КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

КАФЕДРА ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Отчёт по учебной практике

Сервер для сервиса продажи DVD-дисков

Выполнил студент

группы 09-652

Сойлу Мурат

Проверил

*Прокопьев Н.А.*

Казань 2018

**Тема работы: Создание серверной части сервиса продажи DVD-дисков, реализующий функционал доступа к покупателям, продавцам, фильмам и сделкам**

**Задания:**

1. Рассмотреть реализацию тренировочного задания «Аггрегатор RSS» для ознакомления с принципами конструирования ПО и реализации веб-сервера по принципам REST.
2. Создать базу данных блога, реализовать программный слой доступа к созданной базе данных.
3. Реализовать слой бизнес-логики по принципам REST для продавцов, покупателей, фильмов и сделок в БД: GET, POST, PUT, DELETE запросы, соответствующие CRUD операциям над данными.
4. Сгенерировать данные для БД таким образом, чтобы суммарно в ней оказалось как минимум 10000 записей.

**Реализация:**

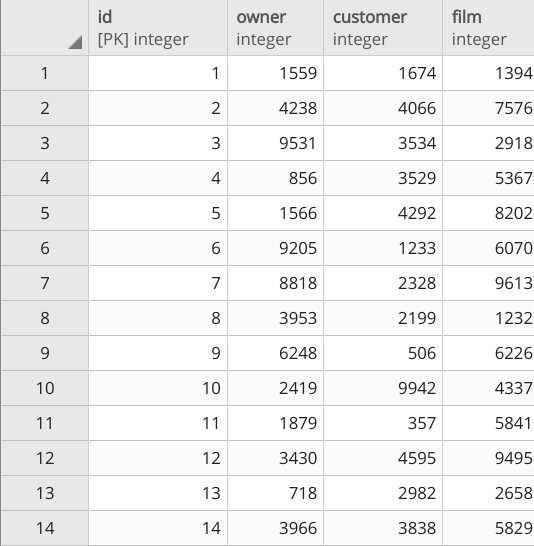
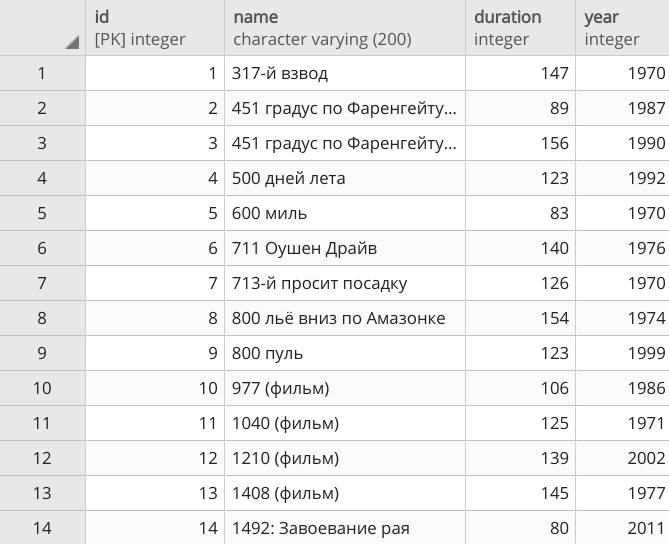
Слой бизнес логики был реализован для ресурсов блога. Были созданы обработчики запросов для совершения операций Get-для вывода всех ресурсов данной таблицы, Get id-вывод конкретного ресурса , Post-добавление ресурсов ,DELETE id-удаление конкретного ресурса.

Была создана БД для:

Таблица “owner” Таблица “customer”



Таблица “film” Таблица “deal”



Было рассмотрено приложение Агрегатор RSS.

Для реализации веб-сервера был выбран язык Python и фреймворк Flask а также библиотеки:

1. flask\_sqlalchemy
2. Psycopg2
3. Sqlalchemy

Поскольку Flask относится к категории так называемых микрофреймворков

— минималистичных каркасов веб-приложений, сознательно предоставляющих лишь самые базовые возможности, что полезно для понимания основных принципов создания серверной части web-приложения.

Слой доступа к БД был реализован ORM flask-sqlalchemy для управления нашим приложением мы будем использовать расширение Flask-SQLAlchemy. Это расширение предоставляет собой обертку для проекта SQLAlchemy, который является ORM или Объектно-реляционным отображением (англ. Object-relational mapping).  
  
ORM позволяет приложениям БД работать с объектами вместо таблиц или SQL. Операции выполняются над объектами, а потом прозрачно транслируются в команды БД при помощи ORM. Фактически это означает, что мы не будем изучать SQL в этом руководстве, а позволим Flask-SQLAlchemy говорить на SQL за нас.

Для генерации данных в БД был использован sql-запрос к БД PostgreSQL

**Листинг кода**

@app.route('/api/customer', methods=['GET'])

def api\_customer\_get():

customers = Customer.query.all()

customers\_json = [{"id": customer.id, "name": customer.name}

for customer in customers]

return jsonify(customers\_json)

@app.route('/api/customer/<id>', methods=['GET'])

def api\_customer\_get\_id(id):

customers = Customer.query.filter\_by(id=id)

if not customers:

abort(404)

customer = customers[0]

customer\_json = {"id": customer.id, "name": customer.name}

return jsonify(customer\_json)

@app.route('/api/customer', methods=['POST'])

def api\_customer\_insert():

new\_customer = request.get\_json()

customer = Customer(id=new\_customer['id'], name=new\_customer['name'])

db.session.add(customer)

db.session.commit()

customer\_json = {"id": customer.id, "name": customer.name}

return jsonify(customer\_json)

@app.route('/api/customer/<id>', methods=['DELETE'])

def api\_customer\_delete(id):

customers = Customer.query.filter\_by(id=id)

if not customers:

abort(404)

customer = customers[0]

db.session.delete(customer)

db.session.commit()

return jsonify()

@app.route('/api/customer/<id>', methods=['PUT'])

def api\_customer\_update(id):

updated\_customer = request.get\_json()

customers\_to\_update = Customer.query.filter\_by(id=id)

data = json.loads(request.get\_data())

customer\_to\_update = customers\_to\_update[0]

customer\_to\_update = db.session.query(Customer).filter\_by(id = id).first()

customer\_to\_update.name = data['name']

db.session.commit()

return jsonify(customer\_to\_update.to\_dict()