Государственное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и ИТ»

Р А С Ч Е Т Н О - П О Я С Н И Т Е Л Ь Н А Я З А П И С К А

к курсовому проекту на тему:

Распределённая система поиска и оценки деталей для ремонта мобильных телефонов

Студент_____ (Аринкин Н.А.)
(ф.и.о.)

Руководитель курсового проекта_____ (Ковтушенко А. П.)
(ф.и.о.)

Москва, 2015

Оглавление

Введение

В современном мире почти у каждого человека имеется мобильный телефон или смартфон, а иногда и несколько. Являясь устройством, который используются для многих задач по несколько часов в день, мобильные телефоны часто ломаются, разбиваются, перестают работать. Перед людьми, занимающимися ремонтом смартфонов, или тем, кто самостоятельно решил отремонтировать своё устройство, стоит задача поиска запчастей для дальнейшей их замены. Выбор и поиск деталей для мобильных телефонов осложняется отсутствием информации о них, о производителе, о качестве запчасти.

1.1. Постановка задачи

Целью данной курсовой работы разработка и создание сложной многосоставной программной системы, предоставляющая пользователю информацию о запасных частях для мобильных телефонов, отзывы о них от других пользователей, а также инструкции по замене необходимой детали. Среди основных задач работы можно выделить несколько основных:

- Архитектура системы должна следовать принципам SOA;
- Взаимодействие сервисов должно происходить по HTTP-протоколу, учитывая рекомендации REST;

Также система должна иметь пользовательский интерфейс, при помощи которой пользователей может оставить свой отзыв о детали или перейти на магазин, в котором она представлена для покупки.

1.2. Актуальность и новизна разработки

В интернете есть много сайтов, предлагающие купить различные детали для смартфонов, однако нет ресурса, дающего полную информацию о качестве, совместимости детали с тем или иным устройством, а также способ её замены.

2. Конструкторская часть

2.1. Проектирование системы

2.1.1. Сценарии использования

П1. Регистрация

Исполнитель: Пользователь.

Предусловие: Пользователь авторизован.

Основной сценарий:

1. Пользователь осуществляет ввод следующей информации: имя пользователя, адрес электронной почты, пароль, подтверждение пароля;
2. Система сохраняет соответствующую информацию о пользователе.

Альтернативные сценарии:

2а. Недопустимые пользовательские данные

1. Система уведомляет пользователя об ошибке.

2б. Имя пользователя не уникально

1. Система уведомляет пользователя об ошибке.

2в. Адрес электронной почты не уникально

1. Система уведомляет пользователя об ошибке.

П2. Добавление нового устройства

Исполнитель: Администратор.

Предусловие: Администратор авторизован.

Основной сценарий:

1. Администратор осуществляет ввод следующей информации: производитель, название модели, номер модели, дата выпуска, характеристики;
2. Система сохраняет соответствующую информацию об устройстве.

Альтернативные сценарии:

2а. Недопустимые данные об устройстве

1. Система уведомляет администратора об ошибке.

2б. Название и номер модели не уникальны

1. Система уведомляет администратора об ошибке.

2в. Ошибка при сохранении

1. Система уведомляет администратора об ошибке.

П3. Добавление отзыва и оценки

Исполнитель: Пользователь.

Предусловие: Пользователь авторизован, выбрана деталь.

Основной сценарий:

1. Пользователь для выбранной детали вводит оценку и пишет отзыв;
2. Система сохраняет соответствующую информацию об устройстве.

Альтернативные сценарии:

2а. Недопустимые данные при вводе

1. Система уведомляет пользователя об ошибке.

2б. Ошибка при сохранении

1. Система уведомляет пользователя об ошибке.

П4. Просмотр отзывов и оценок к детали

Исполнитель: Пользователь.

Предусловие: Пользователь авторизован.

Основной сценарий:

1. Пользователь запрашивает оценки и отзывы по нужной детали;
2. Система возвращает пользователю отзывы и оценки по выбранной детали.

Альтернативные сценарии:

2а. Для детали нет отзывов и оценок

1. Система сообщает об этом пользователю.

П5. Поиск и просмотр информации о производителе устройства

Исполнитель: Пользователь.

Предусловие: Пользователь авторизован.

Основной сценарий:

1. Пользователь запрашивает названия производителя;
2. Система возвращает информацию о производителе.

Альтернативные сценарии:

2а. Ошибка в запросе

1. Система сообщает об этом пользователю об ошибке.

2б. Нет такого производителя

1. Система сообщает об этом пользователю.

П6. Поиск и просмотр информации об устройстве

Исполнитель: Пользователь.

Предусловие: Пользователь авторизован.

Основной сценарий:

1. Пользователь запрашивает названия модели, или номер модели;
2. Система возвращает подходящие запросу модели устройств и информацию о них.

Альтернативные сценарии:

2а. Ошибка в запросе

1. Система сообщает об этом пользователю об ошибке.

2б. Нет такого устройства

2. Система сообщает об этом пользователю.

Отношения между прецедентами и актерами представлены на Рис.1.

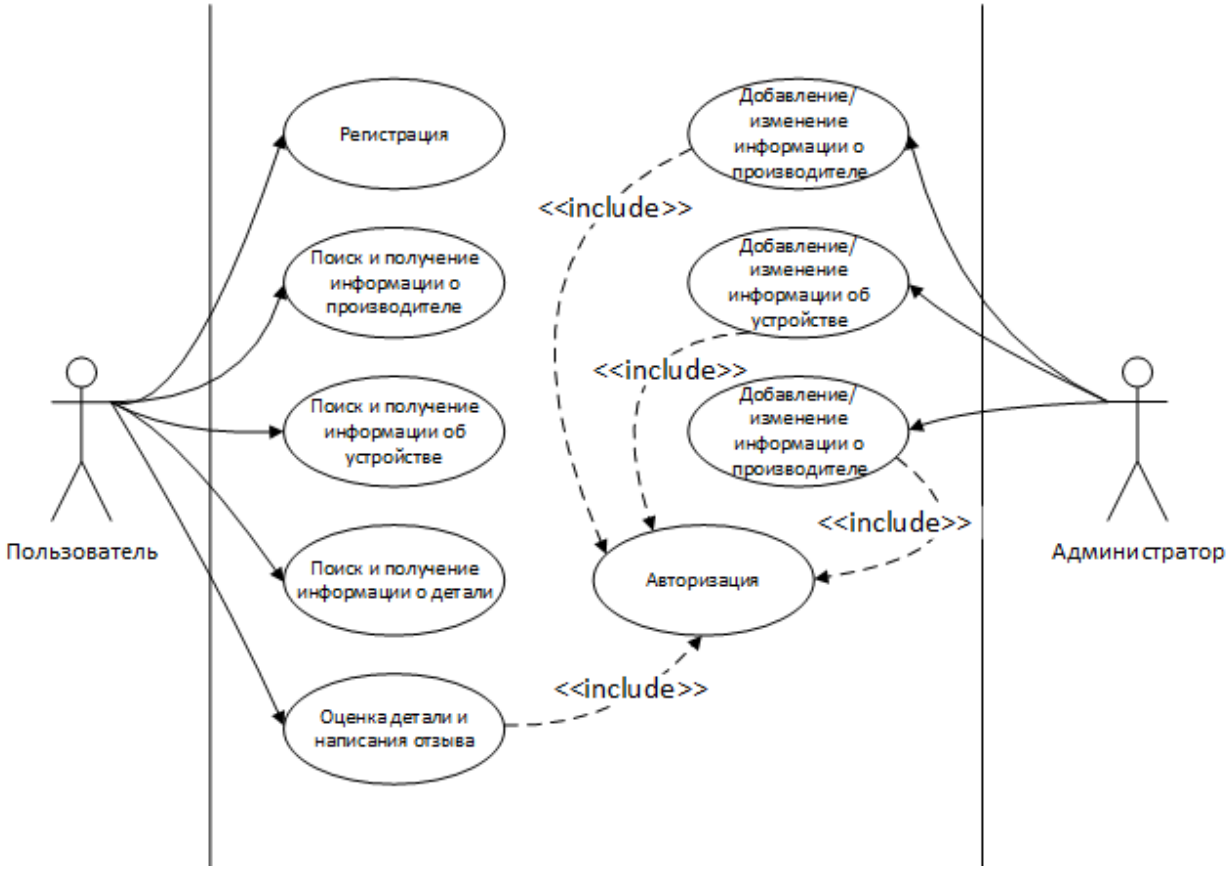


Рис 1. Диаграмма прецедентов

2.1.2. Основные сущности

Выделены следующие основные сущности системы:

- Пользователь
- Производитель
- Устройство
- Деталь
- Отзыв
- Оценка

Рассмотрим подробнее описанные выше сущности.
Пользователь – информация о пользователе:

- Имя
- Фамилия
- Адрес электронной почты
- Пароль

Производитель – информации о производителе устройств:

- Название
- Страна
- Год основания
- Краткая справка

Устройство – информация об устройстве:

- Название модели
- Номер модели
- Производитель
- Дата выхода
- Характеристики

Деталь – информация об детали для устройства:

- Тип детали
- Устройства
- Номер детали
- Производитель
- Оригинал/копия/подделка
- Информация
- Средняя оценка

Отзыв – информация, оставленная пользователем о детали для устройства

- Пользователь
- Деталь
- Оценка
- Описание(отзыв)

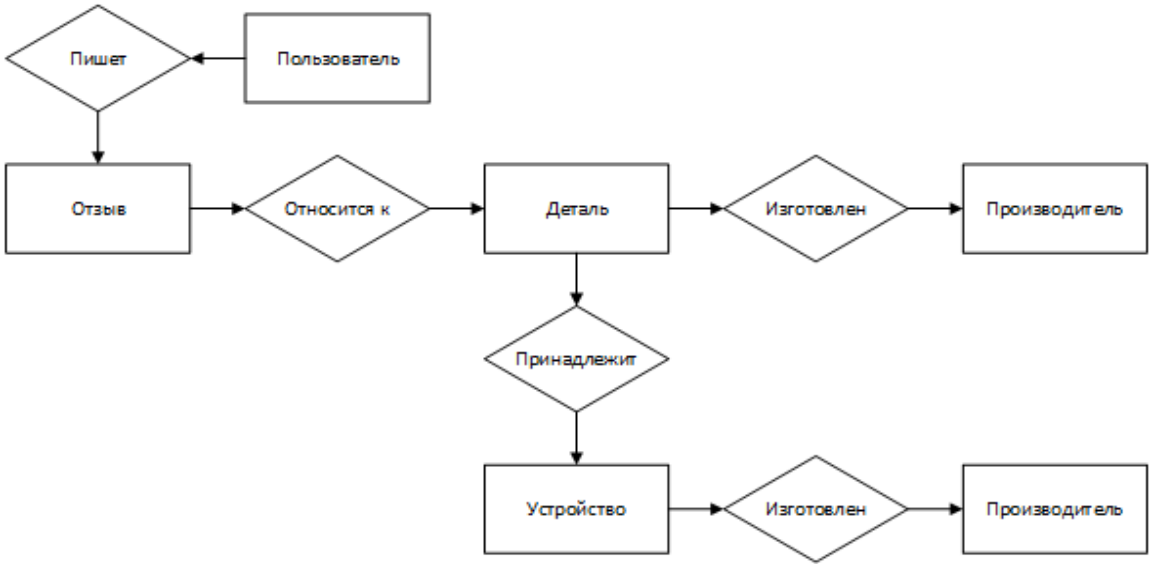


Рис 2. ER-диаграмма

2.2. Архитектура системы

Согласно требованиям, описанных в Пункте 1.1, к данному курсовому проекту, система является сервис-ориентированной.

Выделены следующие сервисы:

1. Фронтенд;
2. Сессия;
3. Пользователи;
4. Производители;
5. Устройства;
6. Детали;
7. Отзывы

2.2.1. Сервис “Сессия”

Задачами данного сервиса являются:

1. Авторизация пользователей в системе. Представляет собой проверку соответствия между именем пользователя и паролем. Если авторизация успешна, то создается сессия с уникальным идентификатором, который передается клиенту. Этот идентификатор сессии в дальнейшем используется для проверки состояния авторизации. Время жизни сессии ограничено по времени, по истечению которого соответствующая идентификатору сессия будет уничтожена.
2. Управление пользовательскими сессиями.

Для данного сервиса можно выделить следующие сущности:

- Пользователь
- Сессия

Отметим, что в сервисе сессии не хранится полная информация о пользователе, а лишь информация необходимая для его авторизации – имя пользователя и пароль.

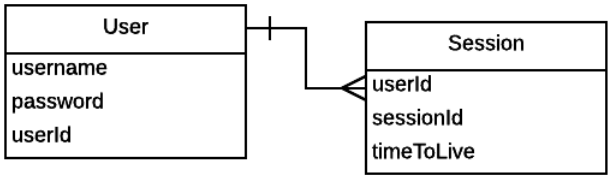


Рис 3. Сущности сервиса “Сессия”

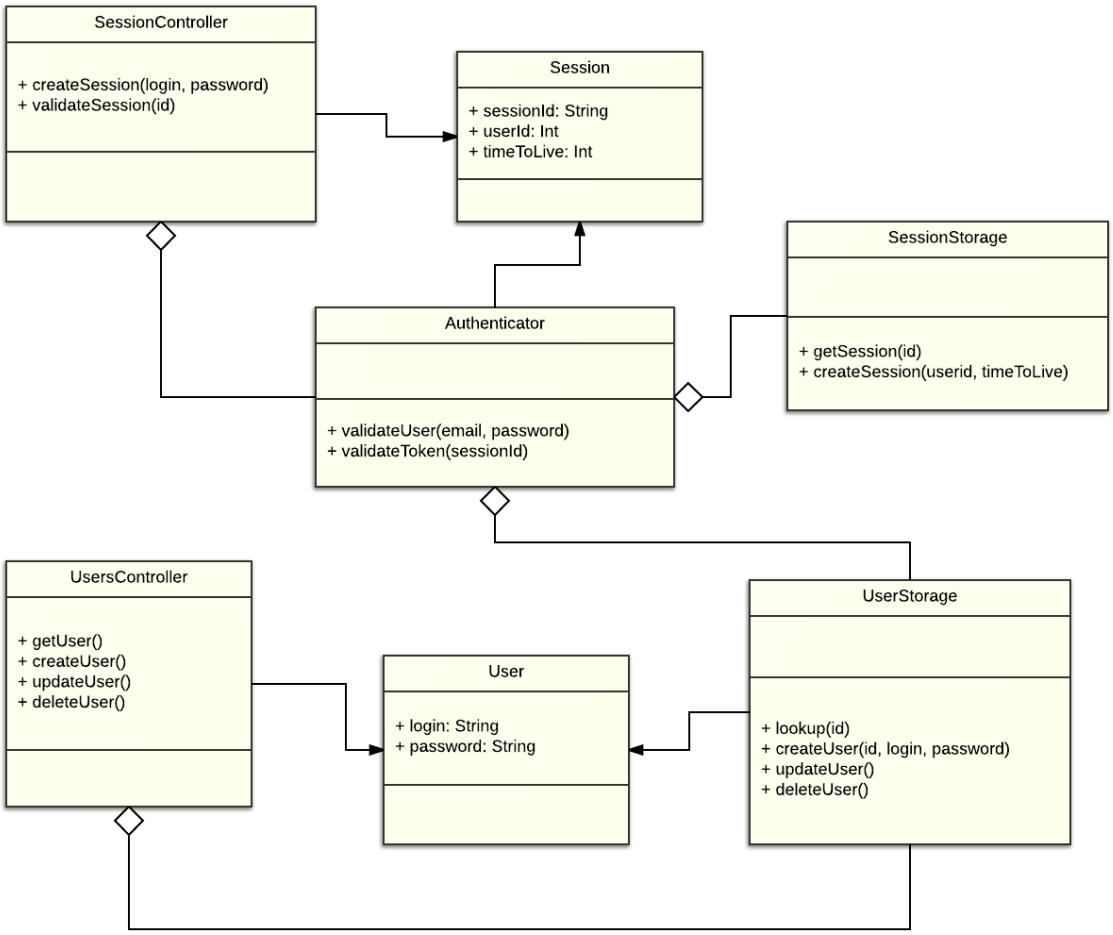


Рис 4. Диаграмма классов UML для бекэнда “Сессия”

На рисунке «**Ошибка! Источник ссылки не найден.** 4» представлены основные классы данного сервиса. Рассмотрим некоторые из них:

Классы SessionController и UsersController принимают и отвечают на HTTP запросы. SessionController отвечает за запросы на создание и валидацию сессии. UsersController отвечает за запросы на получение, создание, обновление и удаление пользователя. Отметим, что реализация всех этих операций находится в других классах, (UserStorage, SessionStorage и Authenticator). Сами контроллеры реализуют только получение параметров из HTTP запроса и отправку ответа.

Authenticator отвечает за проверку авторизационных данных пользователя и создание сессии. А также проверку идентификатора сессии на действительность.

UserStorage – осуществляет взаимодействие с базой данных пользователей. Реализует такие операции как поиск, создание, обновление и удаление пользователей.

SessionStorage – осуществляет взаимодействие с хранилищем сессий. Предоставляет возможность создания и получения сессии по идентификатору.

2.2.2. Сервис “Устройства”

Данный сервис служит для хранения и предоставления пользователям информации об устройствах. Так же администратором предоставляется возможность добавления новых устройств, удаления, изменения информации о устройствах.

2.2.3. Сервис “Детали”

Данный сервис служит для хранения и предоставления пользователям информации о деталях для устройств. Так же администратором предоставляется возможность добавления новых деталей, удаления, изменения информации о деталях.

2.2.4. Сервис “Отзывы”

Сервис “Отзывы” служит для проставления пользователям возможности оценить и оставить отзыв для выбранной детали

2.2.5. Фронтэнд

Данный сервис реализует высокоуровневую логику работы система. Основные задачи, реализуемые на данном сервисе:

1. Обработка запросов пользователей;
2. Формирование ответы;
3. Агрегирование результатов;
4. Взаимодействие с сервисами(бекэндами), описанными выше.

Классы Controller на бекэндах обрабатывают HTTP-запросы:

1. Считывают параметры запросов, при условии их существования;
2. Делигируют сформированные запросы соответствующим классам;
3. Сериализуют полученные результаты в JSON-формате и формируют ответ.

3. Технологическая часть

3.1. Языки и средства разработки

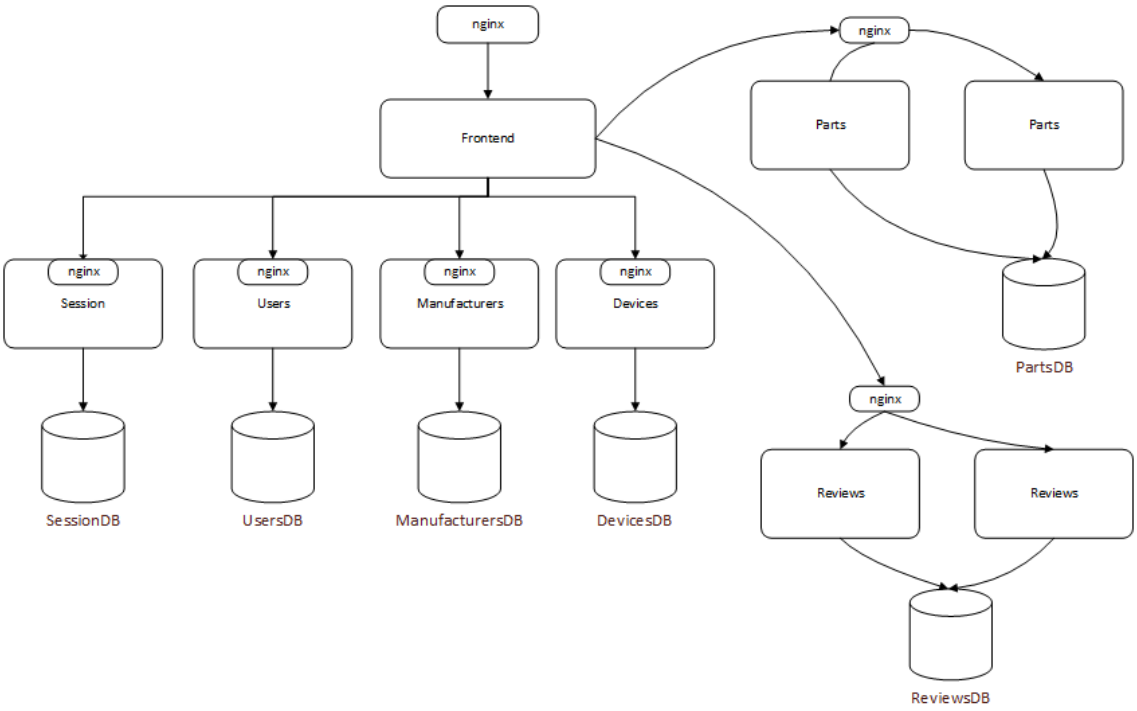
В качестве основного языка был выбран язык Python 3.5. В качестве фреймворка использовался Django 1.8.7 - свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC.

Для разработки баз данных использовался СУБД – SQLite компактная встраиваемая реляционная база данных.

Для хранения информации о сессии используется хранилище Redis. Redis является нереляционным хранилищем типа «ключ-значение», которое хранит данные в памяти. Несмотря на это, redis позволяет восстанавливать данные в случае непредвиденного отключения, так как он позволяет дублировать данные на диске. Для этого Redis предоставляет два механизма. Первый - механизм снимков, в котором данные асинхронно переносятся из оперативной памяти в файл. Второй - механизм журналирования, в котором все операции, изменявшие данные в памяти, записываются в специальный файл, на основе которого можно в дальнейшем восстановить состояние.

3.2. Структура системы

На рисунке изображена структура системы. Одним из преимуществ сервис-ориентированной архитектуры является возможность запуска нескольких экземпляров одного и того же сервиса и балансировки нагрузки между ними. В данном случае балансировка была произведена для бекэндов фильмов и рекомендаций. Балансировка производится при помощи http сервера nginx. Nginx был выбран в связи с простотой его установки и настройки. Балансировка происходит по дисциплине round-robin, то есть сервисы выбираются один за другим циклически.



Заключение

При выполнении курсового проекта был получен опыт проектирования и разработки распределенной системы обработки информации. Была разработана распределённая система, предоставляющая пользователям получать информацию о деталях для смартфонов, а также оценивать их. В будущем планирует добавить возможность перехода в магазины с лучшей ценой для покупки деталей.

Список использованных источников

1. . Django documentation.[Электронный ресурс]. https://docs.djangoproject.com/
2. Django book.[Электронный ресурс]. http://djbook.ru/

Приложение А

Фрагменты кода

```
from django.shortcuts import render
from django.views.decorators.csrf import csrf_exempt
from django.http import HttpResponse
from django.core.exceptions import ObjectDoesNotExist
import logging
from manufacturers.models import Manufacturer
import json
import requests
import os
from django.conf import settings

@csrf_exempt
def list(request):
      if request.method == "GET":
            manufacturers = Manufacturer.objects.all()
            results = [ob.as_json() for ob in manufacturers]
            result = {"count": Manufacturer.objects.count(),
"manufacturers": results}
            return HttpResponse(json.dumps(result))
      return HttpResponse("Ok")

def list_num(request, manufacturer_id):
      if request.method == "GET":
            logger = logging.getLogger('backend_device')
            try:
                  manufacturer =
Manufacturer.objects.get(manufacturer_id)
                  results = manufacturer.as_json()
                  result = {"manufacturer": results}
                  data = json.dumps(result)
                  logger.info(data)
                  return HttpResponse(data)
            except ObjectDoesNotExist:
                  data = json.dumps({"manufacturer": 0})
                  logger.info(data)
                  return HttpResponse(json.dumps(result), status=404)
      return HttpResponse("Ok")

@csrf_exempt
def remove(request):
      if request.method == "POST":
            logger = logging.getLogger('backend_manufacturer')
            data = request.body
            data = json.loads(data.decode('utf8'))
            frontend_key = data["frontend_key"]
            manufacturer_id = data["manufacturer_id"]
```

```
            post_data = {"session_key": session_key}
            headers = {'Content-type': 'application/json'}
            check =
requests.post("http://localhost:8000/session/check/",
data=json.dumps(post_data), headers=headers)
            if check.status_code == requests.codes.ok:
                  try:
                        manufacturer =
Manufacturer.objects.get(pk=manufacturer_id)
                        manufacturer.delete()
                        data = json.dumps({"info":"Success deleting"})
                        logger.info(data)
                        return HttpResponse(data)
                  except ObjectDoesNotExist:
                        with open(os.path.join(settings.BASE_DIR,
"static/jsons/manufacturer_not_found.json")) as data_file:
                              data = json.load(data_file)
                        logger.info(data)
                        return HttpResponse(json.dumps(data), status=404)
            else:
                  with open(os.path.join(settings.BASE_DIR,
"static/jsons/error_check.json")) as data_file:
                        data = json.load(data_file)
                  logger.info(data)
                  return HttpResponse(json.dumps(data), status=401)

      return HttpResponse("Ok")

@csrf_exempt
def add(request):
      if request.method == "POST":
            logger = logging.getLogger('backend_manufacturer')
            data = request.body
            data = json.loads(data.decode('utf8'))
            session_key = data["session_key"]
            name = data["name"]
            established = data["established"]
            country = data["country"]
            post_data = {"session_key": session_key}
            headers = {'Content-type': 'application/json'}
            check =
requests.post("http://localhost:8000/session/check/",
data=json.dumps(post_data), headers=headers)
            if check.status_code == requests.codes.ok:
                  try:
                        manufacturer =
Manufacturer.objects.create(name=name,established=established,country=
country)
                        manufacturer.save()
                        data = json.dumps({"info":"Success adding"})
                        logger.info(data)
```

```
                    return HttpResponse(data)
                except ObjectDoesNotExist:
                    with open(os.path.join(settings.BASE_DIR,
"static/jsons/manufacturer_not_found.json")) as data_file:
                        data = json.load(data_file)
                    logger.info(data)
                    return HttpResponse(json.dumps(data), status=404)
          else:
                with open(os.path.join(settings.BASE_DIR,
"static/jsons/error_check.json")) as data_file:
                    data = json.load(data_file)
                logger.info(data)
                return HttpResponse(json.dumps(data), status=401)

     return HttpResponse("Ok")
```

Листинг 1. Код бэкенда производители