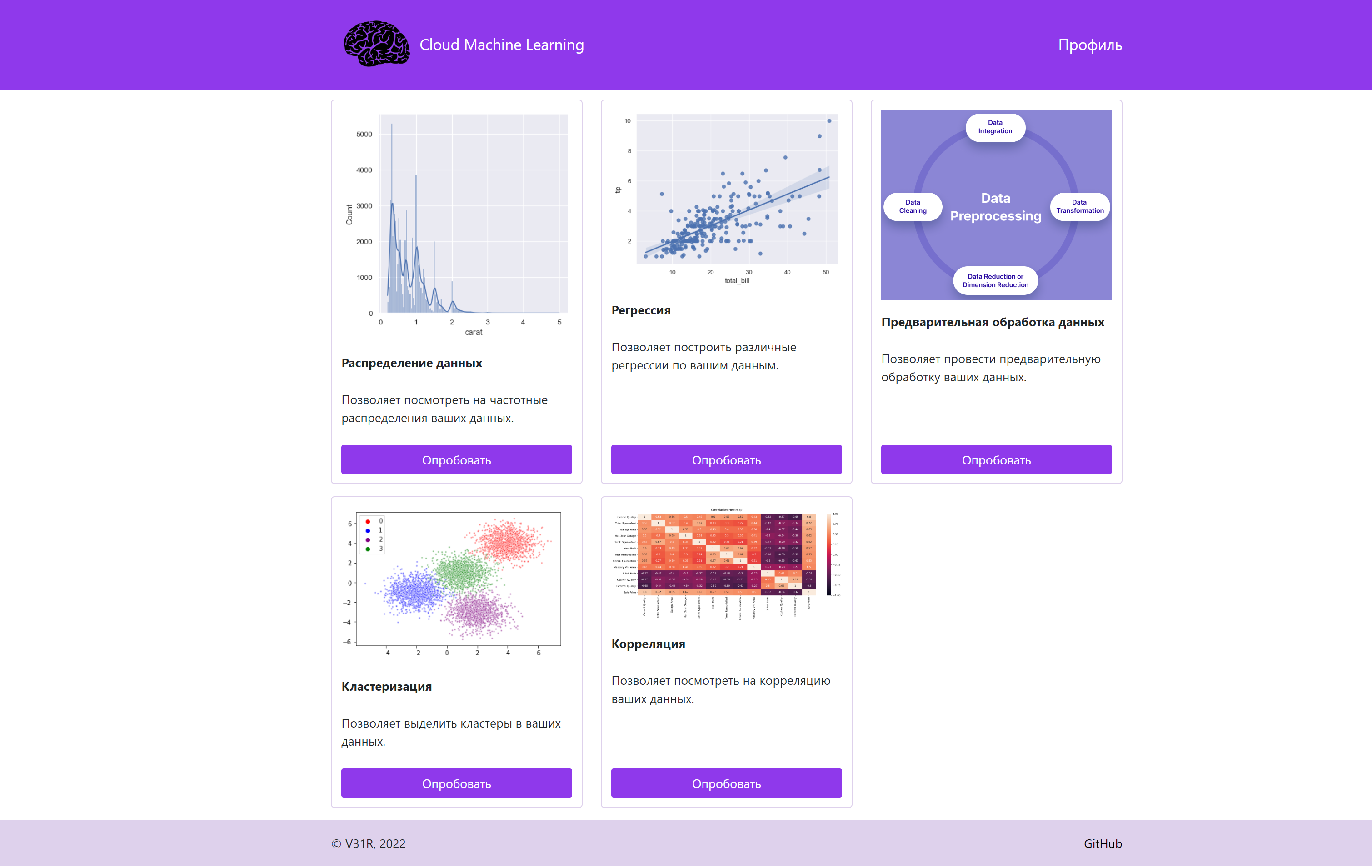
**Цель работы**: создать web приложение с сайтом и сервером для сервиса с облачным машинным обучением, с регистрацией и сохранением историей операций(и их данных).

**Ход работы**

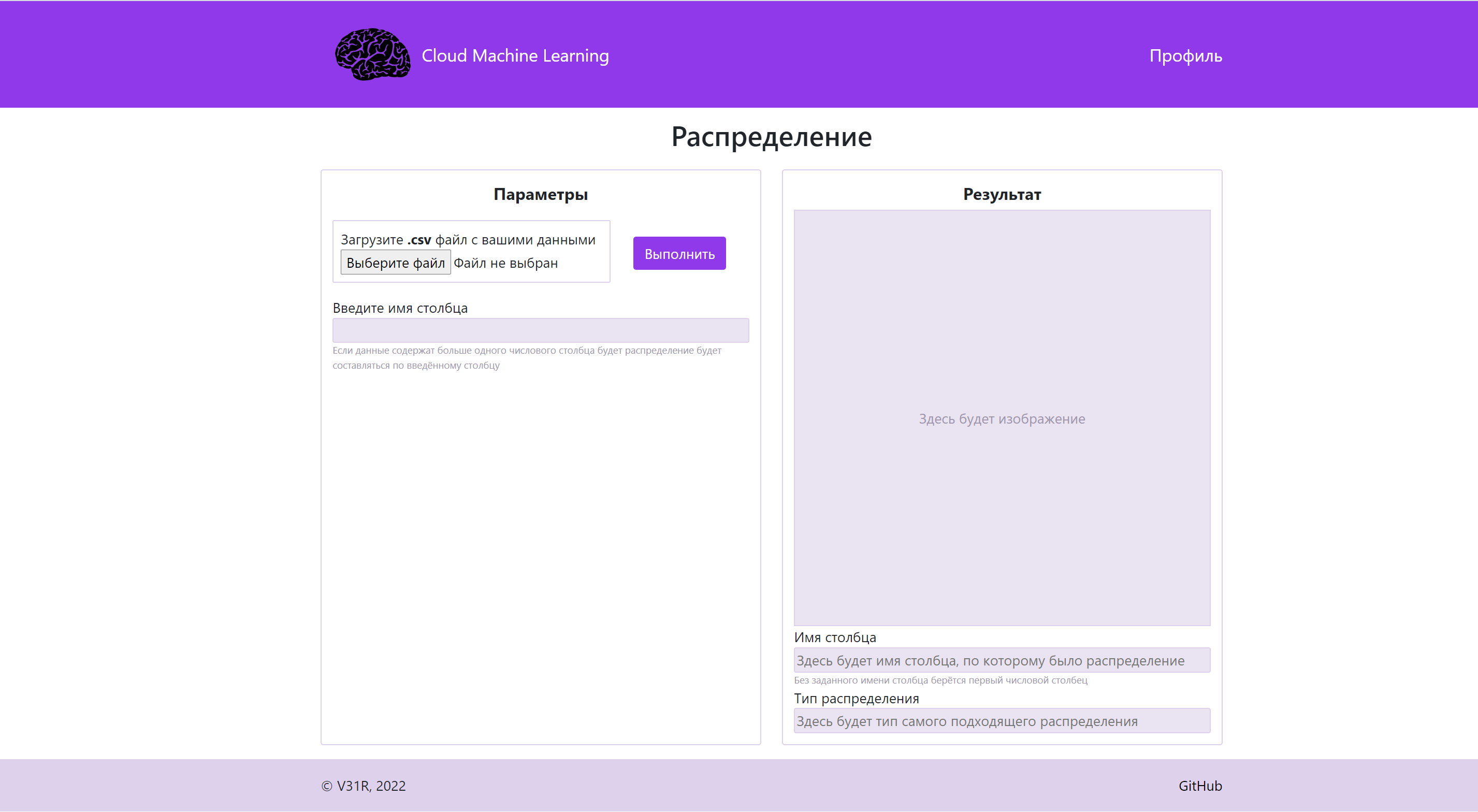
# HTML. Разработка макетов и верстка шаблонов web-приложения с помощью языков HTML и CSS

1. Выбрать шаблон или разработать свой для web-приложения. Шаблон должен включать минимум страницы (главная, контентная страница, страница авторизации/регистрации).
2. Сделать макеты страниц с помощью языка разметки HTML.
3. Стилизовать страницы с помощью языка CSS.
4. Продемонстрировать внешний вид разработанных страниц.

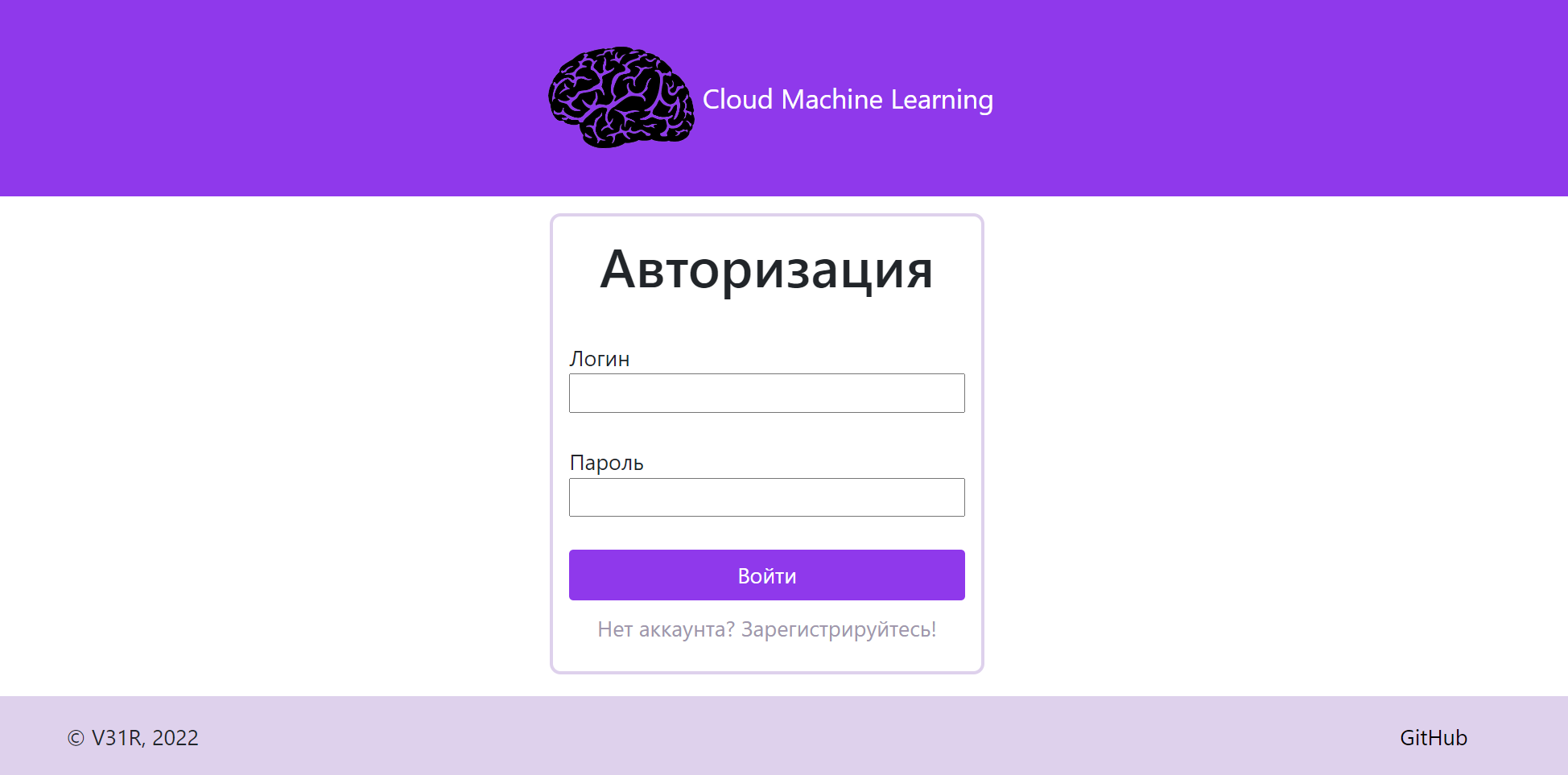
## Главная страница



## Контентная страница



## Страница авторизации



1. Изучить основы разработки на языке JavaScript.
2. Изучить основы разработки frontend-приложения.
3. Развернуть базовое приложение на фреймворке.
4. Добавить необходимые компоненты и перенести в них вёрстку, сделанную в прошлой лабораторной работе.
5. Продемонстрировать работу web-приложения.

# Клиентское программирование

## Главная страница

import React from 'react';

import '../css/home.css';

import operations from '../data\_operations';

import Header from '../components/header';

import OperationComponent from '../components/operationComponent';

import 'bootstrap/dist/css/bootstrap.min.css';

function Home() {

    let operationsComponents: JSX.Element[] = operations.map(op =>

        <OperationComponent

        key={op.id}

        id={op.id}

        img\_name={op.img\_name}

        name={op.name}

        description={op.description}

        link={op.link}

        page\_link={op.page\_link}

    />);

    return (

        <React.Fragment>

            <Header />

            <main className="container">

                <section className="products-list row">

                    {operationsComponents}

                </section>

            </main>

        </React.Fragment>

    );

}

export default Home;

## Контентная страница

import React from 'react';

import '../css/operationTemplateComponent.css';

import '../css/correlation.css';

import MethodPanel from '../components/methodPanel';

import Header from "../components/header";

import ImagePlace from '../components/imagePlace';

import Matrix from '../components/matrix';

import StringInput from '../components/stringInput';

import InputData from '../models/inputData';

import { getImage } from '../apis/fileApi';

import { postCorrelation } from '../apis/taskApi';

interface CorrelationData {

    image\_name: string,

    names: string[],

    values: number[]

}

function Correlation() {

    const [correlationData, setCorrelationData] = React.useState<CorrelationData>({ image\_name: "", names: [], values: [] });

    const [selectedFile, setSelectedFile] = React.useState<null | any>(null);

    const [image, setImage] = React.useState<null | any>(null);

    const [colorMap, setColorMap] = React.useState<string | null>(null);

    React.useEffect(() => {

        if (correlationData!.image\_name === "") {

                return;

        }

        const get\_image = async () => {

            const image = await getImage(correlationData!.image\_name, sessionStorage.getItem('user'));

            setImage(image)

        }

        get\_image()

    },

        [correlationData]

    );

    const handleSubmit = async (event: any) => {

        event.preventDefault()

        if (selectedFile == null) {

            alert('Загрузите файл формата \*.csv');

            return;

        }

        const formData = new FormData();

        formData.append(`${selectedFile.name}`, selectedFile);

        const result: CorrelationData | null = await postCorrelation(formData, sessionStorage.getItem('user'), colorMap);

        if(result !== null){

            setCorrelationData((oldData: Object) => ({ ...oldData, ...result }))

        }

    }

    const handleFileSelect = (event: any) => {

        if (event.target.files[0] !== undefined) {

            setSelectedFile(event.target.files[0]);

            console.log(event.target.files[0], event.target.files[0].name);

        } else {

            setSelectedFile(null);

        }

    }

    const handleColorMap = (event: any) => {

        setColorMap(event.target.value.trim())

    }

    const inputColormap: InputData = {

        mainLabel: 'Введите название colormap',

        fieldName: 'color\_map',

        defaultValue: '',

        tipLabel: 'Допустимые colormap',

        tipLabelLink: 'https://matplotlib.org/stable/tutorials/colors/colormaps.html',

        onChangeHandle: handleColorMap

    };

    return (

        <React.Fragment>

            <Header />

            <main className="container">

                <div className="template-section">

                    <div>

                        <h2>Корреляция</h2>

                    </div>

                </div>

                <section className="functions-list row mt-2 mb-3">

                    <div className="col-md-12 col-lg-6 mt-1">

                        <div className="function">

                            <div className='template-title mb-1'>

                                Параметры

                            </div>

                            <MethodPanel onChange={handleFileSelect} onSend={handleSubmit} />

                            <StringInput {...inputColormap} />

                        </div>

                    </div>

                    <div className="col-md-12 col-lg-6 mt-1">

                        <div className="function">

                            <div className='template-title mb-1'>

                                Результат

                            </div>

                            <ImagePlace image={image} />

                            <Matrix names={correlationData.names} values={correlationData.values} />

                        </div>

                    </div>

                </section>

            </main>

        </React.Fragment>

    );

}

export default Correlation;

## Страница авторизации

import React from "react";

import '../css/button\_default.css';

import '../css/login.css';

import HeaderCentral from "../components/headerCentral";

import { useNavigate, NavigateFunction, Link } from "react-router-dom";

import authContext from "../authContext";

import 'bootstrap/dist/css/bootstrap.min.css';

import { User } from "../models/user";

import { getAuth } from "../apis/authApi";

function Login() {

    const navigate: NavigateFunction = useNavigate();

    const [login, setLogin] = React.useState<string>('');

    const [password, setPassword] = React.useState<string>('');

    const [isPressed, setPressed] = React.useState<boolean>(false);

    const [user, setUser] = React.useState<User>({ token: '' });

    const [error, setError] = React.useState<string | null>(null);

    const auth\_context = React.useContext(authContext);

    React.useEffect(() => {

        if(user.token === ''){

            return;

        }

        console.log(user);

        sessionStorage.setItem('user', user.token);

        navigate('/profile')

        auth\_context!.setAuthenticated(true);

    },

    // eslint-disable-next-line

        [user]

    );

    function handleSubmit(event: any) {

        event.preventDefault()

        setError(null)

        setPressed(true)

        if (login !== '' && password !== '') {

            getAuth(setUser, setError, login, password)

        }

    };

    function handleLogin(event: any) {

        event.preventDefault()

        setLogin(event.target.value.trim())

    }

    function handlePassword(event: any) {

        event.preventDefault()

        setPassword(event.target.value.trim())

    }

    return (

        <React.Fragment>

            <HeaderCentral />

            <main className="container">

                <section className="login-form mb-3">

                    <div className="title">

                        <h1>Авторизация</h1>

                    </div>

                    <form className="mt-4" method="post">

                        <div className="mb-4">

                            <div><label>Логин</label></div>

                            <div><input type="text" className="w-100" name="login" required onChange={evt => handleLogin(evt)} /></div>

                            {(isPressed && login === '') && <div className="errors">Заполните "Логин"</div>}

                        </div>

                        <div className="mb-4">

                            <div><label>Пароль</label></div>

                            <div><input type="password" className="w-100" name="password" required onChange={evt => handlePassword(evt)} /></div>

                            {(isPressed && password === '') && <div className="errors">Заполните "Пароль"</div>}

                        </div>

                        {error !== null &&

                            <div className="mb-2 reg-tip">

                                <div className="errors">Неверный логин или пароль!</div>

                            </div>

                        }

                        <div className="mb-2">

                            <button type="submit" onClick={(event) => handleSubmit(event)} className="btn button-default btn-login">Войти</button>

                        </div>

                        <div className="mb-2 reg-tip">

                            <Link className='reg-link' to='/registration'>Нет аккаунта? Зарегистрируйтесь!</Link>

                        </div>

                    </form>

                </section>

            </main>

        </React.Fragment>

    );

}

export default Login;

# Серверное программирование

1. Развернуть базовое приложение.
2. Настроить конфигурацию работы приложения с docker.
3. Добавить модуль для работы с API.
4. Добавить несколько контроллеров со статическими данными.
5. Продемонстрировать работу API в Postman.

## docker-compose.yaml

version: "3"

services:

  web:

    build: ./nginx

    container\_name: web-nginx

    depends\_on:

      - aiohttp\_cloud\_ml\_server

    volumes:

      - ./nginx/conf.d/:/etc/nginx/conf.d/

    ports:

      - "8080:80"

  aiohttp\_cloud\_ml\_server:

    build: ./server

    container\_name: aiohttp\_cloud\_ml\_server

    depends\_on:

      - postgres\_db

    env\_file:

      - .env

    restart: on-failure

    volumes:

      - data:/data

    ports:

      - "8081:5000"

  postgres\_db:

    build: ./database

    container\_name: postgres\_database

    restart: always

    env\_file:

      - .env

    volumes:

      - postgres\_data:/var/lib/postgresql/data/

    ports:

      - "5432:5432"

volumes:

  data:

  postgres\_data:

## Dockerfile программы сервера

FROM python:latest

WORKDIR /

RUN /usr/local/bin/python -m pip install --upgrade pip

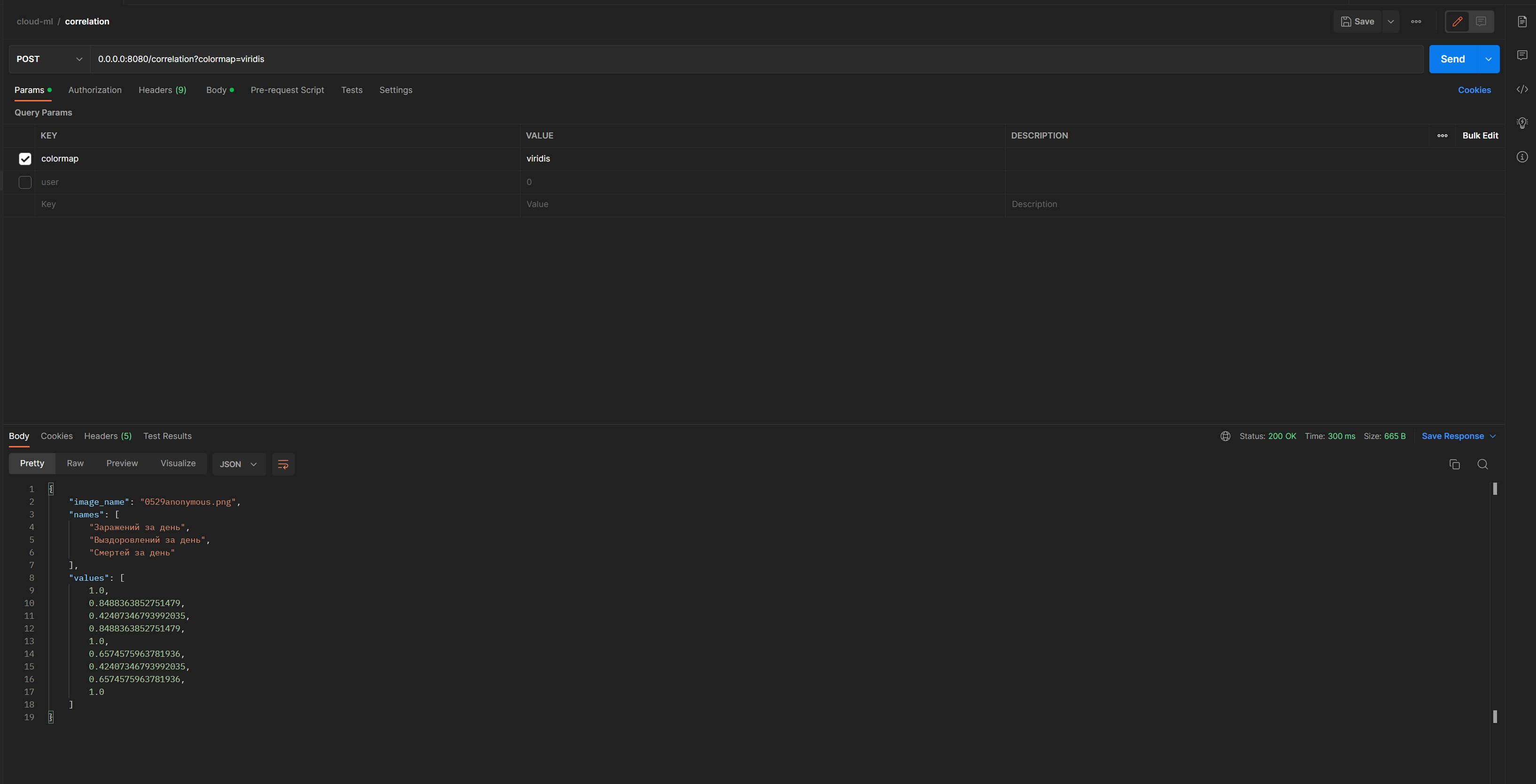
COPY requirements.txt /

RUN pip3 install --upgrade pip -r requirements.txt

COPY . .

CMD ["python","main.py"]

## Запрос в Postman

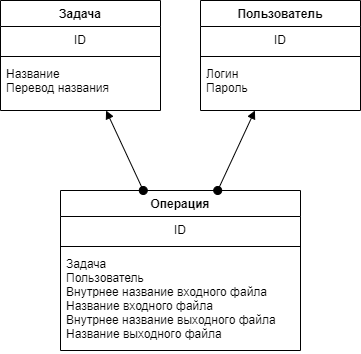


# Разработка и проектирование базы данных

1. Выбрать подходящую СУБД.
2. Изучить методы взаимодействия web-приложения с базой данных.
3. Разработать структуру базы данных.
4. Разработать соответствующие модели в приложении.
5. В отчёт приложить схему базы данных, а так же код одной из моделей (на своё усмотрение).

**СУБД - PostrgeSQL**

## Схема базы данных в нотации IDEF1X



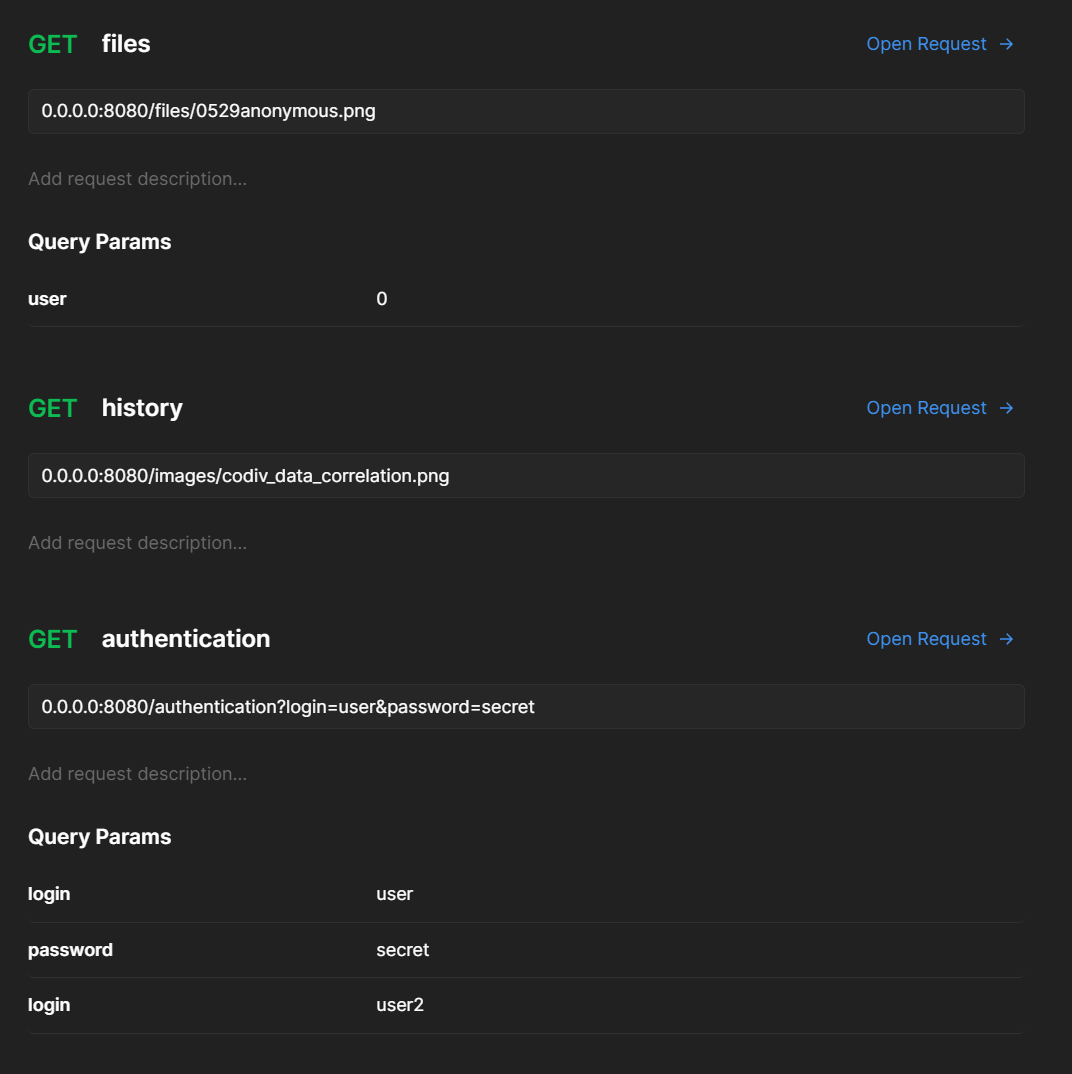
## Код модели Операции

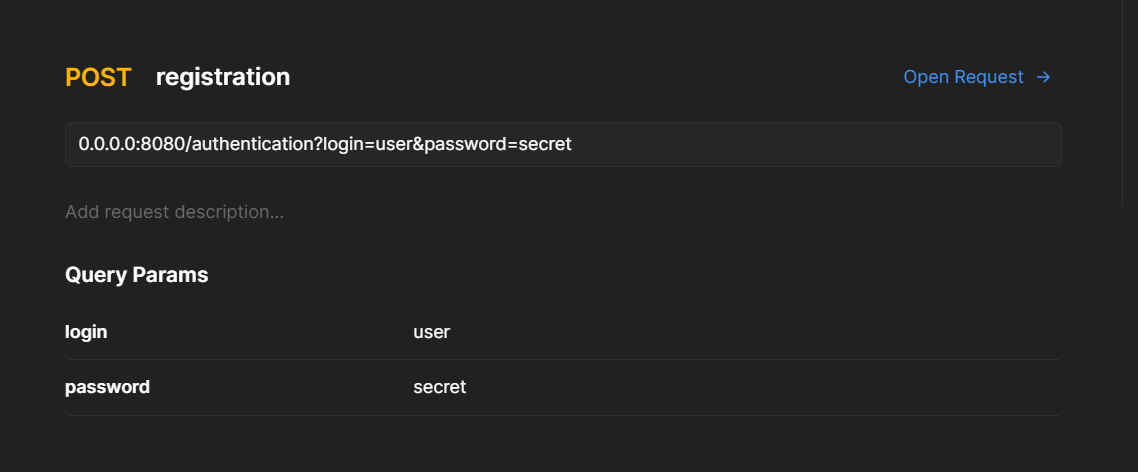
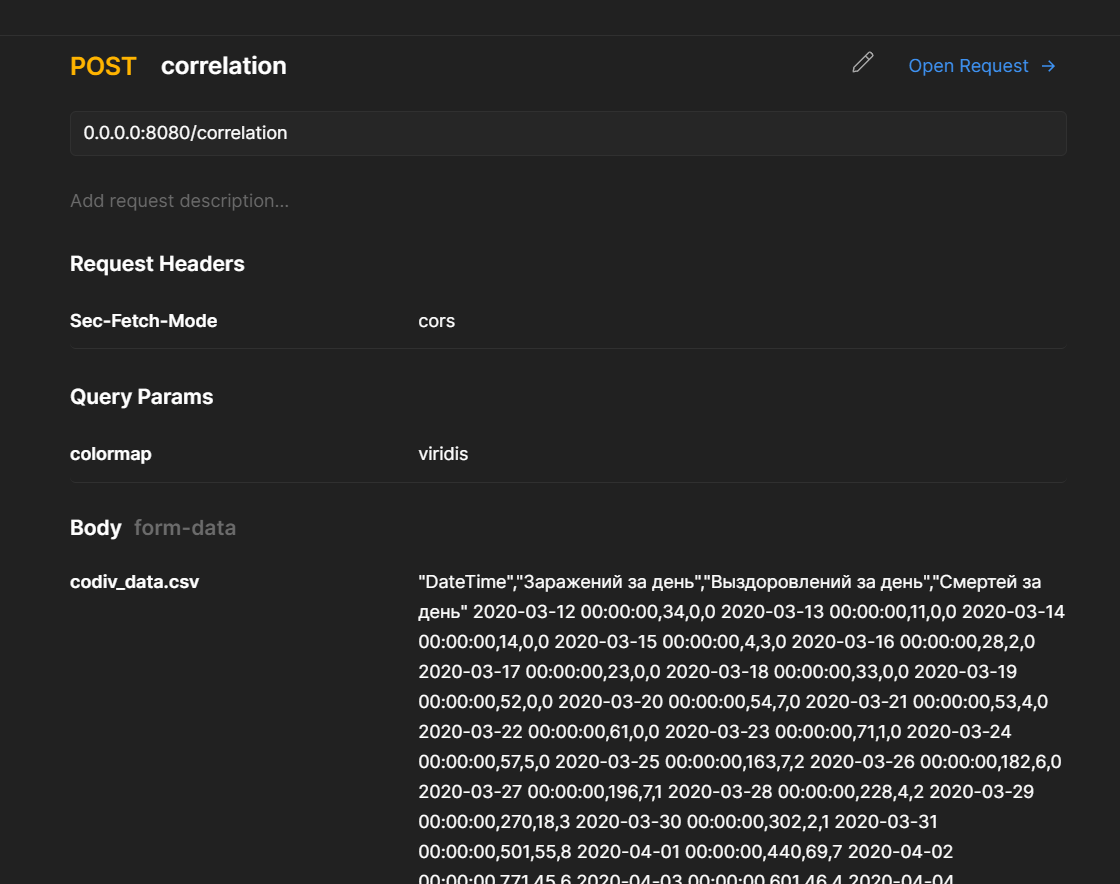
class Operation(Base):  
 \_\_tablename\_\_ = "operation"  
 id = sa.Column(sa.Integer, primary\_key=True)  
 user\_id = sa.Column(sa.Integer, sa.ForeignKey(user\_table.c.id))  
 task\_id = sa.Column(sa.Integer, sa.ForeignKey(task\_table.c.id))  
 input = sa.Column(sa.String(255))  
 input\_name = sa.Column(sa.String(255))  
 result = sa.Column(sa.String(255))  
 result\_name = sa.Column(sa.String(255))  
  
operation\_table = Operation.\_\_table\_\_

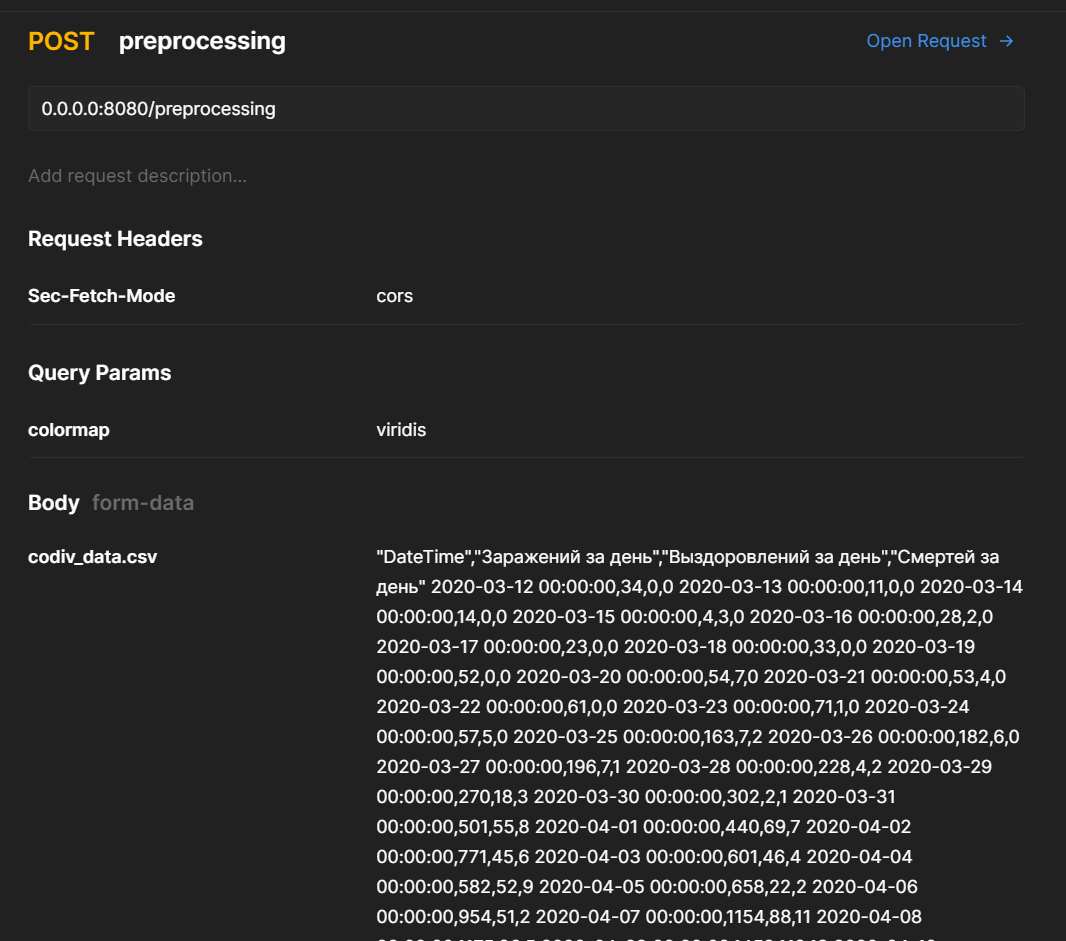
# REST API

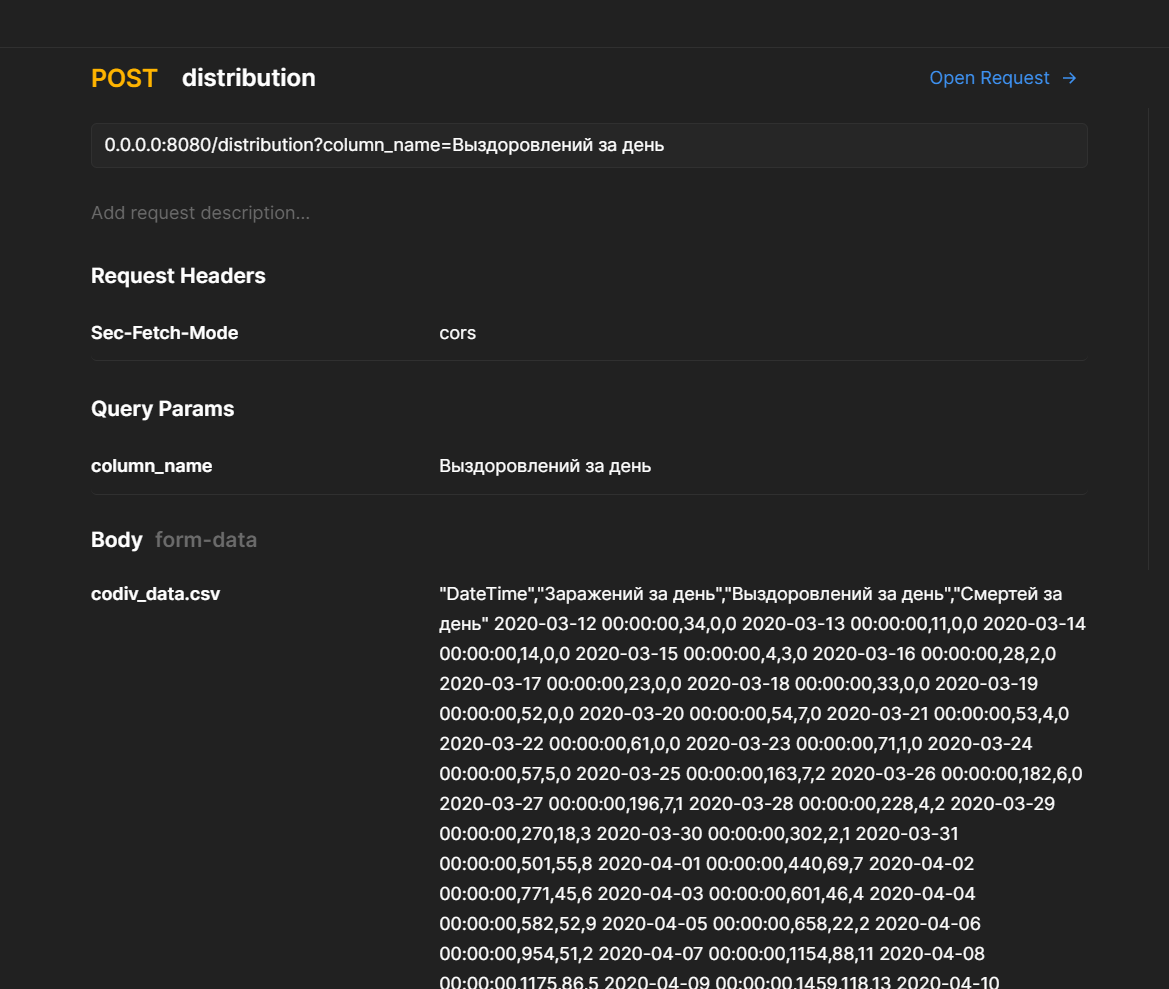
1. Изучить структуру формата представления данных JSON.
2. Изучить типы запросов к API: HEAD, GET, POST, PUT, DELETE.
3. Спроектировать и реализовать собственное REST API (Получение, создание, изменение и удаление каких-либо объектов).
4. В отчёт необходимо предоставить документацию к использованию методов. (Либо словесным описание, либо через Swagger)

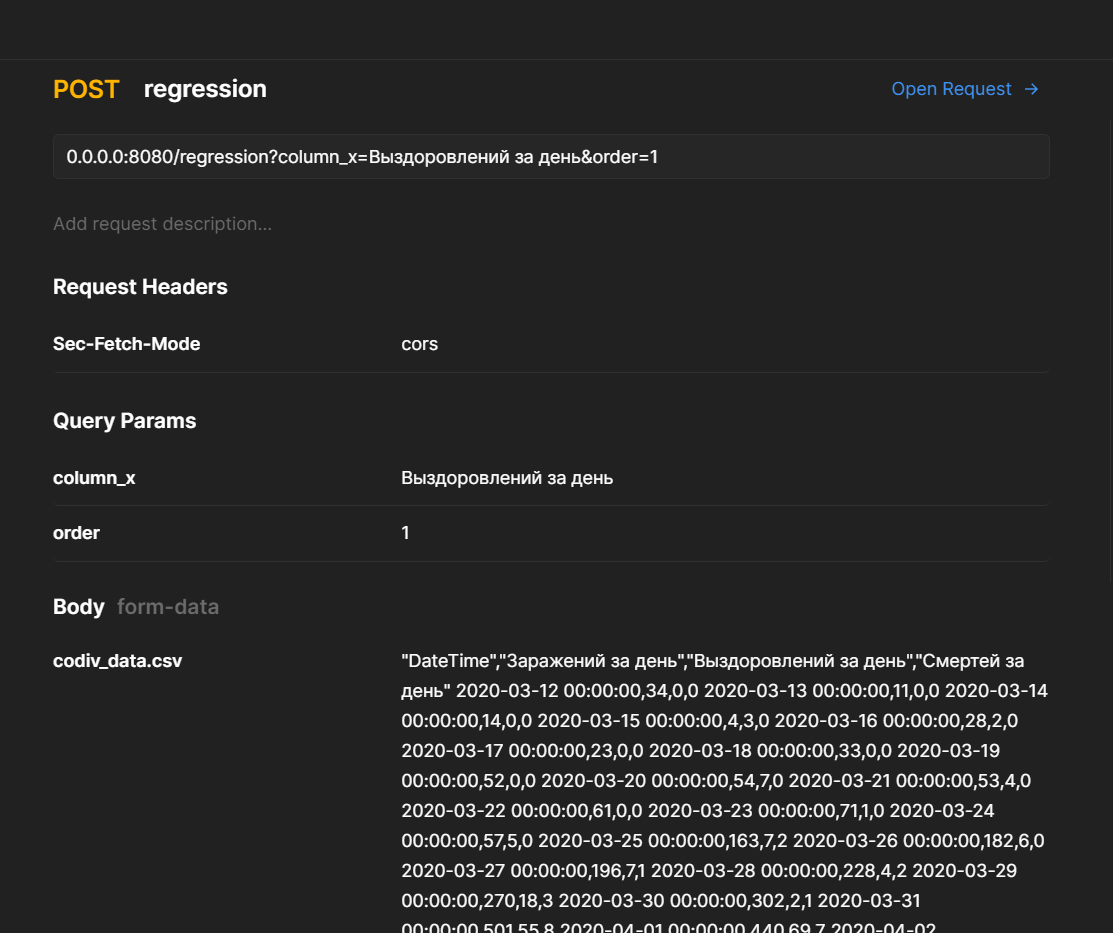
## Схема API (в Postman)

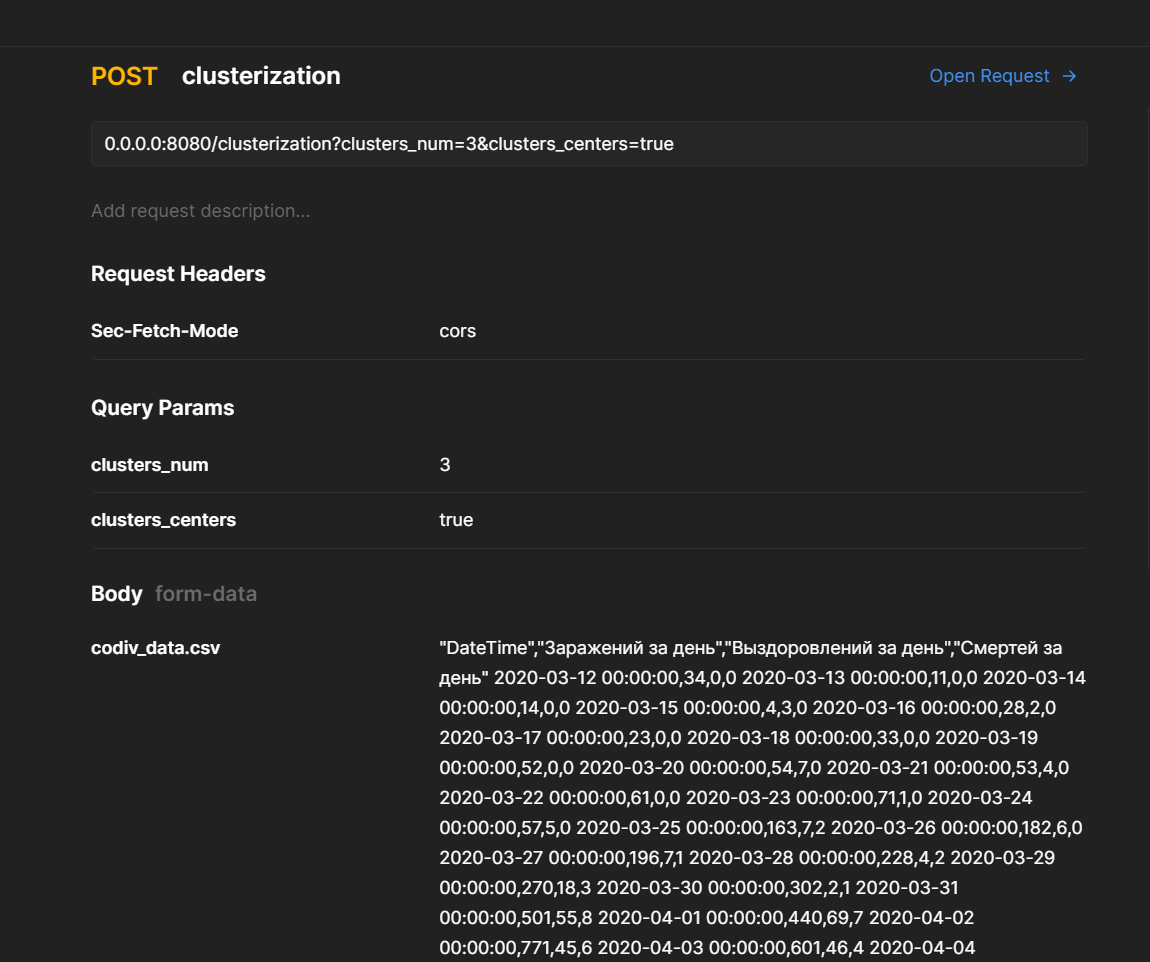










# Работа с HTTP запросами

1. Изучить возможности js для отправки http запросов.
2. Выбрать подходящую библиотеку для работы с запросами.
3. Реализовать взаимодействие фронтенда с REST API, спроектированном в прошлой лабораторной работе.
4. Продемонстрировать работу взаимодействия фронтенд приложения с REST API.

## Код запроса

import Axios, {AxiosError} from 'axios';

const authenticationBasePath:string = 'http://localhost:8080/authentication';

export function getAuth(resultHandler: (data: any)=>void, errorHandler: (data: any)=>void, login: string, password: string){

    Axios.get(authenticationBasePath,

        {

            params: { login:login, password:password},

            responseType: 'json'

        }

    ).then

    (response => {

        console.log(response);

        resultHandler((oldData: Object) => ({ ...oldData, ...response.data }));

    })

    .catch((error: AxiosError) => {

        errorHandler(error.message);

    });

}

export function postAuth(resultHandler: (data: any)=>void, errorHandler: (data: any)=>void, login: string, password: string){

    Axios.post(authenticationBasePath,

        null,

        {

            params: { login:login, password:password },

            responseType: 'json'

        }

    ).then

    (response => {

        resultHandler((oldData: Object) => ({ ...oldData, ...response.data }));

    })

    .catch((error: AxiosError) => {

        errorHandler(error.message);

    });

}

## Результат работы запроса

