МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

КУРСОВАЯ РАБОТА (ПРОЕКТ)
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

Старший преподаватель		А.В. Аксенов
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОЙ РАБОТЕ

Веб-приложение «Танцевальная студия»

по дисциплине: БАЗЫ ДАННЫХ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. № 4142		М.С. Мясникова
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2023

1. **Тема**

   Веб-приложение «Танцевальная студия»

2. **Постановка задачи**

   Целью курсовой работы является разработка веб-приложения танцевальной студии, позволяющего пользователям просматривать информацию о танцевальной студии, преподавателях и записываться на занятия.

   Администратор имеет право изменять и создавать новую информацию о занятиях и преподавателях и назначать других пользователей администраторами.

3. **Словесное описание предметной области и актуальность**

   В наше время танцевальные студии имеют огромную популярность среди молодежи. Различные танцевальные направления развиваются, люди учатся новому, и чтобы реализовать свои способности им требуется студия. Танцевальная студия предоставляет людям пространство для творчества, наставников и единомышленников.

4. **Описание данных, хранящихся в БД**

   БД должна содержать данные о:

   - пользователях, их логин, пароль и роль (ученик, админ);
   - занятии (преподаватель, стиль);
   - стилях (название, описание);
   - расписании (занятие, дата, время);
   - преподавателях (ФИО, навыки);
   - навыках (название, описание);

5. **Роли пользователей приложения**

   Гость;

   Ученик;

   Администратор;

6. **Развернутое описание функционала приложения для каждой из ролей**

   Система доступна для гостей (незарегистрированных пользователей). Без

авторизации пользователь может просмотреть информацию о предстоящих занятиях, о преподавателях и стилях.

Зарегистрированному пользователю-ученику доступна та же информация, что и гостю, но он может записаться на занятие.

Зарегистрированному пользователю-администратору доступна та же информация, что и пользователю. Но дополнительно к этому, он может редактировать информацию о расписании, преподавателях и их навыках.

7. **Предполагаемые технологии и платформа реализации**

   СУБД: PostgreSQL;

   ОС: Windows;

   Язык программирования: Python;

   Фреймворк: Flask;

   Тип приложения: веб-приложение.

8. **Срок представления курсовой работы**

   22.12.2023

ER-диаграмма базы данных

На рисунке 1 представлена ER-диаграмма базы данных, разработанная для работы с веб-сервисом.

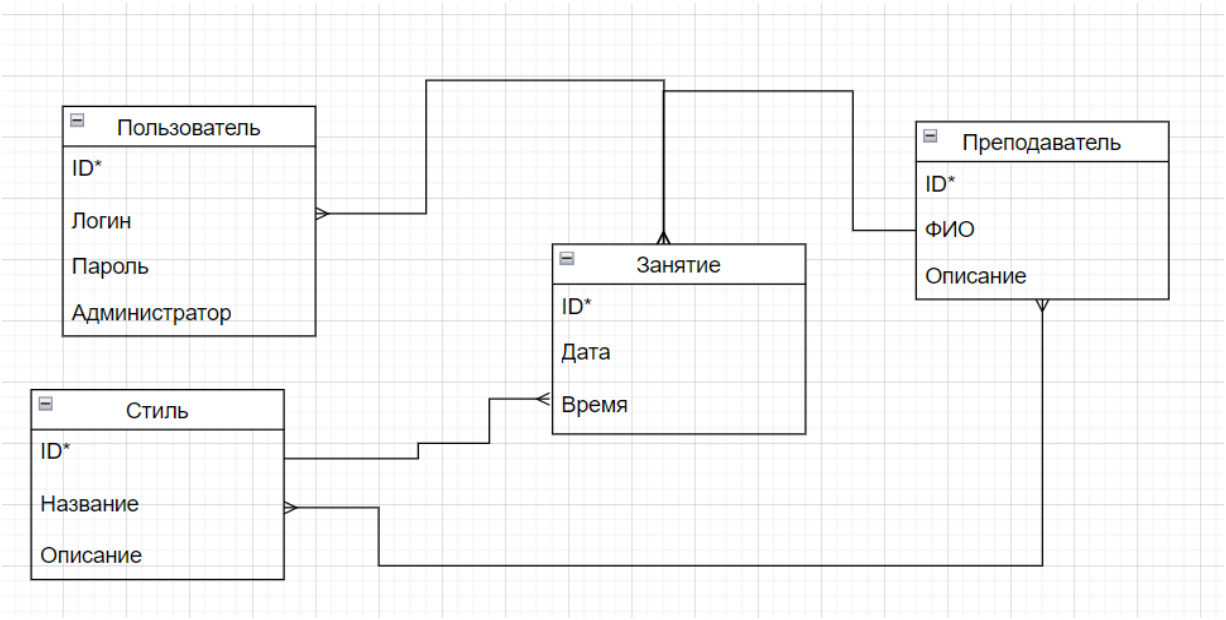

Рисунок 1 – ER – диаграмма

Реляционная схема

На рисунке 2 представлена реляционная схема базы данных, разработанная для работы с веб-сервисом.

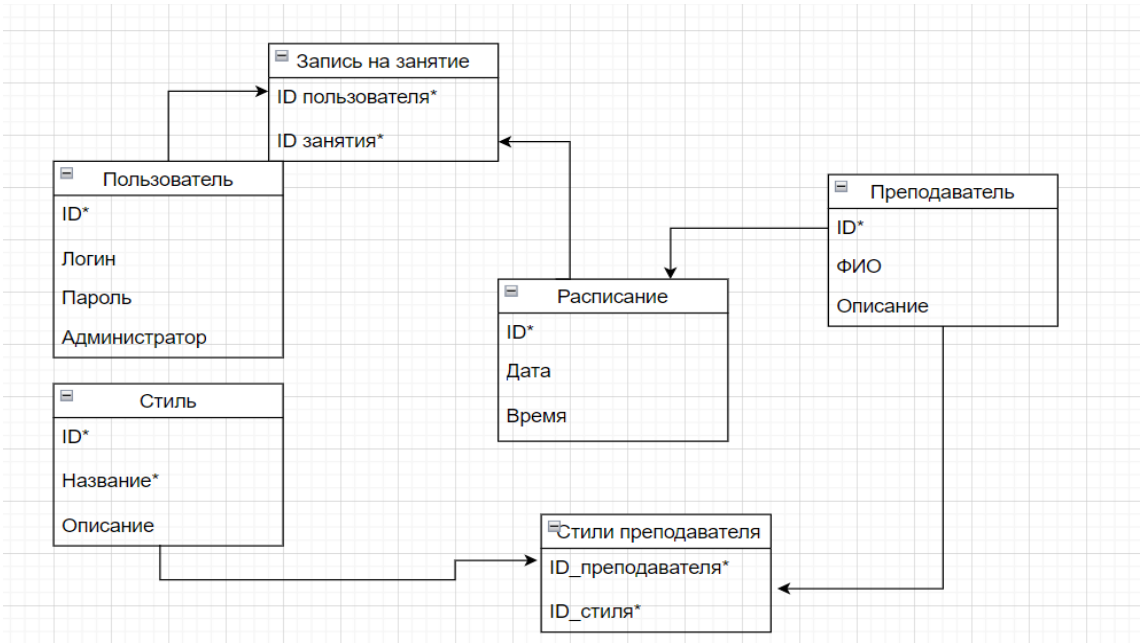

Рисунок 2 – Реляционная схема

Диаграмма вариантов использования

На рисунке 3 представлена диаграмма использования для базы данных, разработанная для работы с веб-сервисом.

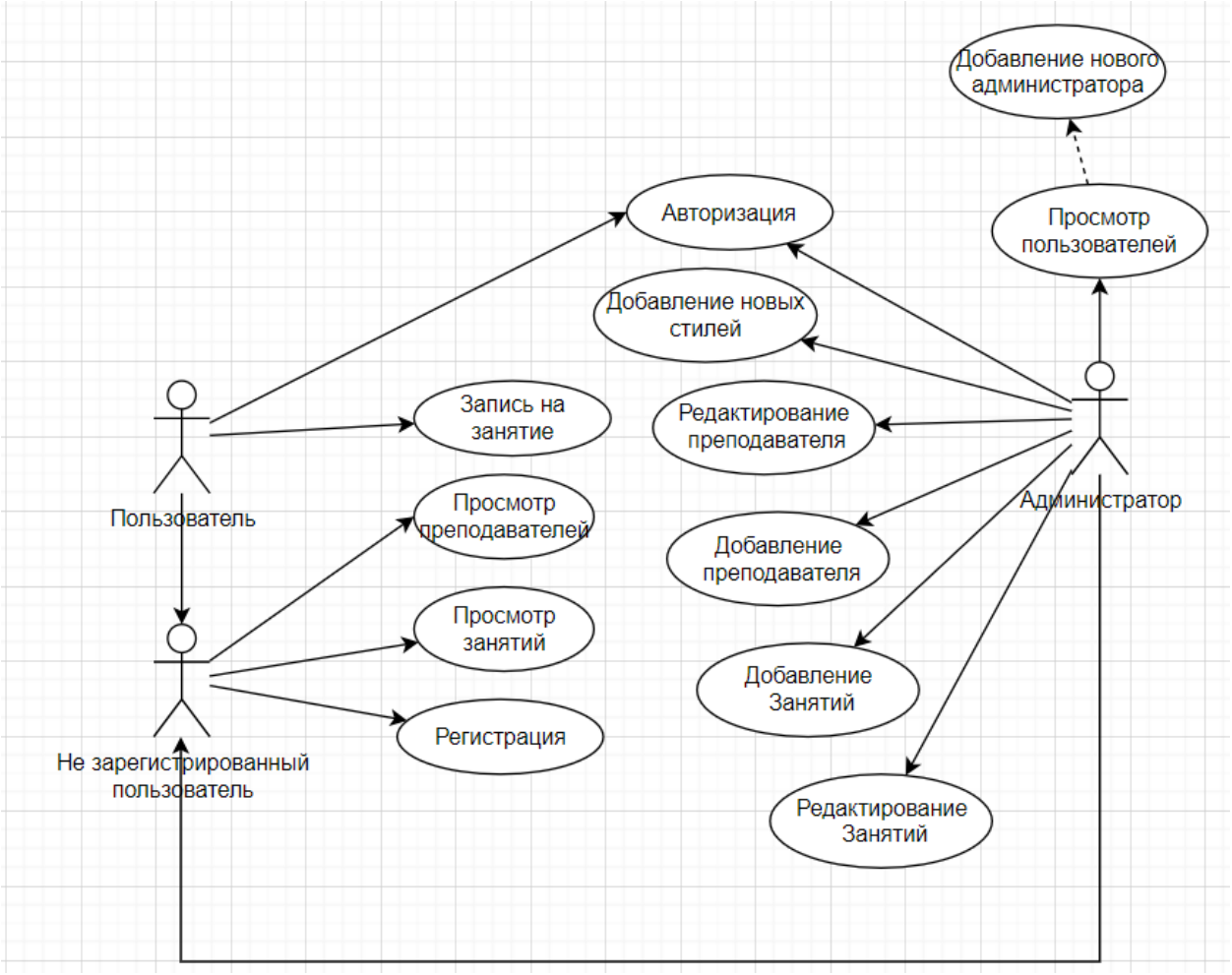

Рисунок 3 – Диаграмма вариантов использования

Описание технологий реализации

Для разработки веб-приложения были использованы следующие программные средства:

- Язык программирования Python с фреймворком Flask
- Система управления базами данных PostgreSQL

Flask – фреймворк для создания веб-приложений на языке программирования Python, использующий набор инструментов Werkzeug, а также шаблонизатор Jinja2. Относится к категории так называемых микрофреймворков – минималистичных каркасов веб – приложений, сознательно предоставляющих лишь самые базовые возможности.

Встроенные возможности Flask:

- Сервер разработки и отладчик
- Интегрированная поддержка модульного тестирования
- Отправка запросов RESTful
- Использование шаблонизатора Jinja2
- Поддержка безопасных файлов cookie (сеансы на стороне клиента)
- Обширная документация
- Совместимость с Google App Engine
- Расширения для улучшения желаемых функций

PostgreSQL – это свободно распространяемая объектно-реляционная система управления базами данных (ORDBMS), наиболее развитая из открытых СУБД в мире. PostgreSQL базируется на языке SQL и поддерживает многие из возможностей стандарта SQL:2011.

Сильными сторонами PostgreSQL считаются:

- Высокопроизводительные и надёжные механизмы транзакций и репликации;
- Наследование;
- Возможность индексирования геометрических (в частности, географических) объектов и наличие базирующегося на ней расширения PostGIS;
- Встроенная поддержка слабоструктурированных данных в формате JSON с возможностью их индексации

Описание используемых методов взаимодействия с БД

База данных подключается в файле __init__.py проекта. Для подключения необходимо импортировать psycopg2 в проект и создать переменную, с помощью которой мы будем обращается к СУБД. Используя SQL подключаемся к БД postrges, указывая имя пользователя, пароль, ip-адрес с портом, а также название подключаемой БД. Благодаря SQL, мы можем делать запросы, изменять БД, а также сохранять в ней результаты запроса.

Описание архитектуры приложения

Проект состоит из большого числа файлов. Рассмотрим основные из них.

- views.py – основная логика работы веб-приложения – ответ на запросы пользователя по конкретным веб-адресам в виде html – страницы;
- models.py – представление сущностей БД в виде классов;
- _init_.py – инициализация приложения;
- forms.py – содержит формы для упрощения работы с html – страницами;
- config.py – содержит значение секретного ключа и параметры для подключения к базе данных;
- decorators.py – содержит декораторы, которые реализует кастомную логику проверки аутентификации пользователя. Это мощный механизм, позволяющий изменять или расширять поведение функций или классов без изменения их исходного кода.

Текст программы

main.py

```
from app import app, models

if __name__ == '__main__':
    with app.app_context():
        models.create_tables()
    app.run()
```

init.py

```
from flask import Flask
from flask_login import LoginManager
app = Flask(__name__)
app.config.from_object('config')

import psycopg2

# Функция для установки соединения с PostgreSQL
def get_db():
    conn = psycopg2.connect(
        host=app.config['DB_HOST'],
        port=app.config['DB_PORT'],
        user=app.config['DB_USER'],
        password=app.config['DB_PASSWORD'],
        database=app.config['DB_NAME']
    )
    return conn
login = LoginManager(app)
login.login_view = 'login'
from app import views
```

views.py

```python
from app import app
from app.models import Users, create_tables, Style, Teacher, Teacher_style,
Lesson, User_lesson
from flask import render_template, request, redirect, url_for, flash
from app.forms import *
from flask_login import login_user, logout_user, current_user, login_required
from app.decorators import custom_login_required, admin_required

@app.route('/')
def index():
     users = Users.get_all()
     return render_template('index.html', users=users)

@app.route('/registration', methods=['GET', 'POST'])
def registration():
    id = None
    password = None
    admin_user = Users(id, 'admin', password, True)
    admin_user.set_password('admin')
    if admin_user.is_username_unique():
        admin_user.save()
        print('Админ создан')
    else:
        print('Админ уже есть в бд')

    form = UserForm()

    if form.validate_on_submit():
        username = form.username.data
        id=None
        password = None
        new_user = Users(id, username, password)
        new_user.set_password(form.password.data)
        if new_user.is_username_unique():
            new_user.save()
            login_user(new_user) #новый пользователь авторизован
            flash('Пользователь успешно добавлен', 'success')
            return redirect(url_for('index'))
        else:
            flash('Имя пользователя уже существует', 'error')

    return render_template('registration.html', form=form)

@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()

    if form.validate_on_submit():
        user = Users.get_by_username(form.username.data)
        if user:
            if user.check_password(form.password.data):
                login_user(user, remember=form.remember_me.data)
                flash('Вход выполнен успешно', 'success')
                return redirect(url_for('index'))
            else:
                flash('Неверный пароль', 'error')
        else:
            flash('Неверное имя пользователя или пароль', 'error')

    return render_template('login.html', form=form)

@app.route('/logout')
@custom_login_required
def logout():
```

```
    logout_user()
    flash('Вы успешно вышли из системы', 'success')
    return redirect(url_for('login'))

@app.route ('/addstyle', methods=['GET', 'POST'])
@admin_required
def addstyle ():
    form = StyleForm()

    if form.validate_on_submit():
        new_style = Style(name=form.stylename.data,
description=form.description.data)
        if new_style.is_name_unique():
            new_style.save()
            flash('Стиль успешно добавлен', 'success')
            return redirect(url_for('index'))
        else:
            flash('Такой стиль уже существует')

    return render_template('add_style.html', form=form)

@app.route ('/addteacher', methods=['GET', 'POST'])
@admin_required
def addteacher ():
    form = TeacherForm()

    if form.validate_on_submit():
        #инициализация класса
        new_teacher = Teacher(FIO=form.teacherFIO.data,
description=form.description.data)
        if new_teacher.is_FIO_unique():
            new_teacher.save()
            flash('Преподаватель успешно добавлен', 'success')
            return redirect(url_for('index'))
        else:
            flash('Такой преподаватель уже существует')

    return render_template('add_teacher.html', form=form)

@app.route('/teachers')
def view_teachers():
    teachers = Teacher.get_all_teacher()
    return render_template('view_teacher.html', teachers=teachers)

@app.route ('/editteacher/<id>', methods=['GET', 'POST'])
@admin_required
def editteacher (id):
    teacher_name = Teacher.get_teacher_name_by_id(id)
    teacher = Teacher.get_by_name(teacher_name)
    form = TeacherForm()
    form.submit.label.text = 'Изменить'
    #изначальные данные
    form.teacherFIO.data = teacher.FIO
    form.description.data = teacher.description
    if form.validate_on_submit():
        teacher_edit = Teacher(
            id=id,
            FIO=request.form.get('teacherFIO'),
            description=request.form.get('description')
        )
        if teacher.FIO != teacher_edit.FIO:
            if teacher_edit.is_FIO_unique():
                teacher_edit.update()
                flash('Преподаватель успешно отредактирован', 'success')
```

```
                return redirect(url_for('view_teachers'))
            else:
                flash('Такой преподаватель уже существует')
        elif teacher_edit.description != teacher.description:
            teacher_edit.update()
            flash('Преподаватель успешно отредактирован', 'success')
            return redirect(url_for('view_teachers'))
        else:
            flash('Вы ничего не изменили', 'success')

    return render_template('edit_teacher.html', form=form)

@app.route('/infoteacher/<id>')
def infoteacher(id):
    ids_styles= Teacher_style.get_styles_for_teacher(id)
    styles_names=Teacher_style.get_style_names(ids_styles)
    return render_template('view_styles_teacher.html', styles=styles_names,
id=id)

@app.route ('/addstyleteacher/<id>', methods=['GET', 'POST'])
@admin_required
def addstyleteacher(id):
    form = TeacherStyleForm()
    styles = Style.get_all_styles()
    style = [i[1] for i in styles]
    form.teacherStyle.choices = style

    if form.validate_on_submit():
        new_style = Style.get_by_name(form.teacherStyle.data)
        style_for_teacher = Teacher_style(teacherID=id, styleID=new_style.id)
        if style_for_teacher.is_TeacherStyle_unique(id, new_style.id):
            style_for_teacher.save()
            flash('Стиль для преподавателя успешно добавлен', 'success')
            return redirect(url_for('infoteacher', id = str(id)))
        else:
            flash('Такой стиль уже существует')

    return render_template('add_style_teacher.html', form=form)

@app.route ('/addlesson', methods=['GET', 'POST'])
@admin_required
def addlesson ():
    form = LessonForm()

    form.teacherFIO.choices = [teacher[1] for teacher in
Teacher.get_all_teacher()]#помещаем в SelectField
    form.styleName.choices = [style[1] for style in Style.get_all_styles()]

    if form.validate_on_submit():
        teacher = Teacher.get_by_name (form.teacherFIO.data)

        style = Style.get_by_name (form.styleName.data)
        date = form.date.data
        time = form.time.data
        new_lesson = Lesson (date=date, time=time, teacher_id=teacher.id,
style_id=style.id)
        if new_lesson.is_lesson_unique(date, time, teacher.id):
            new_lesson.save()
            flash('Занятие успешно добавлено', 'success')
            return redirect(url_for('view_lessons'))
        else:
            flash('Такое занятие уже существует')
```

```
    return render_template('add_lesson.html', form=form)

@app.route('/lesson', methods=['GET', 'POST'])
def view_lessons():
    form = RecordLessonForm()
    lessons = Lesson.get_all_lessons()
    formatted_lessons = [] #список

    for lesson in lessons:
        teacher_name = Teacher.get_teacher_name_by_id(lesson[3]) #из каждой
строки, которые находятся в списке, берем третье значение
        style_name = Style.get_style_name_by_id(lesson[4])

        formatted_lesson = {
            'teacher_name': teacher_name,
            'style_name': style_name,
            'date': lesson[1],
            'time': lesson[2],
            'id': lesson[0],
        }
        formatted_lessons.append(formatted_lesson)

    if form.validate_on_submit():
        lesson_id = request.form['lesson_id']
        user_id = current_user.id
        if not User_lesson.is_user_registered_for_lesson(user_id, lesson_id):
            user_lesson = User_lesson(userID=user_id, lessonID=lesson_id)
            user_lesson.save()
            flash('Вы успешно записались на занятие', 'success')

            return redirect(url_for('index'))
        else:
            flash('Вы уже записаны на это занятие', 'error')


    return render_template('view_lessons.html', lessons=formatted_lessons,
form=form)

@app.route ('/editlesson/<id>', methods=['GET', 'POST'])
@admin_required
def editlesson(id):
    lesson = Lesson.get_lesson_by_id(id)
    form = LessonForm()
    form.submit.label.text = 'Изменить'

    form.teacherFIO.choices = [teacher[1] for teacher in
Teacher.get_all_teacher()]
    form.styleName.choices = [style[1] for style in Style.get_all_styles()]
    teacher_name = Teacher.get_teacher_name_by_id(lesson.teacher_id)
    style_name = Style.get_style_name_by_id(lesson.style_id)
    if form.validate_on_submit():
        teacher = Teacher.get_by_name(form.teacherFIO.data)
        style = Style.get_by_name(form.styleName.data)
        #новые значения
        lesson_edit = Lesson(
            id=id,
            date=request.form.get('date'),
            time=request.form.get('time'),
            teacher_id=teacher.id,
            style_id=style.id,
        )

        if lesson.date != lesson_edit.date:
            if lesson_edit.is_lesson_unique(date=request.form.get('date'),
```

```
time=request.form.get('time'), teacher_id=teacher.id):
                lesson_edit.update()
                flash('Занятие успешно отредактировано', 'success')
                return redirect(url_for('view_lessons'))
            else:
                flash('Такое занятие уже существует')
        elif lesson_edit.time != lesson.time:
            lesson_edit.update()
            flash('Преподаватель успешно отредактирован', 'success')
            return redirect(url_for('view_lessons'))
        elif lesson_edit.teacher_id != lesson.teacher_id:
            lesson_edit.update()
            flash('Занятие успешно отредактировано', 'success')
            return redirect(url_for('view_lessons'))
        elif lesson_edit.style_id != lesson.style_id:
            lesson_edit.update()
            flash('Занятие успешно отредактировано', 'success')
            return redirect(url_for('view_lessons'))
        else:
            flash('Вы ничего не изменили', 'success')

    return render_template('edit_lesson.html', form=form, lesson=lesson,
nameT=teacher_name, nameS=style_name)

@app.route('/styles')
def view_styles():
    styles = Style.get_all_styles()
    return render_template('view_styles.html', styles=styles)

@app.route('/addadmin/<id>')
@custom_login_required
def addadmin(id):
    user = Users.get_by_id(id)
    user.update_admin(True)
    flash('Новый администратор добавлен', 'success')
    return redirect(url_for('index'))

@app.route ('/editstyle/<id>', methods=['GET', 'POST'])
@admin_required
def editstyle (id):
    style_name = Style.get_style_name_by_id(id)
    style = Style.get_by_name(style_name)
    form = StyleForm()
    form.submit.label.text = 'Изменить'
    form.stylename.data = style.name
    form.description.data = style.description
    if form.validate_on_submit():
        style_edit = Style(
            id=id,
            name=request.form.get('stylename'),
            description=request.form.get('description')
        )
        if style.name != style_edit.name:
            if style_edit.is_name_unique():
                style_edit.update()
                flash('Стиль успешно отредактирован', 'success')
                return redirect(url_for('view_styles'))
            else:
                flash('Такой стиль уже существует')
        elif style_edit.description != style.description:
            style_edit.update()
            flash('Стиль успешно отредактирован', 'success')
            return redirect(url_for('view_styles'))
        else:
```

```
            flash('Вы ничего не изменили', 'success')

    return render_template('edit_style.html', form=form)
```

models.py

```
from app import get_db
from werkzeug.security import generate_password_hash, check_password_hash
from flask_login import UserMixin
from app import login


def create_tables():
    conn = get_db()
    cursor = conn.cursor()
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS users(
            id SERIAL PRIMARY KEY,
            username VARCHAR(80) NOT NULL,
            password VARCHAR NOT NULL,
            administrator bool
        );
    ''')
    cursor.execute('''
            CREATE TABLE IF NOT EXISTS style (
                id SERIAL PRIMARY KEY,
                name VARCHAR(80) NOT NULL,
                description VARCHAR(120) NOT NULL
            );
        ''')
    cursor.execute('''
               CREATE TABLE IF NOT EXISTS teacher (
                   id SERIAL PRIMARY KEY,
                   FIO VARCHAR(80) NOT NULL,
                   description VARCHAR(120) NOT NULL
               );
           ''')
    cursor.execute('''
                   CREATE TABLE IF NOT EXISTS teacher_style (
                       teacher_id INTEGER,
                       style_id INTEGER,
                       FOREIGN KEY (teacher_id) REFERENCES teacher (id),
                       FOREIGN KEY (style_id) REFERENCES style (id)
                   );
               ''')
    cursor.execute('''
                       CREATE TABLE IF NOT EXISTS lesson (
                           id SERIAL PRIMARY KEY,
                           date DATE NOT NULL,
                           time TIME NOT NULL,
                           teacher_id INTEGER REFERENCES teacher (id) NOT
NULL,
                           style_id INTEGER REFERENCES style (id) NOT NULL
                       );
                   ''')
    cursor.execute('''
                       CREATE TABLE IF NOT EXISTS user_lesson (
                           user_id INTEGER,
                           lesson_id INTEGER,
                           FOREIGN KEY (user_id) REFERENCES users (id),
                           FOREIGN KEY (lesson_id) REFERENCES lesson (id)
                       );
                   ''')
    conn.commit()
    cursor.close()
```

```
    conn.close()


class Users(UserMixin):
    def __init__(self, id, username, password, administrator=False):
        #шаблон
        self.id = id
        self.username = username
        self.password = password
        self.administrator = administrator

    def save(self):
        conn = get_db()
        cursor = conn.cursor()
        cursor.execute(
            "INSERT INTO users (username, password, administrator) VALUES
(%s, %s, %s) RETURNING id;",
            (self.username, self.password, self.administrator)
        )
        self.id = cursor.fetchone()[0]
        conn.commit()
        cursor.close()
        conn.close()

    @staticmethod
    def get_all():
        conn = get_db()
        cursor = conn.cursor()
        cursor.execute('SELECT * FROM users;')
        users = cursor.fetchall()
        cursor.close()
        conn.close()
        return users

    def is_administrator(self):
        return self.administrator

    def is_username_unique(self):
        conn = get_db()
        cursor = conn.cursor()
        cursor.execute('SELECT COUNT(*) FROM users WHERE username = %s;',
(self.username,))
        count = cursor.fetchone()[0] #кол-во
        cursor.close()
        conn.close()
        return count == 0

    def set_password(self, password):
        self.password = generate_password_hash(password)

    def check_password(self, passw):
        return check_password_hash(self.password, passw)

    @staticmethod
    def get_by_id(user_id):
        try:
            user_id = int(user_id)
        except ValueError:
            return None

        conn = get_db()
        cursor = conn.cursor()
        cursor.execute('SELECT * FROM users WHERE id = %s;', (user_id,))
        user_data = cursor.fetchone() #для одного пользователя
```

```
        cursor.close()
        conn.close()

        if user_data:
            return Users(*user_data)
        return None

    @staticmethod
    def get_by_username(username):
        conn = get_db()
        cursor = conn.cursor()
        cursor.execute('SELECT * FROM users WHERE username = %s;',
(username,))
        user_data = cursor.fetchone()
        cursor.close()
        conn.close()

        if user_data:
            return Users(*user_data)
        return None

    def update_admin(self, new_admin):
        conn = get_db()
        cursor = conn.cursor()
        cursor.execute('UPDATE users SET administrator = %s WHERE username =
%s RETURNING id;',
                       (new_admin, self.username))
        updated_user_id = cursor.fetchone()[0]
        conn.commit()
        cursor.close()
        conn.close()

        return updated_user_id

class Style:
    def __init__(self, name, description, id=None):
        self.id = id
        self.name = name
        self.description = description

    def save(self):
        conn = get_db()
        cursor = conn.cursor()
        cursor.execute(
            "INSERT INTO style (name, description) VALUES (%s, %s) RETURNING
id;",
            (self.name, self.description)
        )
        self.id = cursor.fetchone()[0]
        conn.commit()
        cursor.close()
        conn.close()

    @staticmethod
    def get_all_styles():
        conn = get_db()
        cursor = conn.cursor()
        cursor.execute('SELECT * FROM style;')
        styles = cursor.fetchall()
        cursor.close()
        conn.close()
        return styles

    def is_name_unique(self):
```

```
        conn = get_db()
        cursor = conn.cursor()
        cursor.execute('SELECT COUNT(*) FROM style WHERE name = %s;',
(self.name,))
        count = cursor.fetchone()[0]
        cursor.close()
        conn.close()
        return count == 0

    @staticmethod
    def get_by_name(stylename):
        conn = get_db()
        cursor = conn.cursor()
        cursor.execute('SELECT * FROM style WHERE name = %s;', (stylename,))
        style_data = cursor.fetchone()
        cursor.close()
        conn.close()
        # id = style_data[0]
        # stylename = style_data[1]
        # description = style_data[2]
        if style_data:
            id, stylename, description = style_data
            return Style(stylename, description, id)
        return None

    @staticmethod
    def get_style_name_by_id(style_id):
        conn = get_db()
        cursor = conn.cursor()
        cursor.execute('SELECT name FROM style WHERE id = %s;', (style_id,))
        style_name = cursor.fetchone()[0]
        cursor.close()
        conn.close()
        return style_name

    def update(self):
        if not self.id:
            # Если id не задан, обновление невозможно
            return

        conn = get_db()
        cursor = conn.cursor()
        cursor.execute(
            "UPDATE style SET name = %s, description = %s WHERE id = %s;",
            (self.name, self.description, self.id)
        )
        conn.commit()
        cursor.close()
        conn.close()


@login.user_loader
def load_user(user_id):
    return Users.get_by_id(user_id)


class Teacher:
    def __init__(self, FIO, description, id=None):
        self.id = id
        self.FIO = FIO
        self.description = description

    def save(self):
        conn = get_db()
```

```
        cursor = conn.cursor()
        cursor.execute(
            "INSERT INTO teacher (FIO, description) VALUES (%s, %s) RETURNING
id;",
            (self.FIO, self.description)
        )
        self.id = cursor.fetchone()[0]
        conn.commit()
        cursor.close()
        conn.close()

    @staticmethod
    def get_all_teacher():
        conn = get_db()
        cursor = conn.cursor()
        cursor.execute('SELECT * FROM teacher;')
        teachers = cursor.fetchall()
        cursor.close()
        conn.close()
        return teachers

    def is_FIO_unique(self):
        conn = get_db()
        cursor = conn.cursor()
        cursor.execute('SELECT COUNT(*) FROM teacher WHERE FIO = %s;',
(self.FIO,))
        count = cursor.fetchone()[0]
        cursor.close()
        conn.close()
        return count == 0

    @staticmethod
    def get_by_name(teachername):
        conn = get_db()
        cursor = conn.cursor()
        cursor.execute('SELECT * FROM teacher WHERE FIO = %s;',
(teachername,))
        teacher_data = cursor.fetchone()
        cursor.close()
        conn.close()
        if teacher_data:
            id, FIO, description = teacher_data
            return Teacher(FIO, description, id)
        return None

    @staticmethod
    def get_teacher_name_by_id(teacher_id):
        conn = get_db()
        cursor = conn.cursor()
        cursor.execute('SELECT FIO FROM teacher WHERE id = %s;',
(teacher_id,))
        teacher_name = cursor.fetchone()[0]
        cursor.close()
        conn.close()
        return teacher_name

    def update(self):
        if not self.id:
            # Если id не задан, обновление невозможно
            return

        conn = get_db()
        cursor = conn.cursor()
        cursor.execute(
```

```
            "UPDATE teacher SET FIO = %s, description = %s WHERE id = %s;",
            (self.FIO, self.description, self.id)
        )
        conn.commit()
        cursor.close()
        conn.close()

class Teacher_style:
    def __init__(self, teacherID, styleID):
        self.teacherID = teacherID
        self.styleID = styleID

    def save(self):
        conn = get_db()
        cursor = conn.cursor()
        cursor.execute(
            "INSERT INTO teacher_style (teacher_id, style_id) VALUES (%s,
%s);",
            (self.teacherID, self.styleID)
        )
        conn.commit()
        cursor.close()
        conn.close()

    @staticmethod
    def get_styles_for_teacher(id_teacher):
        conn = get_db()
        cursor = conn.cursor()
        cursor.execute('SELECT style_id FROM teacher_style WHERE teacher_id =
%s;', (id_teacher,))
        styles = [row[0] for row in cursor.fetchall()]
        cursor.close()
        conn.close()
        return styles

    @staticmethod
    def get_style_names(style_ids):
        if not style_ids:
            return []
        conn = get_db()
        cursor = conn.cursor()
        # плейсхолдеры для всех элементов style_ids
        placeholders = ', '.join(['%s' for _ in style_ids])
        # выбор строк, у которых id находится в списке style_ids
        cursor.execute(f'SELECT id, name FROM style WHERE id IN
({placeholders});', style_ids)
        style_results = cursor.fetchall()
        cursor.close()
        conn.close()

        # словарь (id стиля-название)
        style_dict = {style_id: style_name for style_id, style_name in
style_results}
        return style_results

    @staticmethod
    def is_TeacherStyle_unique(teacher_id, style_id):
        conn = get_db()
        cursor = conn.cursor()
        # Проверка есть ли уже такой стиль у данного преподавателя
        cursor.execute('SELECT COUNT(*) FROM teacher_style WHERE teacher_id =
%s AND style_id = %s;',
                       (teacher_id, style_id))
        count = cursor.fetchone()[0]
```

```
        cursor.close()
        conn.close()
        return count == 0


class Lesson:
    def __init__(self, date, time, teacher_id, style_id, id=None):
        self.id = id
        self.date = date
        self.time = time
        self.teacher_id = teacher_id
        self.style_id = style_id

    # save все
    def save(self):
        conn = get_db()
        cursor = conn.cursor()
        cursor.execute("INSERT INTO lesson (date, time, teacher_id, style_id)
VALUES (%s, %s, %s, %s) RETURNING id;",
                       (self.date, self.time, self.teacher_id,
self.style_id))
        self.id = cursor.fetchone()[0]
        conn.commit()
        cursor.close()
        conn.close()

    @staticmethod
    def get_all_lessons():
        conn = get_db()
        cursor = conn.cursor()
        cursor.execute('SELECT * FROM lesson;')
        lessons = cursor.fetchall()
        cursor.close()
        conn.close()
        return lessons

    @staticmethod
    def is_lesson_unique(date, time, teacher_id):
        conn = get_db()
        cursor = conn.cursor()
        # проверка есть ли уже урок с такой же датой, временем и учителем
        cursor.execute('SELECT COUNT(*) FROM lesson WHERE date = %s AND time
= %s AND teacher_id = %s;',
                       (date, time, teacher_id))
        count = cursor.fetchone()[0]
        cursor.close()
        conn.close()
        return count == 0

    @staticmethod
    def get_lesson_by_id(lesson_id):
        conn = get_db()
        cursor = conn.cursor()
        cursor.execute('SELECT * FROM lesson WHERE id = %s;', (lesson_id,))
        lesson_data = cursor.fetchone()
        cursor.close()
        conn.close()

        if lesson_data:
            id, date, time, teacher_id, style_id = lesson_data
            return Lesson(date, time, teacher_id, style_id, id)
        return None

    def update(self):
```

```
        if not self.id:
            # Если id не задан, обновление невозможно
            return

        conn = get_db()
        cursor = conn.cursor()
        cursor.execute(
            "UPDATE lesson SET date = %s, time = %s, teacher_id = %s,
style_id = %s WHERE id = %s;",
            (self.date, self.time, self.teacher_id, self.style_id, self.id)
        )
        conn.commit()
        cursor.close()
        conn.close()


class User_lesson:
    def __init__(self, userID, lessonID):
        self.userID = userID
        self.lessonID = lessonID

    def save(self):
        conn = get_db()
        cursor = conn.cursor()
        cursor.execute(
            "INSERT INTO user_lesson (user_id, lesson_id) VALUES (%s, %s);",
            (self.userID, self.lessonID))
        conn.commit()
        cursor.close()
        conn.close()

    @staticmethod
    def is_user_registered_for_lesson(userID, lessonID):
        if not lessonID:
            # когда lessonID не задан
            return False
        conn = get_db()
        cursor = conn.cursor()
        cursor.execute(
            "SELECT COUNT(*) FROM user_lesson WHERE user_id = %s AND
lesson_id = %s;",
            (userID, lessonID)
        )
        count = cursor.fetchone()[0]
        cursor.close()
        conn.close()
        return count > 0
```

forms.py

```
from flask_wtf import FlaskForm
from wtforms import (StringField, PasswordField, SubmitField, BooleanField,
TextAreaField, SelectField, DateField,
                     TimeField)
from wtforms.validators import DataRequired, EqualTo, Length

class UserForm(FlaskForm):
    username = StringField('Имя пользователя', validators=[DataRequired()])
    password = PasswordField('Пароль', validators=[DataRequired()])
    password2 = PasswordField(
        'Повторите Пароль', validators=[DataRequired(), EqualTo('password')])
    submit = SubmitField('Add User')

class LoginForm(FlaskForm):
```

```
    username = StringField('Имя пользователя', validators=[DataRequired()])
    password = PasswordField('Пароль', validators=[DataRequired()])
    remember_me = BooleanField('Запомнить меня')
    submit = SubmitField('Авторизоваться')

class StyleForm (FlaskForm):
    stylename = StringField('Название стиля', validators=[DataRequired()])
    description = TextAreaField('Описание', validators=[DataRequired(),
Length(max=120)])
    submit = SubmitField('Добавить')

class TeacherForm (FlaskForm):
    teacherFIO = StringField('ФИО преподавателя',
validators=[DataRequired()])
    description = TextAreaField('Описание', validators=[DataRequired(),
Length(max=120)])
    submit = SubmitField('Добавить')

class TeacherStyleForm (FlaskForm):
    teacherStyle = SelectField('Назначить стиль преопдавателю',
validators=[DataRequired()])
    submit = SubmitField('Добавить стиль преподавателю')

class LessonForm (FlaskForm):
    teacherFIO = SelectField('Назначить преподавателя',
validators=[DataRequired()])
    styleName = SelectField('Назначить стиль', validators=[DataRequired()])
    date = DateField('Назначить дату', validators=[DataRequired()])
    time = TimeField('Назначить время', validators=[DataRequired()])
    submit = SubmitField('Добавить занятие')

class RecordLessonForm (FlaskForm):
    submit = SubmitField('Записаться на занятие')
```

decorators.py

```
from functools import wraps
from flask import flash, redirect, url_for, abort
from flask_login import current_user

def custom_login_required(func):
    @wraps(func)
    def login_view(*args, **kwargs):
        if not current_user.is_authenticated:
            flash('Пожалуйста, войдите в систему, чтобы просмотреть эту
страницу', 'danger')
            return redirect(url_for('login'))
        return func(*args, **kwargs)
    return login_view

def admin_required(func):
    @wraps(func)
    def admin(*args, **kwargs):
        if current_user.is_authenticated and current_user.administrator:
            #если пользователь аутентифицирован и является администратором
            return func(*args, **kwargs)
        else:
            flash('Вы не администратор сайта')
            return redirect(url_for('index'))
    return admin
```

HTML – файлы

base.html

```
<!DOCTYPE html>
<html>
<head>
    <title>{% block title %}Танцевальная студия{% endblock %}</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='styles.css') }}">
</head>
<body>
    <nav>
        <a href="{{ url_for('index') }}">Главная</a>
        <a href="{{ url_for('view_lessons') }}">Информация о занятиях</a>
        <a href="{{ url_for('view_styles') }}">Информация о стилях</a>
        <a href="{{ url_for('view_teachers') }}">Информация о преподавателях
и стилях</a>
        {% if current_user.is_anonymous %}
        <a href="{{ url_for('login') }}">Авторизация</a>
        <a href="{{ url_for('registration') }}">Регистрация</a>

        {% else%}
        <a href="{{ url_for('logout') }}">Выйти</a><br>
        {% endif %}

        {% if current_user.is_authenticated %}


        {% if current_user.is_administrator() %}
            <p class="admpanel">Панель администратора:</p>
         <a href="{{ url_for('addstyle') }}">Создать стиль</a>
         <a href="{{ url_for('addteacher') }}">Добавить преподавателя</a>
        <a href="{{ url_for('addlesson') }}">Добавить занятие</a>
        {% endif %}
        {% endif %}
    </nav>
    <div class="content">
        {% block content %}{% endblock %}
    </div>
{% with messages = get_flashed_messages() %}
  {% if messages %}
    <ul class="flash-messages">
      {% for message in messages %}
        <li>{{ message }}</li>
      {% endfor %}
    </ul>
  {% endif %}
{% endwith %}
</body>
</html>
```

add_lesson.html

```
{% extends "base.html" %}
{% block title %}Добавление занятия{% endblock %}
{% block content %}
<link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='styles.css') }}">
<style>
        body {
            font-size: 20px;
            background-image: url('/static/fon_login.jpg');
            background-size: cover;
            color: #FFDAB9;
            text-align: center;
        }

        h2 {
```

```
            font-size: 1.5em;
        }
</style>
    <h2>Добавление занятия</h2>

    <form action="" method="post">
        {{ form.hidden_tag() }}
        <p>
            {{ form.teacherFIO.label }}<br>
            {{ form.teacherFIO(size=1) }}<br>
            {% for error in form.teacherFIO.errors %}
            <span style="color: red;">[{{ error }}]</span>
            {% endfor %}
        </p>
        <p>
            {{ form.styleName.label }}<br>
            {{ form.styleName(size=1) }}<br>
            {% for error in form.styleName.errors %}
            <span style="color: red;">[{{ error }}]</span>
            {% endfor %}
        </p>
        <p>
            {{ form.date.label }}<br>
            {{ form.date(size=1) }}<br>
            {% for error in form.date.errors %}
            <span style="color: red;">[{{ error }}]</span>
            {% endfor %}
        </p>
        <p>
            {{ form.time.label }}<br>
            {{ form.time(size=1) }}<br>
            {% for error in form.time.errors %}
            <span style="color: red;">[{{ error }}]</span>
            {% endfor %}
        </p>

        <p>{{ form.submit() }}</p>
    </form>

{% endblock %}
```

add_style.html

```
{% extends "base.html" %}
{% block title %}Добавление стиля{% endblock %}
{% block content %}
<link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='styles.css') }}">
<style>
        body {
            font-size: 20px;
            background-image: url('/static/fon_login.jpg');
            background-size: cover;
            color: #FFDAB9;
            text-align: center;
        }

        h2 {
            font-size: 1.5em;
        }

</style>
    <h2>Создание стиля</h2>
    <form action="" method="post">
```

```
        {{ form.hidden_tag() }}
        <p>
            {{ form.stylename.label }}<br>
            {{ form.stylename(size=32) }}<br>
            {% for error in form.stylename.errors %}
            <span style="color: red;">[{{ error }}]</span>
            {% endfor %}
        </p>
        <p>
            {{ form.description.label }}<br>
            {{ form.description(cols=50, rows=20) }}<br>
            {% for error in form.description.errors %}
            <span style="color: red;">[{{ error }}]</span>
            {% endfor %}
        </p>
        <p>{{ form.submit() }}</p>
    </form>
{% endblock %}
```

add_style_teacher.html

```
{% extends "base.html" %}

{% block content %}
<link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='styles.css') }}">
<style>
        body {
            font-size: 20px;
            background-image: url('/static/fon_edit.jpg');
            background-size: cover;
            color: #FFDAB9;
            text-align: center;
        }

        h2 {
            font-size: 1.5em;
        }
</style>
    <h2>Добавление стиля преподавателя</h2>
    <form action="" method="post">
        {{ form.hidden_tag() }}
        <p>
            {{ form.teacherStyle.label }}<br>
            {{ form.teacherStyle(size=1) }}<br>
            {% for error in form.teacherStyle.errors %}
            <span style="color: red;">[{{ error }}]</span>
            {% endfor %}
        </p>
        <p>{{ form.submit() }}</p>
    </form>
{% endblock %}
```

add_teacher.html

```
{% extends "base.html" %}
{% block title %}Добавление преподавателя{% endblock %}
{% block content %}
<link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='styles.css') }}">
<style>
        body {
            font-size: 20px;
            background-image: url('/static/fon_login.jpg');
```

```
            background-size: cover;
            color: #FFDAB9;
            text-align: center;
        }

        h2 {
            font-size: 1.5em;
        }

</style>
    <h2>Добавление преподавателя</h2>
    <form action="" method="post">
        {{ form.hidden_tag() }}
        <p>
            {{ form.teacherFIO.label }}<br>
            {{ form.teacherFIO(size=32) }}<br>
            {% for error in form.teacherFIO.errors %}
            <span style="color: red;">[{{ error }}]</span>
            {% endfor %}
        </p>
        <p>
            {{ form.description.label }}<br>
            {{ form.description(cols=50, rows=20) }}<br>
            {% for error in form.description.errors %}
            <span style="color: red;">[{{ error }}]</span>
            {% endfor %}
        </p>
        <p>{{ form.submit() }}</p>
    </form>
{% endblock %}
```

edit_lesson.html

```
{% extends "base.html" %}

{% block title %}Редактирование информации о занятии{% endblock %}

{% block content %}
<link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='styles.css') }}">
<style>
        body {
            font-size: 20px;
            background-image: url('/static/fon_edit.jpg');
            background-size: cover;
            color: #FFDAB9;
            text-align: center;
        }

        h2 {
            font-size: 1.5em;
        }
</style>
    <h2>Редактирование информации о занятии</h2>
    <form action="" method="post">
        {{ form.hidden_tag() }}
        <p>
            {{ form.teacherFIO.label }} (текущий преподаватель: {{nameT}})
<br>
            {{ form.teacherFIO(size=1) }}<br>
            {% for error in form.teacherFIO.errors %}
            <span style="color: red;">[{{ error }}]</span>
            {% endfor %}
        </p>
```

```
        <p>
            {{ form.styleName.label }} (текущий стиль: {{nameS}}) <br><br>
            {{ form.styleName(size=1) }}<br>
            {% for error in form.styleName.errors %}
            <span style="color: red;">[{{ error }}]</span>
            {% endfor %}
        </p>
        <p>
            {{ form.date.label }} (текущая дата: {{lesson.date}})<br>
            {{ form.date(size=1) }}<br>
            {% for error in form.date.errors %}
            <span style="color: red;">[{{ error }}]</span>
            {% endfor %}
        </p>
        <p>

            {{ form.time.label }} (текущее время: {{lesson.time}})<br>
            {{ form.time(size=1) }}<br>
            {% for error in form.time.errors %}
            <span style="color: red;">[{{ error }}]</span>
            {% endfor %}
        </p>

        <p>{{ form.submit() }}</p>
    </form>

    <a href="{{ url_for('view_lessons') }}">Назад к списку занятий</a>

{% endblock %}
```

edit_style.html

```
{% extends "base.html" %}

{% block title %}Редактирование информации о стиле{% endblock %}

{% block content %}
<link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='styles.css') }}">
<style>
        body {
            font-size: 20px;
            background-image: url('/static/fon_edit.jpg');
            background-size: cover;
            color: #FFDAB9;
            text-align: center;
        }

        h2 {
            font-size: 1.5em;
        }
</style>
    <h2>Редактирование информации о стиле</h2>
    <form action="" method="post">
        {{ form.hidden_tag() }}
        <p>
            {{ form.stylename.label }}<br>
            {{ form.stylename(size=32) }}<br>
            {% for error in form.stylename.errors %}
            <span style="color: red;">[{{ error }}]</span>
            {% endfor %}
        </p>
        <p>
            {{ form.description.label }}<br>
```

```
            {{ form.description(cols=50, rows=20) }}<br>
            {% for error in form.description.errors %}
            <span style="color: red;">[{{ error }}]</span>
            {% endfor %}
        </p>
        <p>{{ form.submit() }}</p>
    </form>

    <a href="{{ url_for('view_styles') }}">Назад к списку стилей</a>

{% endblock %}
```

edit_teacher.html

```
{% extends "base.html" %}

{% block title %}Редактирование информации о преподавателе{% endblock %}

{% block content %}
<link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='styles.css') }}">
<style>
        body {
            font-size: 20px;
            background-image: url('/static/fon_edit.jpg');
            background-size: cover;
            color: #FFDAB9;
            text-align: center;
        }

        h2 {
            font-size: 1.5em;
        }
</style>
    <h2>Редактирование информации о преподавателях</h2>
    <form action="" method="post">
        {{ form.hidden_tag() }}
        <p>
            {{ form.teacherFIO.label }}<br>
            {{ form.teacherFIO(size=32) }}<br>
            {% for error in form.teacherFIO.errors %}
            <span style="color: red;">[{{ error }}]</span>
            {% endfor %}
        </p>
        <p>
            {{ form.description.label }}<br>
            {{ form.description(cols=50, rows=20) }}<br>
            {% for error in form.description.errors %}
            <span style="color: red;">[{{ error }}]</span>
            {% endfor %}
        </p>
        <p>{{ form.submit() }}</p>
    </form>

    <a href="{{ url_for('view_teachers') }}">Назад к списку
преподавателей</a>

{% endblock %}
```

index.html

```
{% extends "base.html" %}

{% block title %}Главная{% endblock %}
```

```
{% block content %}
<link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='styles.css') }}">
<style>
        body {
            font-size: 20px;
            background-image: url('/static/fon.jpg');
            background-size: cover;
            color: #FFDAB9;

        }

        h1 {
            font-size: 2.5em; /* Измененный размер заголовка h1 */
            color: #FFFFFF;
            font-family: 'Arial', sans-serif; /* Шрифт */
            text-align: center; /* Выравнивание по центру */
        }
    </style>
{% if current_user.is_authenticated %}
    <p class="welcome">Добро пожаловать, {{current_user.username}}</p>

{% if current_user.is_administrator() %}

    <h2>Список пользователей:</h2>
    <ul>
        {% for user in users %}
        <li>{{ user[1] }}</li>
        <a href="{{url_for('addadmin',id=user[0])}}" >Сделать
администратором</a><br>
        {% endfor %}
    </ul>
{%endif%}
{% else %}
        <h1>Dance Studio</h1>{%endif%}
{% endblock %}
```

login.html

```
{% extends 'base.html' %}

{% block content %}
<link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='styles.css') }}">
<style>
        body {
            font-size: 20px;
            background-image: url('/static/fon_login.jpg');
            background-size: cover;
            color: white;
            text-align: center;
        }

        h2 {
            font-size: 2em;
        }

    </style>
  <h2>Вход в систему</h2>
  <form method="POST" action="{{ url_for('login') }}">
    {{ form.hidden_tag() }}
    <div>
      {{ form.username.label }}<br>
```

```
      {{ form.username(size=32) }}<br>
      {% for error in form.username.errors %}
        <p class="error">{{ error }}</p>
      {% endfor %}
    </div>
    <div>
      {{ form.password.label }}<br>
      {{ form.password(size=32) }}<br>
      {% for error in form.password.errors %}
        <p class="error">{{ error }}</p>
      {% endfor %}
    </div>
     <p>{{ form.remember_me() }} {{ form.remember_me.label }}</p>
    <div>
      {{ form.submit() }}
    </div>
  </form>
 <p>Новый пользователь? <a href="{{ url_for('registration')
}}">Регистрация</a></p>
{% endblock %}
```

registration.html

```
{% extends "base.html" %}

{% block title %}Регистрация{% endblock %}

{% block content %}
<link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='styles.css') }}">
<style>
        body {
            font-size: 20px;
            background-image: url('/static/fon_login.jpg');
            background-size: cover;
            color: white;
            text-align: center;
        }

        h2 {
            font-size: 2em;
        }

    </style>
  <h2>Регистрация</h2>
  <form method="POST" action="{{ url_for('registration') }}">
    {{ form.hidden_tag() }}
    <div>
      {{ form.username.label }}<br>
      {{ form.username(size=32) }}<br>
      {% for error in form.username.errors %}
        <p class="error">{{ error }}</p>
      {% endfor %}
    </div>
    <div>
      {{ form.password.label }}<br>
      {{ form.password(size=32) }}<br>
      {% for error in form.password.errors %}
        <p class="error">{{ error }}</p>
      {% endfor %}
    </div>
    <div>
      {{ form.password2.label }}<br>
      {{ form.password2(size=32) }}<br>
```

```
      {% for error in form.password2.errors %}
        <p class="error">{{ error }}</p>
      {% endfor %}
    </div>
    <div><br>
      {{ form.submit() }}
    </div>
  </form>
{% endblock %}
```

view_lessons.html

```
{% extends "base.html" %}

{% block title %}Занятия{% endblock %}

{% block content %}
    <link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='styles.css') }}">
    <style>
       body {
            background-image: url('/static/fon_styles.jpg');
            background-size: cover;
        }
    </style>
    <h1>Занятия</h1>
    <ul>
        {% for lesson in lessons %}
        <li>
            <span class="teacher-name">ФИО преподавателя: {{
lesson['teacher_name'] }}</span><br>
            <span class="style-name">Стиль занятия: {{ lesson['style_name']
}}</span><br>
            <span class="date">Дата: {{ lesson['date'] }}</span><br>
            <span class="time">Время: {{ lesson['time'] }}</span><br>

            {% if current_user.is_authenticated %}
                <form method="post" action="{{ url_for('view_lessons') }}">
                    {{ form.hidden_tag() }}
                    <input type="hidden" name="lesson_id" value="{{
lesson['id'] }}">
                    {{ form.submit() }}
                </form>
                {% if current_user.is_administrator() %}
                    <a href="{{url_for('editlesson',
id=lesson['id'])}}">Редактировать занятие</a>{% endif %}{% endif %}
            <h3>----------------------------------------------------------</h3>
        </li>

    {% endfor %}
    </ul>
{% endblock %}
```

view_styles.html

```
{% extends "base.html" %}

{% block title %}Стили{% endblock %}

{% block content %}
<link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='styles.css') }}">
<style>
        body {
```

```
            background-image: url('/static/fon_styles.jpg');
            background-size: cover;
        }

    </style>

    <h1>Стили</h1>
    <ul>
        {% for style in styles %}
        <h2>Название: {{ style[1] }}</h2>
        <p class="style-description">Описание: {{style[2]}}</p>
        {% if current_user.is_authenticated %}
        {% if current_user.is_administrator() %}
        <a href="{{url_for('editstyle',id=style[0])}}" >Редактировать
стиль</a><br>
        {%else%}
        {%endif%}
        {% endif %}
        <h3>-----------------------------------------------------</h3>
        {% endfor %}
    </ul>
{% endblock %}
```

view_styles_teacher.html

```
{% extends "base.html" %}

{% block content %}
<link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='styles.css') }}">
<style>
        body {
            background-image: url('/static/fon_styles.jpg');
            background-size: cover;
        }

    </style>
    <h1>Стили</h1>
    <h2>Список стилей преподавателя:</h2>
    <ul>
        {% for style in styles %}
            <li>{{ style[1] }}</li>
        {% endfor %}
    </ul>
{% if current_user.is_authenticated %}
{% if current_user.is_administrator() %}
        <a href="{{url_for('addstyleteacher',id=id)}}" >Добавить стиль
преподавателю</a><br>
        {%else%}
        {%endif%}
{%endif%}
{% endblock %}
```

view_teacher.html

```
{% extends "base.html" %}

{% block title %}Преподаватели{% endblock %}

{% block content %}
<link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='styles.css') }}">
<style>
        body {
```

```
            background-image: url('/static/fon_styles.jpg');
            background-size: cover;
        }

    </style>
{% if current_user.is_authenticated %}

{% else %}
{% endif %}
    <h1>Преподаватели</h1>
    <ul>
        {% for teacher in teachers %}
        <h2>ФИО: {{ teacher[1] }}</h2>
        <p class="teacher-description">Информация о себе:
{{teacher[2]}}</p><br>
        <a href="{{url_for('infoteacher',id=teacher[0])}}" >Информация о
стилях преподавателя</a> <br>
        {% if current_user.is_authenticated %}
        {% if current_user.is_administrator() %}
        <a href="{{url_for('addstyleteacher',id=teacher[0])}}" >Добавить
стиль преподавателю</a><br>
        <a href="{{url_for('editteacher',id=teacher[0])}}" >Редактировать
информацию о преподавателе</a><br>
        {%else%}
        {%endif%}
        {%endif%}
        <h3>-----------------------------------------------------</h3>
        {% endfor %}
    </ul>
{% endblock %}
```

Результат работы

При входе на сайт пользователь видит главную страницу приложения (рисунок 4)

Рисунок 4 – Главная страница

На рисунке 5 представлена страница регистрации.

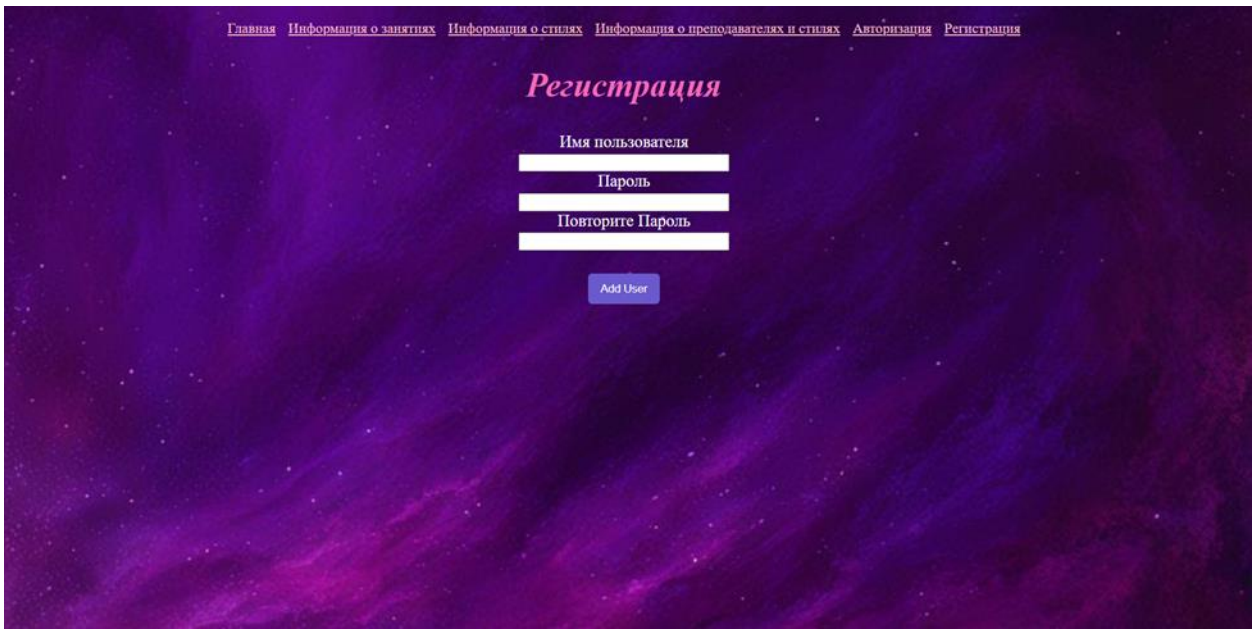

Рисунок 5 – Страница регистрации

Страница авторизации пользователя (рисунок 6) с нее можно перейти на страницу регистрации.

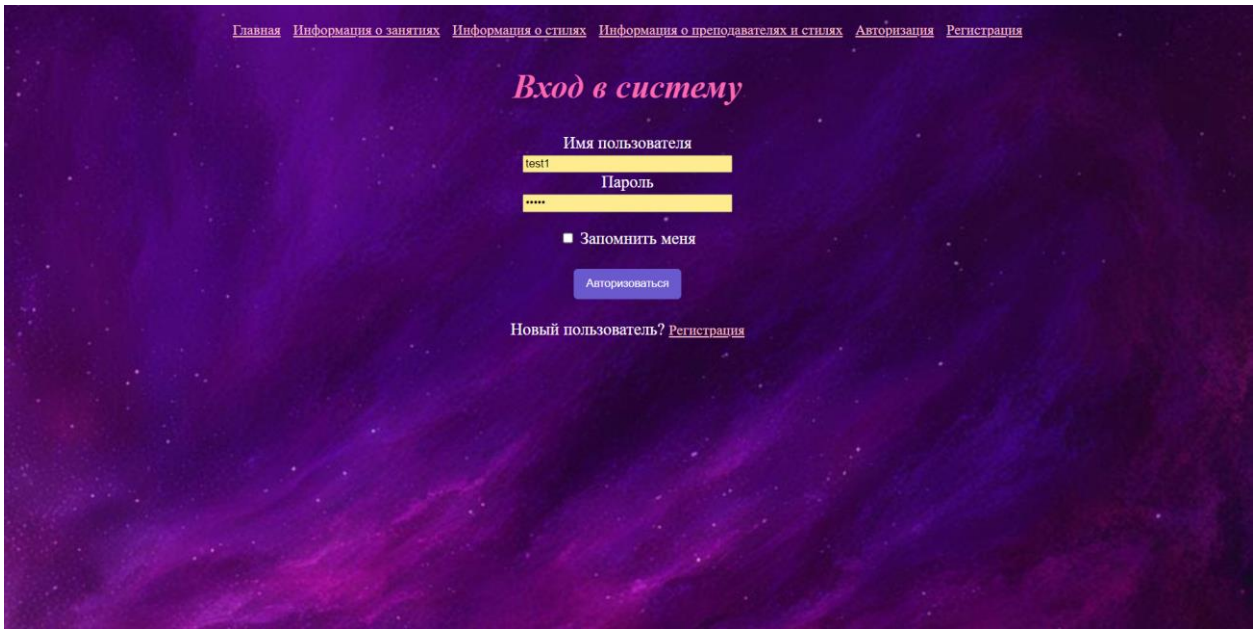

Рисунок 6 – Страница авторизации

Главная страница для авторизованного пользователя (рисунок 7)

Рисунок 7 – Главная страница после авторизации

Информация о занятиях для авторизованного пользователя (рисунок 8).

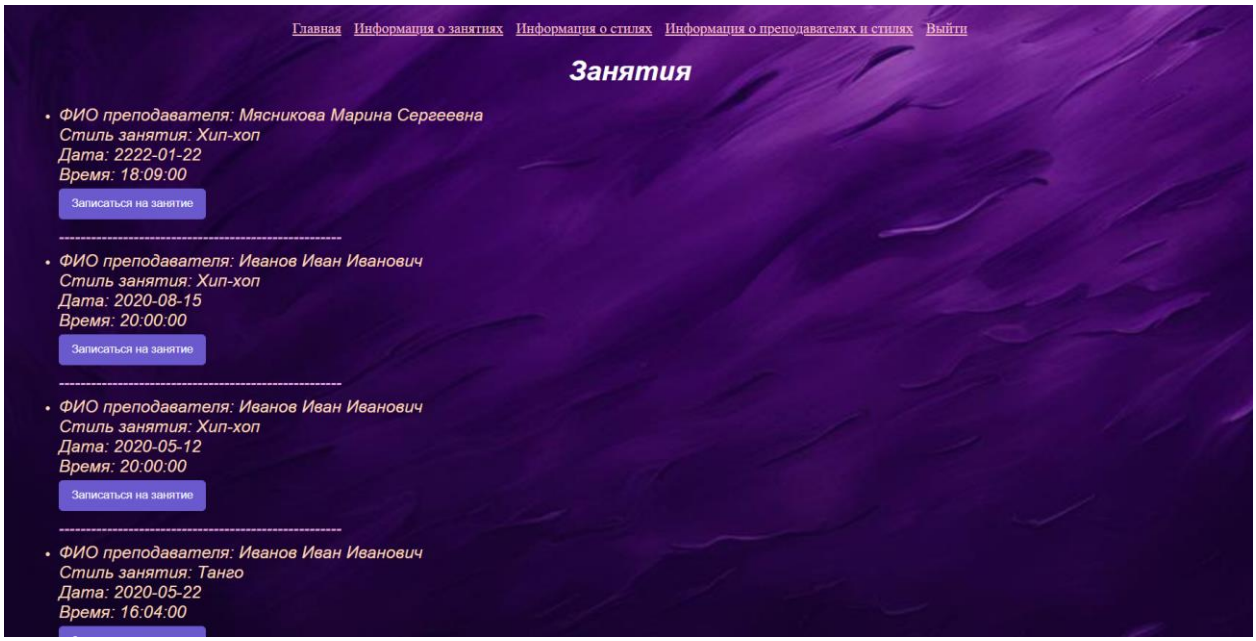

Рисунок 8 – Информация о занятиях для авторизованного пользователя

Страница после нажатия кнопки записаться на занятие (рисунок 9).

Рисунок 9 – Успешная запись на занятие

Рисунок 10 – Не успешная запись на занятие

Страница с информацией о стилях (рисунок 11).

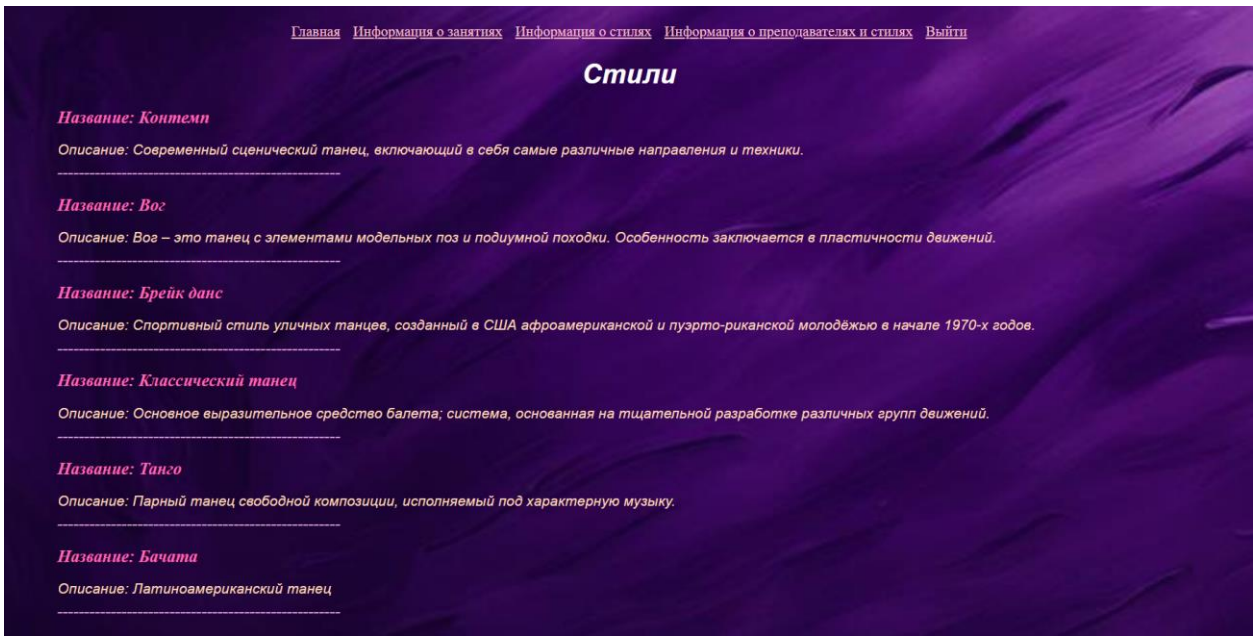

Рисунок 11 – Информация о стилях

На рисунке 12 представлена страница с информацией о преподавателях, также можно посмотреть стили, которыми владеет конкретный преподаватель (рисунок 13).

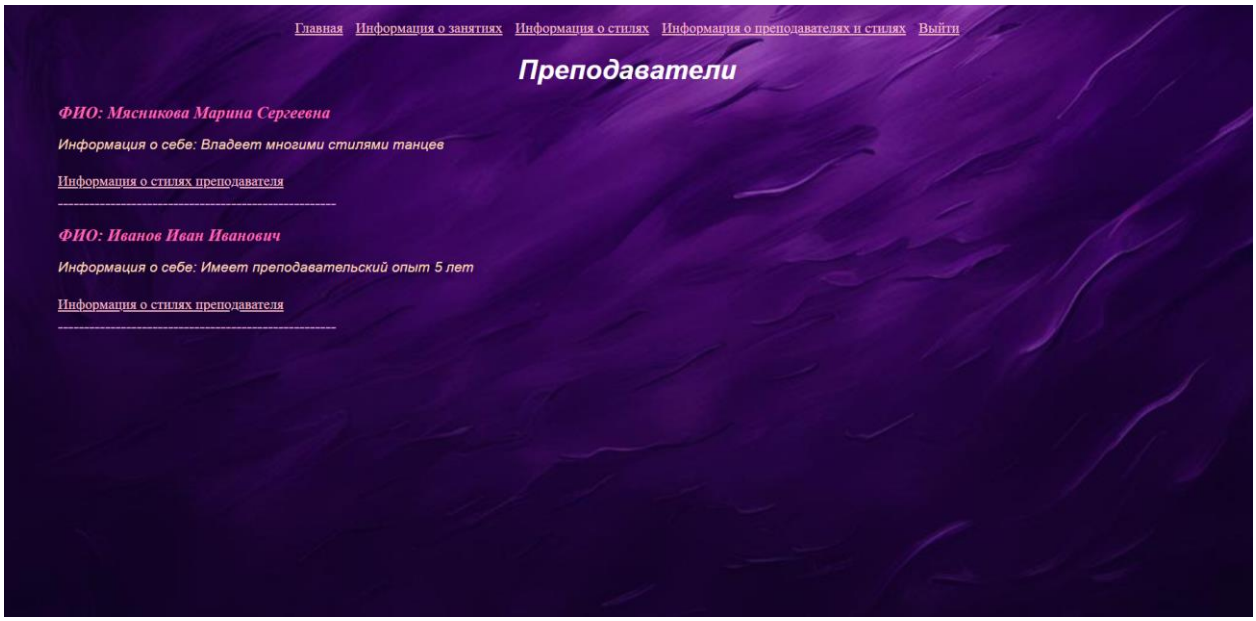

Рисунок 12 – Информация о преподавателях

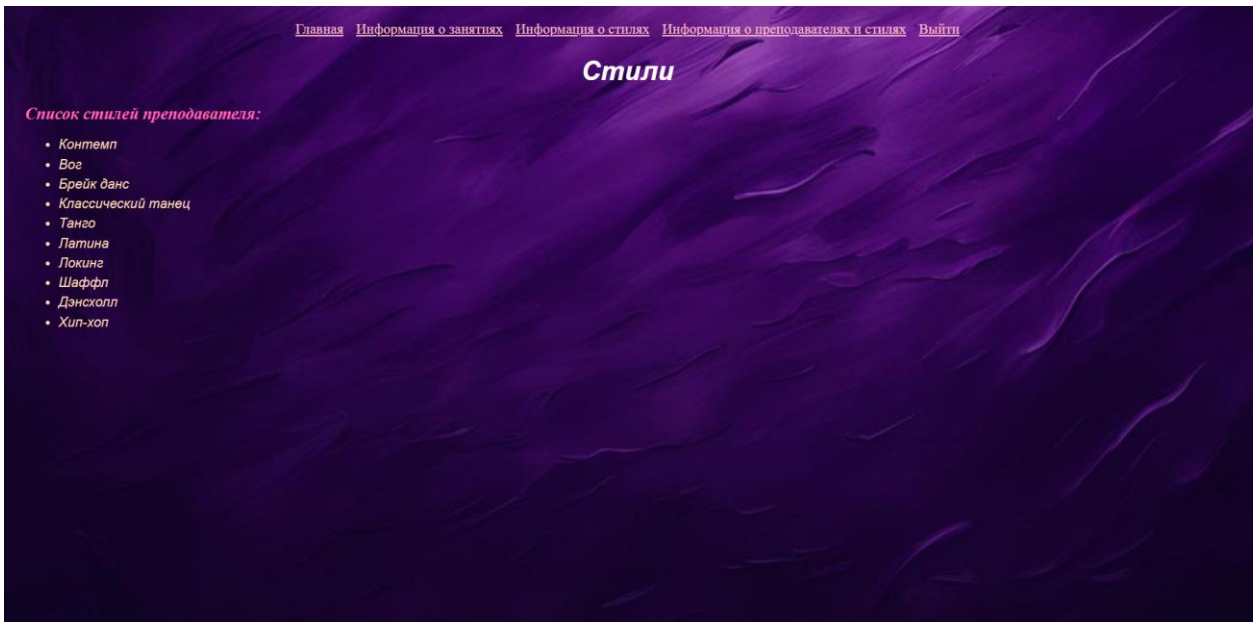

Рисунок 13 – Информация о стилях преподавателя

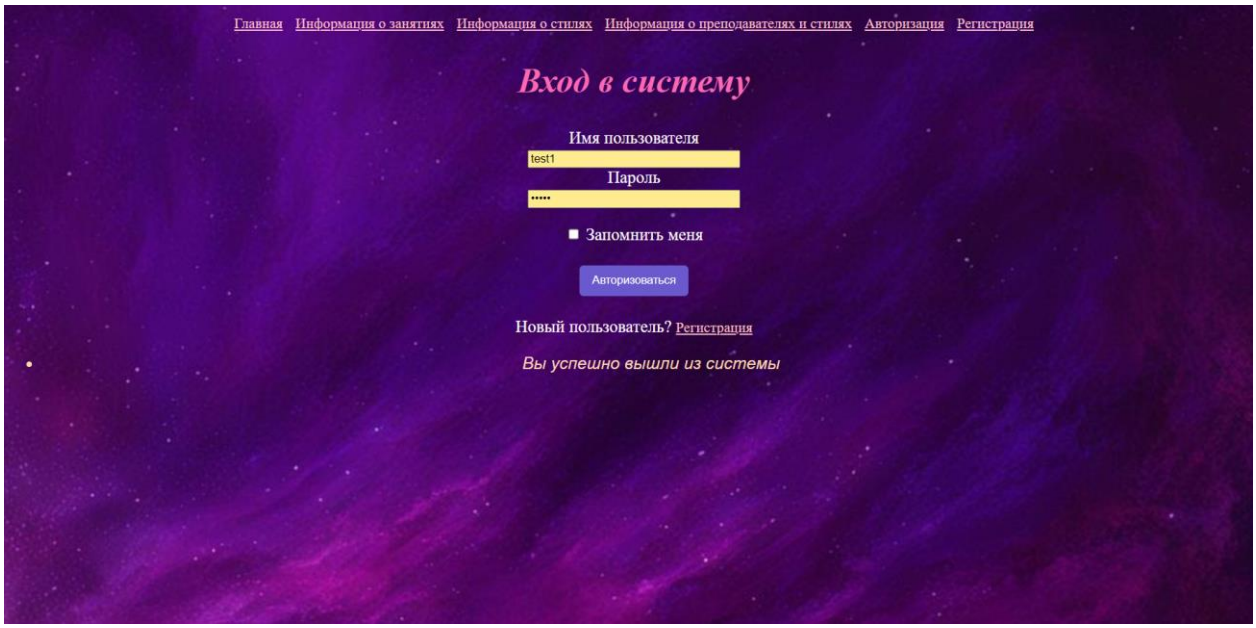

Рисунок 14 – Выход из системы

На рисунке 15 представлена главная страница администратора.

Рисунок 15 – Главная страница администратора

Администратору доступно редактирование занятий (рисунок 16, 17).

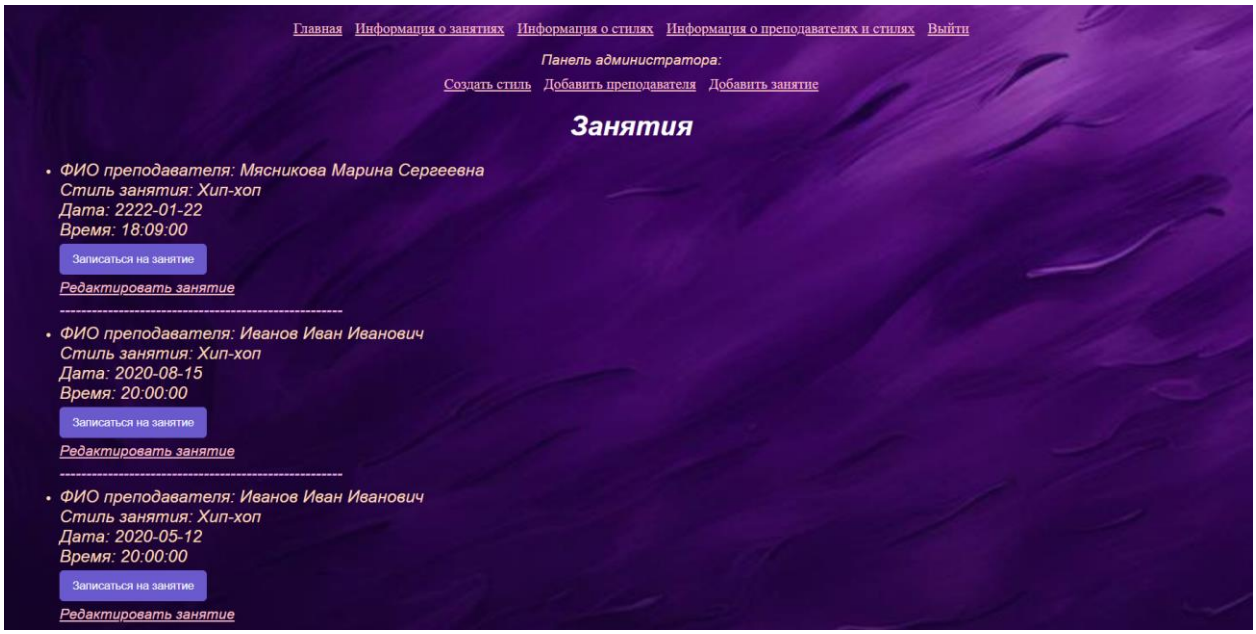

Рисунок 16 – Редактирование занятия

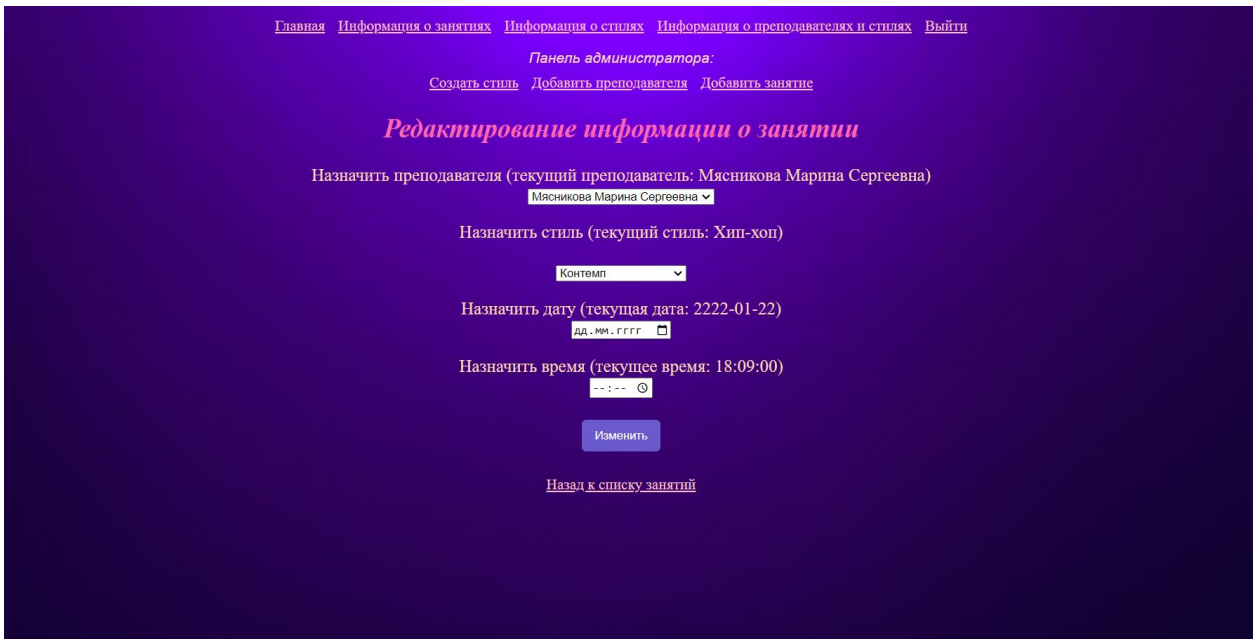

Рисунок 17 – Редактирование занятия

На странице о преподавателях (рисунок 18) администратор может просмотреть информацию как обычный пользователь, добавить стиль преподавателю (рисунок 19) и отредактировать информацию о преподавателе (рисунок 20).

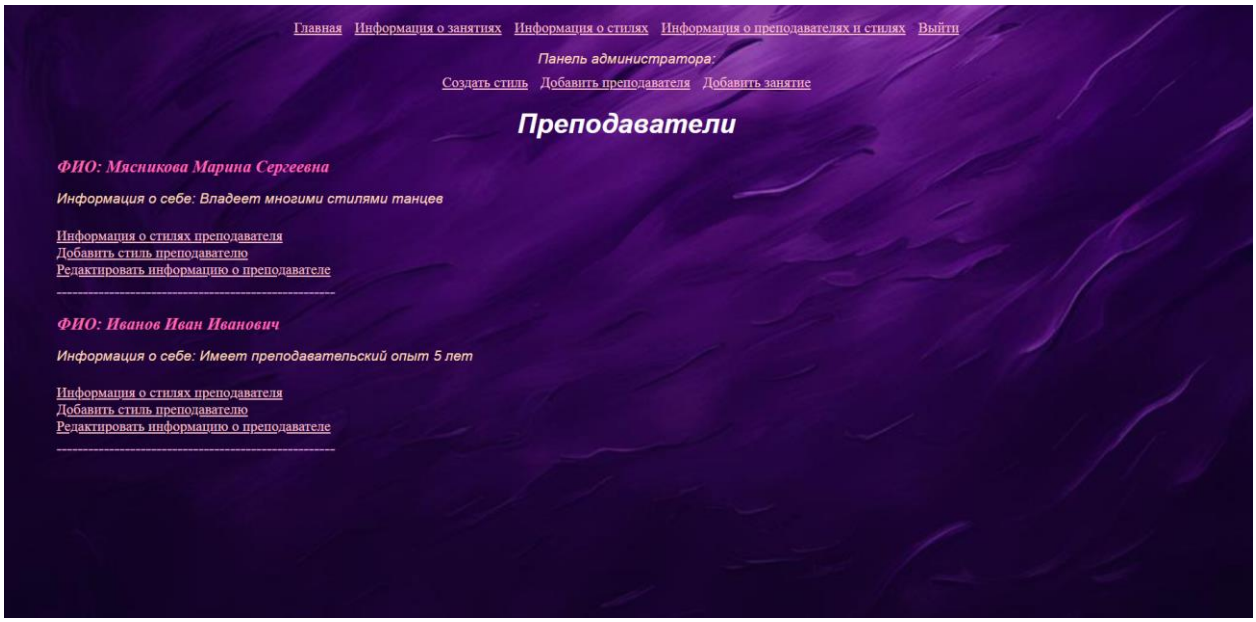

Рисунок 18 – Информация о преподавателях

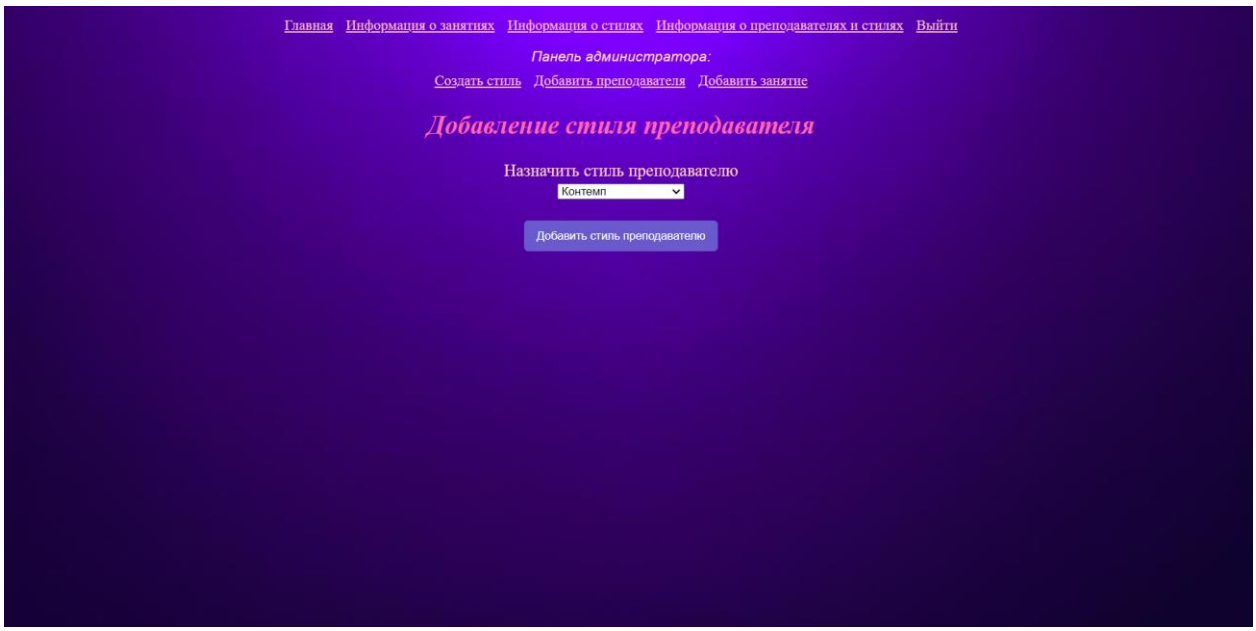

Рисунок 19 – Добавление стиля преподавателю

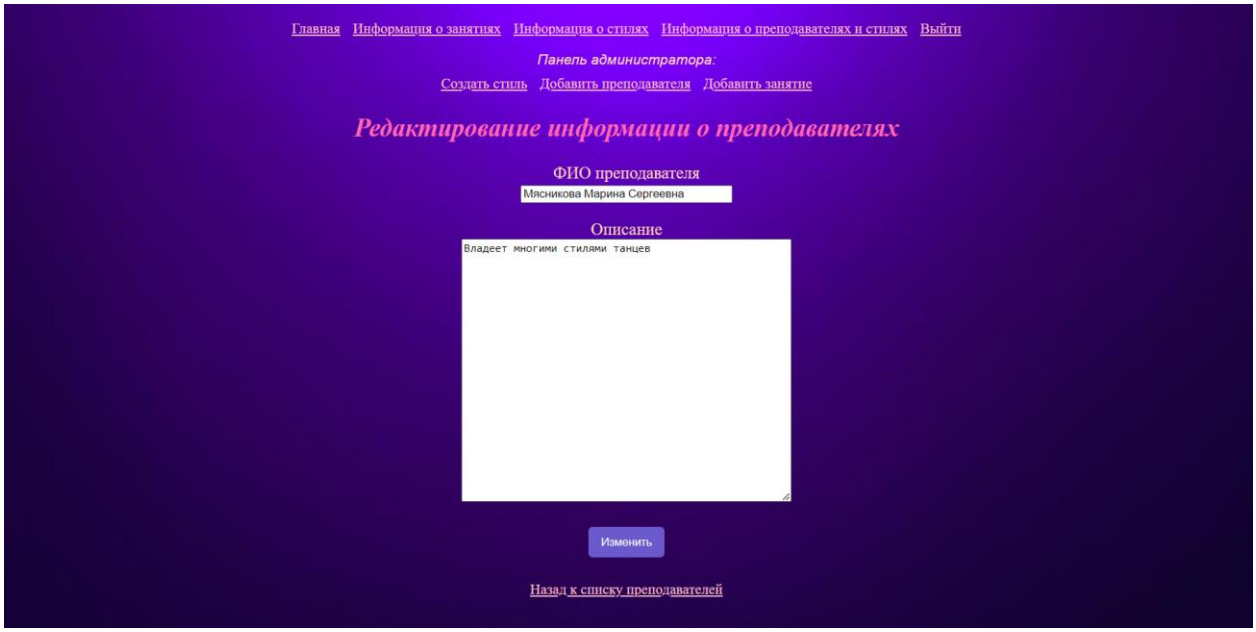

Рисунок 20 – Редактирование информации о преподавателе

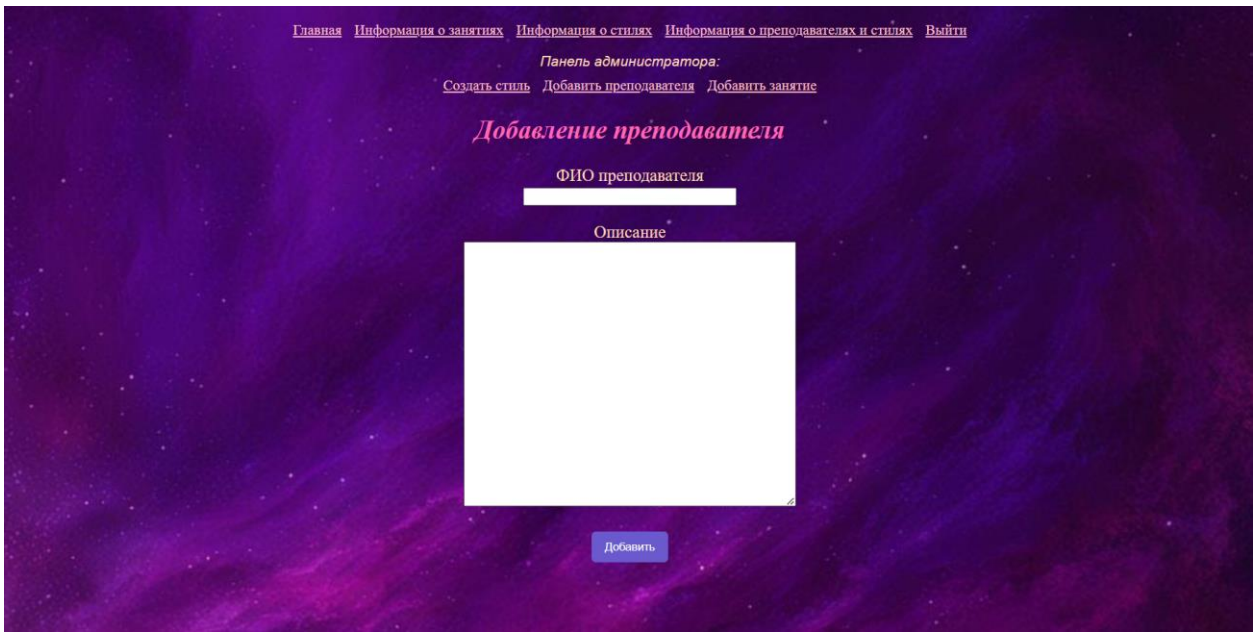

Рисунок 21 – Страница добавления преподавателя

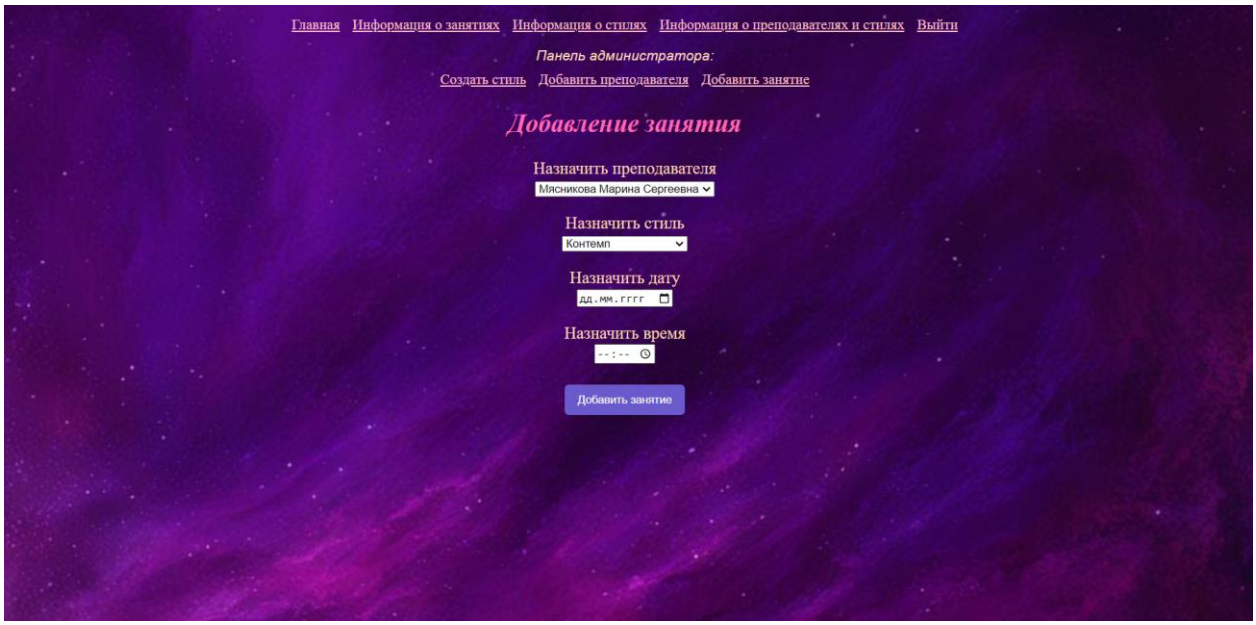

Рисунок 21 – Страница добавления занятия

Заключение

В работе описан процесс создания веб – приложения, представляющего из себя танцевальную студию. Веб – приложение написано на языке программирования Phyton с использованием фреймворка Flask и системы управления базой данных PostgreSQL. В ходе курсовой работы также были усовершенствованы навыки написания sql запросов.

Список используемых источников

1. https://habr.com/ru/companies/otus/articles/727590/

2. https://pythonru.com/biblioteki/vvedenie-v-postgresql-s-python-psycopg2
3. https://www.youtube.com/watch?v=Wd4BeJxFLbQ
4. https://www.youtube.com/watch?v=qVw9K9Jl7G8