**Исходный код**

**from datetime import datetime**

**from flask import Flask, render\_template, request, redirect, url\_for, session, flash**

**from flask\_sqlalchemy import SQLAlchemy**

**import os**

**from werkzeug.utils import secure\_filename**

**import matplotlib**

**matplotlib.use('Agg')**

**import matplotlib.pyplot as plt**

**import io**

**import base64**

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

**from flask\_restx import Api, Resource, fields, Namespace**

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

**app.secret\_key = 'super\_secret\_key'**

**app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///data.db'**

**app.config['UPLOAD\_FOLDER'] = 'static/img'**

**app.config['MAX\_CONTENT\_LENGTH'] = 16 \* 1024 \* 1024**

**api = Api(**

**app,**

**version='1.0',**

**title='ELITE SHOP API',**

**description='API для интернет-магазина ELITE SHOP',**

**doc='/docs',**

**prefix='/api'**

**)**

**products\_ns = Namespace('products', description='Операции с товарами')**

**product\_model = api.model('Product', {**

**'id': fields.Integer(readOnly=True, description='ID продукта'),**

**'name': fields.String(required=True, description='Название'),**

**'price': fields.Float(required=True, description='Цена'),**

**'description': fields.String(description='Описание'),**

**'image': fields.String(description='URL изображения'),**

**'stock': fields.Integer(description='Количество на складе')**

**})**

**api.add\_namespace(products\_ns)**

**db = SQLAlchemy(app)**

**class User(db.Model):**

**\_\_tablename\_\_ = 'user'**

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

**username = db.Column(db.String(80), unique=True, nullable=False)**

**password = db.Column(db.String(120), nullable=False)**

**role = db.Column(db.String(20), default='user')**

**balance = db.Column(db.Float, default=0.0)**

**address = db.Column(db.String(200))**

**given\_ratings = db.relationship('ProductRating', back\_populates='rating\_user')**

**given\_comments = db.relationship('ProductComment', back\_populates='comment\_user')**

**def set\_password(self, password):**

**self.password = generate\_password\_hash(password)**

**def check\_password(self, password):**

**return check\_password\_hash(self.password, password)**

**class ProductRating(db.Model):**

**\_\_tablename\_\_ = 'product\_rating'**

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

**product\_id = db.Column(db.Integer, db.ForeignKey('product.id'))**

**user\_id = db.Column(db.Integer, db.ForeignKey('user.id'))**

**rating = db.Column(db.Integer)**

**created\_at = db.Column(db.String(50), default=lambda: datetime.now().strftime("%Y-%m-%d %H:%M:%S"))**

**rated\_product = db.relationship('Product', back\_populates='ratings')**

**rating\_user = db.relationship('User', back\_populates='given\_ratings')**

**class ProductComment(db.Model):**

**\_\_tablename\_\_ = 'product\_comment'**

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

**product\_id = db.Column(db.Integer, db.ForeignKey('product.id'))**

**user\_id = db.Column(db.Integer, db.ForeignKey('user.id'))**

**text = db.Column(db.Text)**

**created\_at = db.Column(db.String(50), default=lambda: datetime.now().strftime("%Y-%m-%d %H:%M:%S"))**

**commented\_product = db.relationship('Product', back\_populates='comments')**

**comment\_user = db.relationship('User', back\_populates='given\_comments')**

**class Product(db.Model):**

**\_\_tablename\_\_ = 'product'**

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

**name = db.Column(db.String(100), nullable=False)**

**price = db.Column(db.Float, nullable=False)**

**description = db.Column(db.Text)**

**image = db.Column(db.String(100))**

**stock = db.Column(db.Integer, default=10)**

**ratings = db.relationship('ProductRating', back\_populates='rated\_product')**

**comments = db.relationship('ProductComment', back\_populates='commented\_product')**

**class CartItem(db.Model):**

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

**user\_id = db.Column(db.Integer, db.ForeignKey('user.id'))**

**product\_id = db.Column(db.Integer, db.ForeignKey('product.id'))**

**quantity = db.Column(db.Integer, default=1)**

**product = db.relationship('Product')**

**class Order(db.Model):**

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

**user\_id = db.Column(db.Integer, db.ForeignKey('user.id'))**

**total = db.Column(db.Float, nullable=False)**

**address = db.Column(db.String(200), nullable=False)**

**date = db.Column(db.String(50), nullable=False)**

**items = db.relationship('OrderItem', backref='order')**

**user = db.relationship('User', backref='orders')**

**class OrderItem(db.Model):**

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

**order\_id = db.Column(db.Integer, db.ForeignKey('order.id'))**

**product\_id = db.Column(db.Integer, db.ForeignKey('product.id'))**

**quantity = db.Column(db.Integer)**

**price = db.Column(db.Float)**

**product = db.relationship('Product')**

**with app.app\_context():**

**db.create\_all()**

**if not User.query.filter\_by(username='manager').first():**

**manager = User(username='manager', role='manager', balance=0)**

**manager.set\_password('manager123')**

**db.session.add(manager)**

**if not User.query.filter\_by(username='admin').first():**

**admin = User(username='admin', role='admin', balance=0)**

**admin.set\_password('admin123')**

**db.session.add(admin)**

**db.session.commit()**

**@app.route('/')**

**def index():**

**return render\_template('index.html')**

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

**def register():**

**if request.method == 'POST':**

**username = request.form['username']**

**password = request.form['password']**

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

**flash('Это имя пользователя уже занято!', 'error')**

**return redirect(url\_for('register'))**

**new\_user = User(username=username, balance=1000)**

**new\_user.set\_password(password)**

**db.session.add(new\_user)**

**db.session.commit()**

**flash('Регистрация успешна! Войдите в систему.', 'success')**

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

**return render\_template('register.html')**

**@products\_ns.route('/')**

**class ProductList(Resource):**

**@products\_ns.marshal\_list\_with(product\_model)**

**@products\_ns.doc(description='Получить список всех товаров')**

**def get(self):**

**"""Список всех товаров"""**

**products = Product.query.all()**

**return products**

**@products\_ns.route('/<int:id>')**

**@products\_ns.param('id', 'ID товара')**

**class ProductResource(Resource):**

**@products\_ns.marshal\_with(product\_model)**

**@products\_ns.doc(description='Получить товар по ID')**

**def get(self, id):**

**"""Получить один товар"""**

**product = Product.query.get\_or\_404(id)**

**return product**

**@api.route('/ping')**

**class Ping(Resource):**

**def get(self):**

**"""Проверка работоспособности API"""**

**return {'status': 'ok', 'message': 'API работает'}**

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

**def login():**

**if request.method == 'POST':**

**username = request.form['username']**

**password = request.form['password']**

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

**if user and user.check\_password(password):**

**session['user\_id'] = user.id**

**session['username'] = user.username**

**session['role'] = user.role**

**flash(f'Добро пожаловать, {username}!', 'success')**

**if user.role == 'manager':**

**return redirect(url\_for('manager\_panel'))**

**elif user.role == 'admin':**

**return redirect(url\_for('admin\_panel'))**

**else:**

**return redirect(url\_for('shop'))**

**else:**

**flash('Неверное имя пользователя или пароль!', 'error')**

**return render\_template('login.html')**

**@app.route('/logout')**

**def logout():**

**session.clear()**

**flash('Вы успешно вышли из системы.', 'info')**

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

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

**def shop():**

**if 'user\_id' not in session:**

**flash('Пожалуйста, войдите в систему.', 'warning')**

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

**search\_query = request.args.get('search', '').strip().lower()**

**if search\_query:**

**products = Product.query.filter(**

**db.or\_(**

**db.func.lower(Product.name).contains(search\_query),**

**db.func.lower(Product.description).contains(search\_query)**

**)**

**).all()**

**else:**

**products = Product.query.all()**

**return render\_template('shop.html', products=products, search\_query=search\_query)**

**@app.route('/add\_to\_cart/<int:product\_id>')**

**def add\_to\_cart(product\_id):**

**if 'user\_id' not in session:**

**flash('Пожалуйста, войдите в систему.', 'warning')**

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

**user\_id = session['user\_id']**

**cart\_item = CartItem.query.filter\_by(user\_id=user\_id, product\_id=product\_id).first()**

**if cart\_item:**

**cart\_item.quantity += 1**

**else:**

**cart\_item = CartItem(user\_id=user\_id, product\_id=product\_id)**

**db.session.add(cart\_item)**

**db.session.commit()**

**flash('Товар добавлен в корзину!', 'success')**

**return redirect(url\_for('shop'))**

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

**def cart():**

**if 'user\_id' not in session:**

**flash('Пожалуйста, войдите в систему.', 'warning')**

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

**user\_id = session['user\_id']**

**if request.method == 'POST' and 'coupon' in request.form:**

**coupon = request.form['coupon']**

**user = User.query.get(user\_id)**

**if coupon == 'ELITE500':**

**user.balance += 500**

**flash('Купон на 500 рублей применен!', 'success')**

**elif coupon == 'ELITE1000':**

**user.balance += 1000**

**flash('Купон на 1000 рублей применен!', 'success')**

**else:**

**flash('Неверный купон!', 'error')**

**db.session.commit()**

**return redirect(url\_for('cart'))**

**if request.method == 'POST' and 'address' in request.form:**

**address = request.form['address']**

**user = User.query.get(user\_id)**

**cart\_items = CartItem.query.filter\_by(user\_id=user\_id).all()**

**if not cart\_items:**

**flash('Ваша корзина пуста!', 'error')**

**return redirect(url\_for('cart'))**

**total = sum(item.product.price \* item.quantity for item in cart\_items)**

**if user.balance < total:**

**flash('Недостаточно средств на балансе!', 'error')**

**return redirect(url\_for('cart'))**

**from datetime import datetime**

**order = Order(**

**user\_id=user\_id,**

**total=total,**

**address=address,**

**date=datetime.now().strftime("%Y-%m-%d %H:%M:%S")**

**)**

**db.session.add(order)**

**db.session.commit()**

**for item in cart\_items:**

**order\_item = OrderItem(**

**order\_id=order.id,**

**product\_id=item.product\_id,**

**quantity=item.quantity,**

**price=item.product.price**

**)**

**db.session.add(order\_item)**

**item.product.stock -= item.quantity**

**db.session.delete(item)**

**user.balance -= total**

**db.session.commit()**

**flash(f'Заказ оформлен! Номер заказа: {order.id}', 'success')**

**return redirect(url\_for('profile'))**

**cart\_items = CartItem.query.filter\_by(user\_id=user\_id).all()**

**total = sum(item.product.price \* item.quantity for item in cart\_items)**

**user = User.query.get(user\_id)**

**return render\_template('cart.html', cart\_items=cart\_items, total=total, user=user)**

**@app.route('/remove\_from\_cart/<int:item\_id>')**

**def remove\_from\_cart(item\_id):**

**if 'user\_id' not in session:**

**flash('Пожалуйста, войните в систему.', 'warning')**

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

**cart\_item = CartItem.query.get(item\_id)**

**if cart\_item and cart\_item.user\_id == session['user\_id']:**

**db.session.delete(cart\_item)**

**db.session.commit()**

**flash('Товар удален из корзины.', 'info')**

**return redirect(url\_for('cart'))**

**@app.route('/profile')**

**def profile():**

**if 'user\_id' not in session:**

**flash('Пожалуйста, войдите в систему.', 'warning')**

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

**user\_id = session['user\_id']**

**user = User.query.get(user\_id)**

**orders = Order.query.filter\_by(user\_id=user\_id).order\_by(Order.id.desc()).all()**

**return render\_template('profile.html', user=user, orders=orders)**

**@app.route('/manager')**

**def manager\_panel():**

**if 'user\_id' not in session or session.get('role') != 'manager':**

**flash('Доступ запрещен!', 'error')**

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

**try:**

**orders = Order.query.all()**

**from collections import defaultdict**

**sales\_by\_date = defaultdict(float)**

**for order in orders:**

**date = order.date.split()[0]**

**sales\_by\_date[date] += order.total**

**dates = sorted(sales\_by\_date.keys())**

**amounts = [sales\_by\_date[date] for date in dates]**

**fig, ax = plt.subplots(figsize=(10, 5))**

**ax.plot(dates, amounts, marker='o')**

**ax.set\_title('Продажи по дням')**

**ax.set\_xlabel('Дата')**

**ax.set\_ylabel('Сумма (руб)')**

**plt.xticks(rotation=45)**

**plt.tight\_layout()**

**img = io.BytesIO()**

**plt.savefig(img, format='png', bbox\_inches='tight')**

**img.seek(0)**

**plot\_url = base64.b64encode(img.getvalue()).decode('utf8')**

**plt.close(fig)**

**products = Product.query.all()**

**orders = db.session.query(Order, User).join(User, Order.user\_id == User.id)\**

**.order\_by(Order.id.desc()).all()**

**return render\_template('manager.html',**

**products=products,**

**orders=orders,**

**plot\_url=plot\_url)**

**except Exception as e:**

**print(f"Ошибка при создании графика: {e}")**

**flash('Ошибка при создании отчета', 'error')**

**return redirect(url\_for('manager\_panel'))**

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

**def add\_product():**

**if 'user\_id' not in session or session.get('role') != 'manager':**

**flash('Доступ запрещен!', 'error')**

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

**if request.method == 'POST':**

**name = request.form['name']**

**price = float(request.form['price'])**

**description = request.form['description']**

**stock = int(request.form['stock'])**

**if 'image' not in request.files:**

**flash('Не выбрано изображение товара!', 'error')**

**return redirect(request.url)**

**image = request.files['image']**

**if image.filename == '':**

**flash('Не выбрано изображение товара!', 'error')**

**return redirect(request.url)**

**if image:**

**filename = secure\_filename(image.filename)**

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

**image.save(image\_path)**

**new\_product = Product(**

**name=name,**

**price=price,**

**description=description,**

**image=filename,**

**stock=stock**

**)**

**db.session.add(new\_product)**

**db.session.commit()**

**flash('Товар успешно добавлен!', 'success')**

**return redirect(url\_for('product\_page', product\_id=new\_product.id))**

**return render\_template('add\_product.html')**

**@app.route('/delete\_product/<int:product\_id>')**

**def delete\_product(product\_id):**

**if 'user\_id' not in session or session.get('role') != 'manager':**

**flash('Доступ запрещен!', 'error')**

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

**product = Product.query.get(product\_id)**

**if product:**

**if product.image:**

**try:**

**os.remove(os.path.join(app.config['UPLOAD\_FOLDER'], product.image))**

**except:**

**pass**

**db.session.delete(product)**

**db.session.commit()**

**flash('Товар успешно удален!', 'success')**

**return redirect(url\_for('manager\_panel'))**

**@app.route('/delete\_order/<int:order\_id>')**

**def delete\_order(order\_id):**

**if 'user\_id' not in session or session.get('role') != 'manager':**

**flash('Доступ запрещен!', 'error')**

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

**order = Order.query.get(order\_id)**

**if order:**

**OrderItem.query.filter\_by(order\_id=order.id).delete()**

**db.session.delete(order)**

**db.session.commit()**

**flash('Заказ успешно удален!', 'success')**

**return redirect(url\_for('manager\_panel'))**

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

**def product\_page(product\_id):**

**product = Product.query.get\_or\_404(product\_id)**

**return render\_template('page/product.html', product=product)**

**@app.route('/product/<int:product\_id>/comment', methods=['POST'])**

**def add\_comment(product\_id):**

**if 'user\_id' not in session:**

**flash('Пожалуйста, войдите в систему.', 'warning')**

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

**text = request.form.get('text')**

**if not text:**

**flash('Комментарий не может быть пустым', 'error')**

**return redirect(url\_for('product\_page', product\_id=product\_id))**

**comment = ProductComment(**

**product\_id=product\_id,**

**user\_id=session['user\_id'],**

**text=text**

**)**

**db.session.add(comment)**

**db.session.commit()**

**flash('Комментарий добавлен', 'success')**

**return redirect(url\_for('product\_page', product\_id=product\_id))**

**@app.route('/product/<int:product\_id>/rate', methods=['POST'])**

**def rate\_product(product\_id):**

**if 'user\_id' not in session:**

**flash('Пожалуйста, войдите в систему.', 'warning')**

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

**rating = int(request.form.get('rating', 0))**

**if not 1 <= rating <= 5:**

**flash('Неверная оценка', 'error')**

**return redirect(url\_for('product\_page', product\_id=product\_id))**

**existing\_rating = ProductRating.query.filter\_by(**

**product\_id=product\_id,**

**user\_id=session['user\_id']**

**).first()**

**if existing\_rating:**

**existing\_rating.rating = rating**

**else:**

**new\_rating = ProductRating(**

**product\_id=product\_id,**

**user\_id=session['user\_id'],**

**rating=rating**

**)**

**db.session.add(new\_rating)**

**db.session.commit()**

**flash('Спасибо за вашу оценку!', 'success')**

**return redirect(url\_for('product\_page', product\_id=product\_id))**

**@app.template\_filter('average')**

**def average\_filter(values):**

**if not values:**

**return 0**

**values\_list = list(values)**

**return sum(values\_list) / len(values\_list)**

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

**def delete\_comment(comment\_id):**

**if 'user\_id' not in session:**

**flash('Пожалуйста, войдите в систему.', 'warning')**

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

**comment = ProductComment.query.get\_or\_404(comment\_id)**

**product\_id = comment.product\_id**

**if session['user\_id'] == comment.user\_id or session.get('role') == 'manager':**

**db.session.delete(comment)**

**db.session.commit()**

**flash('Комментарий удален', 'success')**

**else:**

**flash('Недостаточно прав для удаления', 'error')**

**return redirect(url\_for('product\_page', product\_id=product\_id))**

**@app.route('/admin')**

**def admin\_panel():**

**if 'user\_id' not in session or session.get('role') != 'admin':**

**flash('Доступ запрещен!', 'error')**

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

**users = User.query.all()**

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

**@api.route('/hello')**

**class HelloWorld(Resource):**

**def get(self):**

**"""Тестовый endpoint"""**

**return {'message': 'API работает!'}**

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

**def delete\_user(user\_id):**

**if 'user\_id' not in session or session.get('role') != 'admin':**

**flash('Доступ запрещен!', 'error')**

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

**user = User.query.get(user\_id)**

**if user:**

**if user.id == session['user\_id']:**

**flash('Вы не можете удалить себя!', 'error')**

**return redirect(url\_for('admin\_panel'))**

**db.session.delete(user)**

**db.session.commit()**

**flash('Пользователь успешно удален!', 'success')**

**return redirect(url\_for('admin\_panel'))**

**@app.route('/admin/change\_role/<int:user\_id>', methods=['POST'])**

**def change\_role(user\_id):**

**if 'user\_id' not in session or session.get('role') != 'admin':**

**flash('Доступ запрещен!', 'error')**

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

**user = User.query.get(user\_id)**

**if user:**

**if user.id == session['user\_id']:**

**flash('Вы не можете изменить свою роль!', 'error')**

**return redirect(url\_for('admin\_panel'))**

**if user.role == 'user':**

**user.role = 'manager'**

**elif user.role == 'manager':**

**user.role = 'user'**

**db.session.commit()**

**flash('Роль пользователя изменена!', 'success')**

**return redirect(url\_for('admin\_panel'))**

**@app.route('/api/docs')**

**def swagger\_ui():**

**return api.render\_page()**

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

**app.run(debug=True)**