**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Текстовое поле

Экзаменационная работа

**«Разработка веб-приложений»**

|  |  |
| --- | --- |
| Группа | 231-3211 |
| Студент | Егорова Ксения Валерьевна |

Москва 2025 г.

О структуре проекта:

Есть 4 файла типа “\*.py”:

app.py - главный модуль веб-приложения, включающий:

* маршрутизацию URL
* обработчики для работы с пользователями, книгами и отзывами
* систему модерации контента
* функцию генерации тестовых данных (30 книг, пользователей, жанров, ролей и статусов)

Код:

from flask import Flask

from config import Config

from models import db

from flask\_login import LoginManager

from flask import render\_template, redirect, url\_for, flash, request, abort, send\_from\_directory

from flask\_login import login\_user, logout\_user, login\_required, current\_user

from models import db, User, Role, Book, Genre, Cover, Review, ReviewStatus

from forms import LoginForm, BookForm, ReviewForm, RegisterForm

from werkzeug.security import check\_password\_hash, generate\_password\_hash

import os, hashlib, markdown, bleach

from sqlalchemy import func

from flask\_migrate import Migrate

from flask\_wtf import FlaskForm

from wtforms import StringField, SelectField, SubmitField, PasswordField

from wtforms.validators import DataRequired, Length

app = Flask(\_\_name\_\_)

app.config.from\_object(Config)

db.init\_app(app)

migrate = Migrate(app, db)

login\_manager = LoginManager(app)

login\_manager.login\_view = 'login'

@login\_manager.unauthorized\_handler

def custom\_unauthorized():

    flash('Для выполнения данного действия необходимо пройти процедуру аутентификации', 'error')

    return redirect(url\_for('login'))

def allowed\_file(filename):

    return '.' in filename and filename.rsplit('.', 1)[1].lower() in {'png', 'jpg', 'jpeg', 'gif'}

def md5\_for\_file(file):

    hash\_md5 = hashlib.md5()

    for chunk in iter(lambda: file.read(4096), b""):

        hash\_md5.update(chunk)

    file.seek(0)

    return hash\_md5.hexdigest()

def sanitize\_html(text):

    return bleach.clean(

        markdown.markdown(text),

        tags=list(bleach.sanitizer.ALLOWED\_TAGS) + ['p', 'pre', 'span'],

        attributes=bleach.sanitizer.ALLOWED\_ATTRIBUTES

    )

@login\_manager.user\_loader

def load\_user(user\_id):

    return db.session.get(User, int(user\_id))

@app.route('/')

def index():

    page = request.args.get('page', 1, type=int)

    search = request.args.get('search', '').strip()

    query = Book.query

    if search:

        query = query.filter(Book.title.ilike(f'%{search}%'))

    books = query.order\_by(Book.id.desc()).paginate(page=page, per\_page=10)

    for book in books.items:

        book.avg\_rating = db.session.query(func.avg(Review.rating)).filter(Review.book\_id==book.id, Review.status.has(name='approved')).scalar()

        book.reviews\_count = Review.query.filter\_by(book\_id=book.id, status\_id=ReviewStatus.query.filter\_by(name='approved').first().id).count()

    return render\_template('index.html', books=books, search=search)

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

def login():

    if current\_user.is\_authenticated:

        return redirect(url\_for('index'))

    form = LoginForm()

    if form.validate\_on\_submit():

        user = User.query.filter\_by(username=form.username.data).first()

        if user and check\_password\_hash(user.password\_hash, form.password.data):

            login\_user(user, remember=form.remember\_me.data)

            return redirect(url\_for('index'))

        flash('Невозможно аутентифицироваться с указанными логином и паролем', 'error')

    return render\_template('login.html', form=form)

@app.route('/logout')

@login\_required

def logout():

    logout\_user()

    return redirect(url\_for('index'))

@app.route('/register', methods=['GET', 'POST'])

def register():

    if current\_user.is\_authenticated:

        return redirect(url\_for('index'))

    form = RegisterForm()

    if form.validate\_on\_submit():

        if User.query.filter\_by(username=form.username.data).first():

            flash('Пользователь с таким логином уже существует', 'error')

        else:

            user\_role = Role.query.filter\_by(name='user').first()

            user = User(

                username=form.username.data,

                password\_hash=generate\_password\_hash(form.password.data),

                last\_name=form.last\_name.data,

                first\_name=form.first\_name.data,

                middle\_name=form.middle\_name.data,

                role\_id=user\_role.id

            )

            db.session.add(user)

            db.session.commit()

            flash('Регистрация успешна. Теперь вы можете войти.', 'success')

            return redirect(url\_for('login'))

    return render\_template('register.html', form=form)

@app.route('/book/add', methods=['GET', 'POST'])

@login\_required

def add\_book():

    if current\_user.role.name != 'admin':

        flash('У вас недостаточно прав для выполнения данного действия', 'error')

        return redirect(url\_for('index'))

    form = BookForm()

    form.genres.choices = [(g.id, g.name) for g in Genre.query.all()]

    if form.validate\_on\_submit():

        try:

            safe\_description = bleach.clean(form.description.data)

            book = Book(

                title=form.title.data,

                description=safe\_description,

                year=form.year.data,

                publisher=form.publisher.data,

                author=form.author.data,

                pages=form.pages.data

            )

            for genre\_id in form.genres.data:

                genre = db.session.get(Genre, genre\_id)

                if genre:

                    book.genres.append(genre)

            db.session.add(book)

            db.session.flush()

            file = form.cover.data

            if file and hasattr(file, "filename") and file.filename:

                if allowed\_file(file.filename):

                    md5 = md5\_for\_file(file)

                    cover = Cover.query.filter\_by(md5\_hash=md5).first()

                    if cover:

                        book.cover\_id = cover.id

                    else:

                        cover = Cover(

                            filename='',

                            mime\_type=file.mimetype,

                            md5\_hash=md5

                        )

                        db.session.add(cover)

                        db.session.flush()

                        filename = f"{cover.id}.jpg"

                        file.save(os.path.join(app.config['UPLOAD\_FOLDER'], filename))

                        cover.filename = filename

                        db.session.commit()

                        book.cover\_id = cover.id

            db.session.commit()

            flash('Книга успешно добавлена', 'success')

            return redirect(url\_for('index'))

        except Exception as e:

            db.session.rollback()

            import traceback

            print('Ошибка при добавлении книги:', e)

            traceback.print\_exc()

            flash(f'При сохранении данных возникла ошибка: {e}. Проверьте корректность введённых данных.', 'error')

    return render\_template('book\_form.html', form=form, book=None)

@app.route('/book/<int:book\_id>/edit', methods=['GET', 'POST'])

@login\_required

def edit\_book(book\_id):

    book = db.session.get(Book, book\_id)

    if not book:

        abort(404)

    if current\_user.role.name not in ['admin', 'moderator']:

        flash('У вас недостаточно прав для выполнения данного действия', 'error')

        return redirect(url\_for('index'))

    form = BookForm(obj=book)

    form.genres.choices = [(g.id, g.name) for g in Genre.query.all()]

    if request.method == 'GET':

        form.genres.data = [g.id for g in book.genres]

    if form.validate\_on\_submit():

        try:

            book.title = form.title.data

            book.description = bleach.clean(form.description.data)

            book.year = form.year.data

            book.publisher = form.publisher.data

            book.author = form.author.data

            book.pages = form.pages.data

            book.genres = [db.session.get(Genre, gid) for gid in form.genres.data]

            file = form.cover.data

            if file and hasattr(file, "read") and hasattr(file, "filename") and file.filename:

                if allowed\_file(file.filename):

                    md5 = md5\_for\_file(file)

                    cover = Cover.query.filter\_by(md5\_hash=md5).first()

                    if cover:

                        book.cover\_id = cover.id

                    else:

                        if book.cover:

                            old\_cover = book.cover

                            if Book.query.filter(Book.cover\_id == old\_cover.id, Book.id != book.id).count() == 0:

                                old\_path = os.path.join(app.config['UPLOAD\_FOLDER'], old\_cover.filename)

                                if os.path.exists(old\_path):

                                    os.remove(old\_path)

                                db.session.delete(old\_cover)

                                db.session.flush()

                        cover = Cover(filename='', mime\_type=file.mimetype, md5\_hash=md5)

                        db.session.add(cover)

                        db.session.flush()

                        filename = f"{cover.id}.jpg"

                        file.save(os.path.join(app.config['UPLOAD\_FOLDER'], filename))

                        cover.filename = filename

                        db.session.commit()

                        book.cover\_id = cover.id

            db.session.commit()

            flash('Книга успешно обновлена', 'success')

            return redirect(url\_for('book\_view', book\_id=book.id))

        except Exception as e:

            db.session.rollback()

            import traceback

            print('Ошибка при обновлении книги:', e)

            traceback.print\_exc()

            flash(f'Ошибка при обновлении книги: {e}', 'error')

    return render\_template('book\_form.html', form=form, book=book)

@app.route('/book/<int:book\_id>')

def book\_view(book\_id):

    book = Book.query.get\_or\_404(book\_id)

    book.description\_html = sanitize\_html(book.description)

    reviews = Review.query.filter\_by(book\_id=book.id).join(ReviewStatus).filter(ReviewStatus.name=='approved').order\_by(Review.created\_at.desc()).all()

    for r in reviews:

        r.text\_html = sanitize\_html(r.text)

    can\_review = False

    if current\_user.is\_authenticated and current\_user.role.name in ['user', 'moderator', 'admin']:

        exists = Review.query.filter\_by(book\_id=book.id, user\_id=current\_user.id).first()

        if not exists:

            can\_review = True

    return render\_template('book\_view.html', book=book, reviews=reviews, can\_review=can\_review)

@app.route('/book/<int:book\_id>/delete')

@login\_required

def delete\_book(book\_id):

    if current\_user.role.name != 'admin':

        flash('У вас недостаточно прав для выполнения данного действия', 'error')

        return redirect(url\_for('index'))

    book = Book.query.get\_or\_404(book\_id)

    try:

        if book.cover:

            cover\_path = os.path.join(app.config['UPLOAD\_FOLDER'], book.cover.filename)

            if os.path.exists(cover\_path):

                os.remove(cover\_path)

        db.session.delete(book)

        db.session.commit()

        flash('Книга удалена', 'success')

    except Exception as e:

        db.session.rollback()

        flash('Ошибка при удалении книги', 'error')

    return redirect(url\_for('index'))

@app.route('/book/<int:book\_id>/review', methods=['GET', 'POST'])

@login\_required

def add\_review(book\_id):

    book = Book.query.get\_or\_404(book\_id)

    if Review.query.filter\_by(book\_id=book.id, user\_id=current\_user.id).first():

        flash('Вы уже оставляли рецензию на эту книгу', 'error')

        return redirect(url\_for('book\_view', book\_id=book.id))

    form = ReviewForm()

    if form.validate\_on\_submit():

        try:

            status = ReviewStatus.query.filter\_by(name='pending').first()

            review = Review(

                book\_id=book.id,

                user\_id=current\_user.id,

                rating=form.rating.data,

                text=form.text.data,

                status\_id=status.id

            )

            db.session.add(review)

            db.session.commit()

            flash('Рецензия отправлена на модерацию', 'success')

            return redirect(url\_for('book\_view', book\_id=book.id))

        except Exception as e:

            db.session.rollback()

            flash('Ошибка при добавлении рецензии', 'error')

    return render\_template('review\_form.html', form=form)

@app.route('/my-reviews')

@login\_required

def my\_reviews():

    reviews = Review.query.filter\_by(user\_id=current\_user.id).order\_by(Review.created\_at.desc()).all()

    for r in reviews:

        r.text\_html = sanitize\_html(r.text)

    return render\_template('my\_reviews.html', reviews=reviews)

@app.route('/moderate')

@login\_required

def moderate():

    if current\_user.role.name != 'moderator':

        flash('У вас недостаточно прав для выполнения данного действия', 'error')

        return redirect(url\_for('index'))

    page = request.args.get('page', 1, type=int)

    status = ReviewStatus.query.filter\_by(name='pending').first()

    reviews = Review.query.filter\_by(status\_id=status.id).order\_by(Review.created\_at).paginate(page=page, per\_page=10)

    for r in reviews.items:

        r.text\_html = sanitize\_html(r.text)

    return render\_template('moderate.html', reviews=reviews)

@app.route('/moderate/<int:review\_id>', methods=['GET', 'POST'])

@login\_required

def moderate\_review(review\_id):

    if current\_user.role.name != 'moderator':

        flash('У вас недостаточно прав для выполнения данного действия', 'error')

        return redirect(url\_for('index'))

    review = Review.query.get\_or\_404(review\_id)

    review.text\_html = sanitize\_html(review.text)

    if request.method == 'POST':

        action = request.form.get('action')

        try:

            if action == 'approve':

                review.status\_id = ReviewStatus.query.filter\_by(name='approved').first().id

            elif action == 'reject':

                review.status\_id = ReviewStatus.query.filter\_by(name='rejected').first().id

            db.session.commit()

            flash('Статус рецензии обновлён', 'success')

            return redirect(url\_for('moderate'))

        except Exception as e:

            db.session.rollback()

            flash('Ошибка при обновлении статуса', 'error')

    return render\_template('moderate\_review.html', review=review)

class UserEditForm(FlaskForm):

    last\_name = StringField('Фамилия', validators=[DataRequired(), Length(max=64)])

    first\_name = StringField('Имя', validators=[DataRequired(), Length(max=64)])

    middle\_name = StringField('Отчество', validators=[Length(max=64)])

    role\_id = SelectField('Роль', coerce=int)

    password = PasswordField('Новый пароль')

    submit = SubmitField('Сохранить')

class UserAddForm(FlaskForm):

    username = StringField('Логин', validators=[DataRequired(), Length(max=64)])

    password = PasswordField('Пароль', validators=[DataRequired(), Length(min=4)])

    last\_name = StringField('Фамилия', validators=[DataRequired(), Length(max=64)])

    first\_name = StringField('Имя', validators=[DataRequired(), Length(max=64)])

    middle\_name = StringField('Отчество', validators=[Length(max=64)])

    role\_id = SelectField('Роль', coerce=int)

    submit = SubmitField('Добавить')

@app.route('/users')

@login\_required

def users():

    if current\_user.role.name != 'admin':

        abort(403)

    users = User.query.order\_by(User.id).all()

    return render\_template('users.html', users=users)

@app.route('/users/add', methods=['GET', 'POST'])

@login\_required

def add\_user():

    if current\_user.role.name != 'admin':

        abort(403)

    form = UserAddForm()

    form.role\_id.choices = [(role.id, role.name) for role in Role.query.all()]

    if form.validate\_on\_submit():

        if User.query.filter\_by(username=form.username.data).first():

            flash('Пользователь с таким логином уже существует', 'error')

        else:

            user = User(

                username=form.username.data,

                password\_hash=generate\_password\_hash(form.password.data),

                last\_name=form.last\_name.data,

                first\_name=form.first\_name.data,

                middle\_name=form.middle\_name.data,

                role\_id=form.role\_id.data

            )

            db.session.add(user)

            db.session.commit()

            flash('Пользователь добавлен', 'success')

            return redirect(url\_for('users'))

    return render\_template('user\_add.html', form=form)

@app.route('/users/<int:user\_id>/edit', methods=['GET', 'POST'])

@login\_required

def edit\_user(user\_id):

    if current\_user.role.name != 'admin':

        abort(403)

    user = User.query.get\_or\_404(user\_id)

    form = UserEditForm(obj=user)

    form.role\_id.choices = [(role.id, role.name) for role in Role.query.all()]

    if form.validate\_on\_submit():

        user.last\_name = form.last\_name.data

        user.first\_name = form.first\_name.data

        user.middle\_name = form.middle\_name.data

        user.role\_id = form.role\_id.data

        if form.password.data:

            user.password\_hash = generate\_password\_hash(form.password.data)

        db.session.commit()

        flash('Пользователь обновлён', 'success')

        return redirect(url\_for('users'))

    return render\_template('user\_edit.html', form=form, user=user)

@app.route('/users/<int:user\_id>/delete', methods=['POST'])

@login\_required

def delete\_user(user\_id):

    if current\_user.role.name != 'admin':

        abort(403)

    user = User.query.get\_or\_404(user\_id)

    if user.id == current\_user.id:

        flash('Нельзя удалить самого себя.', 'error')

        return redirect(url\_for('users'))

    db.session.delete(user)

    db.session.commit()

    flash('Пользователь удалён', 'success')

    return redirect(url\_for('users'))

@app.errorhandler(401)

def unauthorized(e):

    flash('Для выполнения данного действия необходимо пройти процедуру аутентификации', 'error')

    return redirect(url\_for('login'))

@app.errorhandler(403)

def forbidden(e):

    flash('У вас недостаточно прав для выполнения данного действия', 'error')

    return redirect(url\_for('index'))

@app.route('/all-reviews')

@login\_required

def all\_reviews():

    if current\_user.role.name != 'admin':

        abort(403)

    reviews = Review.query.order\_by(Review.created\_at.desc()).all()

    for r in reviews:

        r.text\_html = sanitize\_html(r.text)

    return render\_template('all\_reviews.html', reviews=reviews)

if \_\_name\_\_ == '\_\_main\_\_':

    with app.app\_context():

        db.create\_all()

        from werkzeug.security import generate\_password\_hash

        from models import User, Role, ReviewStatus, Genre

        roles\_data = [

            ('admin', 'Администратор: полный доступ'),

            ('moderator', 'Модератор: редактирование книг, модерация рецензий'),

            ('user', 'Пользователь: может оставлять рецензии')

        ]

        for name, desc in roles\_data:

            if not Role.query.filter\_by(name=name).first():

                db.session.add(Role(name=name, description=desc))

        db.session.commit()

        users\_data = [

            ('admin', 'admin', 'Админ', 'Админ', '', 'admin'),

            ('moder', 'moder', 'Модер', 'Модератор', '', 'moderator'),

            ('user', 'user', 'Пользователь', 'Обычный', '', 'user')

        ]

        for username, password, last\_name, first\_name, middle\_name, role\_name in users\_data:

            role = Role.query.filter\_by(name=role\_name).first()

            if role and not User.query.filter\_by(username=username).first():

                user = User(

                    username=username,

                    password\_hash=generate\_password\_hash(password),

                    last\_name=last\_name,

                    first\_name=first\_name,

                    middle\_name=middle\_name,

                    role\_id=role.id

                )

                db.session.add(user)

        db.session.commit()

        for status\_name in ['pending', 'approved', 'rejected']:

            if not ReviewStatus.query.filter\_by(name=status\_name).first():

                db.session.add(ReviewStatus(name=status\_name))

        db.session.commit()

        default\_genres = [

            'Фантастика', 'Детектив', 'Роман', 'Поэзия', 'Научная литература',

            'Приключения', 'Фэнтези', 'История', 'Биография', 'Драма',

            'Триллер', 'Ужасы', 'Комедия', 'Психология', 'Детская литература',

            'Энциклопедия', 'Публицистика', 'Мемуары', 'Любовный роман', 'Саморазвитие'

        ]

        for genre\_name in default\_genres:

            if not Genre.query.filter\_by(name=genre\_name).first():

                db.session.add(Genre(name=genre\_name))

        db.session.commit()

    app.run(debug=True)

models.py - файл с определениями моделей базы данных:

* Книги
* Пользователи
* Жанры
* Обложки книг
* Рецензии
* Статусы публикаций

Код:

from flask\_sqlalchemy import SQLAlchemy

from flask\_login import UserMixin

db = SQLAlchemy()

class Role(db.Model):

    \_\_tablename\_\_ = 'roles'

    id: int = db.Column(db.Integer, primary\_key=True)

    name: str = db.Column(db.String(64), unique=True, nullable=False)

    description: str = db.Column(db.Text, nullable=False)

    users = db.relationship('User', backref='role', lazy=True)

class User(UserMixin, db.Model):

    \_\_tablename\_\_ = 'users'

    id: int = db.Column(db.Integer, primary\_key=True)

    username: str = db.Column(db.String(64), unique=True, nullable=False)

    password\_hash: str = db.Column(db.String(128), nullable=False)

    last\_name: str = db.Column(db.String(64), nullable=False)

    first\_name: str = db.Column(db.String(64), nullable=False)

    middle\_name: str = db.Column(db.String(64))

    role\_id: int = db.Column(db.Integer, db.ForeignKey('roles.id'), nullable=False)

    reviews = db.relationship('Review', backref='user', lazy=True)

class Genre(db.Model):

    \_\_tablename\_\_ = 'genres'

    id: int = db.Column(db.Integer, primary\_key=True)

    name: str = db.Column(db.String(64), unique=True, nullable=False)

    books = db.relationship('Book', secondary='books\_genres', back\_populates='genres')

class Book(db.Model):

    \_\_tablename\_\_ = 'books'

    id: int = db.Column(db.Integer, primary\_key=True)

    title: str = db.Column(db.String(255), nullable=False)

    description: str = db.Column(db.Text, nullable=False)

    year: int = db.Column(db.Integer, nullable=False)

    publisher: str = db.Column(db.String(128), nullable=False)

    author: str = db.Column(db.String(128), nullable=False)

    pages: int = db.Column(db.Integer, nullable=False)

    genres = db.relationship('Genre', secondary='books\_genres', back\_populates='books')

    cover\_id: int = db.Column(db.Integer, db.ForeignKey('covers.id', ondelete='SET NULL'))

    cover = db.relationship('Cover', backref='books', foreign\_keys=[cover\_id])

    reviews = db.relationship('Review', backref='book', lazy=True, cascade="all, delete-orphan")

class BooksGenres(db.Model):

    \_\_tablename\_\_ = 'books\_genres'

    book\_id: int = db.Column(db.Integer, db.ForeignKey('books.id', ondelete='CASCADE'), primary\_key=True)

    genre\_id: int = db.Column(db.Integer, db.ForeignKey('genres.id', ondelete='CASCADE'), primary\_key=True)

class Cover(db.Model):

    \_\_tablename\_\_ = 'covers'

    id: int = db.Column(db.Integer, primary\_key=True)

    filename: str = db.Column(db.String(255), nullable=False)

    mime\_type: str = db.Column(db.String(64), nullable=False)

    md5\_hash: str = db.Column(db.String(32), nullable=False, unique=True)

class ReviewStatus(db.Model):

    \_\_tablename\_\_ = 'review\_statuses'

    id: int = db.Column(db.Integer, primary\_key=True)

    name: str = db.Column(db.String(32), unique=True, nullable=False)

    reviews = db.relationship('Review', backref='status', lazy=True)

class Review(db.Model):

    \_\_tablename\_\_ = 'reviews'

    id: int = db.Column(db.Integer, primary\_key=True)

    book\_id: int = db.Column(db.Integer, db.ForeignKey('books.id', ondelete='CASCADE'), nullable=False)

    user\_id: int = db.Column(db.Integer, db.ForeignKey('users.id', ondelete='CASCADE'), nullable=False)

    rating: int = db.Column(db.Integer, nullable=False)

    text: str = db.Column(db.Text, nullable=False)

    created\_at = db.Column(db.DateTime, server\_default=db.func.now(), nullable=False)

    status\_id: int = db.Column(db.Integer, db.ForeignKey('review\_statuses.id'), nullable=False)

    \_\_table\_args\_\_ = (db.UniqueConstraint('book\_id', 'user\_id', name='\_book\_user\_uc'),)

forms.py - содержит классы форм для:

* авторизации и регистрации пользователей
* добавления новых книг
* создания рецензий

Код:

from flask\_wtf import FlaskForm

from wtforms import StringField, PasswordField, SubmitField, TextAreaField, IntegerField, SelectMultipleField, SelectField, FileField, BooleanField

from wtforms.validators import DataRequired, Length, NumberRange

class LoginForm(FlaskForm):

    username = StringField('Логин', validators=[DataRequired()])

    password = PasswordField('Пароль', validators=[DataRequired()])

    remember\_me = BooleanField('Запомнить меня')

    submit = SubmitField('Войти')

class BookForm(FlaskForm):

    title = StringField('Название', validators=[DataRequired(), Length(max=255)])

    description = TextAreaField('Описание', validators=[DataRequired()])

    year = IntegerField('Год', validators=[DataRequired(), NumberRange(min=1000, max=2100)])

    publisher = StringField('Издательство', validators=[DataRequired(), Length(max=128)])

    author = StringField('Автор', validators=[DataRequired(), Length(max=128)])

    pages = IntegerField('Страниц', validators=[DataRequired(), NumberRange(min=1)])

    genres = SelectMultipleField('Жанры', coerce=int)

    cover = FileField('Обложка')

    submit = SubmitField('Сохранить')

class ReviewForm(FlaskForm):

    rating = SelectField('Оценка', choices=[(5, 'отлично'), (4, 'хорошо'), (3, 'удовлетворительно'), (2, 'неудовлетворительно'), (1, 'плохо'), (0, 'ужасно')], coerce=int, default=5)

    text = TextAreaField('Текст рецензии', validators=[DataRequired()])

    submit = SubmitField('Сохранить')

class RegisterForm(FlaskForm):

    username = StringField('Логин', validators=[DataRequired(), Length(max=64)])

    password = PasswordField('Пароль', validators=[DataRequired(), Length(min=4)])

    last\_name = StringField('Фамилия', validators=[DataRequired(), Length(max=64)])

    first\_name = StringField('Имя', validators=[DataRequired(), Length(max=64)])

    middle\_name = StringField('Отчество', validators=[Length(max=64)])

    submit = SubmitField('Зарегистрироваться')

config.py - файл конфигурации с настройками:

* секретный ключ приложения
* путь к файлу базы данных
* директория для хранения обложек книг

Код:

import os

class Config:

    SECRET\_KEY = 'your\_secret\_key'

    SQLALCHEMY\_DATABASE\_URI = 'sqlite:///exam.db'

    SQLALCHEMY\_TRACK\_MODIFICATIONS = False

    UPLOAD\_FOLDER = os.path.join(os.path.abspath(os.path.dirname(\_\_file\_\_)), 'static', 'covers')

**db\_schema.sql** — SQL-скрипт для инициализации структуры базы данных MySQL, включающий:

* Создание таблиц (книги, пользователи, жанры, обложки, рецензии, статусы)
* Определение первичных и внешних ключей
* Настройку индексов для оптимизации запросов
* Заполнение справочных данных (роли, статусы модерации)

Код:

CREATE TABLE roles (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    name VARCHAR(64) NOT NULL UNIQUE,

    description TEXT NOT NULL

);

CREATE TABLE users (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    username VARCHAR(64) NOT NULL UNIQUE,

    password\_hash VARCHAR(128) NOT NULL,

    last\_name VARCHAR(64) NOT NULL,

    first\_name VARCHAR(64) NOT NULL,

    middle\_name VARCHAR(64),

    role\_id INT NOT NULL,

    FOREIGN KEY (role\_id) REFERENCES roles(id)

);

CREATE TABLE genres (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    name VARCHAR(64) NOT NULL UNIQUE

);

CREATE TABLE books (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    title VARCHAR(255) NOT NULL,

    description TEXT NOT NULL,

    year YEAR NOT NULL,

    publisher VARCHAR(128) NOT NULL,

    author VARCHAR(128) NOT NULL,

    pages INT NOT NULL,

    cover\_id INT,

    FOREIGN KEY (cover\_id) REFERENCES covers(id) ON DELETE SET NULL

);

CREATE TABLE books\_genres (

    book\_id INT NOT NULL,

    genre\_id INT NOT NULL,

    PRIMARY KEY (book\_id, genre\_id),

    FOREIGN KEY (book\_id) REFERENCES books(id) ON DELETE CASCADE,

    FOREIGN KEY (genre\_id) REFERENCES genres(id) ON DELETE CASCADE

);

CREATE TABLE covers (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    filename VARCHAR(255) NOT NULL,

    mime\_type VARCHAR(64) NOT NULL,

    md5\_hash VARCHAR(32) NOT NULL UNIQUE

);

CREATE TABLE review\_statuses (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    name VARCHAR(32) NOT NULL UNIQUE

);

CREATE TABLE reviews (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    book\_id INT NOT NULL,

    user\_id INT NOT NULL,

    rating INT NOT NULL,

    text TEXT NOT NULL,

    created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP NOT NULL,

    status\_id INT NOT NULL,

    FOREIGN KEY (book\_id) REFERENCES books(id) ON DELETE CASCADE,

    FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE,

    FOREIGN KEY (status\_id) REFERENCES review\_statuses(id),

    UNIQUE (book\_id, user\_id)

);

-- Заполнение ролей

INSERT INTO roles (name, description) VALUES

('admin', 'Администратор: полный доступ'),

('moderator', 'Модератор: редактирование книг, модерация рецензий'),

('user', 'Пользователь: может оставлять рецензии');

-- Заполнение статусов рецензий

INSERT INTO review\_statuses (name) VALUES

('pending'),

('approved'),

('rejected');

-- Заполнение жанров

INSERT INTO genres (name) VALUES

('Фантастика'),

('Детектив'),

('Роман'),

('Поэзия'),

('Научная литература'),

('Приключения'),

('Фэнтези'),

('История'),

('Биография'),

('Драма'),

('Триллер'),

('Ужасы'),

('Комедия'),

('Психология'),

('Детская литература'),

('Энциклопедия'),

('Публицистика'),

('Мемуары'),

('Любовный роман'),

('Саморазвитие');

ALTER TABLE books ADD COLUMN cover\_id INTEGER;

ALTER TABLE books ADD CONSTRAINT fk\_books\_cover\_id\_covers FOREIGN KEY (cover\_id) REFERENCES covers(id) ON DELETE SET NULL;

**templates/** — директория с HTML-шаблонами, включая:

* Главную страницу
* Страницы авторизации и регистрации
* Формы добавления и редактирования книг
* Страницы рецензий и модерации
* Другие необходимые шаблоны интерфейса

**static/style.css** — основной файл CSS, содержащий:

* Стили для всех страниц приложения
* Адаптивное оформление под разные устройства
* Визуальные эффекты и анимации

**static/covers/** — папка для хранения обложек книг:

* Создаётся автоматически при первом запуске (если отсутствует)
* Обложки загружаются пользователями или генерируются системой

all\_reviews.html:

{% extends 'base.html' %}

{% block content %}

<h2>Все рецензии</h2>

<table>

    <tr>

        <th>Статус</th>

        <th>Книга</th>

        <th>Пользователь</th>

        <th>Оценка</th>

        <th>Текст</th>

        <th>Дата</th>

    </tr>

    {% for review in reviews %}

    <tr>

        <td>

            {% if review.status.name == 'approved' %}

                <span style="color: #217a2b; font-weight: 600;">Одобрено</span>

            {% elif review.status.name == 'rejected' %}

                <span style="color: #a94442; font-weight: 600;">Отклонено</span>

            {% elif review.status.name == 'pending' %}

                <span style="color: #888; font-weight: 600;">На рассмотрении</span>

            {% else %}

                {{ review.status.name }}

            {% endif %}

        </td>

        <td><a href="/book/{{ review.book.id }}">{{ review.book.title }}</a></td>

        <td>{{ review.user.last\_name }} {{ review.user.first\_name }}</td>

        <td>{{ review.rating }}</td>

        <td>{{ review.text\_html|safe }}</td>

        <td>{{ review.created\_at.strftime('%d.%m.%Y %H:%M') }}</td>

    </tr>

    {% endfor %}

</table>

<div style="margin-top: 24px;">

    <a href="/" class="btn" style="background:#888;">Назад</a>

</div>

{% endblock %}

base.html:

<!DOCTYPE html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <title>Электронная библиотека</title>

    <link rel="stylesheet" href="/static/style.css">

    <style>

    html, body {

        height: 100%;

    }

    body {

        min-height: 100vh;

        display: flex;

        flex-direction: column;

    }

    main {

        flex: 1 0 auto;

    }

    nav {

        display: flex;

        justify-content: space-between;

        align-items: center;

        padding-right: 32px;

        padding-left: 32px;

    }

    .nav-left {

        display: flex;

        align-items: center;

        gap: 18px;

    }

    .nav-user {

        color: #fff;

        font-weight: 500;

        margin-left: 0;

    }

    .nav-right {

        display: flex;

        align-items: center;

        gap: 18px;

    }

    .nav-right a {

        margin-left: 0;

    }

    footer {

        text-align: center;

        padding: 10px 0;

        background-color: #f4f6fb;

        border-top: 1px solid #e9ecef;

        box-sizing: border-box;

    }

    </style>

</head>

<body>

    <nav>

        <div class="nav-left">

            {% if current\_user.is\_authenticated %}

                <span class="nav-user">

                    {{ current\_user.first\_name }} {{ current\_user.last\_name }}

                </span>

            {% else %}

                <span class="nav-user">Гость</span>

            {% endif %}

        </div>

        <div class="nav-right">

            <a href="/">Главная</a>

            {% if current\_user.is\_authenticated %}

                {% if current\_user.role.name == 'admin' %}

                    <a href="/users">Пользователи</a>

                    <a href="/all-reviews">Все рецензии</a>

                {% endif %}

                {% if current\_user.role.name == 'user' %}

                    <a href="/my-reviews">Мои рецензии</a>

                {% elif current\_user.role.name == 'moderator' %}

                    <a href="/moderate">Модерация рецензий</a>

                {% endif %}

                <a href="/logout">Выйти</a>

            {% else %}

                <a href="/login">Войти</a>

            {% endif %}

        </div>

    </nav>

    <main>

        {% with messages = get\_flashed\_messages(with\_categories=true) %}

            {% if messages %}

                <ul class="flashes">

                {% for category, message in messages %}

                    <li class="{{ category }}">{{ message }}</li>

                {% endfor %}

                </ul>

            {% endif %}

        {% endwith %}

        {% block content %}{% endblock %}

    </main>

    <footer>

        <hr>

        <small style="font-size: 18px;">Электронная библиотека <b>Егорова Ксения, группа 231-3211</b></small>

    </footer>

</body>

</html>

book\_form.html:

{% extends 'base.html' %}

{% block content %}

<h2 style="margin-bottom:24px;">{% if book %}Редактирование книги{% else %}Добавление книги{% endif %}</h2>

<form method="post" {% if not book %}enctype="multipart/form-data"{% endif %} class="book-form">

    {{ form.hidden\_tag() }}

    <div class="form-row">

        <div class="form-col">

            <div>{{ form.title.label }}<br>{{ form.title(size=60) }}</div>

            <div>{{ form.author.label }}<br>{{ form.author(size=40) }}</div>

            <div>{{ form.publisher.label }}<br>{{ form.publisher(size=40) }}</div>

            <div>{{ form.year.label }}<br>{{ form.year() }}</div>

            <div>{{ form.pages.label }}<br>{{ form.pages() }}</div>

        </div>

        <div class="form-col">

            <div>{{ form.description.label }}<br>{{ form.description(rows=8, cols=50) }}</div>

            <div>{{ form.genres.label }}<br>{{ form.genres(multiple=True) }}</div>

            {% if not book %}

                <div>{{ form.cover.label }}<br>{{ form.cover() }}</div>

            {% endif %}

            {% if book and book.cover %}

                <div style="margin-top:8px;">

                    <b>Текущая обложка:</b><br>

                    <img src="/static/covers/{{ book.cover.filename }}" style="width:100px; border-radius:8px; margin-top:4px;">

                </div>

            {% endif %}

        </div>

    </div>

    <div class="form-actions">

        <a href="/" class="btn form-back-btn same-width-btn">Назад</a>

        {{ form.submit(class\_="btn same-width-btn") }}

    </div>

</form>

<style>

.book-form .form-row {

    display: flex;

    gap: 32px;

    flex-wrap: wrap;

}

.book-form .form-col {

    flex: 1 1 320px;

    min-width: 280px;

}

.book-form label {

    font-weight: 500;

    color: #2d3e50;

}

.book-form input[type="text"], .book-form textarea, .book-form select, .book-form input[type="number"] {

    margin-bottom: 14px;

}

.book-form input[type="file"] {

    margin-top: 6px;

}

.book-form .form-actions {

    display: flex;

    justify-content: flex-end;

    align-items: center;

    margin-top: 28px;

    gap: 16px;

}

.book-form .btn, .book-form .form-back-btn, .same-width-btn {

    min-width: 160px;

    font-size: 17px;

    padding: 10px 32px;

    text-align: center;

    text-decoration: none !important;

    display: inline-block;

    box-sizing: border-box;

}

.form-back-btn {

    background: #888;

    color: #fff;

    border: none;

    transition: background 0.2s, color 0.2s;

}

.form-back-btn:hover {

    background: #ffd700;

    color: #222;

}

@media (max-width: 900px) {

    .book-form .form-row {

        flex-direction: column;

        gap: 0;

    }

    .book-form .form-actions {

        flex-direction: column-reverse;

        gap: 10px;

        align-items: stretch;

    }

    .book-form .btn, .book-form .form-back-btn, .same-width-btn {

        width: 100%;

        min-width: 0;

        justify-content: center;

    }

}

</style>

{% endblock %}

book\_view.html:

{% extends 'base.html' %}

{% block content %}

<h2>{{ book.title }}</h2>

<p><b>Автор:</b> {{ book.author }}</p>

<p><b>Год:</b> {{ book.year }}</p>

<p><b>Издательство:</b> {{ book.publisher }}</p>

<p><b>Жанры:</b>

    {% for genre in book.genres %}

        {{ genre.name }}{% if not loop.last %}, {% endif %}

    {% endfor %}

</p>

<p><b>Страниц:</b> {{ book.pages }}</p>

{% if book.cover %}

    <img src="/static/covers/{{ book.cover.filename }}" width="200">

{% endif %}

<h3>Описание</h3>

<div>{{ book.description\_html|safe }}</div>

<h3>Рецензии</h3>

{% for review in reviews %}

    <div style="border:1px solid #ccc; margin:10px 0; padding:10px;">

        <b>{{ review.user.last\_name }} {{ review.user.first\_name }}</b> — Оценка: {{ review.rating }}<br>

        <div>{{ review.text\_html|safe }}</div>

        <small>Добавлено: {{ review.created\_at.strftime('%d.%m.%Y %H:%M') }}</small>

    </div>

{% else %}

    <p>Пока нет рецензий.</p>

{% endfor %}

<div style="display: flex; gap: 16px; margin-top: 24px;">

    <a href="/" class="btn" style="background:#888;">Назад</a>

    {% if can\_review %}

        <a href="/book/{{ book.id }}/review" class="btn">Написать рецензию</a>

    {% endif %}

</div>

{% endblock %}

index.html:

{% extends 'base.html' %}

{% block content %}

<h2>Список книг</h2>

<div class="search-add-row">

    <form method="get" class="search-form">

        <input type="text" name="search" value="{{ search or '' }}" placeholder="Поиск по названию">

        <button type="submit">Найти</button>

    </form>

    {% if current\_user.is\_authenticated and current\_user.role.name == 'admin' %}

        <a href="/book/add" class="btn btn-small add-book-btn" style="text-decoration:none;">Добавить книгу</a>

    {% endif %}

</div>

<table>

    <tr>

        <th>Обложка</th>

        <th>Название</th>

        <th>Жанры</th>

        <th>Год</th>

        <th>Средняя оценка</th>

        <th>Рецензий</th>

        <th>Действия</th>

    </tr>

    {% for book in books.items %}

    <tr>

        <td>

            {% if book.cover %}

                <img src="/static/covers/{{ book.cover.filename }}" class="cover-thumb" alt="Обложка">

            {% endif %}

        </td>

        <td>{{ book.title }}</td>

        <td>

            {# Исправленный вывод жанров #}

            {% for genre in book.genres %}

                {{ genre.name }}{% if not loop.last %}, {% endif %}

            {% endfor %}

        </td>

        <td>{{ book.year }}</td>

        <td>{{ book.avg\_rating or '-' }}</td>

        <td>{{ book.reviews\_count }}</td>

        <td>

            <div class="actions-row">

                <a href="/book/{{ book.id }}" class="action-btn view-btn" title="Просмотр">&#128065; Просмотр</a>

                {% if current\_user.is\_authenticated and current\_user.role.name in ['admin', 'moderator'] %}

                    <a href="/book/{{ book.id }}/edit" class="action-btn edit-btn" title="Редактировать">&#9998; Редактировать</a>

                {% endif %}

                {% if current\_user.is\_authenticated and current\_user.role.name == 'admin' %}

                    <a href="#" class="action-btn delete-btn" title="Удалить"

                    onclick="showDeleteModal({{ book.id }}, '{{ book.title|escape }}'); return false;">&#128465; Удалить</a>

                {% endif %}

            </div>

        </td>

    </tr>

    {% endfor %}

</table>

<div class="pagination-row">

    <form method="get" style="display:inline;">

        {% if search %}

            <input type="hidden" name="search" value="{{ search }}">

        {% endif %}

        <button type="submit" name="page" value="1" class="btn-small" {% if books.page == 1 %}disabled{% endif %}>&laquo; Первая</button>

        <button type="submit" name="page" value="{{ books.prev\_num }}" class="btn-small" {% if not books.has\_prev %}disabled{% endif %}>&lt; Назад</button>

    </form>

    <span class="pagination-info">

        Страница {{ books.page }} из {{ books.pages }}

    </span>

    <form method="get" style="display:inline;">

        {% if search %}

            <input type="hidden" name="search" value="{{ search }}">

        {% endif %}

        <button type="submit" name="page" value="{{ books.next\_num }}" class="btn-small" {% if not books.has\_next %}disabled{% endif %}>Вперёд &gt;</button>

        <button type="submit" name="page" value="{{ books.pages }}" class="btn-small" {% if books.page == books.pages %}disabled{% endif %}>Последняя &raquo;</button>

    </form>

</div>

<!-- Модальное окно для удаления книги -->

<div id="deleteModal" style="display:none; position:fixed; left:0; top:0; width:100vw; height:100vh; background:rgba(0,0,0,0.4); z-index:1000;">

<div style="background:#fff; max-width:400px; margin:100px auto; padding:20px; border-radius:8px; position:relative;">

    <h3>Удаление книги</h3>

    <p id="deleteModalText"></p>

    <div style="text-align:right;">

    <button onclick="closeDeleteModal()">Нет</button>

    <button id="deleteModalYesBtn">Да</button>

    </div>

</div>

</div>

<script>

let deleteBookId = null;

function showDeleteModal(bookId, bookTitle) {

    deleteBookId = bookId;

    document.getElementById('deleteModalText').innerText = 'Вы уверены, что хотите удалить книгу "' + bookTitle + '"?';

    document.getElementById('deleteModal').style.display = 'block';

}

function closeDeleteModal() {

    document.getElementById('deleteModal').style.display = 'none';

    deleteBookId = null;

}

document.getElementById('deleteModalYesBtn').onclick = function() {

    if (deleteBookId) {

        window.location.href = '/book/' + deleteBookId + '/delete';

    }

};

</script>

<style>

.search-add-row {

    display: flex;

    align-items: center;

    justify-content: space-between;

    gap: 18px;

    margin-bottom: 15px;

}

.add-book-btn {

    white-space: nowrap;

    margin-left: 18px;

    height: 32px;

    display: flex;

    align-items: center;

    font-size: 14px;

    padding: 4px 16px;

    text-decoration: none !important;

}

.btn-small {

    font-size: 14px;

    padding: 4px 16px;

    height: 32px;

}

.actions-row {

    display: flex;

    gap: 8px;

    flex-wrap: wrap;

    align-items: center;

    justify-content: flex-start;

}

.action-btn {

    display: inline-block;

    padding: 6px 12px;

    border-radius: 5px;

    font-size: 14px;

    font-weight: 500;

    text-decoration: none !important;

    transition: background 0.2s, color 0.2s;

    border: none;

    cursor: pointer;

    margin: 0;

}

.view-btn {

    background: #e9ecef;

    color: #2d3e50;

}

.view-btn:hover {

    background: #d1e7fd;

    color: #1d3557;

}

.edit-btn {

    background: #fff3cd;

    color: #856404;

}

.edit-btn:hover {

    background: #ffe082;

    color: #6d4c00;

}

.delete-btn {

    background: #f8d7da;

    color: #a94442;

}

.delete-btn:hover {

    background: #f5c6cb;

    color: #721c24;

}

table {

    width: 100%;

    border-collapse: collapse;

    margin-bottom: 20px;

}

th, td {

    padding: 12px 15px;

    text-align: left;

    border-bottom: 1px solid #ddd;

}

th {

    background-color: #f2f2f2;

    font-weight: bold;

}

tr:hover {

    background-color: #f5f5f5;

}

.cover-thumb {

    width: 80px;

    height: auto;

    border-radius: 10px;

    box-shadow: 0 2px 8px rgba(44,62,80,0.13);

    background: #f8fafc;

    border: 1px solid #e9ecef;

    display: block;

    margin: 0 auto;

    object-fit: cover;

}

.pagination-row {

    display: flex;

    align-items: center;

    justify-content: center;

    gap: 18px;

    margin: 18px 0 0 0;

}

.pagination-row form {

    display: flex;

    gap: 8px;

}

.pagination-info {

    font-size: 16px;

    color: #444;

    min-width: 120px;

    text-align: center;

}

.btn-small[disabled], .btn-small:disabled {

    background: #e9ecef !important;

    color: #aaa !important;

    cursor: not-allowed;

    border: none;

    opacity: 0.7;

}

@media (max-width: 700px) {

    .search-add-row {

        flex-direction: column;

        align-items: stretch;

        gap: 8px;

    }

    .add-book-btn {

        width: 100%;

        justify-content: center;

        margin-left: 0;

    }

    .action-btn {

        width: 100%;

        margin: 4px 0;

        text-align: center;

    }

    .cover-thumb {

        width: 60px;

    }

    .actions-row {

        flex-direction: column;

        gap: 4px;

        align-items: stretch;

    }

    .pagination-row {

        flex-direction: column;

        gap: 8px;

    }

    .pagination-info {

        margin: 6px 0;

    }

}

</style>

{% endblock %}

login.html:

{% extends 'base.html' %}

{% block content %}

<h2>Вход</h2>

<form method="post">

    {{ form.hidden\_tag() }}

    <div>

        {{ form.username.label }}<br>

        {{ form.username(size=32) }}

    </div>

    <div>

        {{ form.password.label }}<br>

        {{ form.password(size=32) }}

    </div>

    <div>

        {{ form.remember\_me() }} {{ form.remember\_me.label }}

    </div>

    <div>

        {{ form.submit() }}

    </div>

</form>

<p>Нет аккаунта? <a href="{{ url\_for('register') }}">Зарегистрироваться</a></p>

{% endblock %}

moderate\_review.html:

{% extends 'base.html' %}

{% block content %}

<h2>Рассмотрение рецензии</h2>

<p><b>Книга:</b> <a href="/book/{{ review.book.id }}">{{ review.book.title }}</a></p>

<p><b>Пользователь:</b> {{ review.user.last\_name }} {{ review.user.first\_name }}</p>

<p><b>Оценка:</b> {{ review.rating }}</p>

<p><b>Текст:</b></p>

<div>{{ review.text\_html|safe }}</div>

<p><b>Дата:</b> {{ review.created\_at.strftime('%d.%m.%Y %H:%M') }}</p>

<form method="post">

    <button name="action" value="approve">Одобрить</button>

    <button name="action" value="reject">Отклонить</button>

</form>

{% endblock %}

moderate.html:

{% extends 'base.html' %}

{% block content %}

<h2>Модерация рецензий</h2>

<table>

    <tr>

        <th>Книга</th>

        <th>Пользователь</th>

        <th>Дата</th>

        <th>Действие</th>

    </tr>

    {% for review in reviews.items %}

    <tr>

        <td><a href="/book/{{ review.book.id }}">{{ review.book.title }}</a></td>

        <td>{{ review.user.last\_name }} {{ review.user.first\_name }}</td>

        <td>{{ review.created\_at.strftime('%d.%m.%Y %H:%M') }}</td>

        <td><a href="/moderate/{{ review.id }}">Рассмотреть</a></td>

    </tr>

    {% endfor %}

</table>

<div>

    {% if reviews.has\_prev %}<a href="/moderate?page={{ reviews.prev\_num }}">&lt; Назад</a>{% endif %}

    Страница {{ reviews.page }} из {{ reviews.pages }}

    {% if reviews.has\_next %}<a href="/moderate?page={{ reviews.next\_num }}">Вперёд &gt;</a>{% endif %}

</div>

{% endblock %}

my\_reviews.html:

{% extends 'base.html' %}

{% block content %}

<h2>Мои рецензии</h2>

<table>

    <tr>

        <th>Статус</th>

        <th>Книга</th>

        <th>Оценка</th>

        <th>Текст</th>

        <th>Дата</th>

    </tr>

    {% for review in reviews %}

    <tr>

        <td>

            {% if review.status.name == 'approved' %}

                <span style="color: #217a2b; font-weight: 600;">Одобрено</span>

            {% elif review.status.name == 'rejected' %}

                <span style="color: #a94442; font-weight: 600;">Отклонено</span>

            {% elif review.status.name == 'pending' %}

                <span style="color: #888; font-weight: 600;">На рассмотрении</span>

            {% else %}

                {{ review.status.name }}

            {% endif %}

        </td>

        <td><a href="/book/{{ review.book.id }}">{{ review.book.title }}</a></td>

        <td>{{ review.rating }}</td>

        <td>{{ review.text\_html|safe }}</td>

        <td>{{ review.created\_at.strftime('%d.%m.%Y %H:%M') }}</td>

    </tr>

    {% endfor %}

</table>

<div style="margin-top: 24px;">

    <a href="/" class="btn" style="background:#888;">Назад</a>

</div>

{% endblock %}

register.html:

{% extends 'base.html' %}

{% block content %}

<h2>Регистрация</h2>

<form method="post" class="register-form">

    {{ form.hidden\_tag() }}

    <div>{{ form.username.label }}<br>{{ form.username(size=32) }}</div>

    <div>{{ form.password.label }}<br>{{ form.password(size=32, autocomplete="new-password") }}</div>

    <div>{{ form.last\_name.label }}<br>{{ form.last\_name(size=32) }}</div>

    <div>{{ form.first\_name.label }}<br>{{ form.first\_name(size=32) }}</div>

    <div>{{ form.middle\_name.label }}<br>{{ form.middle\_name(size=32) }}</div>

    <div class="form-actions">

        <a href="{{ url\_for('login') }}" class="btn form-back-btn same-width-btn">Назад</a>

        {{ form.submit(class\_="btn same-width-btn") }}

    </div>

</form>

<style>

.register-form .form-actions {

    display: flex;

    justify-content: flex-end;

    align-items: center;

    margin-top: 28px;

    gap: 16px;

}

.register-form .btn, .register-form .form-back-btn, .same-width-btn {

    min-width: 160px;

    font-size: 17px;

    padding: 10px 32px;

    text-align: center;

    text-decoration: none !important;

    display: inline-block;

    box-sizing: border-box;

}

.form-back-btn {

    background: #888;

    color: #fff;

    border: none;

    transition: background 0.2s, color 0.2s;

}

.form-back-btn:hover {

    background: #ffd700;

    color: #222;

}

@media (max-width: 900px) {

    .register-form .form-actions {

        flex-direction: column-reverse;

        gap: 10px;

        align-items: stretch;

    }

    .register-form .btn, .register-form .form-back-btn, .same-width-btn {

        width: 100%;

        min-width: 0;

        justify-content: center;

    }

}

</style>

{% endblock %}

review\_form.html:

{% extends 'base.html' %}

{% block content %}

<h2>Добавить рецензию</h2>

<form method="post">

    {{ form.hidden\_tag() }}

    <div>{{ form.rating.label }}<br>{{ form.rating() }}</div>

    <div>{{ form.text.label }}<br>{{ form.text(rows=5, cols=60) }}</div>

    <div>{{ form.submit() }}</div>

</form>

{% endblock %}

user\_add.html:

{% extends 'base.html' %}

{% block content %}

<h2>Добавить пользователя</h2>

<form method="post" class="user-edit-form" style="margin-bottom:0;">

    {{ form.hidden\_tag() }}

    <div>{{ form.username.label }}<br>{{ form.username(size=32) }}</div>

    <div>{{ form.password.label }}<br>{{ form.password(size=32, autocomplete="new-password") }}</div>

    <div>{{ form.last\_name.label }}<br>{{ form.last\_name(size=32) }}</div>

    <div>{{ form.first\_name.label }}<br>{{ form.first\_name(size=32) }}</div>

    <div>{{ form.middle\_name.label }}<br>{{ form.middle\_name(size=32) }}</div>

    <div>{{ form.role\_id.label }}<br>{{ form.role\_id() }}</div>

    <div class="form-actions">

        <a href="{{ url\_for('users') }}" class="btn btn-small form-back-btn" style="background:#888; display: flex; align-items: center; justify-content: center; min-height: 36px;">Назад</a>

        {{ form.submit(class\_="btn") }}

    </div>

</form>

<style>

.user-edit-form .form-actions {

    display: flex;

    justify-content: flex-end;

    align-items: center;

    margin-top: 24px;

    gap: 12px;

}

.user-edit-form .btn, .user-edit-form .btn-small, .user-edit-form .form-back-btn {

    min-width: 140px;

    font-size: 16px;

    padding: 8px 24px;

    text-align: center;

    text-decoration: none !important;

    display: inline-flex;

    align-items: center;

    justify-content: center;

    box-sizing: border-box;

    min-height: 36px;

}

@media (max-width: 900px) {

    .user-edit-form .form-actions {

        flex-direction: column-reverse;

        gap: 8px;

        align-items: stretch;

    }

    .user-edit-form .btn, .user-edit-form .btn-small, .user-edit-form .form-back-btn {

        width: 100%;

        min-width: 0;

        justify-content: center;

    }

}

</style>

{% endblock %}

user\_edit.html:

{% extends 'base.html' %}

{% block content %}

<h2>Редактировать пользователя</h2>

<form method="post" class="user-edit-form" style="margin-bottom:0;">

    {{ form.hidden\_tag() }}

    <div>{{ form.last\_name.label }}<br>{{ form.last\_name(size=32) }}</div>

    <div>{{ form.first\_name.label }}<br>{{ form.first\_name(size=32) }}</div>

    <div>{{ form.middle\_name.label }}<br>{{ form.middle\_name(size=32) }}</div>

    <div>{{ form.role\_id.label }}<br>{{ form.role\_id() }}</div>

    <div>{{ form.password.label }}<br>{{ form.password(size=32, autocomplete="new-password") }}</div>

    <div class="form-actions">

        <a href="{{ url\_for('users') }}" class="btn btn-small form-back-btn" style="background:#888; display: flex; align-items: center; justify-content: center; min-height: 36px;">Назад</a>

        {{ form.submit(class\_="btn") }}

    </div>

</form>

<style>

.user-edit-form .form-actions {

    display: flex;

    justify-content: flex-end;

    align-items: center;

    margin-top: 24px;

    gap: 12px;

}

.user-edit-form .btn, .user-edit-form .btn-small, .user-edit-form .form-back-btn {

    min-width: 140px;

    font-size: 16px;

    padding: 8px 24px;

    text-align: center;

    text-decoration: none !important;

    display: inline-flex;

    align-items: center;

    justify-content: center;

    box-sizing: border-box;

    min-height: 36px;

}

@media (max-width: 900px) {

    .user-edit-form .form-actions {

        flex-direction: column-reverse;

        gap: 8px;

        align-items: stretch;

    }

    .user-edit-form .btn, .user-edit-form .btn-small, .user-edit-form .form-back-btn {

        width: 100%;

        min-width: 0;

        justify-content: center;

    }

}

</style>

{% endblock %}

users.html:

{% extends 'base.html' %}

{% block content %}

<div style="display: flex; align-items: center; justify-content: space-between; margin-bottom: 18px;">

    <h2 style="margin: 0;">Пользователи</h2>

    <a href="{{ url\_for('add\_user') }}" class="btn btn-small user-add-btn">Добавить пользователя</a>

</div>

<table>

    <tr>

        <th>ID</th>

        <th>Логин</th>

        <th>Фамилия</th>

        <th>Имя</th>

        <th>Отчество</th>

        <th>Роль</th>

        <th>Действия</th>

    </tr>

    {% for user in users %}

    <tr>

        <td>{{ user.id }}</td>

        <td>{{ user.username }}</td>

        <td>{{ user.last\_name }}</td>

        <td>{{ user.first\_name }}</td>

        <td>{{ user.middle\_name or '' }}</td>

        <td>{{ user.role.name }}</td>

        <td>

            <div class="user-actions-row">

                <a href="{{ url\_for('edit\_user', user\_id=user.id) }}" class="btn btn-small user-action-btn">Редактировать</a>

                {% if user.id != current\_user.id %}

                <form method="post" action="{{ url\_for('delete\_user', user\_id=user.id) }}" style="display:inline;" onsubmit="return confirm('Удалить пользователя?');">

                    <button type="submit" class="btn btn-small user-action-btn user-delete-btn">Удалить</button>

                </form>

                {% endif %}

            </div>

        </td>

    </tr>

    {% endfor %}

</table>

<style>

.user-actions-row {

    display: flex;

    gap: 10px;

    align-items: center;

}

.user-action-btn {

    min-width: 120px;

    height: 36px;

    font-size: 15px;

    padding: 6px 0;

    text-align: center;

    box-sizing: border-box;

    display: inline-flex;

    align-items: center;

    justify-content: center;

}

.user-delete-btn {

    background: #a94442;

}

.user-delete-btn:hover {

    background: #f5c6cb;

    color: #721c24;

}

.user-add-btn {

    min-width: 180px;

    height: 36px;

    font-size: 15px;

    padding: 6px 0;

    text-align: center;

    box-sizing: border-box;

    display: inline-flex;

    align-items: center;

    justify-content: center;

}

</style>

{% endblock %}

style.css:

body {

    font-family: 'Segoe UI', Arial, sans-serif;

    margin: 0;

    padding: 0;

    background: #f4f6fb;

    color: #222;

}

nav {

    background: #d32f2f; /\* красный \*/

    color: #fff;

    padding: 22px 0 22px 30px; /\* увеличить высоту \*/

    font-size: 1.35em; /\* увеличить размер шрифта \*/

    box-shadow: 0 2px 6px rgba(0,0,0,0.04);

}

nav a {

    color: #fff;

    margin-right: 18px;

    text-decoration: none;

    font-weight: 500;

    transition: color 0.2s;

}

nav a:hover {

    color: #ffd700;

}

nav span {

    margin-left: 30px;

    font-weight: 400;

    color: #ffd700;

}

main {

    max-width: 1600px;

    margin: 0 auto; /\* убрано 40px сверху и 30px снизу \*/

    background: #fff;

    padding: 32px 32px 24px 32px;

    border-radius: 12px;

    box-shadow: 0 2px 16px rgba(44,62,80,0.07);

}

footer {

    text-align: center;

    margin: 0; /\* убраны внешние отступы \*/

    color: #888;

    font-size: 15px;

}

.flashes {

    list-style: none;

    padding: 0;

    margin-bottom: 18px;

}

.flashes li {

    margin: 7px 0;

    padding: 10px 18px;

    border-radius: 6px;

    font-size: 16px;

}

.flashes .success {

    background: #e6f9e6;

    color: #217a2b;

    border: 1px solid #b6eab6;

}

.flashes .error {

    background: #ffeaea;

    color: #a94442;

    border: 1px solid #f5c6cb;

}

/\* Кнопки \*/

.btn, .btn-small, button, input[type="submit"] {

    background: #502d2d;

    color: #fff;

    border: none;

    padding: 8px 22px;

    border-radius: 6px;

    font-size: 15px;

    cursor: pointer;

    margin: 3px 0;

    transition: background 0.2s, color 0.2s;

    font-family: inherit;

}

/\* Все кнопки цвета "назад" (серые) теперь нежно-розовые \*/

.btn[style\*="background:#888"],

.btn[style\*="background: #888"],

.btn[style\*="background:rgb(136,136,136)"],

.btn[style\*="background: #888;"],

.btn[style\*="background:#888;"],

a.btn[style\*="background:#888"],

a.btn[style\*="background: #888"],

a.btn[style\*="background:rgb(136,136,136)"],

a.btn[style\*="background: #888;"],

a.btn[style\*="background:#888;"],

.btn-back,

a.btn-back,

.add-book-btn {

    background: #f8bbd0 !important; /\* нежно-розовый \*/

    color: #ad1457 !important;

    border: none;

}

.btn[style\*="background:#888"]:hover,

.btn[style\*="background: #888"]:hover,

.btn[style\*="background:rgb(136,136,136)"]:hover,

.btn[style\*="background: #888;"]:hover,

.btn[style\*="background:#888;"]:hover,

a.btn[style\*="background:#888"]:hover,

a.btn[style\*="background: #888"]:hover,

a.btn[style\*="background:rgb(136,136,136)"]:hover,

a.btn[style\*="background: #888;"]:hover,

a.btn[style\*="background:#888;"]:hover,

.btn-back:hover,

a.btn-back:hover,

.add-book-btn:hover {

    background: #f06292 !important; /\* чуть ярче при наведении \*/

    color: #fff !important;

}

/\* Таблицы \*/

table {

    width: 100%;

    border-collapse: collapse;

    margin-bottom: 18px;

    background: #fafbfc;

    border-radius: 8px;

    overflow: hidden;

    box-shadow: 0 1px 4px rgba(44,62,80,0.04);

}

th, td {

    padding: 14px 18px;

    text-align: left;

}

th {

    background: #e9ecef;

    font-weight: 600;

    color: #502d2d;

    border-bottom: 2px solid #d1d7de;

}

tr:nth-child(even) {

    background: #f6f8fa;

}

tr:hover {

    background: #f0f4fa;

}

td img {

    border-radius: 4px;

    box-shadow: 0 1px 4px rgba(44,62,80,0.08);

}

/\* Формы \*/

form input[type="text"], form input[type="password"], form textarea, form select {

    width: 100%;

    padding: 8px 10px;

    margin: 6px 0 14px 0;

    border: 1px solid #cfd8dc;

    border-radius: 5px;

    font-size: 15px;

    background: #f8fafc;

    transition: border 0.2s;

    box-sizing: border-box;

}

form input[type="text"]:focus, form input[type="password"]:focus, form textarea:focus, form select:focus {

    border: 1.5px solid #502d2d;

    outline: none;

    background: #fff;

}

form label {

    font-weight: 500;

    color: #502d2d;

}

form textarea {

    min-height: 80px;

    resize: vertical;

}

/\* Модальное окно \*/

#deleteModal {

    display: none;

    position: fixed;

    left: 0; top: 0;

    width: 100vw; height: 100vh;

    background: rgba(80, 44, 44, 0.18);

    z-index: 1000;

}

#deleteModal > div {

    background: #fff;

    max-width: 400px;

    margin: 120px auto;

    padding: 28px 24px 18px 24px;

    border-radius: 10px;

    position: relative;

    box-shadow: 0 2px 16px rgba(80, 44, 44, 0.13);

}

#deleteModal h3 {

    margin-top: 0;

    color: #ad2f2d;

}

#deleteModal button {

    margin-left: 10px;

    min-width: 70px;

}

/\* Картинки обложек \*/

img[src\*="covers/"] {

    border: 1px solid #e9ecef;

    background: #f8fafc;

    padding: 2px;

    max-height: 200px;

    max-width: 100%;

}

/\* Адаптивная таблица \*/

.table-responsive {

    width: 100%;

    margin-bottom: 18px;

}

/\* Адаптивность \*/

@media (max-width: 700px) {

    main { padding: 10px; }

    table, th, td { font-size: 13px; }

    nav { padding-left: 10px; }

}

/\* Стили для поиска \*/

.search-form {

    display: flex;

    gap: 0;

    max-width: 400px;

    margin-bottom: 15px;

}

.search-form input[type="text"] {

    flex: 1 1 auto;

    border-top-right-radius: 0;

    border-bottom-right-radius: 0;

    margin: 0;

}

.search-form button {

    border-top-left-radius: 0;

    border-bottom-left-radius: 0;

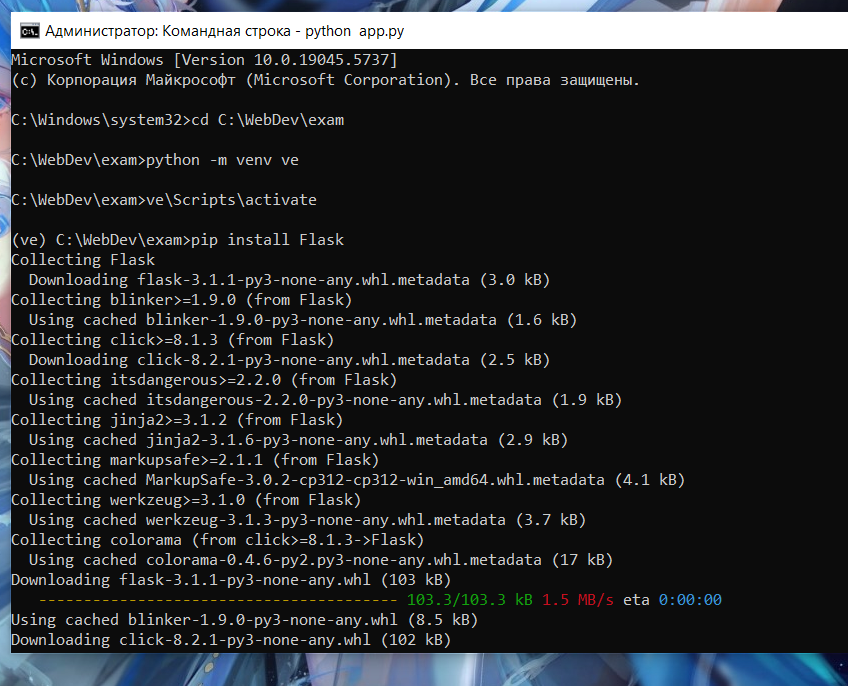
    margin: 0;

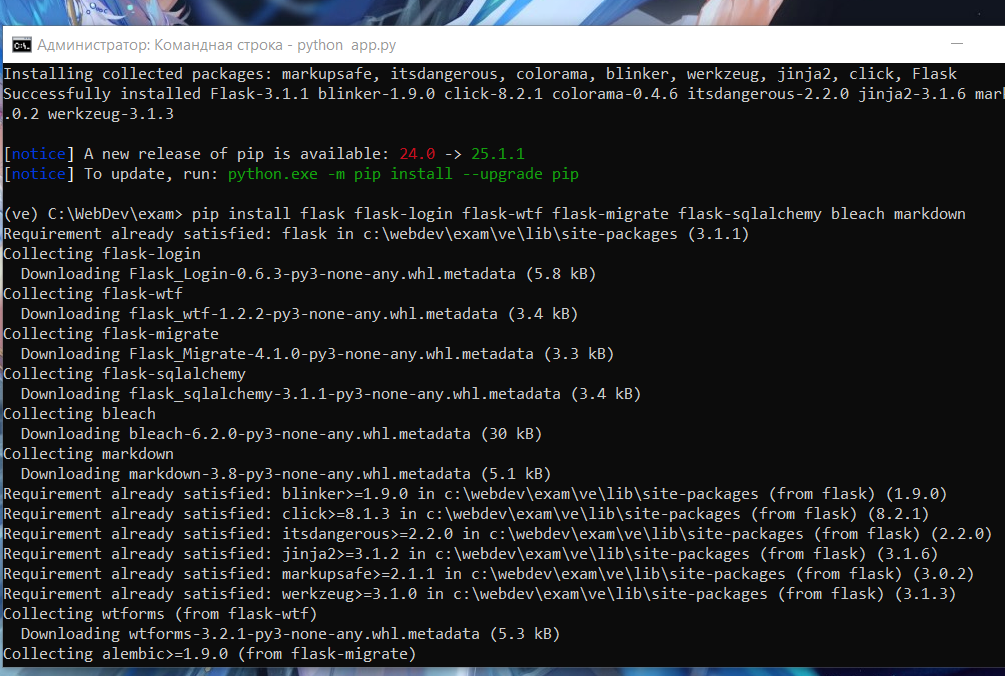
    height: 40px;

    padding: 0 22px;

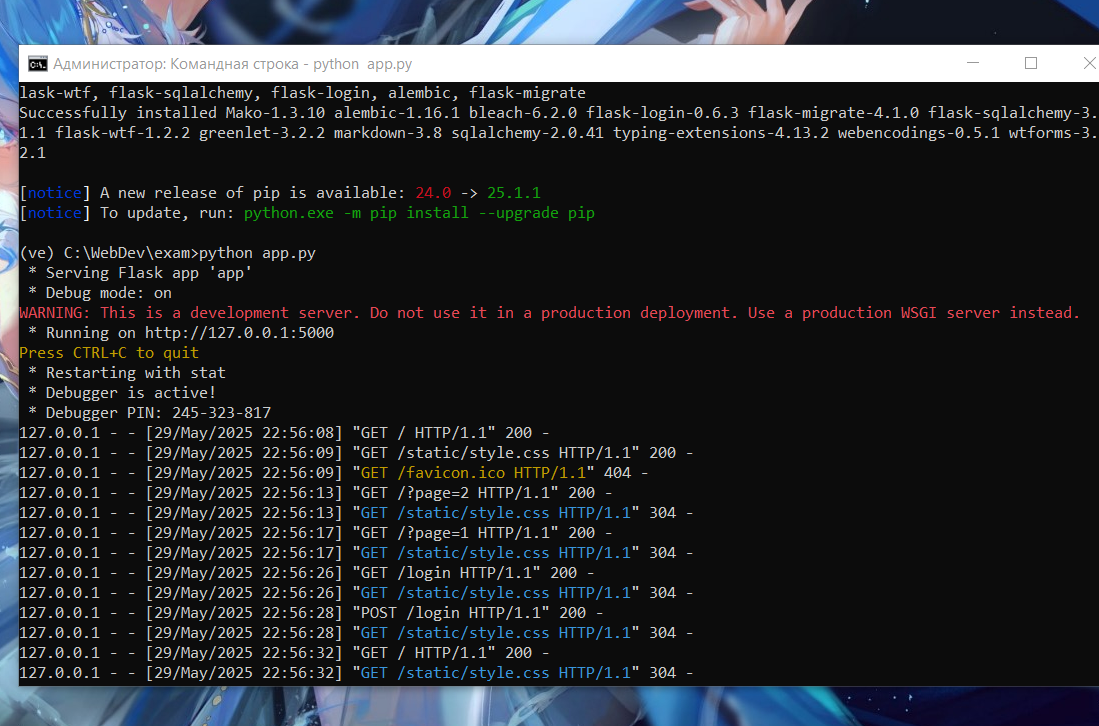
}

Загрузим необходимые библиотеки и активируем виртуальную среду:





Запустим:



Результат:

