Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования     
**Национальный исследовательский университет «МЭИ»**

**Типовой расчёт**по курсу«**Базы данных**»

Выполнил:

Студент группы А-07-21

Тарбеев М.С.

**Москва  
2023**

1. **Тема типового расчета**

Предполагаемая тема для разработки базы данных – магазин цветов.

1. **Постановка задачи**

Необходимо будет наладить работу цветочного магазина, предоставляющего услуги по продаже цветов и букетов. Каждый клиент вносится в базу данных и ему присваивается статус покупателя.

В магазине присутствуют различные виды цветов и букетов с разными ценами. Также введены тарифы доставки: экспресс - доставка в течение 2 часов, стандарт - доставка в течение 24 часов, эконом - доставка в течение 48 часов. При оформлении заказа клиенту предоставляется выбор цветов, букетов и тарифа доставки. Система автоматически высчитывает стоимость заказа и добавляет его в базу данных. Клиент может оплатить заказ онлайн или наличными при получении.

После оплаты клиенту присваивается статус "оплачен". Сотрудник магазина со статусом "работник зала" имеет доступ только к ФИО клиента, выбору цветов и букетов, выбору тарифа доставки и статусу выполнения заказа. Он отвечает за прием и оформление заказа, расчет и выставление кассового чека, а также подготовку заказа для доставки. Пользователь со статусом

"менеджер" имеет доступ ко всем заказам, сделанным в магазине. Он может просмотреть информацию о каждом заказе, включая стоимость, статус и историю оплаты. Менеджер также может отслеживать выполнение заказов и связываться с клиентами в случае необходимости.

Пользователь со статусом "владелец" имеет доступ ко всем заказам и данным о клиентах. Он может добавлять новые виды цветов и букетов, устанавливать и изменять цены, а также наблюдать за общей выручкой магазина за день, неделю и месяц. Владелец также может назначать скидки и акции для клиентов, а также управлять бонусной программой.

Для постоянных клиентов также предусмотрена система бонусов. Клиентам, совершившим покупки на определенную сумму за месяц, предоставляется специальный статус "VIP". Владелец может устанавливать скидки и специальные предложения для VIP-клиентов. Например, цена букетов для них ниже на 25%, а цена доставки ниже на 30%.

Таким образом, бизнес-процесс цветочного магазина включает оформление заказов, расчет и оплату, подготовку к доставке, отслеживание выполнения заказов и управление базой данных клиентов.

1. **Описание данных**

Всего в моей базе данных 6 таблиц:

* *Customers* – таблица с данными о покупателях (содержит имя, фамилия, статус покупателя, месячная сумма покупок, номер телефона)
* *DeliveryRates* – таблица с тарифами доставки (содержит название тарифа, приблизительное время доставки, стоимость доставки)
* Employees – таблица с сотрудниками (содержит имя, фамилия, должность, логин и пароль для авторизации на сайте)
* *Flowers* – таблица с ассортиментом цветов (содержит название цветка и цена)
* *Payments* – таблица c заказами (содержит номер заказа, сумма заказа, тип оплаты, описание к заказу, статус заказа, дата заказа, айди тарифа доставки)
* *Orders* – таблица с позициями заказов и их количеством, т.к. в каждом заказе может быть несколько видов цветов и разное количество (содержит айди покупателя, айди цветка, количество цветков, айди заказа)

**ER – диаграмма:**

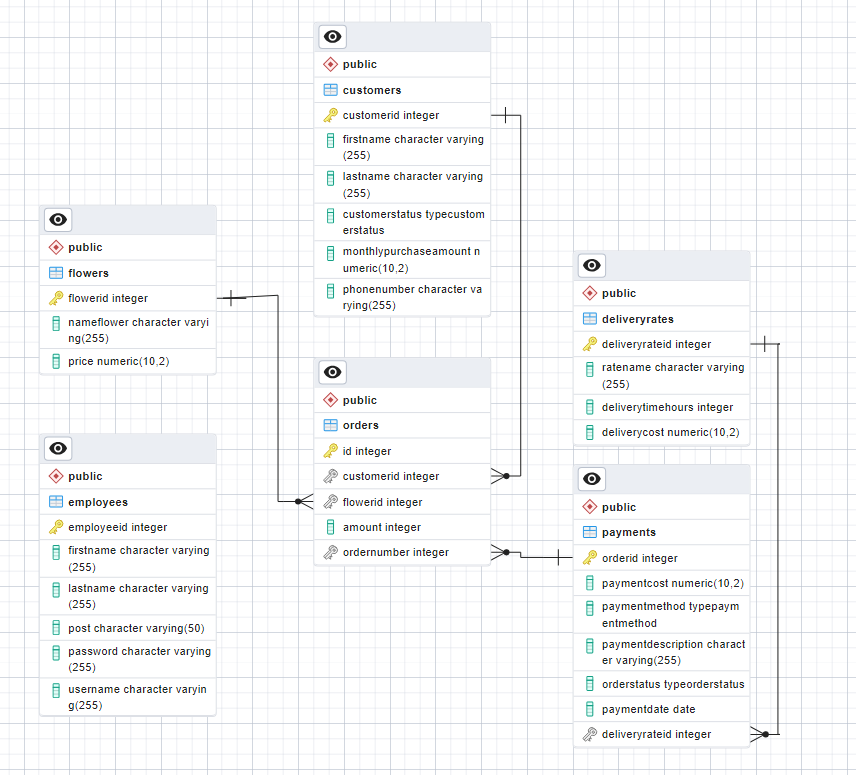


Таблица Customers имеет первичный ключ *customerid*, который отвечает за однозначную идентификацию каждого пользователя.

Таблица *DeliveryRates* имеет первичный ключ *deliveryrateid*, который отвечает за однозначную идентификацию каждого тарифа доставки.

Таблица Employees имеет первичный ключ *employeeid*, который отвечает за однозначную идентификацию каждого сотрудника.

Таблица Flowers имеет первичный ключ *flowerid*, который отвечает за однозначную идентификацию каждого цветка.

Таблица Payments имеет первичный ключ *orderid*, который отвечает за однозначную идентификацию каждого заказа. Также она содержит внешний ключ *deliveryrateid*, который отвечает за тариф доставки, выбранный в заказе.

Таблица *Orders* имеет первичный ключ id, который отвечает за однозначную идентификацию позицию каждого заказа. Также она содержит внешние ключи *customerid*, *flowerid*, *ordernumber*, которые отвечают соответственно за айди покупателя, айди цветка, номер заказа.

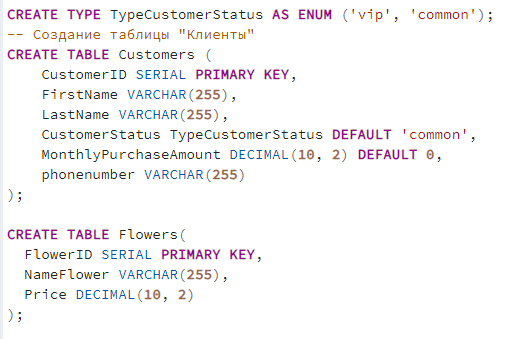
Все используемые типы данных в соответствующих столбцах отображены на ER-диаграмме.

Помимо стандартных типов данных были созданы пользовательские типы данных:

* *typecustomerstatus* – тип статуса покупателя: ‘vip’, либо ‘common’.
* *typeorderstatus* – тип статуса заказа: ‘оплачен’, ‘доставляется’, ‘доставлен’.
* *typepaymentmethod* – тип оплаты заказа: ‘наличные’, ‘карта’.

База данных “*FlowerShop*” была создана с помощью UI-инструментов pgAdmin 4.

Код для создания таблиц и типов:





1. **Группы пользователей**

Можно выделить три группы пользователей:

* *Работник*
* *Менеджер*
* *Владелец*

В возможности *работника* входит:

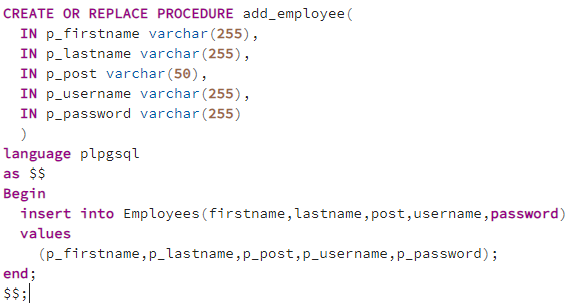
1. Создание заказов, обслуживание покупателей

В возможности *менеджера* входит:

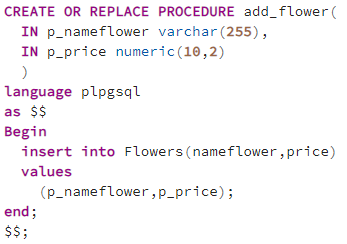
1. Создание заказов, обслуживание покупателей
2. Просмотр заказов и изменение их статуса

В возможности *владельца* входит:

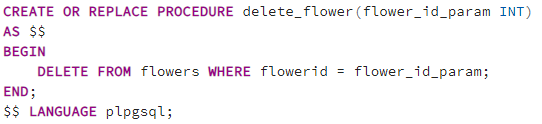
1. Создание заказов, обслуживание покупателей
2. Просмотр заказов и изменение их статуса
3. Добавление/удаление/просмотр цветов из ассортимента
4. Удаление(увольнение)/Добавление(найм)/Просмотр сотрудников
5. Просмотр списка клиентов, информации о них в т.ч. о статусе.
6. Просмотр выручки за день, неделю и месяц.
7. **Функции и процедуры:**
8. Процедура добавления сотрудника

****

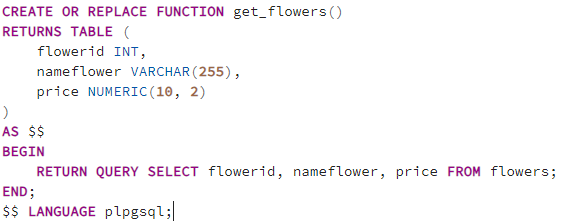
1. Процедура добавления цветка



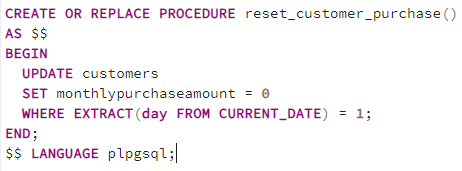
1. Процедура удаления цветка



1. Табличная функция получения списка цветов



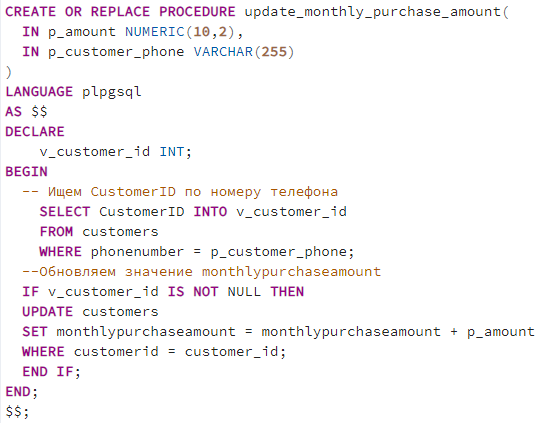
1. Процедура сброса месячных трат пользователей в начале месяца



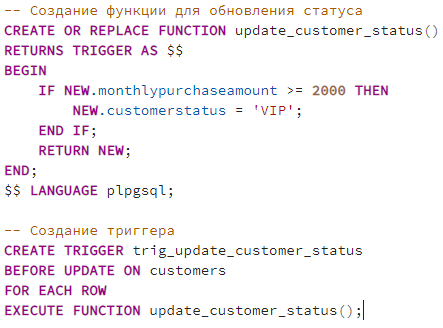
1. Скалярная функция создания заказа (возвращает id заказа)



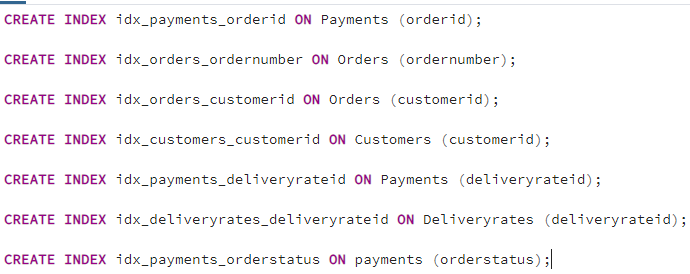
1. Процедура обновления месячный суммы покупок пользователя



1. Триггерная функция для обновления статуса покупателя

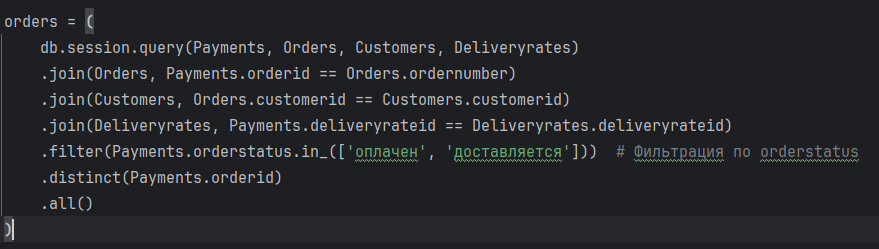


1. **Индексы**

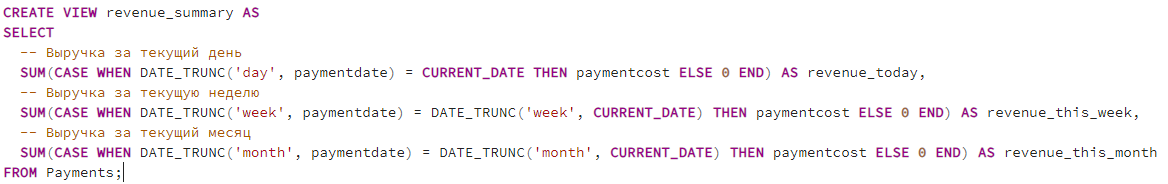
****

Данные индексы необходимы для ускорения последующего соединения таблиц и остальных запросов, к примеру – вывод информации о заказах.

Пример из кода веб-приложения с соединением таблиц:



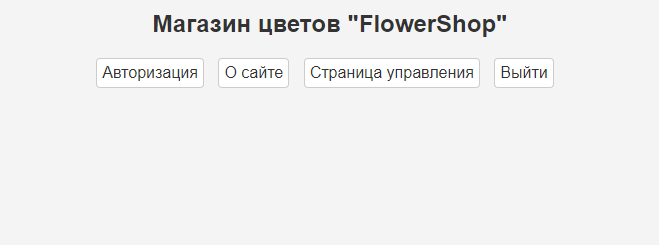
1. **Представления**
2. Представление, отражающее выручку за день, неделю и месяц.

****

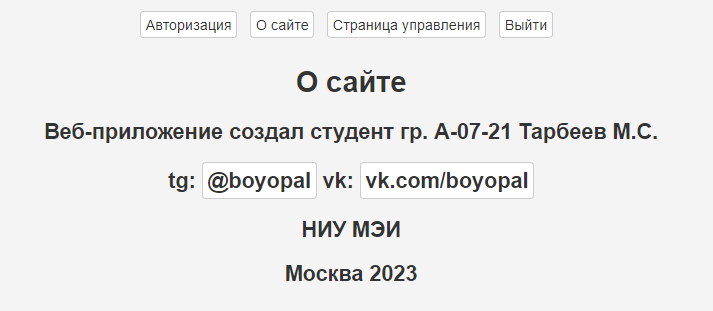
1. **Веб-приложение**

В качестве UI для моей базы данных было разработано веб-приложение на *Python* с использованием фреймворка *Flask*. Из-за некорректного в некоторых местах ТЗ было принято решение, что веб-приложение магазина цветов будет только для сотрудников (т.е. цветочный магазин очный, а покупателей обслуживают сотрудники).

Главная страница сайта выглядит так (/):

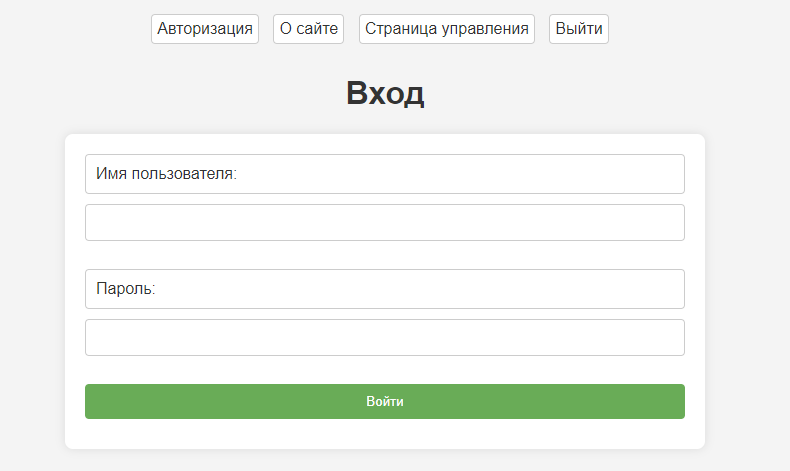


Страница «О сайте»:

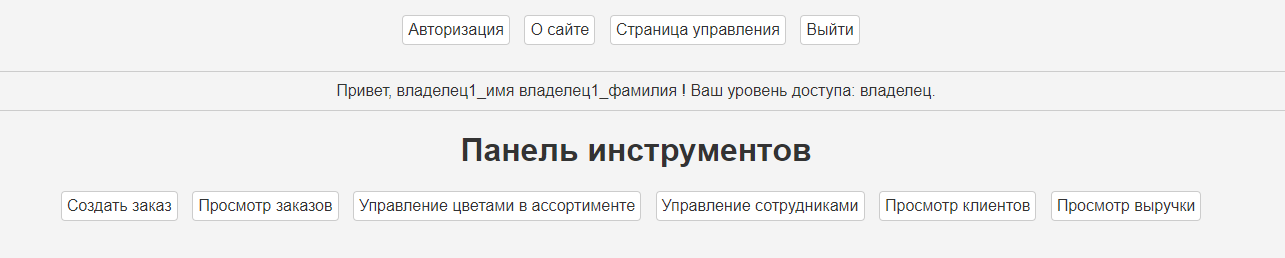


Далее необходимо проследовать на страницу авторизации, для входа сотрудника под своим личным логином и паролем, соответствующие полям *username* и *password* в таблице *Employees.*

При попытке перейти на раздел «Страница управления» без авторизации, автоматически перекидывает на страницу авторизации (/auth):



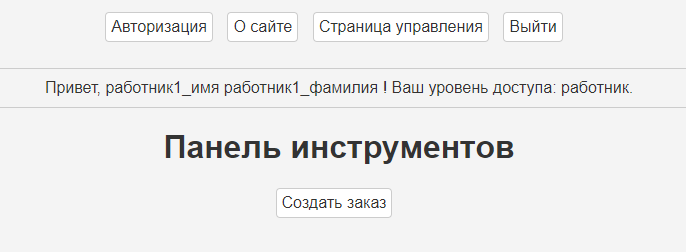
При авторизации, мы попадаем на страницу управления (/dashboard).



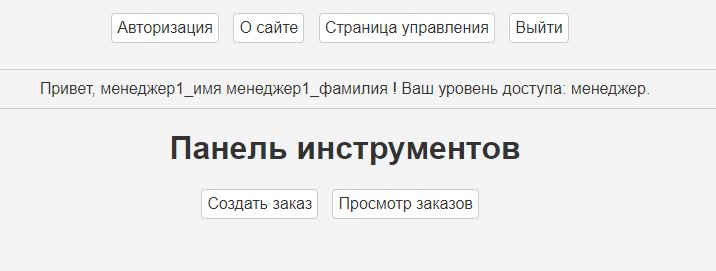
В зависимости от должности сотрудника, имеем разный уровень доступа к функционалу. При попытке воспользоваться функционалом, который не доступен должности нашего сотрудника, высвечивается сообщение о недостатке привелегий. Пример:



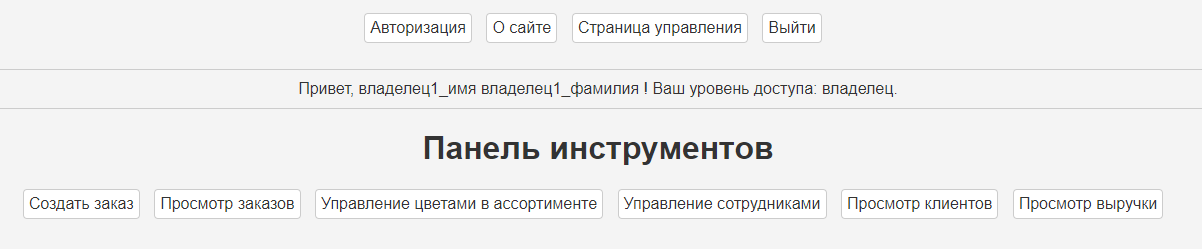
Как выглядит панель инструментов обычного работника:



Как выглядит панель инструментов менеджера:



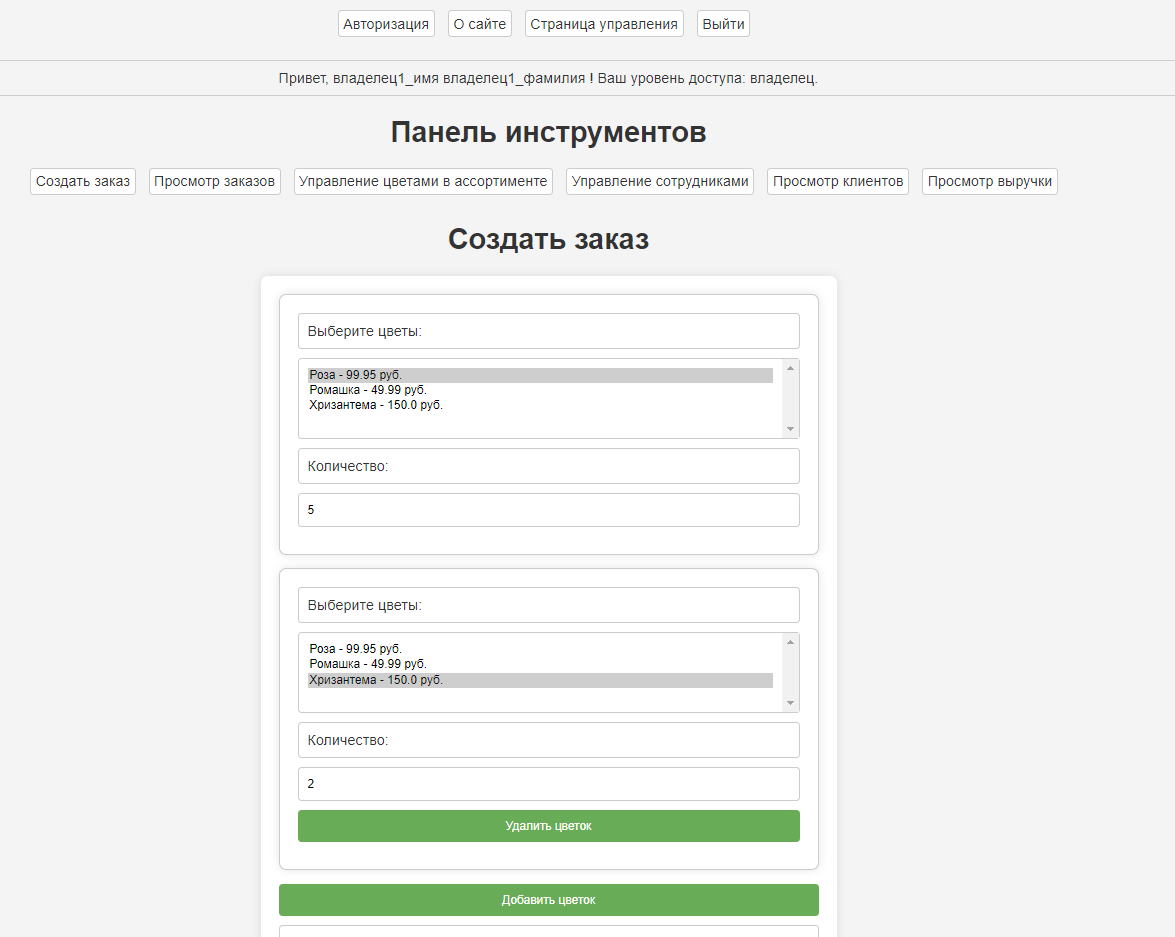
Как выглядит панель инструментов владельца:

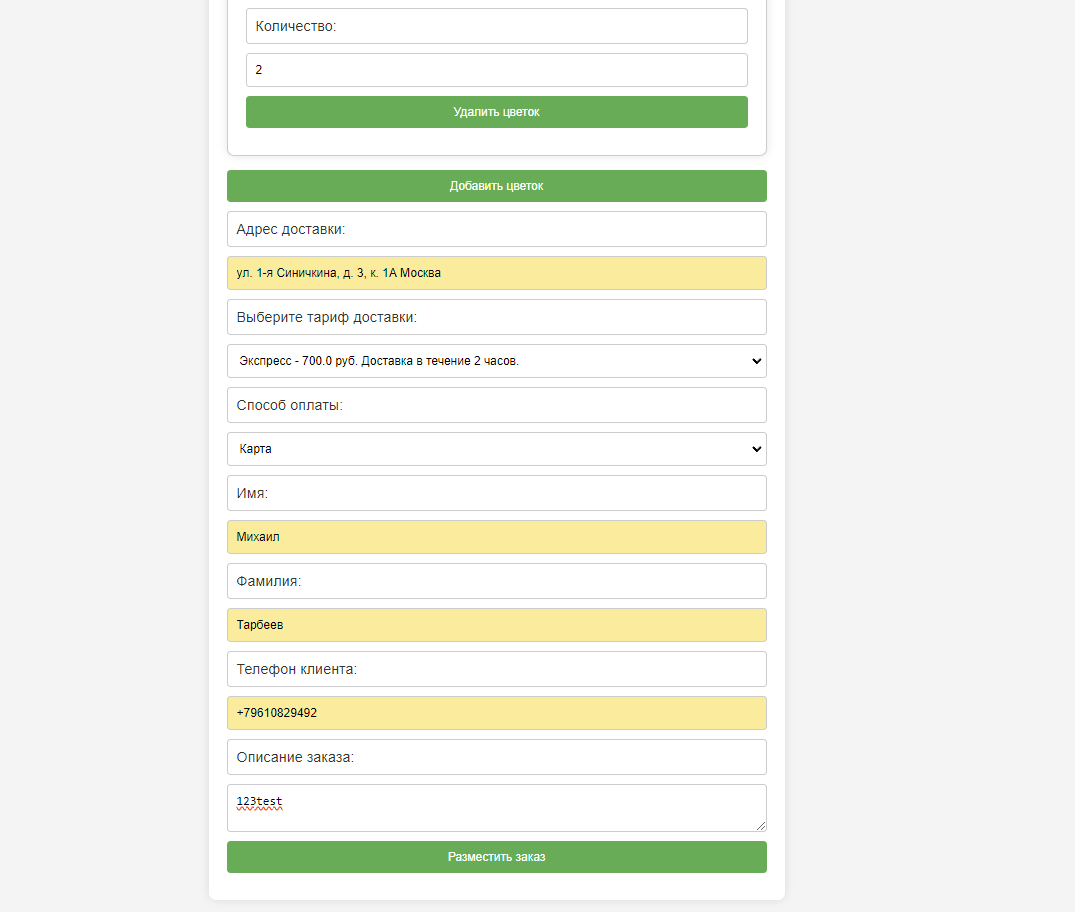


Пройдемся по каждой странице панели инструментов:

1. Страница создания заказа:

На данной странице можно создать заказ. Можно выбрать несколько позиций цветов, их количество, указать адресс доставки, выбрать тариф доставки, способ оплаты, имя, фамилию, номер телефона.

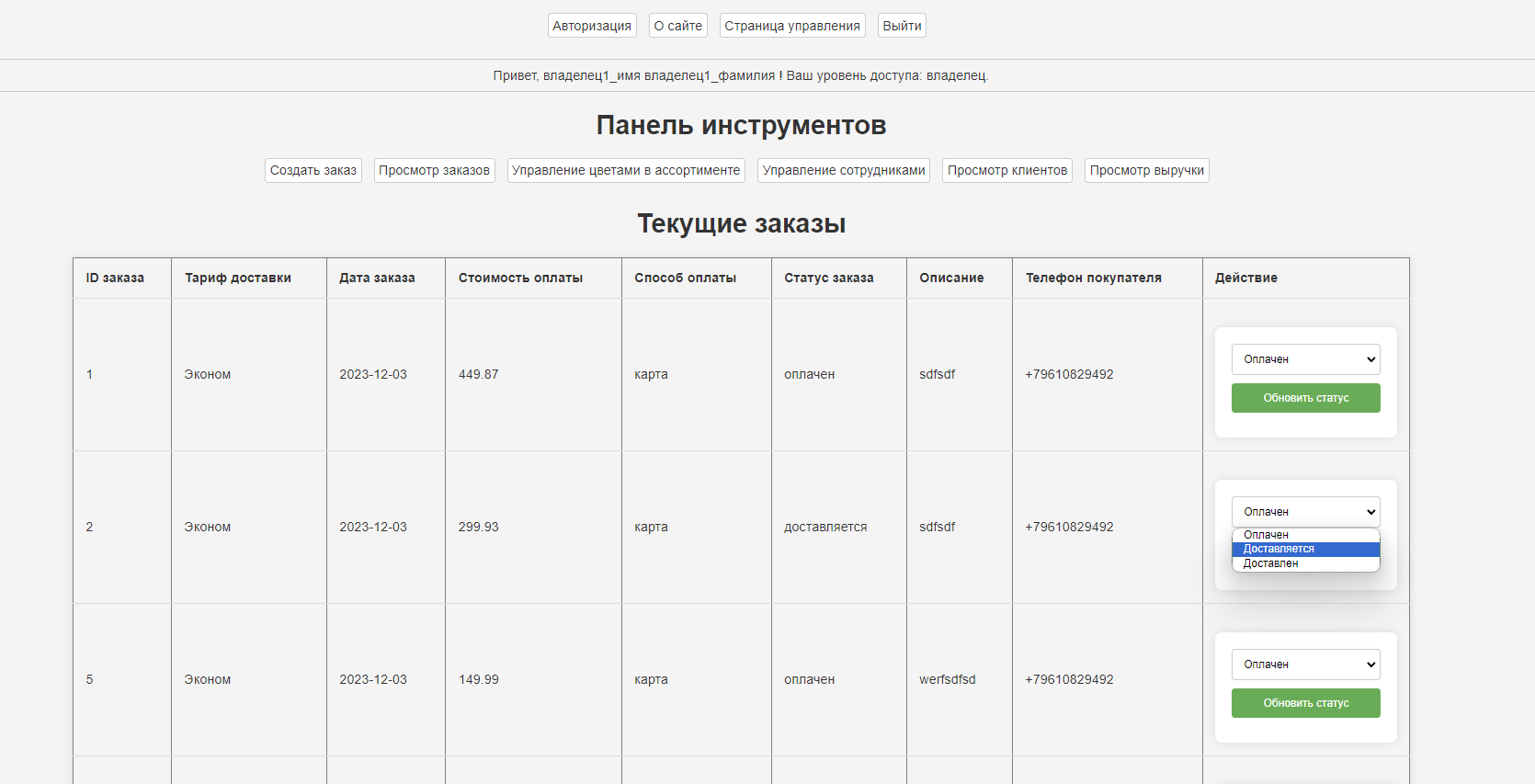




После нажатия кнопки «разместить заказ», наш заказ добавится в базу данных и отобразится на странице «просмотр заказов». Добавление происходит с помощью SQL функции *create\_order\_with\_payment*, которую мы вызываем с помощью db API (библиотека psycopg2).

1. Страница просмотра заказов

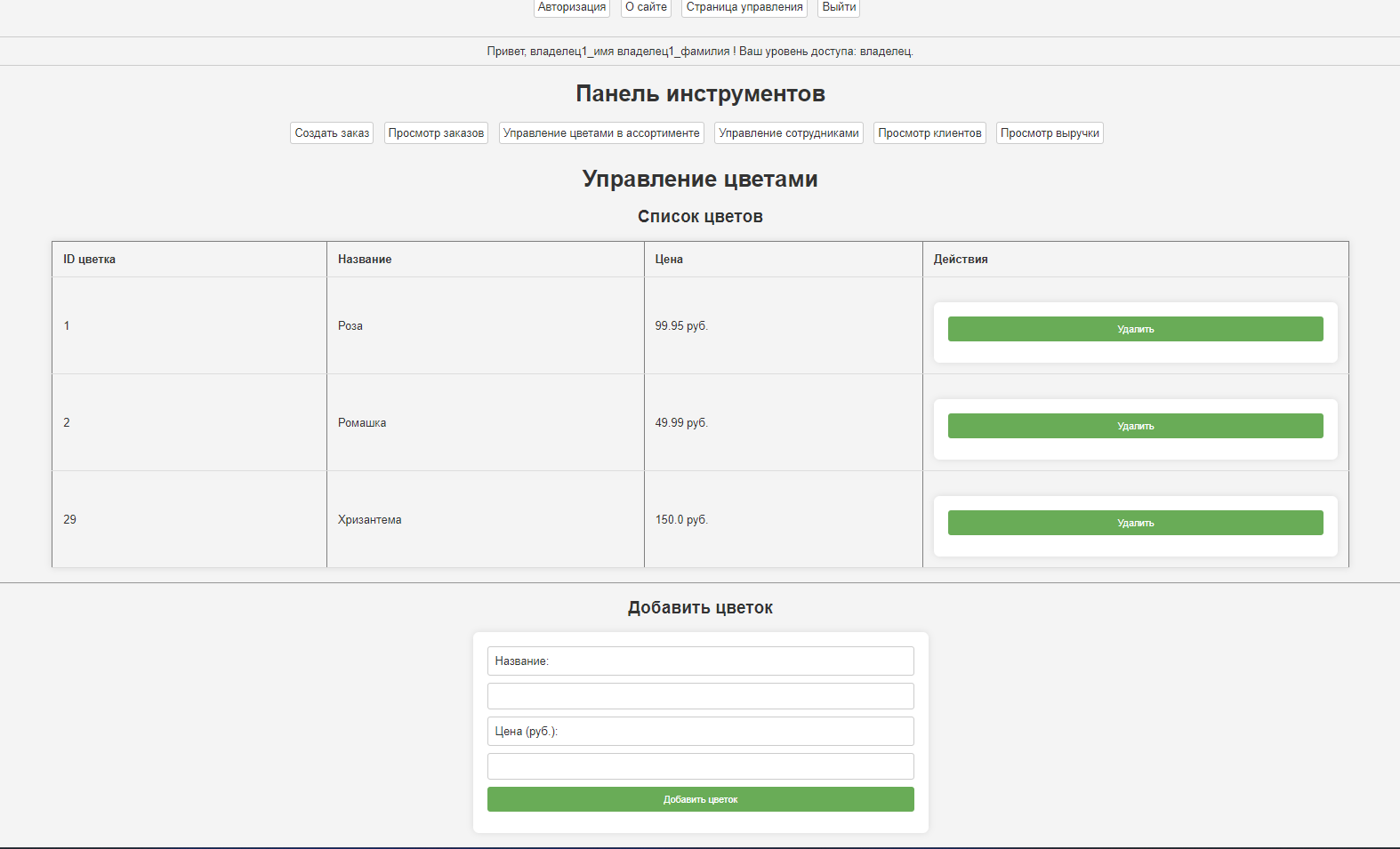
Страница просмотра заказов выглядит следующим образом:



Мы можем обновить статус заказа. При обновлении заказа на статус «доставлен», он перестаёт отображаться на странице. Запрос для вывода таблицы сделан с помощью ORM (библиотека SQLAlchemy).

1. Страница управления цветами

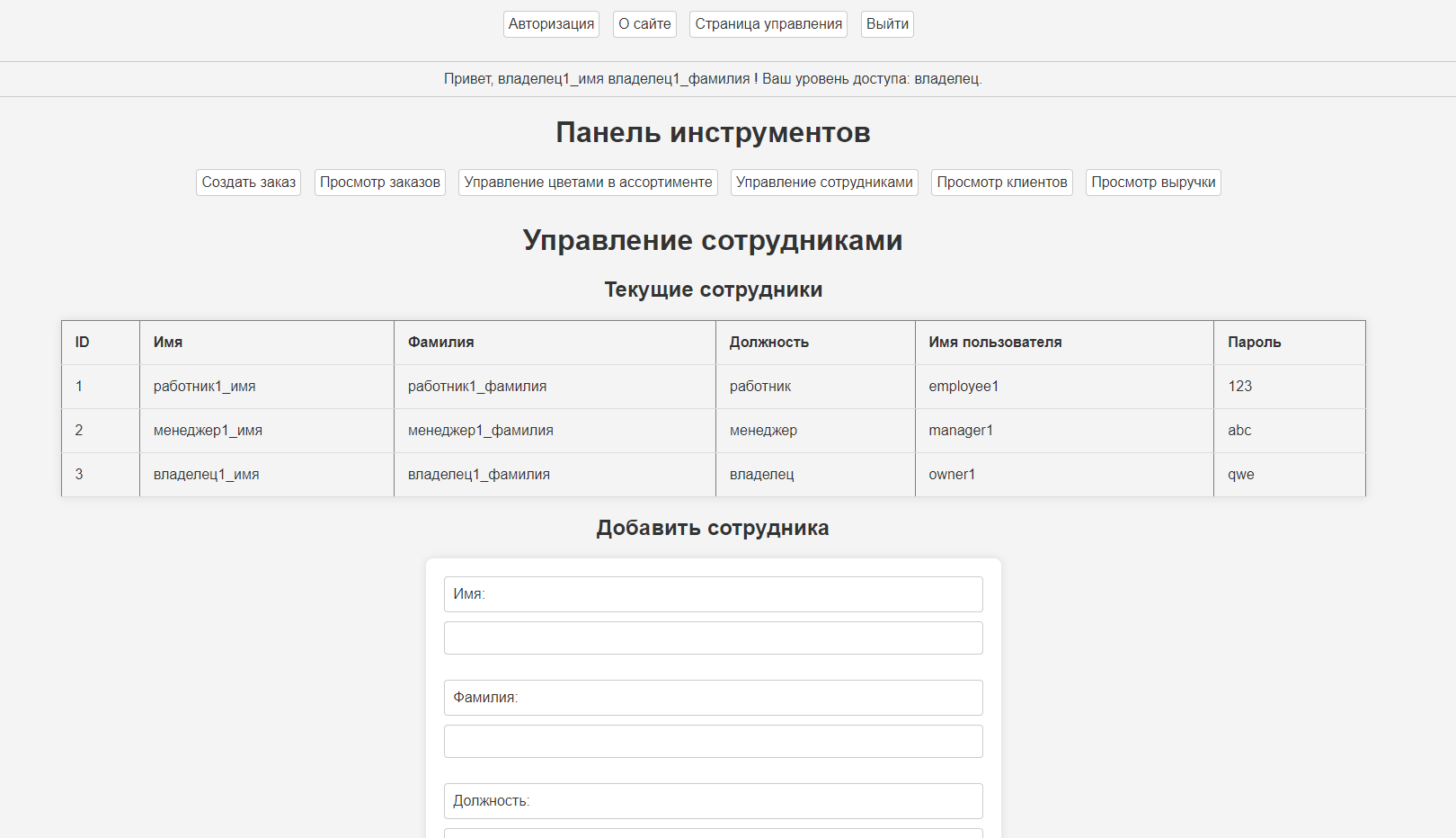
На данной странице мы можем просматривать текущий ассортимент, удалять и добавлять цветы.

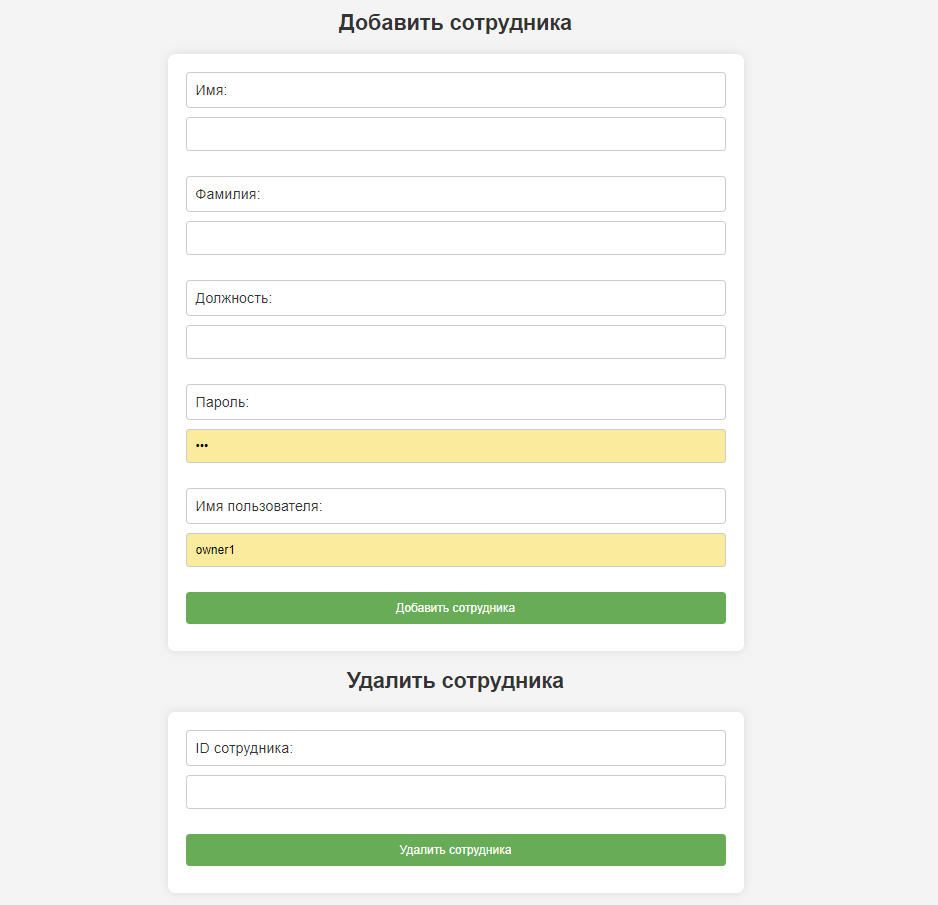


Добавление цветков происходит с помощью функции *add\_flower*, которую мы вызываем при помощи db API, так же, как и получение таблицы цветов с помощью функции *get\_flowers*. Удаление происходит с помощью ORM.

1. Страница управления сотрудниками

На данной странице мы можем просматривать текущий список сотрудников, добавлять и удалять сотрудников.

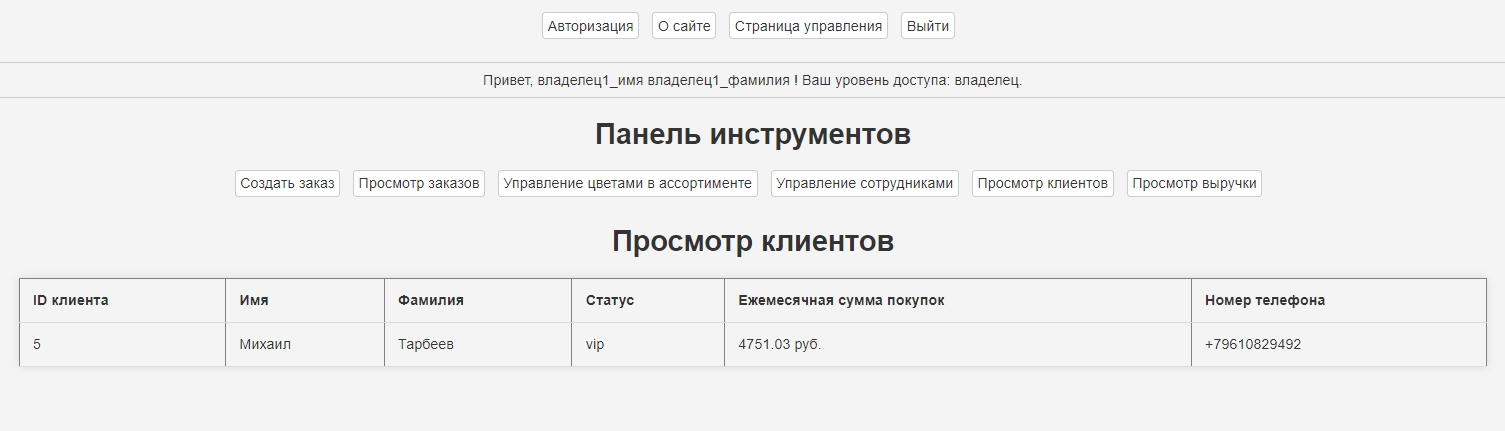




Добавление, удаление и просмотр сотрудников реализован при помощи ORM.

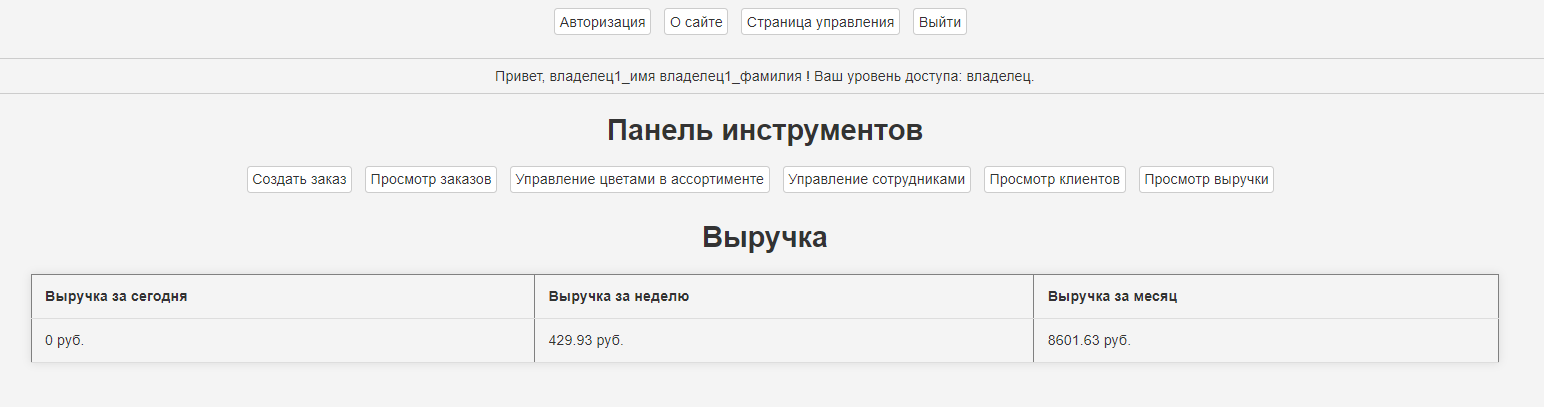
1. Страница просмотра клиентов

На данной странице мы можем просматривать список клиентов и информацию о них. Реализована с помощью ORM.



1. Страница просмотра выручки

На данной странице мы можем просматривать выручку за день, неделю и месяц. Для вывода используется представление *revenue\_summary*, которое вызывается при помощи db API.



Код Python:

from flask import Flask, render\_template, g, request, flash  
from flask import redirect, url\_for  
import psycopg2  
from datetime import datetime  
from flask\_sqlalchemy import SQLAlchemy  
from sqlalchemy import Enum  
from flask\_login import LoginManager, UserMixin, login\_user, login\_required, logout\_user, current\_user  
from sqlalchemy.orm import aliased  
  
# конфигурация  
DATABASE = "FlowerShop"  
DEBUG = True  
SECRET\_KEY = 'fdgfh78@#5?>gfhf89dx,v06k'  
USERNAME = "postgres"  
PASSWORD = "123"  
HOST = "localhost"  
  
app = Flask(\_\_name\_\_)  
app.secret\_key = SECRET\_KEY  
  
app.config['SQLALCHEMY\_DATABASE\_URI'] = 'postgresql://postgres:123@localhost/FlowerShop?client\_encoding=utf8'  
db = SQLAlchemy(app)  
login\_manager = LoginManager(app)  
login\_manager.login\_view = 'login'  
  
  
def connect\_db():  
 conn = psycopg2.connect(dbname=DATABASE, user=USERNAME,  
 password=PASSWORD, host=HOST)  
 return conn  
  
  
def get\_db():  
 *'''Соединение с БД, если оно еще не установлено'''* if not hasattr(g, 'link\_db'):  
 g.link\_db = connect\_db()  
 return g.link\_db  
  
  
@app.teardown\_appcontext  
def close\_db(error):  
 *'''Закрываем соединение с БД, если оно было установлено'''* if hasattr(g, 'link\_db'):  
 g.link\_db.close()  
  
  
@app.route("/FAQ")  
def about():  
 return render\_template('about.html', title="О сайте")  
  
  
class Employees(UserMixin, db.Model):  
 employeeid = db.Column(db.Integer, primary\_key=True)  
 firstname = db.Column(db.String(255), nullable=False)  
 lastname = db.Column(db.String(255), nullable=False)  
 post = db.Column(db.String(50), nullable=False)  
 password = db.Column(db.String(255), nullable=False)  
 username = db.Column(db.String(255), nullable=False, unique=True)  
  
 def get\_id(self):  
 return str(self.employeeid)  
  
  
class Customers(db.Model):  
 \_\_tablename\_\_ = 'customers'  
  
 customerid = db.Column(db.Integer, primary\_key=True)  
 firstname = db.Column(db.String(255))  
 lastname = db.Column(db.String(255))  
 customerstatus = db.Column(Enum('vip', 'common', name='typecustomerstatus'), default='common')  
 monthlypurchaseamount = db.Column(db.Numeric(10, 2), default=0)  
 phonenumber = db.Column(db.String(255))  
  
 def \_\_repr\_\_(self):  
 return f"<Customer(customerid={self.customerid}, firstname={self.firstname}, lastname={self.lastname}, " \  
 f"customerstatus={self.customerstatus}, monthlypurchaseamount={self.monthlypurchaseamount}, " \  
 f"phonenumber={self.phonenumber})>"  
  
  
class Flowers(db.Model):  
 flowerid = db.Column(db.Integer, primary\_key=True)  
 nameflower = db.Column(db.String(255), nullable=False)  
 price = db.Column(db.Float, nullable=False)  
  
  
class Deliveryrates(db.Model):  
 deliveryrateid = db.Column(db.Integer, primary\_key=True)  
 ratename = db.Column(db.String(255), nullable=False)  
 deliverytimehours = db.Column(db.Integer)  
 deliverycost = db.Column(db.Float, nullable=False)  
  
  
class Payments(db.Model):  
 orderid = db.Column(db.Integer, primary\_key=True)  
 paymentcost = db.Column(db.Float, nullable=False)  
 paymentmethod = db.Column(db.String(255), nullable=False)  
 paymentdescription = db.Column(db.String(255), nullable=False)  
 orderstatus = db.Column(db.String(255), nullable=False, default='оплачен')  
 paymentdate = db.Column(db.DateTime, default=datetime.utcnow)  
 deliveryrateid = db.Column(db.Integer, db.ForeignKey('deliveryrates.deliveryrateid'))  
  
  
class Orders(db.Model):  
 id = db.Column(db.Integer, primary\_key=True)  
 customerid = db.Column(db.Integer, nullable=False)  
 flowerid = db.Column(db.Integer, nullable=False)  
 amount = db.Column(db.Integer, nullable=False)  
 ordernumber = db.Column(db.Integer, nullable=False)  
  
  
@login\_manager.user\_loader  
def load\_user(user\_id):  
 return Employees.query.get(int(user\_id))  
  
  
@app.route("/auth", methods=['GET', 'POST'])  
def login():  
 if request.method == 'POST':  
 username = request.form['username']  
 password = request.form['password']  
 user = Employees.query.filter\_by(username=username).first()  
 if user and user.password == password:  
 login\_user(user)  
 return redirect(url\_for('dashboard'))  
 return render\_template('login.html')  
  
  
@app.route('/dashboard', methods=['GET', 'POST'])  
@login\_required  
def dashboard():  
 return render\_template('dashboard.html')  
  
  
@app.route('/view\_revenue', methods=['GET'])  
@login\_required  
def view\_revenue():  
 if current\_user.post == 'владелец':  
 cur = get\_db().cursor()  
  
 # Выполняем запрос к представлению  
 cur.execute("SELECT \* FROM revenue\_summary;")  
  
 # Получаем результат запроса  
 result = cur.fetchall()  
 print(result)  
 # Закрываем курсор и соединение  
 cur.close()  
 get\_db().close()  
 return render\_template('revenue.html', revenue=result)  
 else:  
 return f'''<h2>You have no permission to view revenue</h2><br>Your permission status is {current\_user.post}. <a href="dashboard">Return to Dashboard</a>'''  
  
@app.route('/logout')  
@login\_required  
def logout():  
 logout\_user()  
 return redirect(url\_for('login'))  
  
  
@app.route("/")  
def index():  
 return render\_template('index.html')  
  
  
@app.route('/view\_orders', methods=['GET', 'POST'])  
@login\_required  
def view\_orders():  
 if current\_user.post in ["менеджер", "владелец"]:  
  
 # orders = Payments.query.all()  
  
 orders = (  
 db.session.query(Payments, Orders, Customers, Deliveryrates)  
 .join(Orders, Payments.orderid == Orders.ordernumber)  
 .join(Customers, Orders.customerid == Customers.customerid)  
 .join(Deliveryrates, Payments.deliveryrateid == Deliveryrates.deliveryrateid)  
 .filter(Payments.orderstatus.in\_(['оплачен', 'доставляется'])) # Фильтрация по orderstatus  
 .distinct(Payments.orderid)  
 .all()  
 )  
  
 if request.method == 'POST':  
 order\_id = request.form['order\_id']  
 new\_status = request.form['new\_status']  
  
 order = Payments.query.filter\_by(orderid=order\_id).first()  
 if order:  
 order.orderstatus = new\_status  
 db.session.commit()  
 return redirect(url\_for('view\_orders'))  
 return render\_template('view\_orders.html', orders=orders)  
 else:  
 return f'''<h2>You have no permission to view orders</h2><br>Your permission status is {current\_user.post}. <a href="dashboard">Return to Dashboard</a>'''  
  
  
@app.route('/create\_order', methods=['GET', 'POST'])  
@login\_required  
def create\_order():  
 flowers = Flowers.query.all()  
 delivery\_rates = Deliveryrates.query.all()  
  
 if request.method == 'POST':  
 flower\_ids = request.form.getlist('flowers')  
 flower\_ids.extend(request.form.getlist('flowers[]'))  
 quantities = request.form.getlist('quantities')  
 quantities.extend(request.form.getlist('quantities[]'))  
 delivery\_address = request.form.get('delivery\_address')  
 delivery\_rate\_id = request.form.get('delivery\_rate')  
 payment\_method = request.form.get('payment\_method')  
 order\_firstname = request.form.get('first\_name')  
 order\_lastname = request.form.get('last\_name')  
 order\_description = request.form.get('order\_description')  
 customer\_phone = request.form.get('customer\_phone')  
  
 # Рассчет общей стоимости, учитывая выбранные цветы и тариф доставки  
 total\_cost = calculate\_total\_cost(flower\_ids, quantities, delivery\_rate\_id, customer\_phone)  
  
 # Вызов функции create\_order\_with\_payment  
 order\_id = create\_order\_with\_payment(  
 request.form.get('customer\_phone'),  
 flower\_ids,  
 quantities,  
 delivery\_rate\_id,  
 delivery\_address,  
 payment\_method,  
 order\_description,  
 order\_firstname,  
 order\_lastname,  
 total\_cost  
 )  
 try:  
 cursor = get\_db().cursor()  
 cursor.execute('CALL reset\_customer\_purchase()')  
 cursor.execute('CALL update\_monthly\_purchase\_amount(%s::numeric, %s::varchar)',  
 (total\_cost, customer\_phone))  
 get\_db().commit()  
 except Exception as e:  
 # Обработка ошибок (например, запись в лог или вывод на экран)  
 print(f"Error adding flower: {e}")  
 finally:  
 cursor.close()  
  
 return render\_template('create\_order.html', flowers=flowers, delivery\_rates=delivery\_rates)  
  
  
def create\_order\_with\_payment(customer\_phone, flower\_ids, quantities, delivery\_rate\_id,  
 delivery\_address, payment\_method, order\_description,  
 order\_firstname, order\_lastname, order\_total\_cost):  
 try:  
 # Создание курсора  
 cur = get\_db().cursor()  
 # Формирование строки запроса с плейсхолдерами  
 query = """  
 SELECT create\_order\_with\_payment(  
 %s::VARCHAR, %s::INT[], %s::INT[], %s::INT, %s::VARCHAR,   
 %s::TypePaymentMethod, %s::VARCHAR, %s::VARCHAR, %s::VARCHAR, %s::NUMERIC  
 )  
 """  
 # Вызов функции  
 cur.execute(query, (  
 customer\_phone,  
 flower\_ids,  
 quantities,  
 delivery\_rate\_id,  
 delivery\_address,  
 payment\_method,  
 order\_description,  
 order\_firstname,  
 order\_lastname,  
 order\_total\_cost  
 ))  
  
 # Получение результата (если функция возвращает что-то)  
 result = cur.fetchone()  
 print(result)  
  
 # Подтверждение изменений  
 get\_db().commit()  
  
 # Fetch the result, assuming it's a single integer value  
 order\_id = result[0]  
 return order\_id  
 except Exception as e:  
 print("Error:", e)  
 # Откат изменений в случае ошибки  
 get\_db().rollback()  
  
 finally:  
 # Закрытие курсора (или можете использовать конструкцию with)  
 cur.close()  
  
  
def calculate\_total\_cost(flower\_ids, quantities, delivery\_rate\_id, customer\_phone):  
 try:  
 flower\_prices = {flower.flowerid: flower.price for flower in Flowers.query.all()}  
 delivery\_rate = Deliveryrates.query.get(delivery\_rate\_id).deliverycost  
 customer = Customers.query.filter\_by(phonenumber=customer\_phone).first()  
 if customer.customerstatus == 'vip':  
 mult1 = 0.75  
 mult2 = 0.7  
 else:  
 mult1 = 1  
 mult2 = 1  
 total\_cost = 0.0  
  
 for flower\_id, quantity in zip(flower\_ids, quantities):  
 flower\_id = int(flower\_id)  
 quantity = int(quantity)  
 if flower\_id in flower\_prices:  
 total\_cost += quantity \* flower\_prices[flower\_id] \* mult1  
  
 total\_cost += delivery\_rate \* mult2 # Добавляем стоимость доставки  
  
 return round(total\_cost, 2)  
  
 except Exception as e:  
 print(f"Error calculating total cost: {e}")  
 return None  
  
  
@app.route('/view\_customers')  
def view\_customers():  
 if current\_user.post == "владелец":  
 customers = Customers.query.all()  
 for customer in customers:  
 print(customer.monthlypurchaseamount)  
 return render\_template('view\_customers.html', customers=customers)  
 else:  
 return f'''<h2>You have no permission to view customers</h2><br>Your permission status is {current\_user.post}. <a href="dashboard">Return to Dashboard</a>'''  
  
  
@app.route('/add\_flower', methods=['GET', 'POST'])  
@login\_required  
def add\_flower():  
 if current\_user.post == "владелец":  
 if request.method == 'POST':  
 name = request.form.get('name')  
 price = request.form.get('price')  
 try:  
 cursor = get\_db().cursor()  
 cursor.execute('CALL add\_flower(%s, %s)', (name, price))  
 get\_db().commit()  
 except Exception as e:  
 # Обработка ошибок (например, запись в лог или вывод на экран)  
 print(f"Error adding flower: {e}")  
 finally:  
 cursor.close()  
 return redirect(url\_for('manage\_flowers')) # Перенаправляем на страницу управления цветами  
 else:  
 return f'''<h2>You have no permission to add flowers</h2><br>Your permission status is {current\_user.post}. <a href="dashboard">Return to Dashboard</a>'''  
  
  
# Пример для Flask  
@app.route('/manage\_flowers', methods=['GET', 'POST'])  
@login\_required  
def manage\_flowers():  
 if current\_user.post == "владелец":  
 flowers = Flowers.query.all()  
  
 if request.method == 'POST':  
 # Обработка удаления цветка  
 flower\_id = request.form.get('flower\_id')  
 if flower\_id:  
 try:  
 cursor = get\_db().cursor()  
 cursor.execute('CALL delete\_flower(%s)', (flower\_id,))  
 get\_db().commit()  
 except Exception as e:  
 # Обработка ошибок (например, запись в лог или вывод на экран)  
 print(f"Error deleting flower: {e}")  
 finally:  
 cursor.close()  
  
 return render\_template('manage\_flowers.html', flowers=flowers)  
 else:  
 return f'''<h2>You have no permission to manage flowers</h2><br>Your permission status is {current\_user.post}. <a href="dashboard">Return to Dashboard</a>'''  
  
  
@app.route('/delete\_flower/<int:flower\_id>', methods=['POST'])  
@login\_required  
def delete\_flower(flower\_id):  
 if current\_user.post == "владелец":  
 try:  
 cursor = get\_db().cursor()  
 cursor.execute('CALL delete\_flower(%s)', (flower\_id,))  
 get\_db().commit()  
 except Exception as e:  
 # Обработка ошибок (например, запись в лог или вывод на экран)  
 print(f"Error deleting flower: {e}")  
 finally:  
 cursor.close()  
  
 return redirect(url\_for('manage\_flowers')) # Перенаправляем на страницу управления цветами  
 else:  
 return f'''<h2>You have no permission to delete flowers</h2><br>Your permission status is {current\_user.post}. <a href="dashboard">Return to Dashboard</a>'''  
  
  
@app.route('/add\_employee', methods=['POST', 'GET'])  
@login\_required  
def add\_employee():  
 if current\_user.post == "владелец":  
 if request.method == 'POST':  
 # Обработка добавления сотрудника  
 firstname = request.form['firstname']  
 lastname = request.form['lastname']  
 post = request.form['post']  
 password = request.form['password']  
 username = request.form['username']  
  
 new\_employee = Employees(firstname=firstname, lastname=lastname, post=post, password=password,  
 username=username)  
 db.session.add(new\_employee)  
 db.session.commit()  
  
 # Получите обновленный список сотрудников после добавления  
 employees = Employees.query.all()  
 return render\_template('manage\_employees.html', employees=employees)  
 else:  
 return f'''<h2>You have no permission to add employees</h2><br>Your permission status is {current\_user.post}. <a href="dashboard">Return to Dashboard</a>'''  
  
  
@app.route('/delete\_employee', methods=['POST', 'GET'])  
@login\_required  
def delete\_employee():  
 if current\_user.post == "владелец":  
 if request.method == 'POST':  
 # Обработка удаления сотрудника  
 employee\_id = request.form['employee\_id']  
 employee = Employees.query.get(employee\_id)  
  
 if employee:  
 db.session.delete(employee)  
 db.session.commit()  
  
 employees = Employees.query.all()  
 return render\_template('manage\_employees.html', employees=employees)  
 else:  
 return f'''<h2>You have no permission to delete employees</h2><br>Your permission status is {current\_user.post}. <a href="dashboard">Return to Dashboard</a>'''  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app.run(debug=True)

Полностью веб-приложение, вместе с шаблонами можно скачать:

<https://github.com/tarbeevms/TR_BD_2023>