

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №** 4

з дисципліни “Основи веб-програмування”

|  |  |  |
| --- | --- | --- |
| Виконав  студент II курсу  групи КП-93  Катрич Владислав Сергійович  (*прізвище, ім’я, по батькові*)  варіант №25 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

Київ 2019

**Мета роботи**

Вивчити основні принципи асинхронного програмування в JavaScript.

Навчитись асинхронно взаємодіяти з базою даних.

Налаштувати взаємодію з віддаленою базою даних та сховищем медіа.

Підготувати і опублікувати веб-сервіс в мережі Інтернет.

**Постановка завдання**

**База даних і асинхронне програмування:**

1. Змоделювати за допомогою GUI клієнта бази даних об'єкти сутностей із

попередніх лабораторних робіт та додати нову композиційну сутність

( *{composite}* ) відповідно до варіанту **(див. Додаток ""Структура бази даних"")**.

Називати таблиці\колекції сутностей у множині англійською мовою.

Всі сутності мають містити унікальні ключі-ідентифікатори.

Сутності користувачів мають мати зв'язок із основною або композиційною суністю (залежить від сутностей).

2. Переписати модуль сховища даних для взаємодії з базою даних. Змінити

інтерфейс сховища для використання асинхронних функцій. Весь доступ до бази даних має бути асинхронним.

3. Композиційна сутність

i. Створити нову модель {Composite} у модулі models/{composite}.js для композитних сутностей (за варіантом, замість {Composite} використовувати назву сутності англійською мовою). Реалізувати асинхронні CRUD операції цих сутностей для взаємодії із базою даних.

ii. Додати відповідні веб-сторінки (або Swagger документацію), що дозволяють керувати композиційними сутностями.

4. Видалити з директорії /data всі JSON файли з даними, які тепер будуть

розміщені у БД.

5. Створити у проекті файл конфігурації ( config.js ) і внести у нього шлях

підключення до бази даних. Отримувати шлях зі змінних середовища. Для цього встановити модуль env та створити файл env , який обов'язково додати у .gitignore .

**Віддалена база даних:**

2. Створити Heroku Application для вашого веб-сайту.

3. Створити віддалений екземпляр бази даних (наприклад, на mLab).

4. Налаштувати ваш Heroku Application для доступу до віддаленої БД. Для цього в налаштуванні додатку додайте змінну середовища, що міститиме посилання на віддалену БД.

5. Додати в рішення цієї роботи посилання на створений веб-сайт на Heroku.

**Сховище медіа:**

1. Переписати модуль медіа сховища. Підключити пакет cloudinary та

завантажувати всі файли, що прийшли у запиті з форми туди (див. Додатки). API Cloudinary буде віддавати URL на завантажений файл, який і зберігати у полях сутностей.

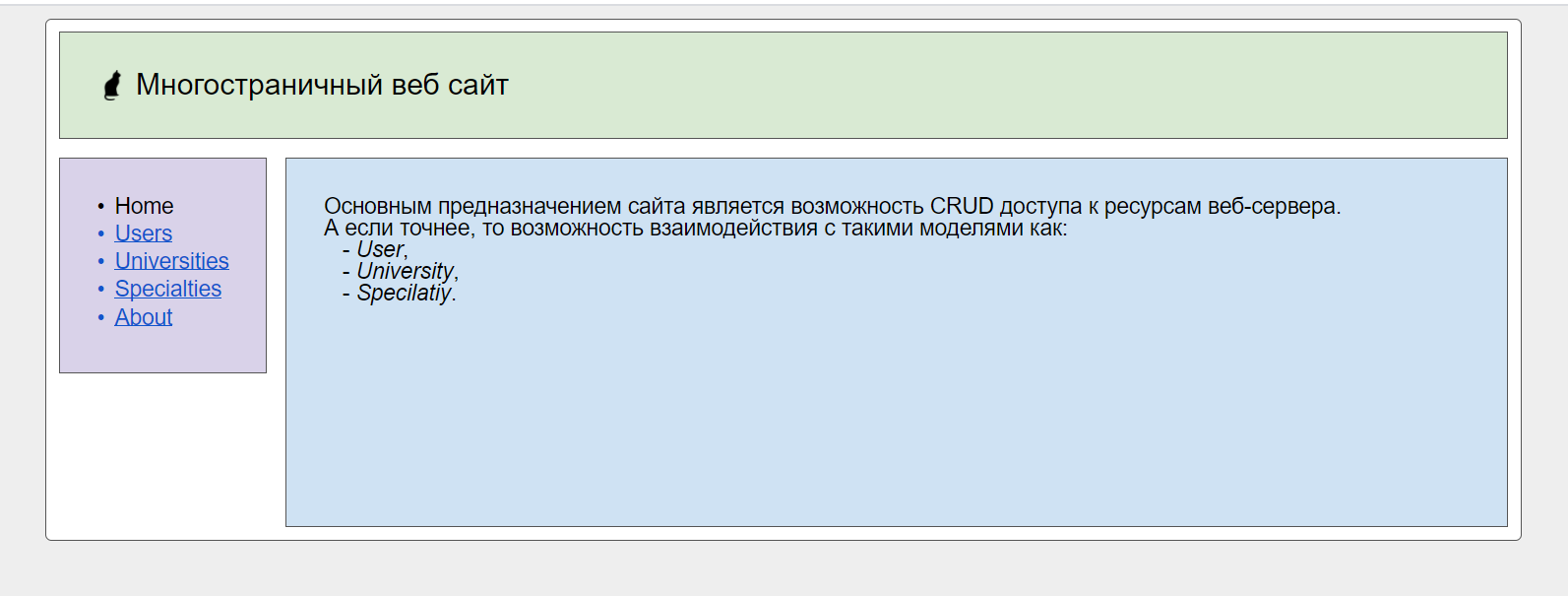
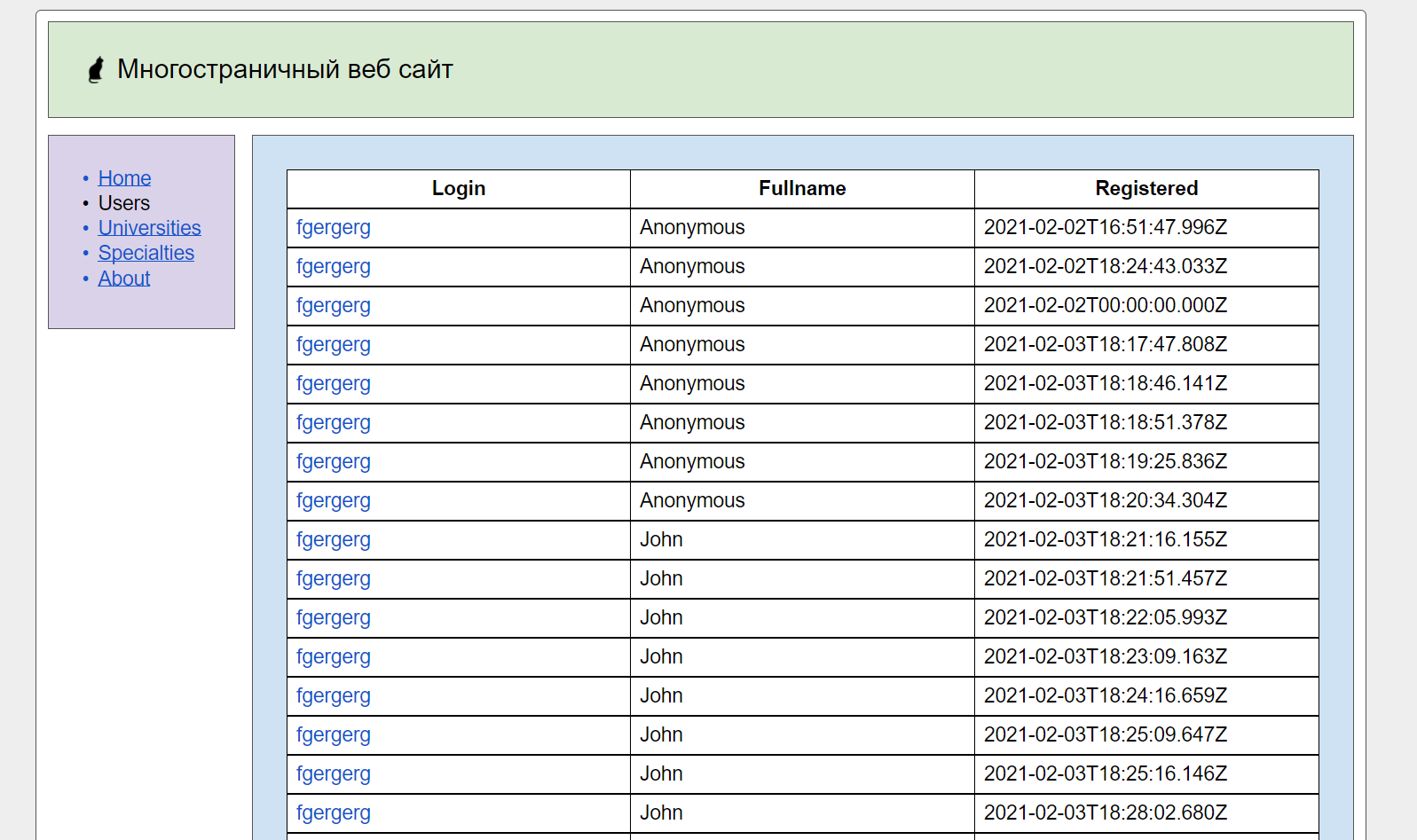
2. Налаштувати ваш Heroku Application для доступу до віддаленого сховища медіа. Для цього в налаштуванні додатку додайте змінну середовища, що міститиме посилання на сховище.

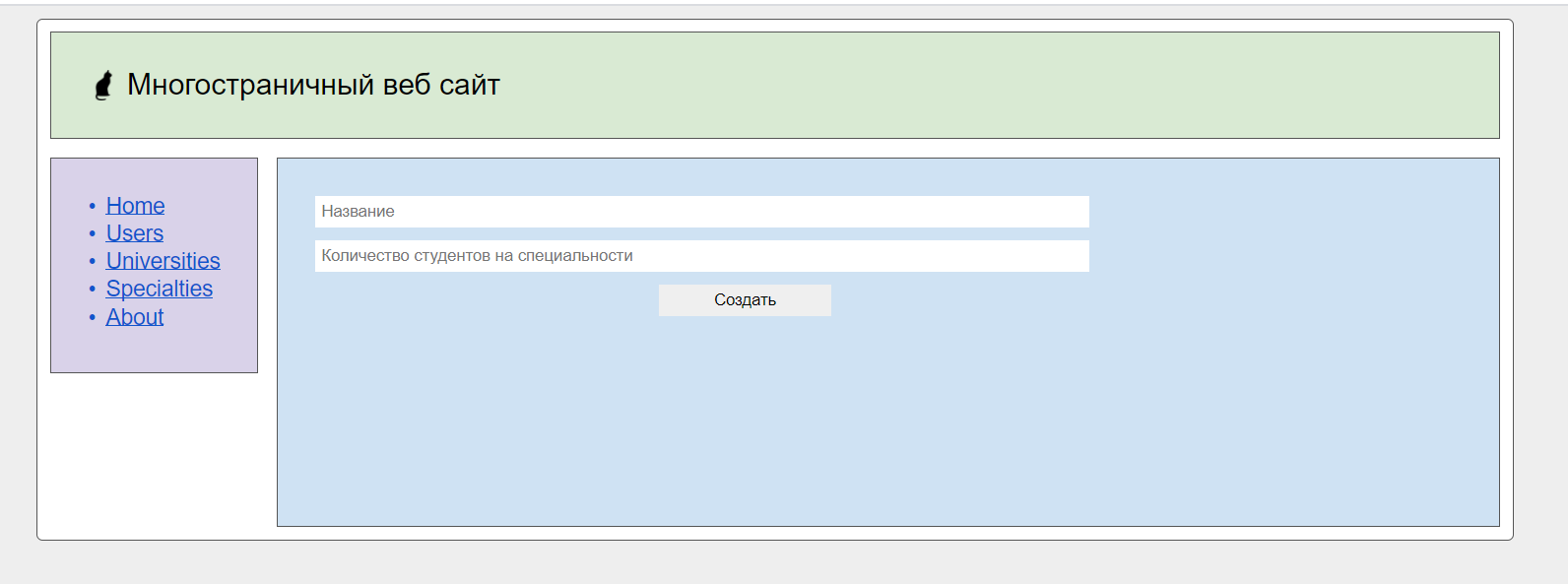
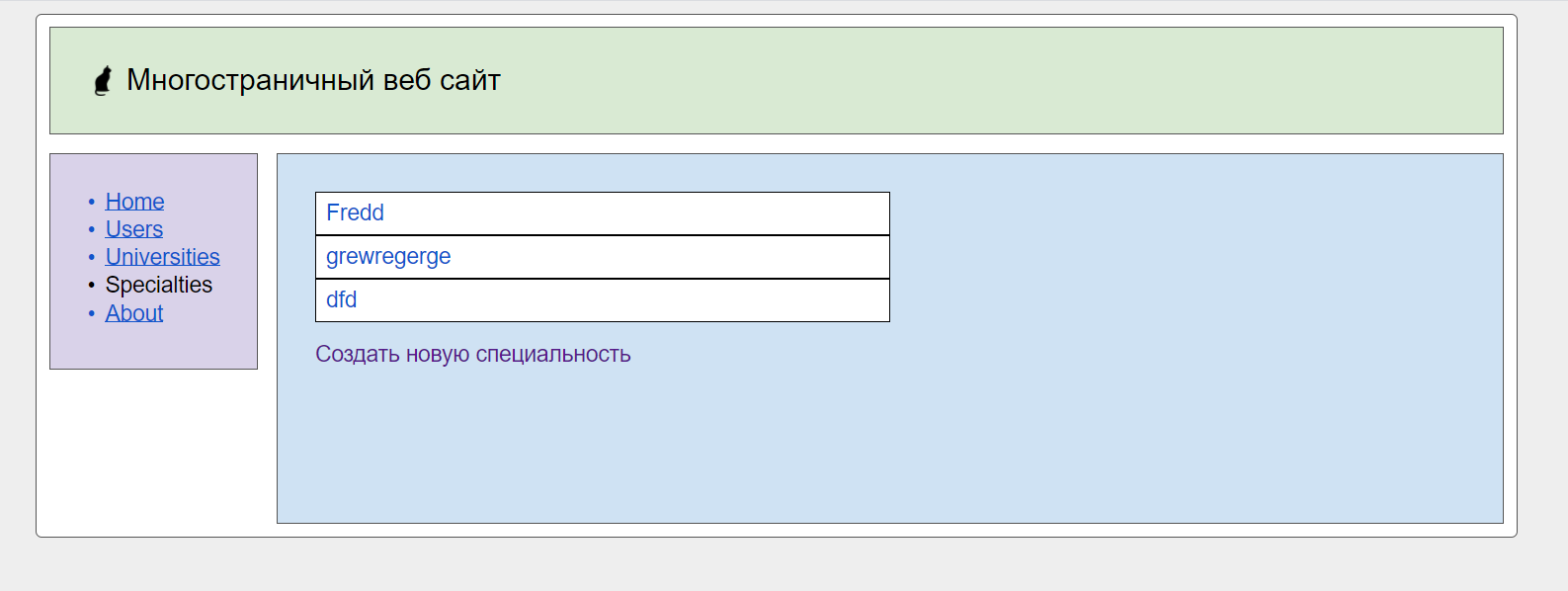
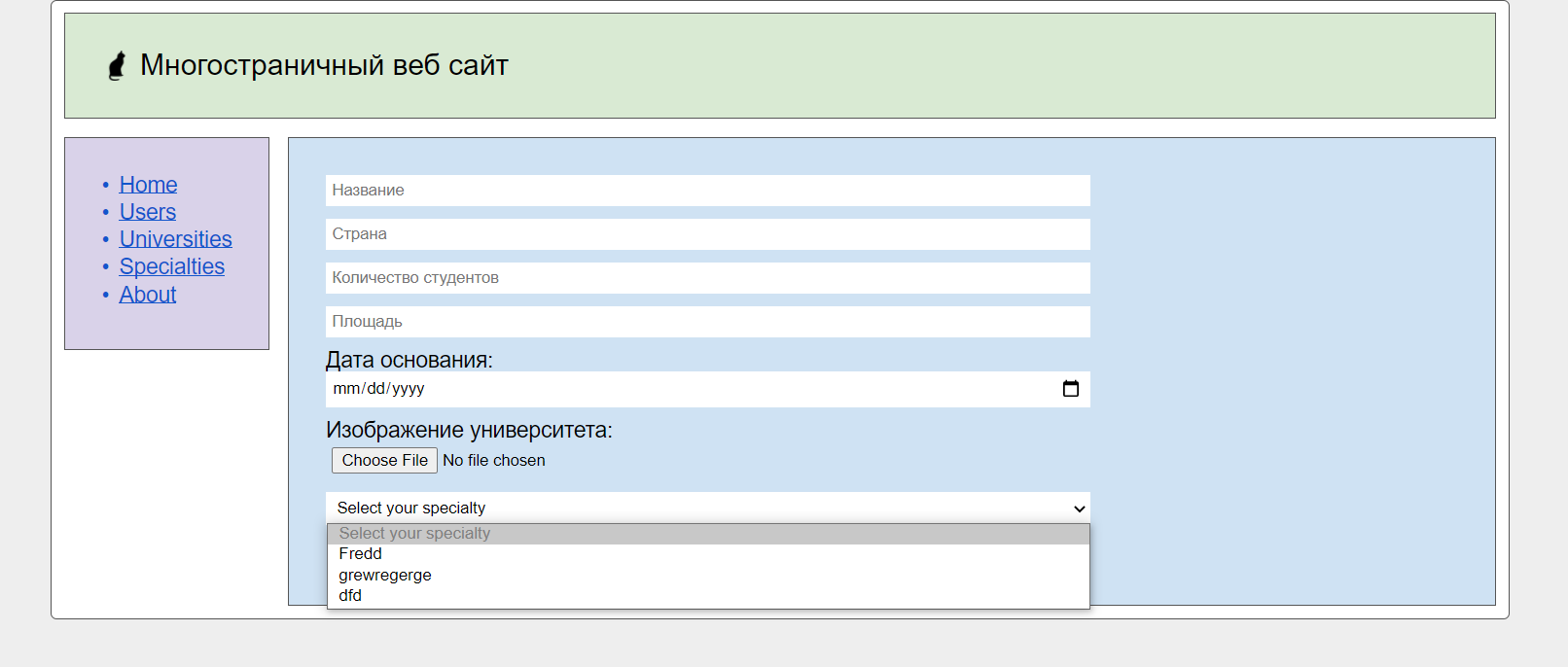
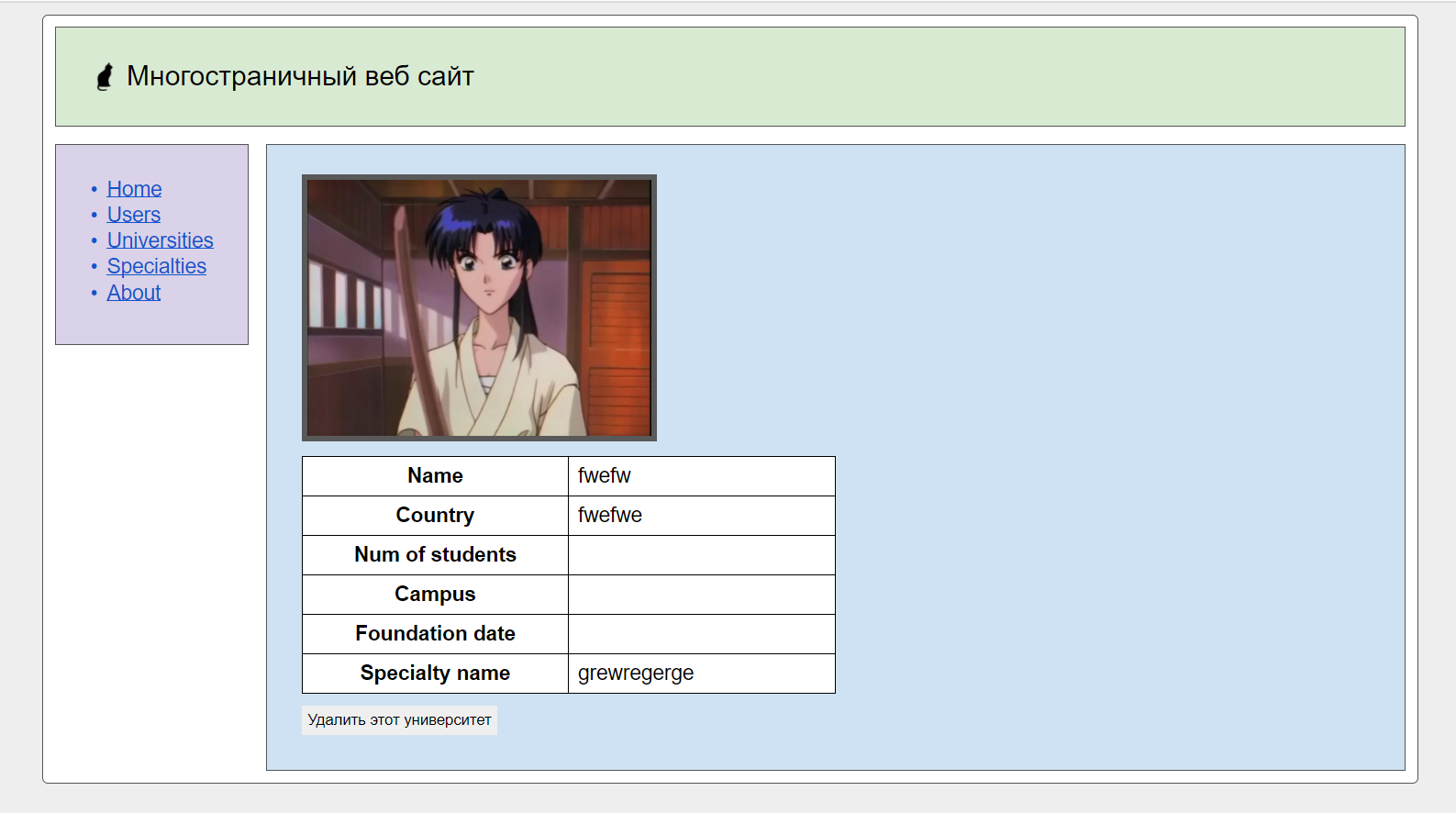
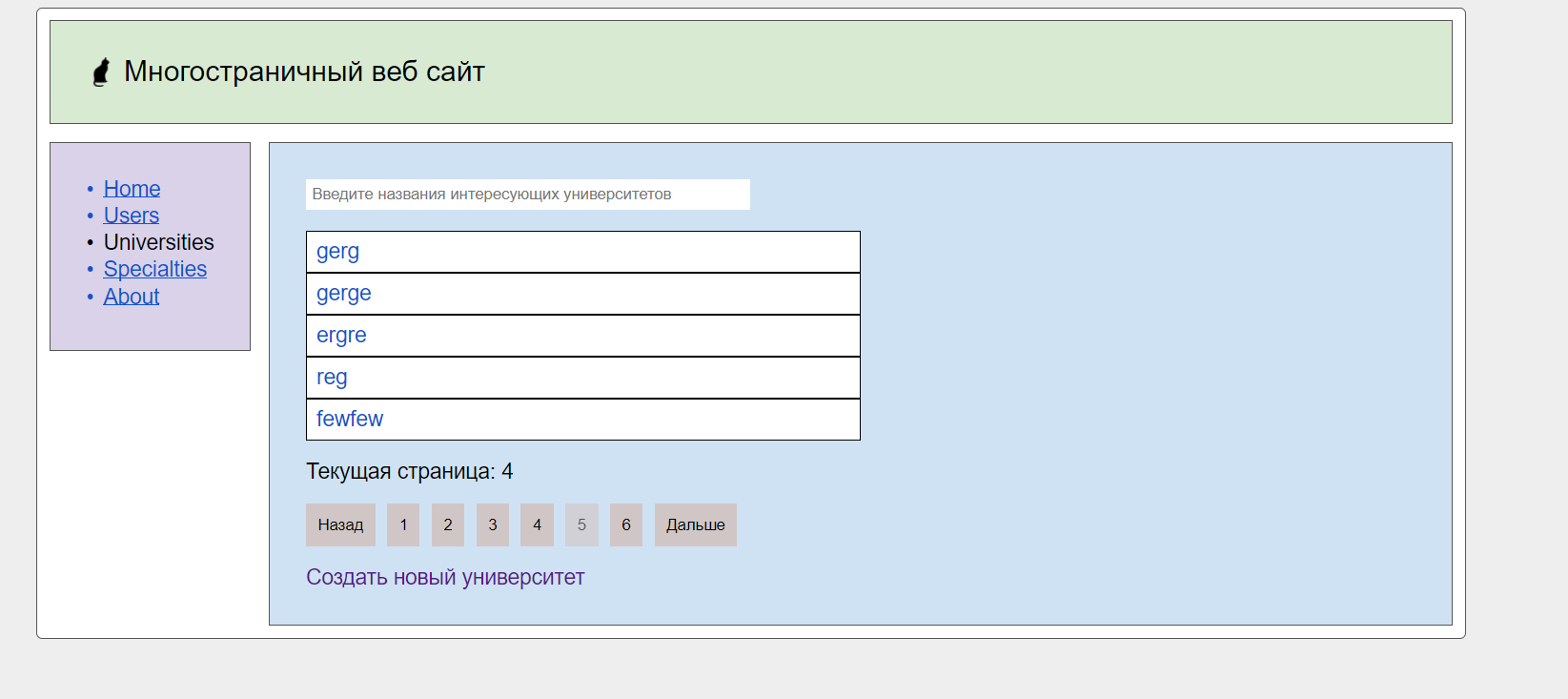
3. Видалити з проекту директорію, у які зберігались динамічні медіа

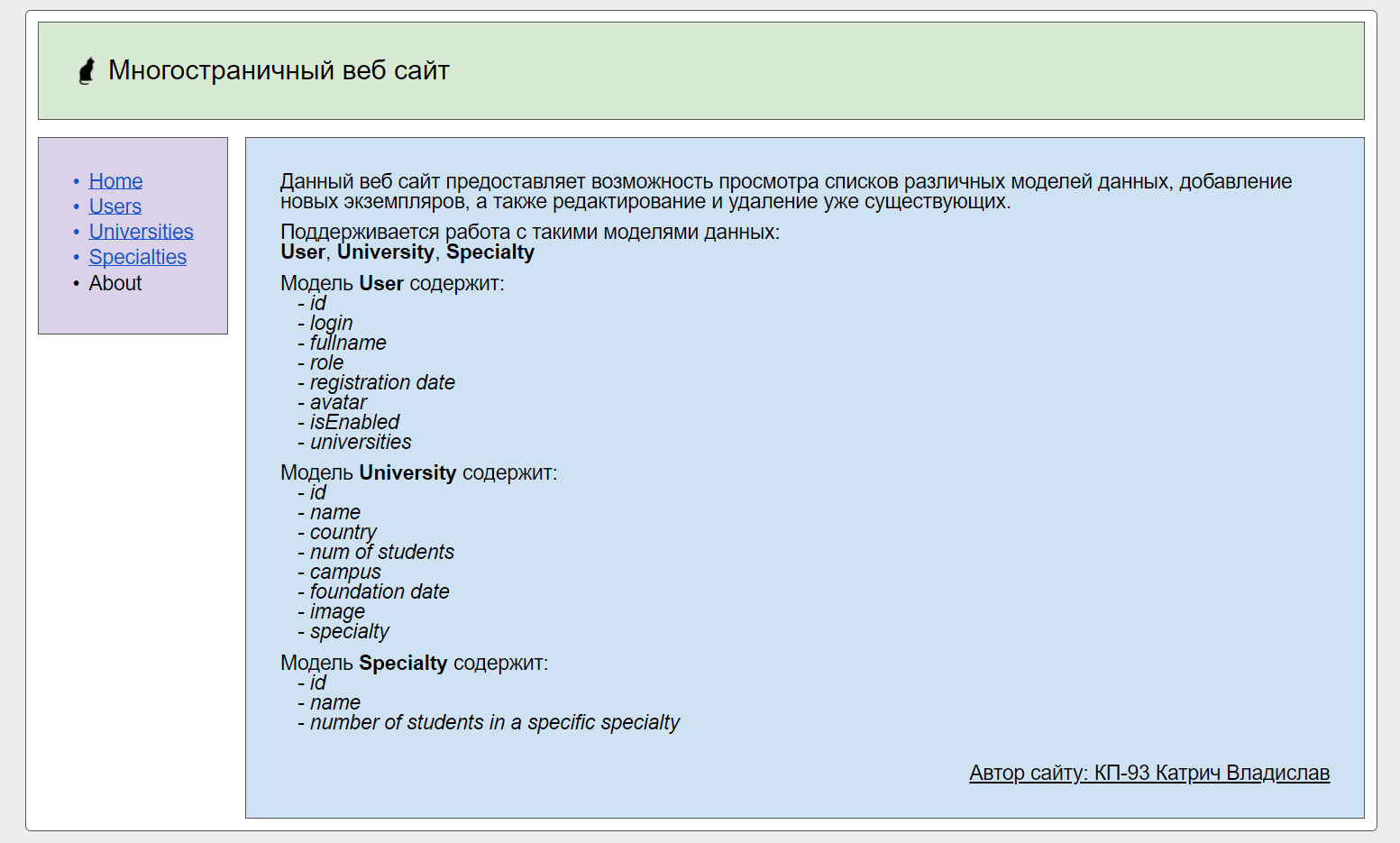
**Код розроблених модулів:**

|  |
| --- |
| **app** |
| const config = require('./config');  const express = require('express');  const fileUpload = require('express-fileupload');  const path = require('path');  const bodyParser = require('body-parser');  const exphbs = require('express-handlebars');  const mongoose = require('mongoose');  const indexRouter = require('./routes/index');  const usersRouter = require('./routes/users');  const universitiesRouter = require('./routes/universities');  const specialtiesRouter = require('./routes/specialties');  const aboutRouter = require('./routes/about');  const apiUsersRouter = require('./routes/apiUsers');  const apiUniversitiesRouter = require('./routes/apiUniversities');  const PORT = config.port;  const mongodbUri = config.mongodbUri;  const app = express();  const hbs = exphbs.create({      defaultLayout: 'main',      extname: 'hbs',      helpers: {          if\_eq: function (a, b, opts) {              if (a === b) return opts.fn(this);              else return opts.inverse(this);          },      },  });  app.engine('hbs', hbs.engine);  app.set('view engine', 'hbs');  app.use(      fileUpload({          createParentPath: true,      })  );  app.use(bodyParser.json());  app.use(bodyParser.urlencoded({ extended: true }));  app.use(express.static(path.join(\_\_dirname, 'public')));  app.use('/', indexRouter);  app.use('/users', usersRouter);  app.use('/universities', universitiesRouter);  app.use('/specialties', specialtiesRouter);  app.use('/about', aboutRouter);  app.use('/api/users', apiUsersRouter);  app.use('/api/universities', apiUniversitiesRouter);  async function start() {      try {          await mongoose.connect(mongodbUri, {              useNewUrlParser: true,              useFindAndModify: false,              useUnifiedTopology: true,          });          // const User = require('./models/user'); \          // const newUser = new User({          //  fullname: 'John',          //  login: 'fgergerg',          //  universities: ['geroigjeoi23'],          // });          // await newUser.save();          app.listen(PORT, () => {              console.log(`Server has been started on http://localhost:${PORT}`);          });      } catch (err) {          console.log('ERROR:', err);      }  }  start(); |

**Зображення всіх розроблених сторінок**





**Висновок**

Виконавши цю лабораторну роботу, ми вивчили основні принципи асинхронного програмування в JavaScript. Навчились асинхронно взаємодіяти з базою даних, налаштували взаємодію з віддаленою базою даних та сховищем даних. Підготували і опублікували веб-сервіс в мережі Інтернет.