Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

Работа выполнена в СПБ «DeCode»

СОГЛАСОВАНО

Начальник отдела ОНиПКРС

_____ Е.М. Димитриади
(подпись)

«06» 06 2024 г.

Декан _____

_____ И.А. Трещев
(подпись)

«06» 06 2024 г.

УТВЕРЖДАЮ

Проректор по научной работе

_____ А.В. Космынин
(подпись)

«07» 06 2024 г.

«Электронная Карта Выпускника»

Комплект *конструкторской / проектной* документации

Руководитель СПБ _____ 06.06.2024 *Е.Б. Абарникова*
(подпись, дата)

Руководитель проекта _____ 06.06.2024 *М.Д. Тимохов*
(подпись, дата)

Комсомольск-на-Амуре 2024

Карточка проекта

Название	*«Электронная Карта Выпускника»*
Тип проекта	Тип проекта: техническое творчество (инициативный)
Исполнители	Студент М.Д. Тимохов – 0ИСб-1 Студент С.А. Валеева – 0ИСб-1 Студент У.С. Михайлова – 2ИБ-1 Студент Ю.С. Цзин – 2ИБ-1
Срок реализации	Февраль 2024 – май 2024

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

ЗАДАНИЕ
на разработку

Название проекта: Разработка веб-сервиса «Карта выпускника»

Назначение: проект направлен на повышение имиджа университета путем внедрения веб-сервиса «Карта выпускника». Данный сервис позволяет визуализировать в наглядной форме места работы выпускников университета

Область использования: проект направлен на создание веб сервиса, который будет внедрен в информационную среду университета. Данный веб-сервис будет содержать информацию о успешном трудоустройстве выпускников университет. Наличие такого веб-сервиса позволит привлечь внимание к университету со стороны будущих абитуриентов и нынешних студентов

Функциональное описание проекта:

- возможность просмотра информации о выпускниках через метки на карте
- возможность фильтрации данных
- возможность получения обратной связи и технической поддержки
- наличие отдельного сервиса для администрации
- возможность редактирования данных о выпускнике

Техническое описание устройства:

Требования:

Функциональные требования:

- сервис должен содержать интерактивную карту;
- сервис должен содержать возможность фильтрации;
- сервис должен содержать обратную связь;
- сервис должен предоставлять информацию о текучем месте работы выпускника;

Нефункциональные требования:

- сервис должен быть легким в использовании и понимании для потенциальных пользователей;
- сервис должен быть доступен на разных устройствах (ПК, планшеты, мобильные устройства);
- сервис должен соответствовать брендбуку университета

План работ:

Наименование работ	Срок
Исследование требований	02.02-03.02
Создание прототипа	04.02-04.03
Разработка клиентской части	05.03-05.04
Разработка серверной части	06.04-06.05
Тестирование и отладка	07.05-10.05
Публикация	11.05

Комментарии:
Разработка проекта велась с использованием agile-методологий, у каждого члена команды была своя задача, благодаря этому выполнение некоторых этапов велось параллельно

Перечень графического материала:

1. Принципиальная схема;

2. Чертежи изделия (или трехмерные модели изделия);

3. Внешний вид изделия;

4. Блок-схема алгоритмов (при наличии управляющих программ);

Отсутствует

Руководитель проекта 10.02.2024 М.Д. Тимохов

(подпись, дата)

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

ПАСПОРТ

Разработка веб-сервиса «Карта выпускника»

Руководитель проекта 06.06.2024 *М.Д. Тимохов*
(подпись, дата)

Комсомольск-на-Амуре 2024

Содержание

1 Общие положения

Настоящий паспорт является документом, предназначенным для ознакомления с основными техническими характеристиками, устройством, правилами установки и эксплуатации проекта «Карта выпускника» (далее «сервис»).

1.1 Наименование изделия

Полное наименование изделия – «Электронная Карта выпускника».

1.2 Наименования документов, на основании которых ведется проектирование изделия

Проектирвание «Карты выпускника» осуществляется на основании требований и положений следующих документов:

- задание на разработку.

1.3 Перечень организаций, участвующих в разработке изделия

Заказчиком проекта «Карта выпускника» является Федеральное государственное бюджетное образовательное учреждение высшего образования «Комсомольский-на-Амуре государственный университет» (далее заказчик), находящийся по адресу: 681013, Хабаровский край, г. Комсомольск-на-Амуре, Ленина пр-кт., д. 17.

Исполнителями проекта «Карты выпускника» являются Конструкторы студенческого конструкторского студенческого проектного бюро «DeCode» (далее СПБ «DeCode»), студенты группы 0ИСб-1 М.Д. Тимохов, С.А. Валеева и 2ИБ-1 Ю.С. Цзин, У.С. Михайлова.

1.4 Сведения об использованных при проектировании нормативно-технических документах

При проектировании использованы следующие нормативно-технические документы:

ГОСТ 2.001-2013. Единая система конструкторской документации. Общие положения.

ГОСТ 2.102-2013. Единая система конструкторской документации. Виды и комплектность конструкторских документов.

ГОСТ 2.105-95. Единая система конструкторской документации. Общие требования к текстовым документам.

ГОСТ 2.610-2006. Единая система конструкторской документации. Правила выполнения эксплуатационных документов.

ГОСТ 2.004-88. Единая система конструкторской документации. Общие требования к выполнению конструкторских технологических документов на печатающих и графических устройствах вывода ЭВМ.

ГОСТ 2.051-2006. Единая система конструкторской документации. Электронные документы. Общие положения.

ГОСТ 2.052-2006. Единая система конструкторской документации. Электронная модель изделия. Общие положения.

ГОСТ 2.601-2013. Единая система конструкторской документации. Эксплуатационные документы.

2 Техническое задание

2.1 Наименование проекта

Разработка веб-сервиса «Карта выпускника».

2.2 Цель проекта

Целью проекта является создание карты выпускника, которая в дальнейшем поможет привлечь внимание и увеличит количество поступающих абитуриетов на 15%.

2.3 Требования к функциональности

Сервис должен содержать следующую информацию : фио, год окончания, какой университет, какой факультет закончил, в какой группе обучался, где работает, в каком городе работает, должность и почту о выпускниках университета, которые на нем отмечены, а также предоставлять данные при нажатии на соответсвующий маркер на карте.

Цветовая гамма и шрифт текста должны соответсвовать фирменному стилю Университета.

2.4 Требования к дизайну

Дизайн сервиса должен быть современным и привлекательным для целевой аудитории. Для этого необходимо использовать сочетание цветовых решений, текстовых блоков и графических элементов.

Также цветовая гамма должна соответствовать фирменному стилю Университета, а использование шрифтов должно соответствовать корпоративному стилю.

2.5 Технические требования

Сервис должен быть разработан с использованием современного стека технологий MEVN. Должна быть поддержка с другими системами. Для удоб-

ства пользователей сервис должен поддерживать отображение на мобильных устройствах и иметь адаптивный дизайн.

2.6 Срок выполнения проекта

Проект должен быть завершен 11.05.2024.

2.7 Требования к аппаратному обеспечению

Для просмотра сервиса необходимо использовать персональный компьютер или мобильное устройство с доступом в Интернет и следующими характеристиками:

- процессор Intel Core i3 или аналогичный;
- оперативная память не менее 4 Гб;
- разрешение экрана не менее 1280 x 720 пикселей.

2.8 Требования к программному обеспечению

Для просмотра сервиса необходимо использовать веб-браузер, поддерживающий HTML3 и CSS3. Рекомендуется использовать один из следующих браузеров:

- Google Chrome версии 80 или выше;
- Mozilla Firefox версии 75 или выше;
- Apple Safari версии 13 или выше;
- Microsoft Edge версии 80 или выше.

3 Руководство разработчика

В качестве инструмента разработки использовался стек MERN, включающий в себя Mongo DB в качестве базы данных, Express.js в качестве сервера для приложения, React.js в качестве библиотеки для клиентской части приложения, Node.js в качестве базы самого приложения. Для визуализации карты использовался сервис 2ГИС MapGL API.

Основные схемы хранящихся данных показана в листингех 3.1 – 3.3

Листинг 3.1 – Схема описыввающая профиль студента

```
const { model, Schema } = require('mongoose');
var schema = new Schema({
    userId: {
        type: String,
    },
    FIO: {
        type: String,
        default: '',
    },
    yearFinish: {
        type: Number,
        default: 2020,
    },
    finishUniversyty: {
        type: String,
        default: '',
    },
    facult: {
        type: String,
        default: '',
    },
    nameGroup: {
        type: String,
        default: '',
    },
    jobTown: {
        type: String,
        default: '',
    },
    jobPlace: {
        type: String,
        default: '',
    },
    jobTitle: {
        type: String,
        default: '',
```

```
    },
    email: {
        type: String,
        default: '',
    },
    photoUrl: {
        type: String,
        default: '',
    },
    isActive: {
        type: Boolean,
        default: true,
    },
    coordinates: {
        x: {
            type: Number,
            default: 0.0,
        },
        y: {
            type: Number,
            default: 0.0,
        },
    },
    dateCreate: {
        type: Date,
        default: new Date(),
    },
    isApplication: {
        type: Boolean,
        default: false,
    },
});
module.exports = model('Profile', schema);
```

Листинг 3.2 – Схема описывающая авторизацию пользователя

```
const { Schema, model } = require('mongoose');

const User = new Schema({
    username: {
        type: String,
        unique: true,
        required: true,
    },
    name: {
        type: String,
        required: true,
        default: 'User',
    },
    password: {
        type: String,
```

```
        required: true,
    },
    roles: [
        {
            type: String,
            ref: 'Role',
        },
    ],
    registrDate: {
        type: Date,
        default: new Date(),
        required: true,
    },
});

module.exports = model('usersauth', User);
```

Листинг 3.3 – Схема, описывающая профиль учебного заведения

```
const { model, Schema } = require('mongoose');
var schema = new Schema({
    nameOrganization: {
        type: String,
        default: '',
    },
    coutGraduates: {
        type: Number,
        default: '',
    },
    countFacult: {
        type: Number,
        default: '',
    },
    townOrganization: {
        type: String,
        default: '',
    },
    linkOrganization: {
        type: String,
        default: '',
    },
    photoUrl: {
        type: String,
        default: '',
    },
    mapId: {
        type: String,
    },
    coordinates: {
        x: {
            type: Number,
```

```
            default: 0,
        },
        y: {
            type: Number,
            default: 0,
        },
    },
});
module.exports = model('MapProfile', schema);
```

Для каждой схемы данных созданы методы поддерживающие CRUD (Создание, чтение, обновление и удаление), данные операции показана для схемы профиля, показаны в листинге 3.4.

Листинг 3.4 – CRUD методы для схемы профиля

```
const { Profile } = require('../model');

const boom = require('boom');
const multer = require('multer');
const path = require('path');
const fs = require('fs');

module.exports = {
    /**
     * Получение одной записи по id
     * @param {*} req
     * @param {*} res
     * @returns
     */
    async getByUserId(req, res) {
        try {
            const userId = req.params.id;

             const item = await Profile.findOne({ userId: userId
});
            return res.status(200).send(item);
        } catch (err) {
              return res.status(400).send({ status: false, err:
boom.boomify(err) });
        }
    },
    /**
     * Получение всех записей
     * @param {*} _
     * @param {*} res
     * @returns
     */
```

```
    async getAll(_, res) {
        try {
            const items = await Profile.find();
            return res.status(200).send(items);
        } catch (err) {
                return res.status(400).send({ status: false, err:
boom.boomify(err) });
        }
    },

    /**
     * Добавление записи
     * @param {*} req
     * @param {*} res
     * @returns
     */
    async create(req, res) {
        try {
            const item = new Profile(req.body);
            const newItem = await item.save();
            return res.status(200).send(newItem);
        } catch (err) {
                return res.status(400).send({ status: false, err:
boom.boomify(err) });
        }
    },
    /**
     * Обновление записи
     * @param {*} param0
     * @param {*} res
     * @returns
     */
    async update(req, res) {
        try {
            const userId = req.params.id;
            const newData = req.body;

            // Найти текущий профиль пользователя
                const currentProfile = await Profile.findOne({
userId: userId });

            // Проверить, были ли внесены изменения в данные
            let dataChanged = false;
            for (const key in newData) {
                if (
                    newData.hasOwnProperty(key) &&
                    currentProfile[key] !== newData[key]
                ) {
                    dataChanged = true;
                    break;
                }
            }
```

```
            if (!dataChanged) {
                 // Если данные не изменились, возвращаем текущие
данные профиля без выполнения обновления
                return res
                    .status(201)
                        .send({ message: 'Данные не изменились',
profile: currentProfile });
            }

            // Если данные изменились, обновляем профиль
              await Profile.findOneAndUpdate({ userId: userId },
newData);

             const item = await Profile.findOne({ userId: userId
});

            return res.status(200).send(item);
        } catch (err) {
              return res.status(400).send({ status: false, err:
boom.boomify(err) });
        }
    },

    /**
     * Удаление записи
     * @param {*} param0
     * @param {*} res
     * @returns
     */
    async delete({ params: { id } }, res) {
        try {
            await Profile.findByIdAndDelete(id);
             return res.status(200).send({ status: 'ok', massage:
'Удалено' });
        } catch (err) {
              return res.status(400).send({ status: false, err:
boom.boomify(err) });
        }
    },

}
```

Одним из важных элементов сервиса является регистрации и авторизация, код реализующий эти возможности показан в листинге 3.5.

Листинг 3.5 – Авторизация и регистрация

```
const User = require('../model/User');
```

```
const Roles = require('../model/Roles');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const { validationResult } = require('express-validator');
const { secret } = require('../../config');
const boom = require('boom');

const generateAccesToken = (id, username) => {
    const payload = {
        id,
        username,
    };
    return jwt.sign(payload, secret, {
        expiresIn: '24h',
    });
};

class authController {
    async registr(req, res) {
        try {
            // const errors = validationResult(req);
            // if (!errors.isEmpty()) {
            //  return res.status(400).json({
            //      message: 'Ошибка при регистрации',
            //      errors,
            //  });
            // }
            const name = req.body.name ?? 'User';
            const { username, password } = req.body;

              // Проверка username с использованием регулярного
выражения
            const usernameRegex =
                        /^([\w]+\.?)+(?<!\.)@(?!\.)[a-za-я0-9ё\.-
]+\.?[a-za-яё]{2,}$/iu;
            if (!usernameRegex.test(username)) {
                return res.status(400).json({
                    message: 'Некорректный формат логина',
                });
            }

            if (password.length < 5) {
                return res.status(400).json({
                     message: 'Длинна пароля должна быть минимум
5 символов',
                });
            }

               //TODO переписать регулярку, чтобы проверяла на
длинну и повтор символов
             // Проверка пароля с использованием регулярного вы-
ражения
```

```
            const passwordRegex = /^[a-zA-Z._0-9]+$/;
            if (!passwordRegex.test(password)) {
                return res.status(400).json({
                    message:
                         'Пароль должен содержать только буквы a-
z, A-Z, цифры 0-9, точку (.) и символ подчеркивания (_)',
                });
            }

            const candidate = await User.findOne({
                username,
            });
            if (candidate) {
                return res.status(400).json({
                     message: `Пользователь с логином ${username}
уже существует`,
                });
            }

            const userRole = await Roles.findOne({
                value: 'USER',
            });

            const hashPassword = bcrypt.hashSync(password, 7);
            const user = new User({
                username,
                name,
                password: hashPassword,
                roles: [userRole.value],
            });
            await user.save();

            return res.status(200).send(user);
        } catch (error) {
              return res.status(400).send({ status: false, err:
boom.boomify(error) });
        }
    }
async login(req, res) {
        try {
            const { username, password } = req.body;

              // Проверка username с использованием регулярного
выражения
            const usernameRegex =
                         /^([\w]+\.?)+(?<!\.)@(?!\.)[a-za-я0-9ё\.-
]+\.?[a-za-яё]{2,}$/iu;
            if (!usernameRegex.test(username)) {
                return res.status(400).json({
                    message: 'Некорректный формат логина',
                });
            }
```

```
            // Проверка пароля с использованием регулярного выражения
            const passwordRegex = /^[a-zA-Z._0-9]+$/;
            if (!passwordRegex.test(password)) {
                return res.status(400).json({
                    message:
                        'Пароль должен содержать только буквы a-z, A-Z, цифры 0-9, точку (.) и символ подчеркивания (_)',
                });
            }

            const user = await User.findOne({
                username,
            });

            if (!user) {
                return res.status(400).json({
                    message: `Пользователь с логином ${username} не найден`,
                });
            }
              const validPassword = bcrypt.compareSync(password, user.password);
            if (!validPassword) {
                return res.status(400).json({
                    message: `Пароль не правильный`,
                });
            }
            const name = user.name;
            const user_id = user._id;
            const roles = user.roles;
            const token = generateAccesToken(user_id, name);
            return res.status(200).send({
                token: token,
                name: name,
                roles: roles,
                id: user_id,
            });
        } catch (error) {
              return res.status(400).send({ status: false, err: boom.boomify(err) });
        }
    }
}
```

Карта и метки, которые выводятся на ней являются самой большой и важной частью проекта, код реализующий вывод карты, меток, модальных окон показан в листингах 3.6 – 3.8.

Листинг 3.6 – Код выводящий карту и метки

```
import React, { useEffect, useState } from 'react';

import { useLoaderData } from 'react-router-dom';

import { load } from '@2gis/mapgl';
import { Clusterer } from '@2gis/mapgl-clusterer';

import Cookies from 'js-cookie';

import { Modal, Card, Flex, Button, Input, Drawer } from 'antd';

import GraduateModal from './GraduateModal';
import FilterDrawer from './FilterDrawer';
import MapProfileModal from './MapProfileModal';

export default function MapComponents() {
    const { graduatesData, mapProfileData } = useLoaderData();

    const [graduateModal, setGraduateModal] = useState(false);
        const   [mapProfileModal,   setMapProfileModal]   =   us-
eState(false);

          const    [graduateData,    setGraduateData]    =    us-
eState(graduatesData);
    const [modalData, setModalData] = useState([]);
    const [filter, setFilter] = useState(false);

    const userId = null;

    useEffect(() => {
        let map;
        let markers = [];
        //Если нет то сохраняем в  куки id карты
        if (!Cookies.get('mapId'))
                                    Cookies.set('mapId',     win-
dow.location.pathname.slice(5, 29));

        for (let i = 0; i < graduateData.length; i++) {
            markers[i] = {
                        coordinates:  [graduateData[i].x,  gradu-
ateData[i].y],
                id: i,
            };
        }

        load().then((mapglAPI) => {
            //Определяем  карту
            map = new mapglAPI.Map('map-container', {
                center: [137.00334687256984, 50.546384863773291,
                zoom: 15,
```

```
                // key: '042b5b75-f847-4f2a-b695-b5f58adc9dfd',
                key: '10153539-2026-4a0c-b7a3-52ddb3fed411',
            });

            const mapProfileMarker = new mapglAPI.Marker(map, {
                coordinates: [
                    mapProfileData.coordinates.x,
                    mapProfileData.coordinates.y,
                ],
                                                                icon:
'https://docs.2gis.com/img/mapgl/marker.svg',
                id: 'profile',
            });

            mapProfileMarker.on('click', () => {
                setMapProfileModal(true);
            });

            //Определяем кластер
            const clusterer = new Clusterer(map, {
                clusterStyle: {
                                                                icon:
'https://docs.2gis.com/img/mapgl/cluster.svg',
                                                             hoverIcon:
'https://docs.2gis.com/img/mapgl/clusterHover.svg',
                    labelColor: '#ffffff',
                    labelFontSize: 16,
                },
                radius: 40,
            });

            clusterer.load(markers);

            clusterer.on('click', (event) => {
                if (event.target.type == 'marker') {
                                                          setModalDa-
ta(graduatesData[event.target.data.id]);
                    setGraduateModal(true);
                } else {
                    map.setCenter(event.lngLat);
                    map.setZoom(map.getZoom() + 0.9);

                     const clusterCenter = event.lngLat; // Центр
кластера

                    // Получаем все кластеры
                    const clusters = event.target.data;
                    console.log(clusters);

                     // Проверяем, есть ли в кластере меньше ме-
ток, чем в общем
                    if (clusters.length < markers.length) {
```

```
                        // Удаляем метки кластера из начального
массива меток
                                const clusterMarkers = clus-
ters.slice(1);
                        clusterMarkers.forEach((clusterMarker)
=> {
                         const index = markers.findIndex(
                               (marker) => marker.id === clus-
terMarker.id
                         );
                         if (index !== -1) {
                             markers.splice(index, 1);
                         }
                        });

                        console.log(clusterMarkers);

                          // Оставляем один маркер в центре кла-
стера
                        const centerMarker = clusters[0];

                          // Распределяем остальные метки в слу-
чайных позициях в пределах области
                         const areaSize = 0.009; // Измените это
значение по вашему усмотрению
                         clusterMarkers.forEach((clusterMarker)
=> {
                             const offsetX = (Math.random() -
0.5) * areaSize;
                             const offsetY = (Math.random() -
0.5) * areaSize;
                         const newMarker = {
                             coordinates: [
                                 clusterCenter[0] + offsetX,
                                 clusterCenter[1] + offsetY,
                             ],
                               id: clusterMarker.id, // сохра-
няем id
                         };
                         markers.push(newMarker);
                        });

                             // Возвращаем центральный маркер в
начальный массив
                        // markers.push(centerMarker);
                    }

                    // Обновляем кластер
                    clusterer.load(markers);
                }
            });
        });
```

```
        // Удаляем карту при размонтировании компонента
        return () => map && map.destroy();
    }, []);

    const showDrawer = () => {
        setFilter(true);
    };
    const onClose = () => {
        setFilter(false);
    };
    // const onChange = (e) => {
    //  setPlacement(e.target.value);
    // };

    return (
        <>
                <Flex style={{ marginInline: '128px' }} justi-
fy={'space-between'}>
                <Button
                    style={{
                        marginTop: '58px',
                        marginBottom: '24px',
                        width: '294px',
                        height: 40,
                         boxShadow: '4px 7px 11px 0 rgba(0, 0, 0,
0.25)',
                        // padding: '16px 52px',
                    }}
                    type="primary"
                    size="large"
                    onClick={showDrawer}>
                    Настроить фильтры
                </Button>
                <Input.Search
                    placeholder="Поиск"
                    // onSearch={onSearch}
                    style={{
                        marginTop: '56px',
                        marginBottom: '24px',
                    }}
                    variant="borderless"
                    className="seacrh-modify"
                    size="large"
                    enterButton
                />
            </Flex>
            <Card bodyStyle={{ padding: 0 }}>
                <div>
                    <div
                        id="map-container"
                        style={{
```

```
                                height: 'calc(100vh - 300px)',
                            }}></div>
                    </div>
                </Card>

                <Modal
                    open={graduateModal}
                    footer={null}
                                    onCancel={()   =>   setGraduateMod-
al(!graduateModal)}
                    centered
                    width={'824px'}>
                    <GraduateModal graduatesData={modalData} />
                </Modal>

                <Modal
                    open={mapProfileModal}
                    footer={null}
                                onCancel={()   =>   setMapProfileMod-
al(!mapProfileModal)}
                    centered
                    width={'909px'}>
                     <MapProfileModal mapProfileData={mapProfileData}
/>
                </Modal>

                <Drawer
                    width={634}
                    styles={{
                         header: { padding: '48px', fontSize: '24px'
},
                        body: { padding: 0 },
                        backgroundColor: '#F8FFF5',
                    }}
                    size="large"
                    title="Настроить фильтры"
                    onClose={onClose}
                    open={filter}>
                    <FilterDrawer />
                </Drawer>
            </>
        );
}
```

Листинг 3.7 – Код задающий вид модального окна

```
import React from 'react';

import { Typography, Flex, Image } from 'antd';
```

```
const styleLabel = {
    padding: '8px 16px 8px 0px',
    // marginRight: '16px',
    fontSize: '16px',
};

const styleInput = {
    padding: '8px 16px',
    borderBottom: '1px solid black',
    width: '447px',
    fontSize: '16px',
};

const styleWorkInput = {
    padding: '8px 16px',
    borderBottom: '1px solid black',
    width: '79%',
    fontSize: '16px',
};

export default function GraduateModal({ graduatesData }) {
    return (
        <>
              <Typography.Title level={4} style={{ marginBottom:
'16px' }}>
                Информация о выпускнике
            </Typography.Title>
            <Flex>
                <Image
                    width={170}
                    height={200}
                     src={'http://localhost:8080/profile/photos/'
+ graduatesData.photoUrl}
                     fallback="
AAANSUhEUgAAAMIAAADDCAYA-
AADQvc6UAAABRWlDQ1BJQ0MgUHJvZmlsZQAAKJFjYGASSSwoyGFhYGDIzSspCnJ3
UoiIjFJgf8LAwSDCIMogwMCcmFxc4BgQ4ANUwgCjUcG3awyMIPqyLsis7PPOq3Qd
DFcvjV3jOD1boQVTPQrgSkktTgbSf4A4LbmgqISBgTEFyFYuLykAsTuAbJEioKOA
7DkgdjqEvQHEToKwj4DVhAQ5A9k3gGyB5IxEoBmML4BsnSQk8XQkNtReEOBxcfXx
UQg1Mjc0dyHgXNJBSWpFCYh2zi+oLMpMzyhRcASGUqqCZ16yno6CkYGRAQMDKMwh
qj/fAIcloxgHQqxAjIHBEugw5sUIsSQpBobtQPdLciLEVJYzMPBHMDBsayhILEqE
O4DxG0txmrERhM29nYGBddr//5/DGRjYNRkY/l7////39v///y4Dmn+LgeHANwDr
kl1AuO+pmgAAADhlWElmTU0AKgAAAAgAAYdpAAQAAAABAAAAGgAAAAAAAqACAAQA
AAABAAAAwqADAAQAAAABAAAAw-
wAAAAD9b/HnAAAHlklEQVR4Ae3dP3PTWBSGcbGzM6GCKqlIBRV0dHRJFarQ0eUT8
LH4BnRU0NHR0UEFVdIlFRV7TzRksomPY8uykTk/zewQfKw/9znv4yvJynLv4uLiV
2dBoDiBf4qP3/ARuCRABEFAoBEgghggQAQZQKAnYEaQBAQaASKIAQJEkAEEegJmB
ElAoBEgghggQAQZQKAnYEaQBAQaASKIAQJEkAEEegJm-
BElAoBEgghggQAQZQKAnYEaQBAQaASKIAQJEkAEEegJm-
BElAoBEgghggQAQZQKAnYEaQBAQaASKIAQJEkAEEegJm-
BElAoBEgghggQAQZQKAnYEaQBAQaASKIAQJEkAEEegJm-
```

```
BElAoBEgghggQAQZQKAnYEaQBAQaASKIAQJEkAEEegJm-
BElAoBEgghggQAQZQKAnYEaQBAQaASKIAQJEkAEEegJm-
BElAoBEgghggQAQZQKAnYEaQBAQaASKIAQJEkAEEegJm-
BElAoBEgghggQAQZQKAnYEaQBAQaASKIAQJEkAEEegJm-
BElAoBEgghggQAQZQKAnYEaQBAQaASKIAQJEkAEEegJm-
BElAoBEgghggQAQZQKAnYEaQBAQaASKIAQJEkAEEegJm-
BElAoBEgghggQAQZQKAnYEaQBAQaASKIAQJEkAEEegJm-
BElAoBEgghgg0Aj8i0JO4OzsrPv69Wv+hi2qPHr0qNvf39+iI97soRIh4f3z58/u
7du3SXX7Xt7Z2enevHmzfQe+oSN2apSAPj09TSrb+XKI/f379+08+A0cNRE2ANku
pk+ACNPvkSPcAAEibACyXUyfABGm3yNHuAECRNgAZLuYPgEirKlHu7u7XdyytGwH
Ad8jjNyng4OD7vnz51dbPT8/7z58+NB9+/bt6jU/TI+AGWHEnrx48eJ/EsSmHzx4
0L18+fLyzxF3ZVMjEyDCiEDjMYZZS5wiPXnyZFbJaxMhQIQRGzHvWR7XCyOCXsOm
iDAi1HmPMMQjDpbpEiDCiL358eNHurW/5SnWdIBbXiDCiA38/Pnzrce2YyZ4//59
F3ePLNMl4PbpiL2J0L979+7yDtHDhw8vtzzvdGnEXdvUigSIsCLAWavHp/+qM0Bc
XMd/q25n1vF57TYBp0a3mUzilePj4+7k5KSLb6gt6ydAhPUzXnoPR0dHl79WGTNC
fBnn1uvSCJdegQhLI1vvCk+fPu2ePXt2tZOYEV6/fn31dz+shwAR1sP1cqvLntbE
N9MxA9xcYjsxS1jWR4AIa2Ibzx0tc44fYX/16lV6NDFLXH+YL32jwiACRBiEbf5K
cXo-
TIsQSpzXx4N28Ja4BQoK7rgXiydbHjx/P25TaQAJEGAguWy0+2Q8PD6/Ki4R8EVl
+bzBOnZY95fq9rj9zAkTI2SxdidBHqG9+skdw43borCXO/ZcJdraPWdv22uIEiLA
4q7nvvCug8WTqzQveOH26fodo7g6uFe/a17W3+nFBAkRYENRdb1vkkz1CH9cPsVy
/jrhr27PqMYvENYNlHAIesRiBYwRy0V+8iXP8+/fvX11Mr7L7ECueb/r48eMqm7F
uI2BGWDEG8cm+7G3NEOfmdcTQw4h9/55lhm7DekRYKQPZF2ArbXTAyu4kDYB2YxU
zwg0gi/41ztHnfQG26HbGel/crVrm7tNY+/1btkOEAZ2M05r4FB7r9GbAIdxaZYr
HdOsgJ/wCEQY0J74TmOKnbxxT9n3FgGGWWsVdowHtjt9Nnvf7yQM2aZU/TIAIAxr
w6dOnAWtZZcoEnBpNuTuObWMEiLAx1HY0ZQJEmHJ3HNvGCBBhY6jtaMoEiJB0Z29
vL6ls58vxPcO8/zfrdo5qvKO+d3Fx8Wu8zf1dW4p/cPzLly/dtv9Ts/EbcvGAHhH
yfBIhZ6NSiIBTo0LNNtScABFyNiqFCBChULMNNSdAhJyNSiECRCjUbEPNCRAhZ6N
SiAARCjXbUHMCRMjZqBQiQIRCzTbUnAARcjYqhQgQoVCzDTUnQIScjUo-
hAkQo1GxDzQkQIWejUogAEQo121BzAkTI2agUIkCEQs021JwAEXI2KoUIEKFQsw0
1J0CEnI1KIQJEKNRsQ80JECFno1KIABEKNdtQcwJEyNmoFCJAhELNNtScABFyNiq
FCBChULMNNSdAhJyNSiECRCjUbEP-
NCRAhZ6NSiAARCjXbUHMCRMjZqBQiQIRCzTbUnAARcjYqhQgQoVCzDTUnQIScjUo
hAkQo1GxDzQkQIWejUogAEQo121BzAkTI2agUIkCEQs021JwAEXI2KoUIEKFQsw0
1J0CEnI1KIQJEKNRsQ80JECFno1KIABEKNdtQcwJEyNmoFCJAhELNNtScABFyNiq
FCBChULMNNSdAhJyNSiECRCjUbEP-
NCRAhZ6NSiAARCjXbUHMCRMjZqBQiQIRCzTbUnAARcjYqhQgQoVCzDTUnQIScjUo
hAkQo1GxDzQkQIWejUogAEQo121BzAkTI2agUIkCEQs021JwAEXI2KoUIEKFQsw0
1J0CEnI1KIQJEKNRsQ80JECFno1KIABEKNdtQcwJEyNmoFCJAhELNNtScABFyNiq
FCBChULMNNSdAhJyNSiEC/wGgKKC4YMA4TAAAAABJRU5ErkJggg=="
                />
                <Flex
                    vertical={'vertical'}
                    // justify={'space-between'}
                    style={{ marginLeft: '16px' }}>
                    <Flex justify={'space-between'}>
                        <div style={styleLabel}>ФИО:</div>
                                                                <div
style={styleInput}>{graduatesData.FIO}</div>
                    </Flex>
                    <Flex justify={'space-between'}>
                                    <div  style={styleLabel}>Год
```

```
выпуска:</div>
                                                                    <div
style={styleInput}>{graduatesData.yearFinish}</div>
                    </Flex>

                    <Flex justify={'space-between'}>
                        <div style={styleLabel}>Окончил:</div>
                                                                    <div
style={styleInput}>{graduatesData.finishUniversyty}</div>
                    </Flex>
                    <Flex justify={'space-between'}>
                        <div style={styleLabel}>Факультет:</div>
                                                                    <div
style={styleInput}>{graduatesData.facult}</div>
                    </Flex>

                    <Flex justify={'space-between'}>
                        <div style={styleLabel}>В группе:</div>
                                                                    <div
style={styleInput}>{graduatesData.nameGroup}</div>
                    </Flex>
                </Flex>
            </Flex>
            <Flex vertical={'vertical'}>
                <Flex justify={'space-between'}>
                    <div style={styleLabel}>Город работы:</div>
                                                                    <div
style={styleWorkInput}>{graduatesData.jobTown}</div>
                </Flex>
                <Flex justify={'space-between'}>
                    <div style={styleLabel}>Место работы:</div>
                                                                    <div
style={styleWorkInput}>{graduatesData.jobPlace}</div>
                </Flex>
                <Flex justify={'space-between'}>
                    <div style={styleLabel}>Должность:</div>
                                                                    <div
style={styleWorkInput}>{graduatesData.jobTitle}</div>
                </Flex>
            </Flex>
        </>
    );
}
```

Листинг 3.8 – Код задающий вид модального окна профиля учебного заведения

```
import React from 'react';
```

```
import { Typography, Flex, Image } from 'antd';

const styleLabel = {
    padding: '8px 16px 8px 0px',
    // marginRight: '16px',
    fontSize: '16px',
};

const styleInput = {
    padding: '8px 16px',
    borderBottom: '1px solid black',
    width: '447px',
    fontSize: '16px',
};

export default function MapProfileModal({ mapProfileData }) {
    return (
        <>
              <Typography.Title level={4} style={{ marginBottom:
'16px' }}>
                Информация об университете
            </Typography.Title>
            <Flex>
                <Image
                    width={192}
                    height={232}
                    src={
                                        'http://localhost:8080/map-
profile/photos/' +
                        mapProfileData.photoUrl
                    }
                />
                <Flex
                    vertical={'vertical'}
                    // justify={'space-between'}
                    style={{ marginLeft: '16px' }}>
                    <Flex justify={'space-between'}>
                        <div style={styleLabel}>Название:</div>
                                                                <div
style={styleInput}>{mapProfileData.nameOrganization}</div>
                    </Flex>
                    <Flex justify={'space-between'}>
                         <div style={styleLabel}>Кол-во выпускни-
ков:</div>
                                                                <div
style={styleInput}>{mapProfileData.coutGraduates}</div>
                    </Flex>

                    <Flex justify={'space-between'}>
                         <div style={styleLabel}>Кол-во факульте-
тов:</div>
                                                                <div
```

```
style={styleInput}>{mapProfileData.countFacult}</div>
                    </Flex>
                    <Flex justify={'space-between'}>
                        <div style={styleLabel}>Город:</div>
                                                                <div
style={styleInput}>{mapProfileData.townOrganization}</div>
                    </Flex>

                    <Flex justify={'space-between'}>
                         <div style={styleLabel}>Сайт университе-
та:</div>
                                                                <div
style={styleInput}>{mapProfileData.linkOrganization}</div>
                    </Flex>
                </Flex>
            </Flex>
        </>
    );
}
```

4 Руководство пользователя

Людей взаимодействующих с сервисом можно разделить на 3 роли:

- неавторизованный пользователь;
- авторизованный пользователь;
- администратор или владелец карты.

Простроим путь пользователя для каждого из этих ролей.

Путь неавторизованного пользователя показан на рисунках 4.1 – 4.7.

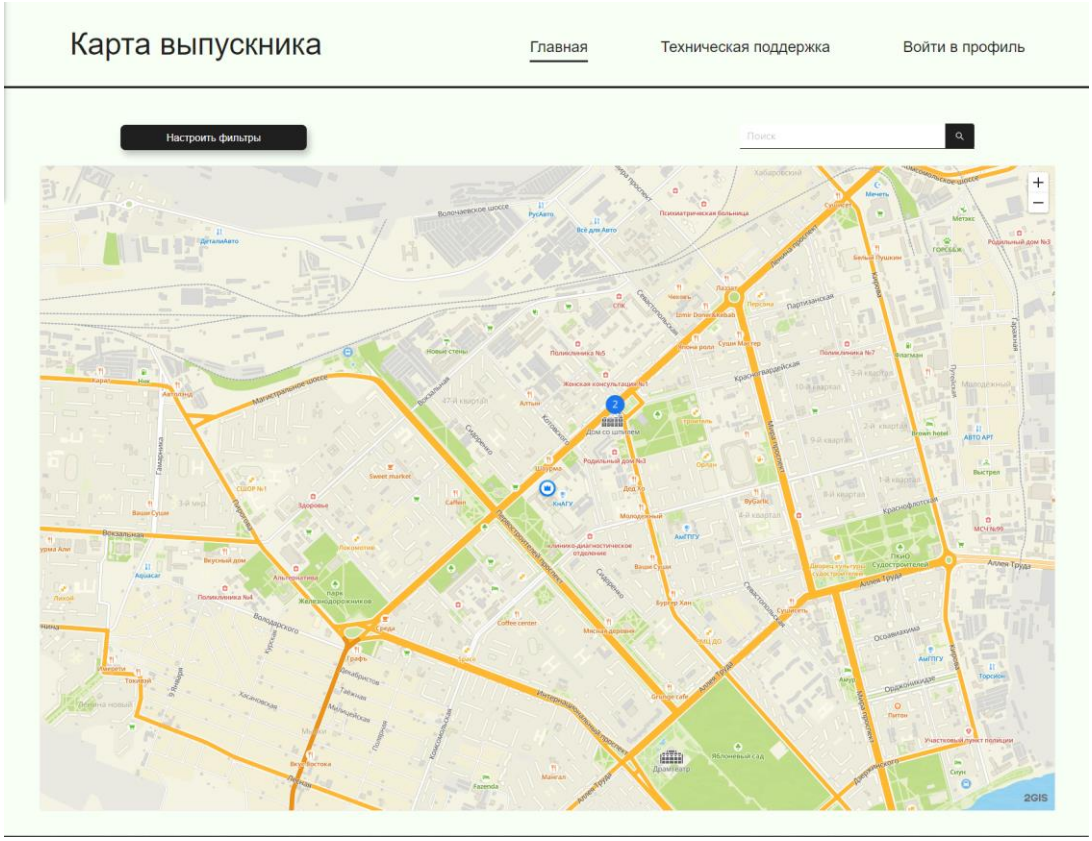

Рисунок 4.1 – Первый этап главная страница сервиса

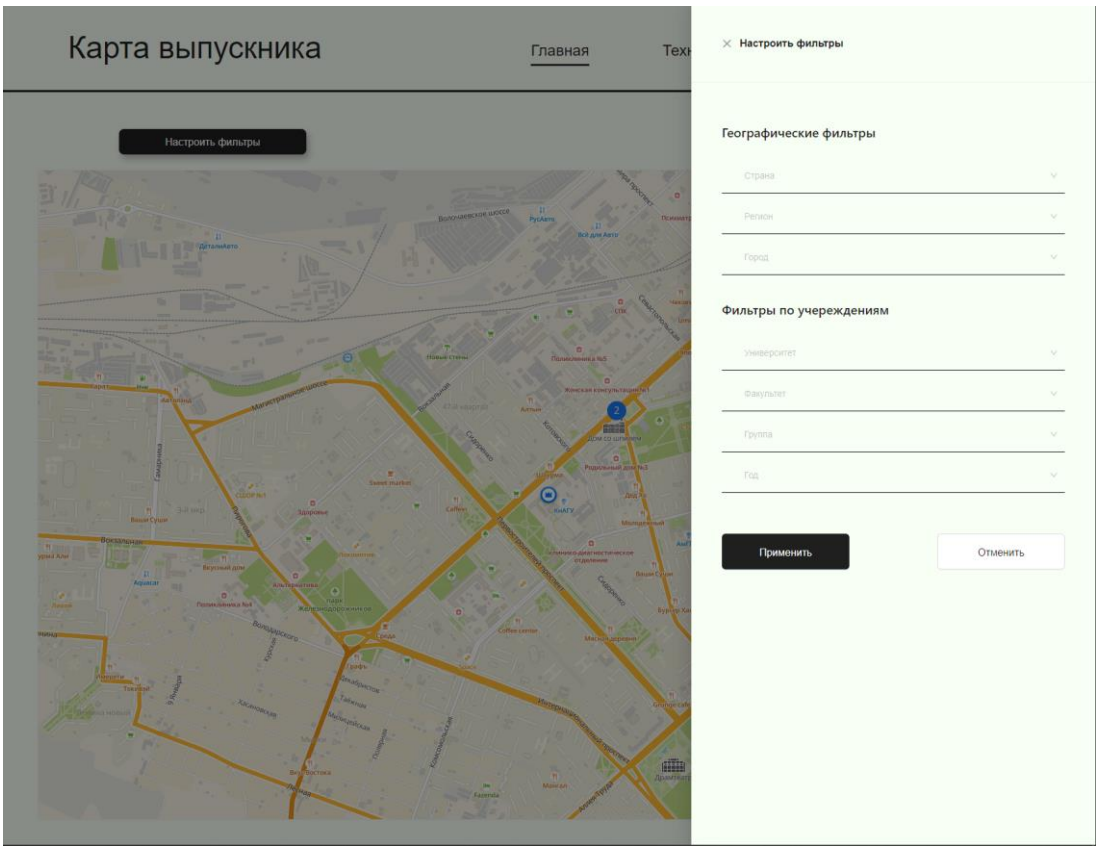

Рисунок 4.2 – Второй этап фильтрация данных на карте

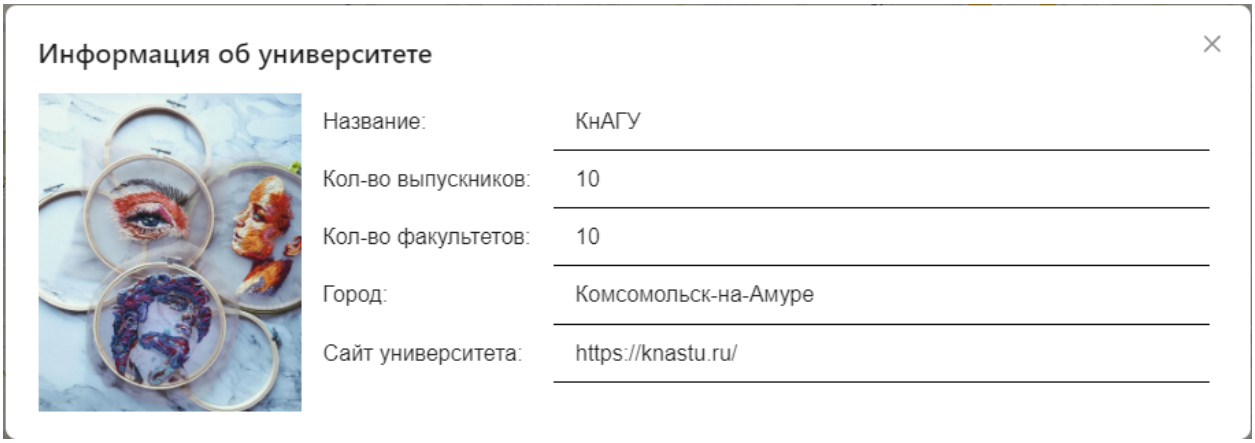

Рисунок 4.3 – Третий этап модальное окно с информацией об учебном заведении

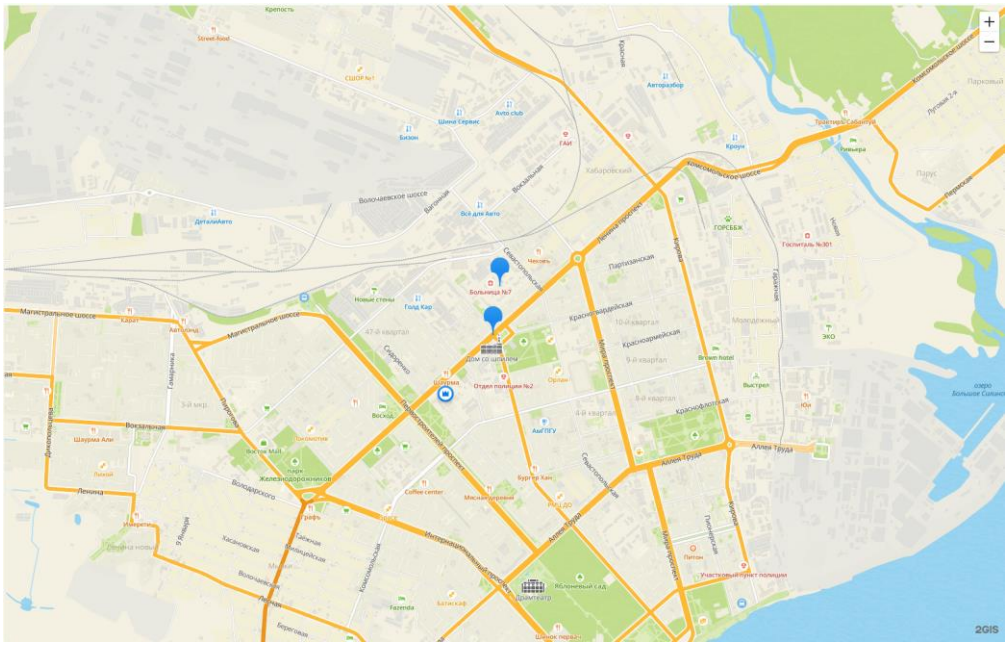

Рисунок 4.4 – Четвертый этап нажатие на кластер, чтобы он распался на метки

Рисунок 4.5 – Пятый этап модальное окно с информацией о выпускнике

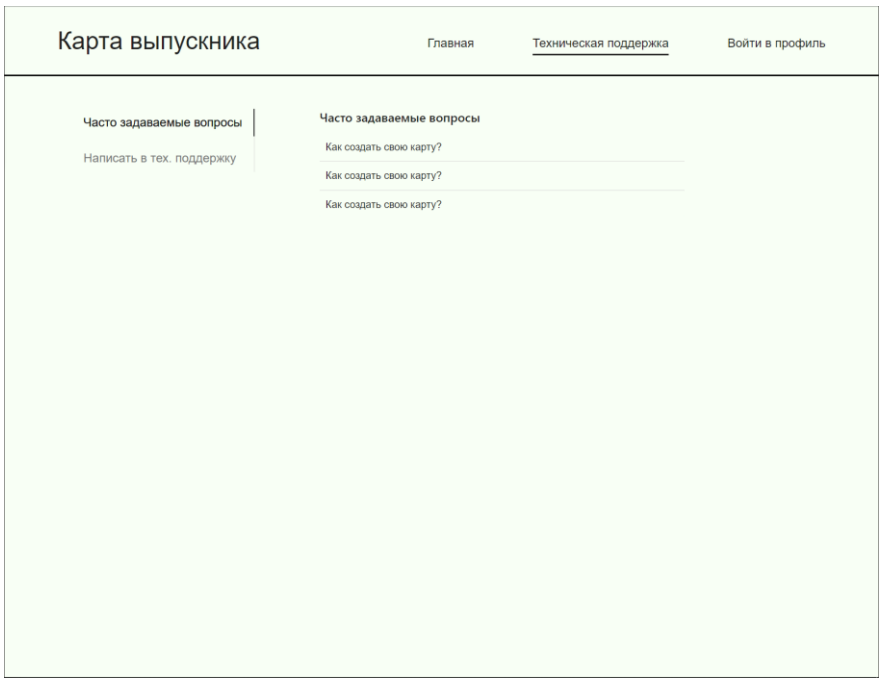

Рисунок 4.6 – Шестой этап страница технической поддержки с часто задаваемыми вопросами

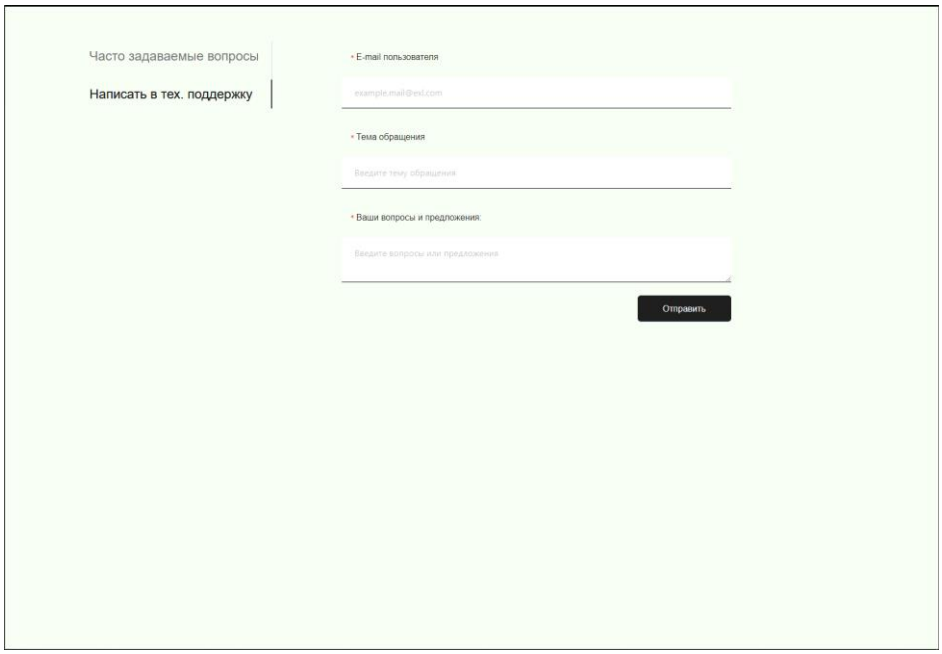

Рисунок 4.7 –Седьмой этап форма, чтобы задать свой вопрос

Путь авторизованного пользователя отличается только возможностью авторизации и заполнении профиля, что показана на рисунках 4.8 – 4.9.

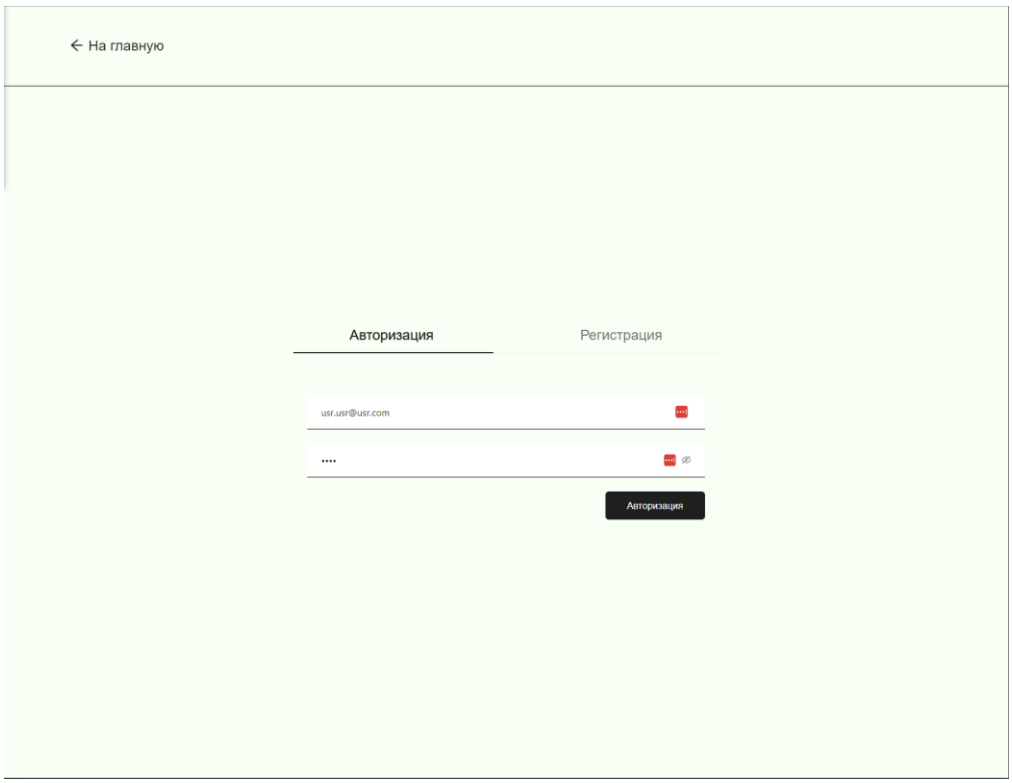

Рисунок 4.8 – Авторизация пользователя в системе

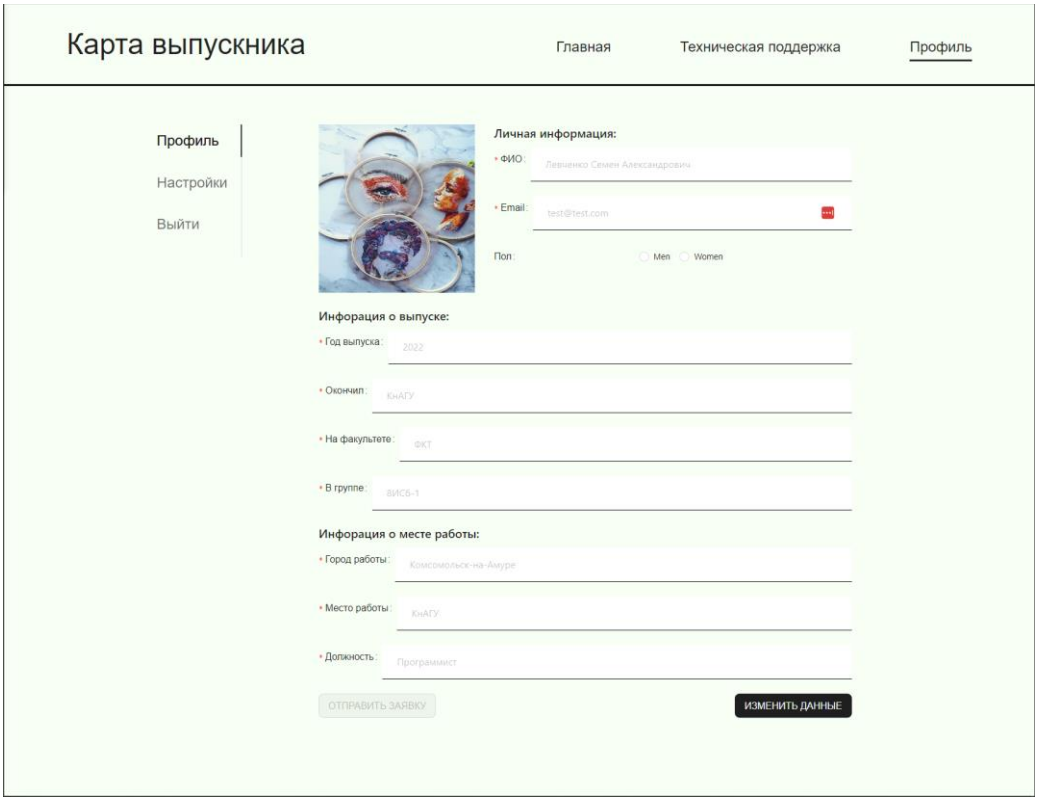

Рисунок 4.9 – Заполнение профиля данными

Путь администратора показана на рисунках 4.10 – 4.15.

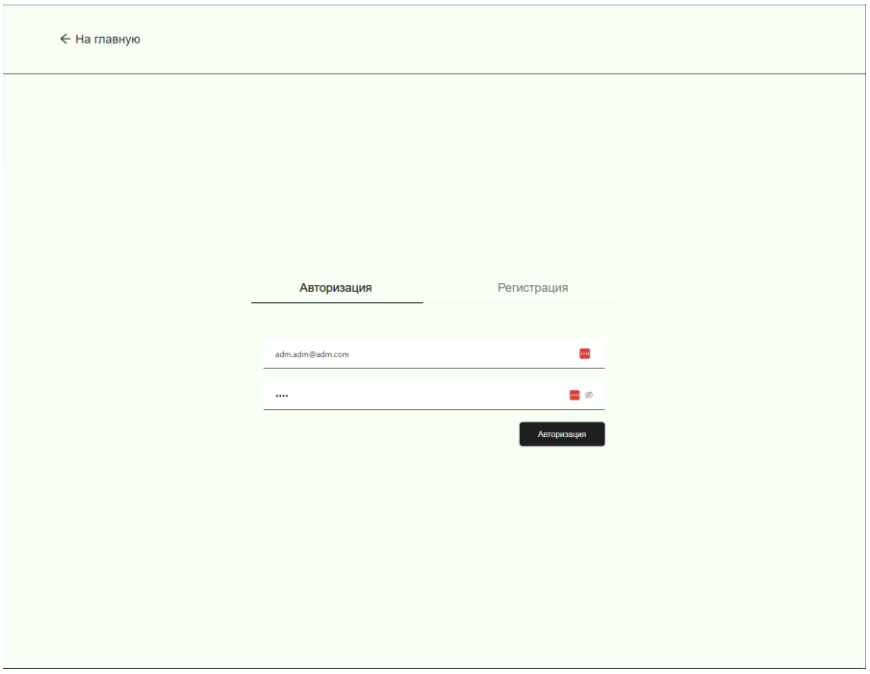

Рисунок 4.10 – Шаг первый авторизация администратора в системе

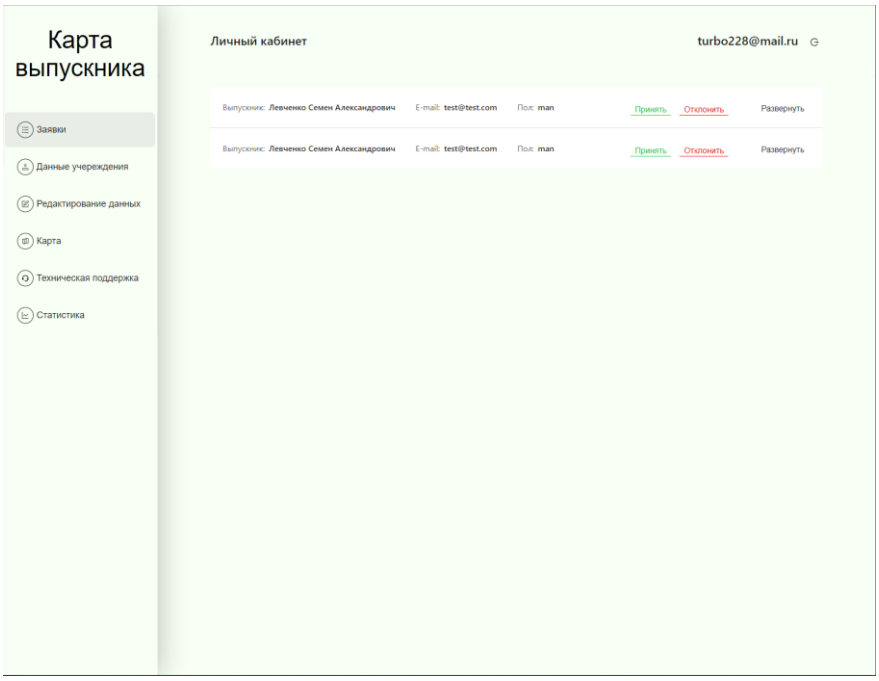

Рисунок 4.11 – Шаг второй заявки отправленные выпускниками

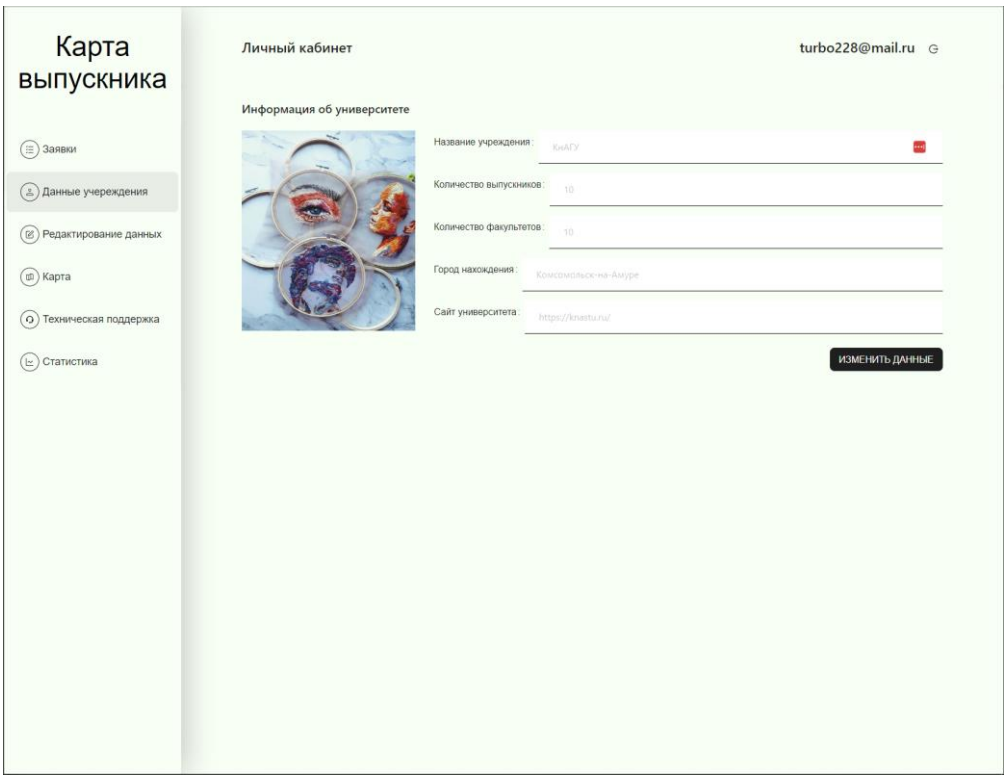

Рисунок 4.12 – Шаг третий заполнение информации об университете

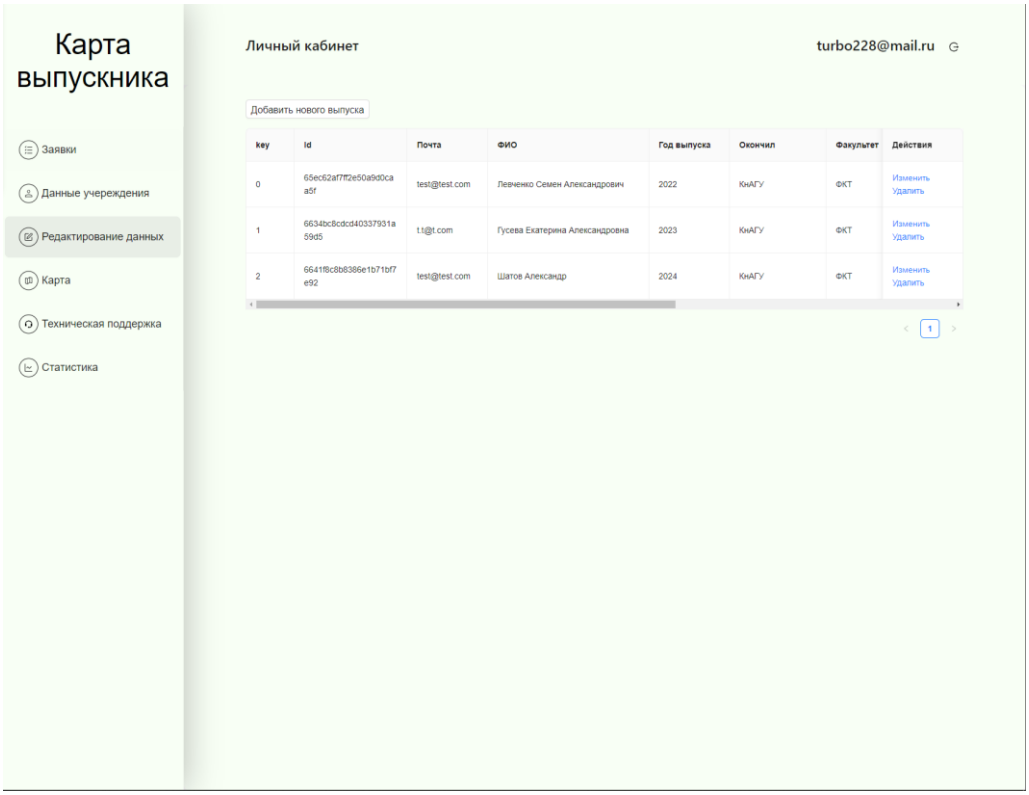

Рисунок 4.13 – Шаг четвертый отредактировать информацию о выпускнике

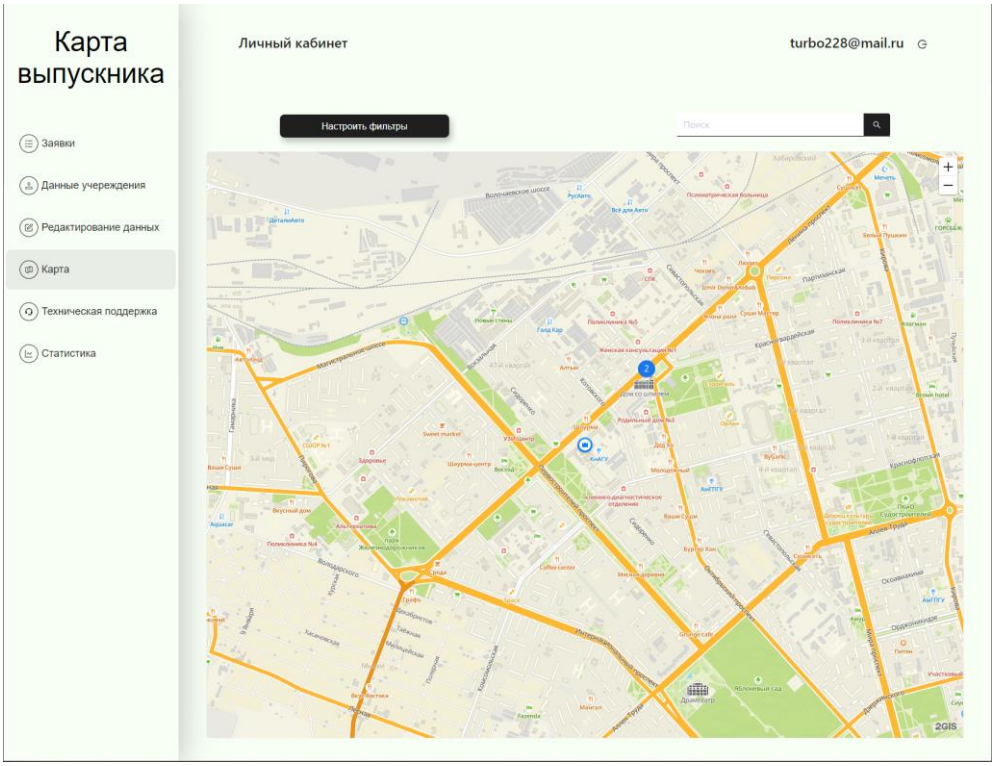

Рисунок 4.14 – Шаг пятый просмотр информации о выпускниках и заведение и на карте

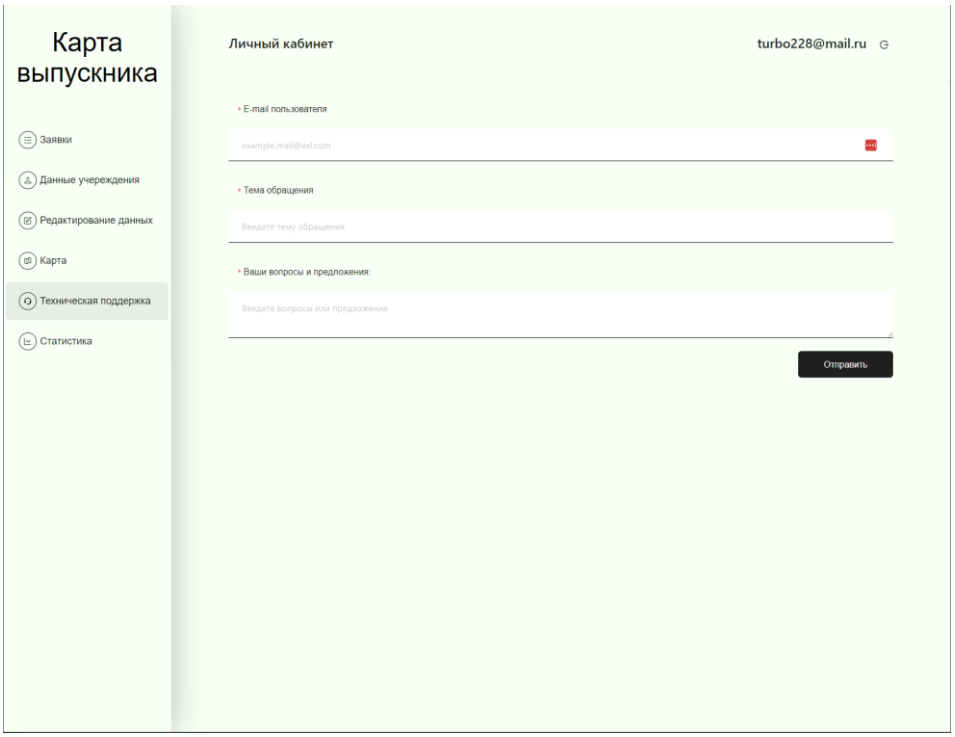

Рисунок 4.15 – Шаг шестой получение технической поддержки по почте

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

СОГЛАСОВАНО

Начальник отдела ОНиПКРС
_____ Е.М. Димитриади
(подпись)
«06» 06 2024 г.

Декан _____
06.06.2024 И.А. Трещев
(подпись)

УТВЕРЖДАЮ

Проректор по научной работе
_____ А.В. Космынин
(подпись)
«07» 06 2024 г.

АКТ
о приемке в эксплуатацию проекта
Разработка веб-сервиса «Карты выпускника»

г. Комсомольск-на-Амуре «06» 06 2024 г.

Комиссия в составе представителей:

со стороны заказчика

- Е.Б. Абарникова – руководитель СПБ «DeCode»,
- И.А. Трещев – декана ФКТ

со стороны исполнителя

- М.Д. Тимохов – руководителя проекта,
- С.А. Валеева – 0ИСб-1,
- Ю.С. Цзин – 2ИБ-1,
- У.С. Михайлова – 2ИБ-1

составила акт о нижеследующем:

«Исполнитель» передает проект *Разработка веб-сервиса «Карты выпускника»*, в составе:

1 Руководство пользователя

2 Руководство программиста

Руководитель проекта 06.06.24 М.Д. Тимохов
(подпись, дата)

Исполнители проекта
(подпись, дата) М.Д. Тимохов
(подпись, дата) С.А. Валеева
(подпись, дата) Ю.С. Цзин
(подпись, дат У.С. Михайлова