

**НИУ ИТМО**  
**ФАКУЛЬТЕТ ИКТ**

**Основы web-программирования**

Отчет

по лабораторной работе №2-3

**«Реализация WEB-сервисов средствами Django REST framework, Vue.js, Muse-UI»**

**Выполнила:**

Суздальцева Маргарита

Студентка группы К3343

**Проверил:**

Говоров Антон Игоревич

Санкт-Петербург

**2020**

### ***Цель работы***

овладеть практическими навыками и умениями реализации web-сервисов средствами Django REST framework, Vue.js, Muse-UI.

### ***Программное обеспечение***

Python 3.6, Django REST framework, Vue.js, Muse-UI

(или аналогичная библиотека), PostgreSQL \*.

Вместо Muse-UI использовала **Vuetify**.

### ***Задание***

#### **Задание 8**

Создать программную систему, предназначенную для отдела маркетинга рекламного агентства.

Одной из задач, решаемых отделом маркетинга рекламного агентства «Луч», является учет работы с клиентами. Для этого необходимо организовать оперативный учет поступивших и выполненных заявок клиентов (рекламодателей).

Рекламное агентство заключает трудовые соглашения с заказчиками на исполнение определенного вида рекламных услуг. Для оформления заявки рекламодатель должен указать контактное лицо, телефон и электронный адрес для связи. Рекламодатель оформляет заявку на рекламу, пользуясь прайс–листом, в котором указаны цены по наименованию рекламных услуг, предоставляемых агентством «Луч». Здесь же оговариваются исполнители изготовления рекламы (сотрудники агентства), стоимость и объем (количество) работ. Для выполнения работ необходимо знать единицы измерения и материалы. Заказчик должен иметь контактные данные исполнителя.

Согласно заявке, выписывается Платежное Поручение Заказчику, которое он обязан оплатить.

После оплаты счета агентство обязуется предоставить рекламные продукты. Заказ считается выполненным, если оплачено Платежное поручение.

Перечень возможных запросов к базе данных:

- список выполненных работ, фиксирующих дату оплаты заявки, заказчиков, код услуги, фамилию исполнителя;

- список платежных поручений, выставленных рекламодателям за любой промежуток времени, фиксирующий заказчика, услугу, состояние заявки (оплачено или нет);
- просмотр номенклатуры рекламных услуг, предлагаемых агентством по видам услуг;
- список заявок, заключенных каждым отдельным заказчиком за любой промежуток времени;
- список сотрудников с указанием количества заявок, которые выполнял каждый сотрудник в заданный период.

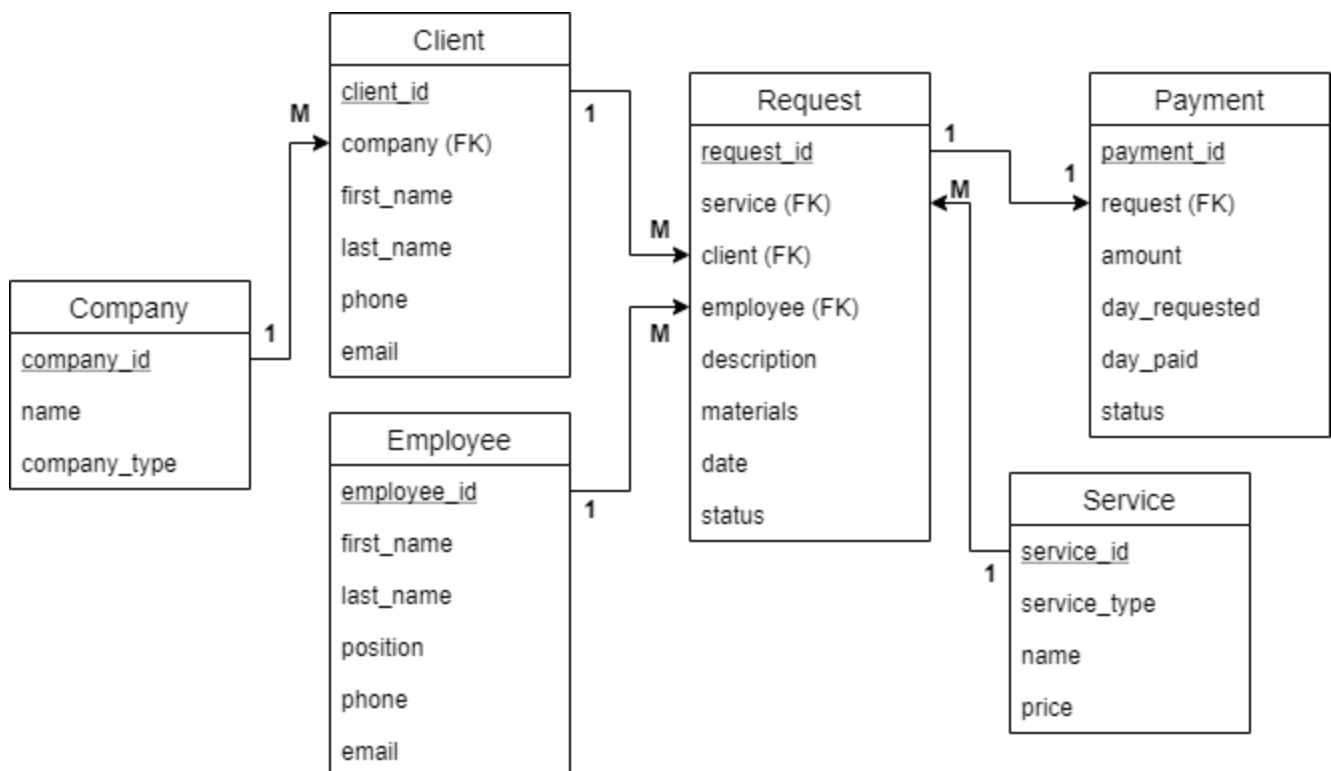
Перечень возможных отчетов:

- отчет об объеме (стоимости) работ, выполненных всеми исполнителями, за последний квартал.

### *Ход работы*

Схема БД:

## marketingdb



### Описание БД:

- Клиент (id, Компания (FK), ФИО, Телефон, Email)
- Услуга (id, Вид, Наименование, Цена)
- Сотрудник (id, ФИО, Должность, Телефон, Email)
- Заявка (id, Услуга (FK), Клиент (FK), Сотрудник (FK), Краткое описание, Материалы, Дата, Статус)
- Платёж (id, Заявка (FK), Сумма, Дата запроса, Дата оплаты, Статус)
- Компания (id, Название, Тип)

### Заявленные интерфейсы:

- 1) Главная страница: краткая информация об агентстве и прайс-лист
- 2) Регистрация, авторизация для клиентов и сотрудников с разными правами
- 3) Форма подачи заявки для зарегистрированных клиентов
- 4) Личный кабинет клиента: информация о заявках
- 5) Личный кабинет сотрудника: информация по текущим и прошлым заявкам
- 6) Страница со статистикой для руководителей (запросы к базе из варианта)

## **Часть 1. Backend: Django API с помощью DRF**

### Установленные приложения в settings.py:

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'corsheaders',  
    'rest_framework',  
    'rest_framework.authtoken',  
    'djoser',  
    'luchapp'  
]
```

### Подключение БД:

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.postgresql',  
        'NAME': 'marketingdb',  
        'USER': 'rita',  
        'PASSWORD': 'Admin666',  
        'HOST': 'localhost',  
        'PORT': '5433'  
    }  
}
```

## Модели в models.py:

```
1  from django.db import models
2  from django.contrib.auth.models import User
3
4  class Company(models.Model):
5      name = models.CharField(max_length=50)
6      TYPES = [
7          ('1', 'ИП'),
8          ('2', 'ООО'),
9          ('3', 'ООО'),
10         ('4', 'ЗАО'),
11         ('5', 'ПТ'),
12         ('6', 'ТНВ'),
13         ('7', 'ГКП')
14         # и другие виды ОПФ
15     ]
16     company_type = models.CharField(max_length=1, choices=TYPES)
17
18
19 class Service(models.Model):
20     name = models.CharField(max_length=100)
21     TYPES = [
22         ('1', 'реклама в СМИ'),
23         ('2', 'наружная реклама'),
24         ('3', 'реклама на транспорте'),
25         ('4', 'реклама на месте продаж'),
26         ('5', 'сувенирка'),
27         ('6', 'печатная реклама'),
28         ('7', 'директ-реклама'),
29         ('8', 'реклама в интернете'),
30         ('9', 'ивент-реклама'),
31         ('10', 'другое')
32         # и т.д. и т.п.
33     ]
34     service_type = models.CharField(max_length=2, choices=TYPES)
35     price = models.IntegerField()
36
37
38
39 class Employee(models.Model):
40     user = models.ForeignKey(User, on_delete=models.CASCADE)
41     POSITIONS = [
42         ('e', 'executor'),
43         ('h', 'head')
44     ]
45     position = models.CharField(max_length=1, choices=POSITIONS, default='e')
46     phone = models.CharField(max_length=14)
47
48
49 class Client(models.Model):
50     user = models.ForeignKey(User, on_delete=models.CASCADE)
51     company = models.ForeignKey(Company, on_delete=models.CASCADE)
52     phone = models.CharField(max_length=14)
53
54
55 class Request(models.Model):
56     service = models.ForeignKey(Service, on_delete=models.CASCADE)
57     client = models.ForeignKey(Client, on_delete=models.CASCADE)
58     employee = models.ForeignKey(Employee, on_delete=models.CASCADE, default="1")
59     description = models.CharField(max_length=280)
60     materials = models.CharField(max_length=280)
61     date = models.DateTimeField(auto_now_add=True, null=True)
62     STATUSES = [
63         ('0', 'in progress'),
64         ('1', 'completed')
65     ]
66     status = models.CharField(max_length=1, choices=STATUSES, default='0')
```

```

66
67
68 class Payment(models.Model):
69     request = models.ForeignKey(Request, on_delete=models.CASCADE)
70     amount = models.IntegerField()
71     day_requested = models.DateTimeField()
72     day_paid = models.DateTimeField()
73     STATUSES = [
74         ('0', 'unpaid'),
75         ('1', 'paid')
76     ]
77     status = models.CharField(max_length=1, choices=STATUSES, default='0')

```

## Сериалайзеры в serializers.py:

```

1  from rest_framework import serializers
2  from .models import *
3  from django.contrib.auth.models import User
4
5
6  class UserSerializer(serializers.ModelSerializer):
7      class Meta:
8          model = User
9          fields = ['id', 'username', 'email', 'first_name', 'last_name']
10
11
12  class NewCompanySerializer(serializers.ModelSerializer):
13      class Meta:
14          model = Company
15          fields = '__all__'
16
17
18  class CompanySerializer(serializers.ModelSerializer):
19      company_type = serializers.CharField(source='get_company_type_display')
20      class Meta:
21          model = Company
22          fields = '__all__'
23
24  class ServiceSerializer(serializers.ModelSerializer):
25      service_type = serializers.CharField(source="get_service_type_display")
26      class Meta:
27          model = Service
28          fields = '__all__'
29
30
31  ~
32  class ClientSerializer(serializers.ModelSerializer):
33      user = UserSerializer(read_only=True)
34      company = CompanySerializer(read_only=True)
35
36      class Meta:
37          model = Client
38          fields = '__all__'
39
40  class NewClientSerializer(serializers.ModelSerializer):
41      class Meta:
42          model = Client
43          fields = '__all__'
44

```

```

45
46 class EmployeeSerializer(serializers.ModelSerializer):
47     user = UserSerializer()
48     position = serializers.CharField(source="get_position_display")
49
50     class Meta:
51         model = Employee
52         fields = '__all__'
53
54
55 class CreateRequestSerializer(serializers.ModelSerializer):
56     class Meta:
57         model = Request
58         fields = '__all__'
59
60
61 class RequestSerializer(serializers.ModelSerializer):
62     status = serializers.CharField(source="get_status_display")
63     service = ServiceSerializer(read_only=True)
64     client = ClientSerializer(read_only=True)
65     employee = EmployeeSerializer(read_only=True)
66     class Meta:
67         model = Request
68         fields = '__all__'
69
70
71 class PaymentSerializer(serializers.ModelSerializer):
72     class Meta:
73         model = Payment
74         fields = '__all__'

```

## views.py:

```

1 from django.shortcuts import render
2 from rest_framework.response import Response
3 from rest_framework.views import APIView
4 from rest_framework import generics
5 from django.db.models import Q
6 from django.contrib.auth import get_user_model
7
8 from .serializers import *
9 from .models import *
10
11 # Requests
12 class CreateRequestView(generics.CreateAPIView):
13     queryset = Request.objects.all()
14     serializer_class = CreateRequestSerializer
15
16
17 class GetRequestView(generics.RetrieveUpdateDestroyAPIView):
18     queryset = Request.objects.all()
19     serializer_class = RequestSerializer
20
21
22 class GetRequestsView(generics.ListAPIView):
23     serializer_class = RequestSerializer
24     def get_queryset(self):
25         queryset = Request.objects.all()
26         user = self.request.query_params.get('user', None)
27         #client = self.request.query_params.get('client', None)
28         #employee = self.request.query_params.get('employee', None)
29         #if client:
30             # queryset = queryset.filter(client=client)
31         if user:
32             queryset = queryset.filter(Q(employee__user__username=user) | Q(client__user__username=user))
33         return queryset

```

```

36 class AdminFilterView(generics.ListAPIView):
37     serializer_class = RequestSerializer
38
39     def get_queryset(self):
40         queryset = Request.objects.all()
41         params = self.request.query_params
42
43         service = params.get('service', None)
44         client = params.get('client', None)
45         employee = params.get('employee', None)
46         status = params.get('status', None)
47         after = params.get('after', None)
48         before = params.get('before', None)
49
50         if service:
51             queryset = queryset.filter(service__id=service)
52
53         if client:
54             queryset = queryset.filter(client__id=client)
55
56         if employee:
57             queryset = queryset.filter(employee__id=employee)
58
59         if status:
60             queryset = queryset.filter(status=status)
61
62         if after:
63             queryset = queryset.filter(date__date__gte=after)
64
65         if before:
66             queryset = queryset.filter(date__date__lte=before)
67
68         return queryset
69
70
71 # Company & Client
72 class CreateClientView(generics.CreateAPIView):
73     queryset = Client.objects.all()
74     serializer_class = NewClientSerializer
75
76
77 class CreateCompanyView(generics.CreateAPIView):
78     queryset = Company.objects.all()
79     serializer_class = NewCompanySerializer
80
81
82 class GetClientView(generics.ListAPIView):
83     serializer_class = ClientSerializer
84     def get_queryset(self):
85         params = self.request.query_params
86         user = params.get('user', None)
87         queryset = Client.objects.all()
88         if user:
89             queryset = queryset.filter(user__username=user)
90         return queryset
91
92
93 # All employees & clients
94 class GetClientsView(generics.ListAPIView):
95     queryset = Client.objects.all()
96     serializer_class = ClientSerializer
97
98
99 class GetEmployeesView(generics.ListAPIView):
100     queryset = Employee.objects.all()
101     serializer_class = EmployeeSerializer
102

```



---

```

104 # Check if employee is head
105 class GetHeaderView(generics.ListAPIView):
106     serializer_class = EmployeeSerializer
107     def get_queryset(self):
108         params = self.request.query_params
109         user = params.get('user', None)
110         queryset = Employee.objects.all()
111         if user:
112             queryset = queryset.filter(Q(user__username=user) & Q(position='h'))
113         return queryset
114
115
116 # Payments
117 class CreatePaymentView(generics.CreateAPIView):
118     queryset = Payment.objects.all()
119     serializer_class = PaymentSerializer
120
121
122 class GetPaymentView(generics.ListAPIView):
123     queryset = Payment.objects.all()
124     serializer_class = PaymentSerializer
125
126
127 class GetPaymentsView(generics.RetrieveUpdateDestroyAPIView):
128     queryset = Payment.objects.all()
129     serializer_class = PaymentSerializer
130
131
132
133 # Services
134 class GetServiceView(generics.RetrieveAPIView):
135     queryset = Service.objects.all()
136     serializer_class = ServiceSerializer
137
138
139 class GetServicesView(generics.ListAPIView):
140     queryset = Service.objects.all()
141     serializer_class = ServiceSerializer

```

---

## urls.py

---

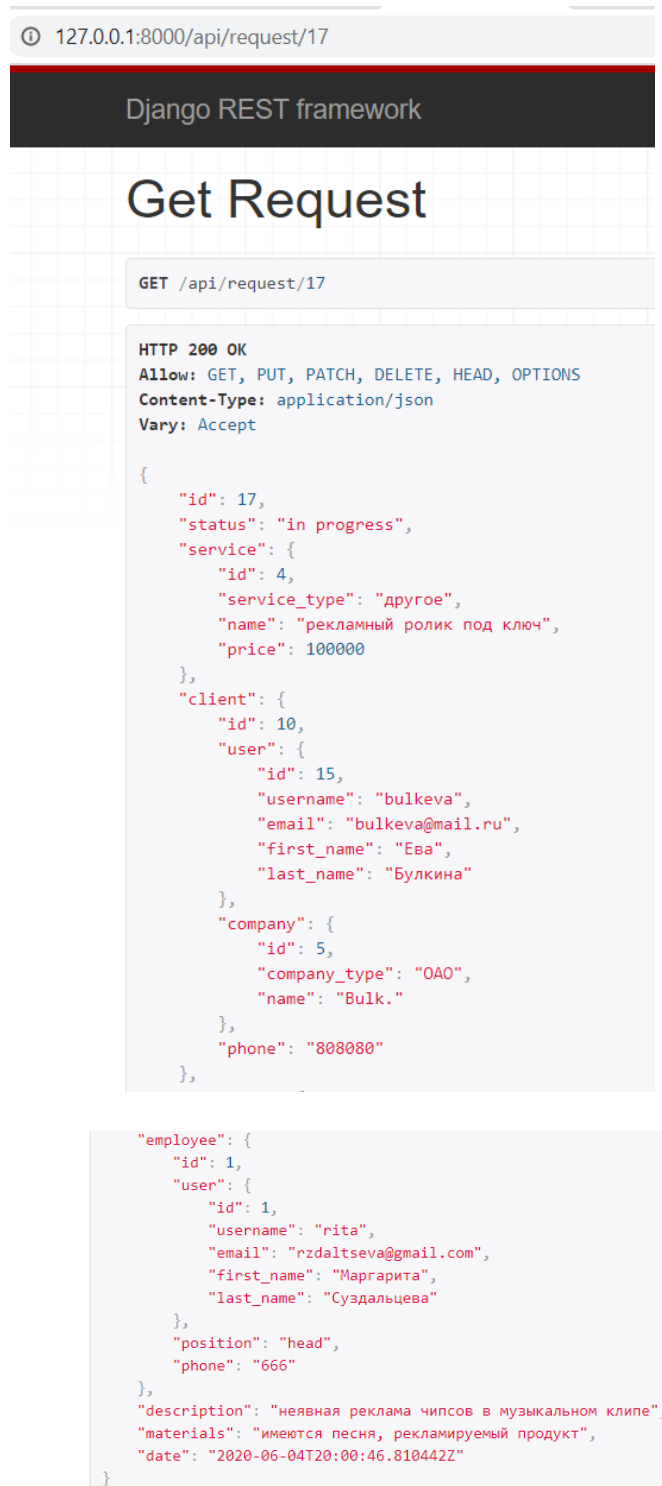
```

1 from django.urls import path, include
2 from .views import *
3 from rest_framework.authtoken.views import obtain_auth_token
4
5 app_name="luchapp"
6
7 urlpatterns = [
8     path('auth/', include('djoser.urls')),
9     path('auth/token', obtain_auth_token, name='token'),
10    path('request/new/', CreateRequestView.as_view()),
11    path('request/<int:pk>', GetRequestView.as_view()),
12    path('request/all', GetRequestsView.as_view()),
13    path('client/new', CreateClientView.as_view()),
14    path('client/all', GetClientsView.as_view()),
15    path('company/new', CreateCompanyView.as_view()),
16    path('payment/new', CreatePaymentView.as_view()),
17    path('payment/<int:pk>', GetPaymentView.as_view()),
18    path('payment/all', GetPaymentsView.as_view()),
19    path('service/<int:pk>', GetServiceView.as_view()),
20    path('service/all', GetServicesView.as_view()),
21    path('gethead/', GetHeaderView.as_view()),
22    path('getclient/', GetClientView.as_view()),
23    path('request/adminfilter', AdminFilterView.as_view()),
24    path('employee/all', GetEmployeesView.as_view()),
25 ]

```

---

Пример: как отображается заявка с id=17:



```
127.0.0.1:8000/api/request/17

Django REST framework

Get Request

GET /api/request/17

HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "id": 17,
  "status": "in progress",
  "service": {
    "id": 4,
    "service_type": "другое",
    "name": "рекламный ролик под ключ",
    "price": 100000
  },
  "client": {
    "id": 10,
    "user": {
      "id": 15,
      "username": "bulkeva",
      "email": "bulkeva@mail.ru",
      "first_name": "Ева",
      "last_name": "Булкина"
    },
    "company": {
      "id": 5,
      "company_type": "ООО",
      "name": "Bulk."
    },
    "phone": "808080"
  },
  "employee": {
    "id": 1,
    "user": {
      "id": 1,
      "username": "rita",
      "email": "rzdaltseva@gmail.com",
      "first_name": "Маргарита",
      "last_name": "Суздальцева"
    },
    "position": "head",
    "phone": "666"
  },
  "description": "неявная реклама чипсов в музыкальном клипе",
  "materials": "имеются песня, рекламируемый продукт",
  "date": "2020-06-04T20:00:46.810442Z"
}
```

## Часть 2. Frontend: Vue.js + Vuetify

Подключенные модули в main.js:

```

1 // The Vue build version to load with the `import` command
2 // (runtime-only or standalone) has been set in webpack.base.conf with an alias.
3 import Vue from 'vue'
4 import App from './App'
5 import router from './router'
6 import vuetify from '@/plugins/vuetify'
7 import axios from 'axios'
8 import VueAxios from 'vue-axios'
9
10 Vue.use(VueAxios, axios)
11 Vue.config.productionTip = false
12
13 /* eslint-disable no-new */
14 new Vue({
15   el: '#app',
16   router,
17   components: { App },
18   template: '<App/>',
19   vuetify,
20   axios
21 })

```

## App.vue:

```

1 <template>
2   <v-app id="app">
3     <v-navigation-drawer v-model="drawer" class="pink darken-4" dark app>
4       <v-list>
5         <v-list-item v-for="item in items" :key="item.title" link :to="item.link">
6           <v-list-item-icon>
7             <v-icon>{{ item.icon }}</v-icon>
8           </v-list-item-icon>
9           <v-list-item-content>
10            <v-list-item-title>{{ item.title }}</v-list-item-title>
11            </v-list-item-content>
12          </v-list-item>
13        </v-list>
14      </v-navigation-drawer>
15      <v-app-bar color="pink darken-4" dense dark app>
16        <v-app-bar-nav-icon @click="drawer = !drawer"></v-app-bar-nav-icon>
17        <v-toolbar-title class="headline">
18          <v-icon>mdi-palette-advanced</v-icon> LUCH Marketing
19        </v-toolbar-title>
20      </v-app-bar>
21      <v-content>
22        <v-container fluid>
23          <router-view></router-view>
24        </v-container>
25      </v-content>
26    </v-app>
27  </template>
28
29 <script>
30 export default {
31   name: 'App',
32   data () {
33     return {
34       items: [
35         { title: 'Главная', icon: 'mdi-palette-swatch', link: '/' },
36         { title: 'Личный кабинет', icon: 'mdi-emoticon', link: '/Cabinet' },
37         { title: 'Для админа', icon: 'mdi-emoticon-cool', link: '/adminCab' }
38       ],
39       drawer: false
40     }
41   }
42 </script>
43 <style scoped>
44 </style>

```

## index.js в router для адресации:

```
1  import Vue from 'vue'
2  import Router from 'vue-router'
3  import Index from '@/components/Index'
4  import Cabinet from '@/components/Cabinet'
5  import AdminCab from '@/components/AdminCab'
6  import Details from '@/components/Details'
7  import ReqForm from '@/components/ReqForm'
8  import RegClient from '@/components/RegClient'
9  import Auth from '@/components/Auth'
10
11  Vue.use(Router)
12
13  export default new Router({
14    routes: [
15      {
16        path: '/',
17        name: 'Index',
18        component: Index
19      },
20      {
21        path: '/Cabinet',
22        name: 'Cabinet',
23        component: Cabinet
24      },
25      {
26        path: '/auth',
27        name: 'Auth',
28        component: Auth
29      },
30      {
31        path: '/adminCab',
32        name: 'AdminCab',
33        component: AdminCab
34      },
35      {
36        path: '/details/:id',
37        name: 'Details',
38        component: Details,
39        props: true
40      },
41      {
42        path: '/newReq',
43        name: 'ReqForm',
44        component: ReqForm
45      },
46      {
47        path: '/newClient',
48        name: 'RegClient',
49        component: RegClient,
50        props: true
51      }
52    ]
53  })
```

## Компонента AdminCab.vue:

(остальные – аналогично, см. github: rzdaltseva)

```

1 <template>
2 <main>
3 <v-col cols="8" class="my-1 mx-auto">
4 <h2>Страница руководителя</h2>
5 <p>Все заявки</p>
6 </v-col>
7 <v-col cols="8" class="my-1 mx-auto">
8 <v-form>
9 <v-container>
10 <v-row>
11 <v-select v-model="req_status" :items="statuses" label="Статус"></v-select>
12 </v-row>
13 <v-row>
14 <v-select v-model="req_service" :items="services" label="Услуга"></v-select>
15 </v-row>
16 <v-row>
17 <v-col cols="6" md="4">
18 <v-menu ref="menu" v-model="menu" :close-on-content-click="false" :return-value.sync="date" transition="scale-transition" off
19 <template v-slot:activator="{ on }">
20 <v-text-field v-model="req_after" label="C" readonly v-on="on"></v-text-field>
21 </template>
22 <v-date-picker v-model="req_after" class="mt-4" @input="menu = false"></v-date-picker>
23 </v-menu>
24 </v-col>
25 <v-col cols="6" md="4">
26 <v-menu ref="menu" v-model="menu" :close-on-content-click="false" :return-value.sync="date" transition="scale-transition" off
27 <template v-slot:activator="{ on }">
28 <v-text-field v-model="req_before" label="По" readonly v-on="on"></v-text-field>
29 </template>
30 <v-date-picker v-model="req_before" class="mt-4" @input="menu = false"></v-date-picker>
31 </v-menu>
32 </v-col>
33 </v-row>
34 <v-row>
35 <v-select v-model="req_employee" :items="employees" label="Сотрудник"></v-select>
36 </v-row>
37 <v-row>
38 <v-select v-model="req_client" :items="clients" label="Клиент"></v-select>
39 </v-row>
40 <v-row>
41 <v-btn dark class="mr-4 pink accent-3" @click="filterReq">
42 <v-icon>mdi-filter</v-icon>Отфильтровать
43 </v-btn>
44 <v-btn dark class="mr-4 pink accent-4" @click="clearFilter">
45 <v-icon>mdi-autorenew</v-icon>Очистить
46 </v-btn>
47 </v-row>
48 </v-container>
49 </v-form>
50 </v-col>
51 <section>
52 <v-col cols="8" class="mx-auto">
53 <h3>Найдено заявок {{ requestcards.length }}</h3>
54 <v-card class="my-2" color="pink darken-3" dark v-for="card in requestcards" :key="card.id">
55 <v-card-title class="headline">{{ card.title }} -- {{ card.status }}</v-card-title>
56 <v-card-subtitle>{{ card.service.name }}</v-card-subtitle>
57 <v-card-text>
58 <p>Заказчик {{ card.client.user.first_name }} {{ card.client.user.last_name }}</p>
59 <p>Исполнитель {{ card.employee.user.first_name }} {{ card.employee.user.last_name }}</p>
60 </v-card-text>
61 <v-card-actions>
62 <v-btn text :to="'/details/' + card.id">Подробнее</v-btn>
63 </v-card-actions>
64 </v-card>
65 </v-col>
66 </section>
67 </main>
68 </template>

```

```

69 <script>
70 export default {
71   name: 'AdminCab',
72   created () {
73     if (sessionStorage.getItem('token') === null || sessionStorage.getItem('user') === null) {
74       this.$router.push('/auth')
75     }
76   },
77   mounted () {
78     this.axios
79       .get(`http://${window.location.hostname}:8000/api/gethead?user=${sessionStorage.getItem('user')}`)
80       .then(response => { this.getRequests(response.data) })
81       .catch(err => { console.error(err) })
82   },
83   data () {
84     return {
85       requestcards: [],
86       statuses: [
87         {
88           text: 'in progress',
89           value: 0
90         },
91         {
92           text: 'completed',
93           value: 1
94         }
95       ],
96       services: [],
97       employees: [],
98       clients: [],
99       req_status: '',
100       req_service: '',
101       req_after: '',
102       req_before: '',
103       req_employee: '',
104       req_client: ''
105     }
106   },
107   methods: {
108     getRequests (data) {
109       if (data.length === 0) {
110         this.$router.push('/')
111       }
112       console.log(data)
113       this.axios
114         .get(`http://${window.location.hostname}:8000/api/request/all`)
115         .then(response => { this.getCards(response.data) })
116         .catch(err => { console.error(err) })
117
118       this.axios
119         .get(`http://${window.location.hostname}:8000/api/service/all`)
120         .then(response => { this.getServices(response.data) })
121         .catch(err => { console.error(err) })
122
123       this.axios
124         .get(`http://${window.location.hostname}:8000/api/employee/all`)
125         .then(response => { this.getEmployees(response.data) })
126         .catch(err => { console.error(err) })
127
128       this.axios
129         .get(`http://${window.location.hostname}:8000/api/client/all`)
130         .then(response => { this.getClients(response.data) })
131         .catch(err => { console.error(err) })
132     },
133     getCards (data) {
134       this.requestcards = []
135       for (let i = 0; i < data.length; i++) {
136         let requestcard = {}
137         requestcard.title = `Заявка № ${data[i].id}`
138         requestcard.id = data[i].id
139         requestcard.service = data[i].service
140         requestcard.client = data[i].client
141         requestcard.employee = data[i].employee
142         requestcard.description = data[i].description
143         requestcard.materials = data[i].materials

```

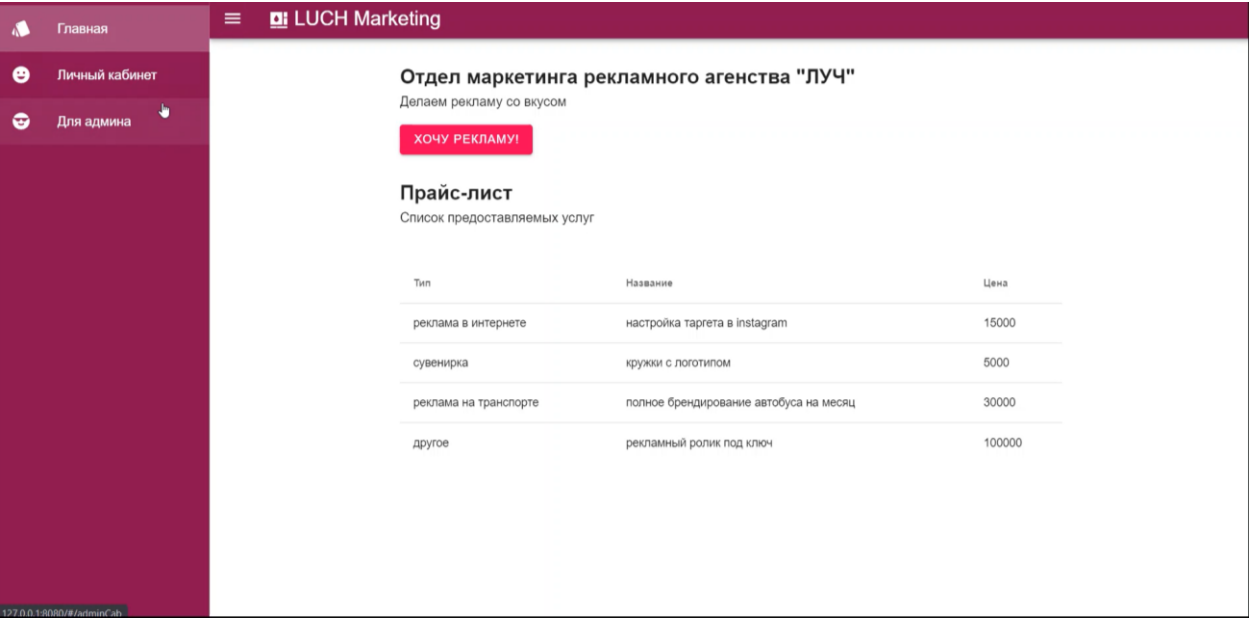
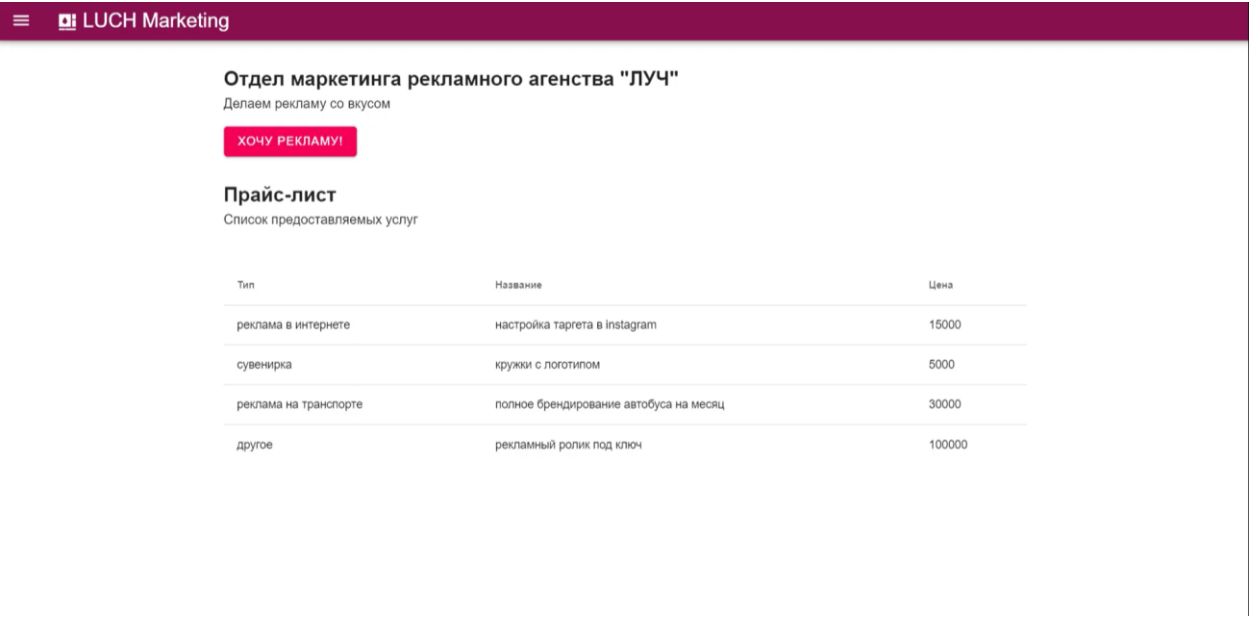
```

144     requestcard.date = data[i].date
145     requestcard.status = data[i].status
146
147     this.requestcards.push(requestcard)
148   }
149 },
150 filterReq () {
151   this.axios
152     .get(`http://${window.location.hostname}:8000/api/request/adminfilter?status=${this.req_status}&service=${this.req_service}&after=${
153       this.req_after
154     }`)
155     .then(response => { this.getCards(response.data) })
156     .catch(err => { console.error(err) })
157 },
158 clearFilter () {
159   this.req_status = ''
160   this.req_service = ''
161   this.req_after = ''
162   this.req_before = ''
163   this.req_employee = ''
164   this.req_client = ''
165
166   this.filterReq()
167 },
168 getServices (data) {
169   for (let i = 0; i < data.length; i++) {
170     let service = {}
171     service.value = data[i].id
172     service.text = data[i].name
173
174     this.services.push(service)
175   }
176 },
177 getEmployees (data) {
178   for (let i = 0; i < data.length; i++) {
179     let employee = {}
180     employee.value = data[i].id
181     employee.text = `${data[i].user.first_name} ${data[i].user.last_name} (${data[i].user.username})`
182
183     this.employees.push(employee)
184   }
185 },
186 getClients (data) {
187   for (let i = 0; i < data.length; i++) {
188     let client = {}
189     client.value = data[i].id
190     client.text = `${data[i].user.first_name} ${data[i].user.last_name} (${data[i].user.username})`
191
192     this.clients.push(client)
193   }
194 }
195 }
196 </script>
197 <style scoped>
198 </style>

```

Часть 3. Готовое приложение

Главная страница:





## Авторизация:

Главная

Личный кабинет

Для админа

ЛУСН Marketing

Логин  
rita

Пароль  
\*\*\*\*\*

ВХОД

ЗАРЕГИСТРИРОВАТЬСЯ

## Регистрация:

ЛУСН Marketing

Регистрация клиента

Логин

Почта

Имя

Фамилия

Название компании

Тип организации

Пароль

Телефон

ЗАРЕГИСТРИРОВАТЬСЯ!

## Форма заявки на рекламу:

Главная

Личный кабинет

Для админа

LUCH Marketing

Оставить заявку на рекламу

Услуга  
кружки с логотипом

Краткое описание  
красивые кружки

Необходимые материалы  
логотип?

ОТПРАВИТЬ

ОЧИСТИТЬ

## Личный кабинет сотрудника / клиента:

Главная

Личный кабинет

Для админа

LUCH Marketing

Личный кабинет пользователя

Все Ваши заявки

Найдено заявок 2

Заявка № 1 -- completed  
настройка таргета в Instagram  
Заказчик Март Горитовна  
Исполнитель Рита Су  
ПОДРОБНЕЕ

Заявка № 18 -- in progress  
кружки с логотипом  
Заказчик Март Горитовна  
Исполнитель Маргарита Суздальцева  
ПОДРОБНЕЕ

## Детали:

ЛУСН Marketing

Детали заявки

Подробная информация о выбранной заявке

Номер заявки	Услуга	Клиент	Сотрудник	Краткое описание	Материалы	Дата	Статус
1	реклама в интернете: настройка таргета в Instagram	Март Горитовна (martgorit) ИП "MG inc" Контакты: ritusyonok@gmail.com 111	Рита Су (marugarita) executor Контакты: margogirlange@mail.ru 555	продаю шопперы в инстаграм	макет? сумки	2020-05-24T19:15:06.394185Z	completed

## Страница руководителя:

Главная

Личный кабинет

Для админа

ЛУСН Marketing

Страница руководителя

Все заявки

Статус

Услуга

СПо

Сотрудник

Клиент

ОФИЛЬТРОВАТЬ

ОЧИСТИТЬ

Найдено заявок 5

Заявка № 15 -- in progress

кружки с логотипом

Главная

Личный кабинет

Для админа

ЛУCH Marketing

Статус

Услуга

С: 2020-06-01По: 2020-06-05

Сотрудник

Клиент

ОФИЛЬТРОВАТЬОЧИСТИТЬ

Найдено заявок 3

Заявка № 15 +- in progress

кружки с логотипом

Заказчик Мари Шан

Исполнитель Алиса Алиева

ПОДРОБНЕЕ

Главная

Личный кабинет

Для админа

ЛУCH Marketing

Страница руководителя

Все заявки

Статус in progress

Услуга

СПо

Сотрудник

Клиент

ОФИЛЬТРОВАТЬОЧИСТИТЬ

Найдено заявок 4

Заявка № 15 -- in progress

кружки с логотипом

Главная

Личный кабинет

Для админа

ЛУCH Marketing

Клиент

ОФИЛЬТРОВАТЬОЧИСТИТЬ

Найдено заявок 5

Заявка № 15 -- in progress

кружки с логотипом

Заказчик Мари Шан

Исполнитель Алиса Алиева

ПОДРОБНЕЕ

Заявка № 16 -- in progress

полное брендирование автобуса на месяц

Заказчик Иван Иванов

Исполнитель Саша Носков

ПОДРОБНЕЕ

Заявка № 17 -- in progress