Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

УНИВЕРСИТЕТ ИТМО

Факультет инфокоммуникационных технологий

Образовательная программа 09.03.03

Направление подготовки (специальность) Мобильные сетевые технологии

О Т Ч Е Т

по курсовой работе

Тема задания: Разработка одностраничного веб-приложения (SPA) с использованием фреймворка Vue.JS

Обучающийся: Шутов Даниил Эдуардович, К33402

Руководитель: Добряков Д. И., преподаватель

Оценка за курсовую работу ____

Подписи членов комиссии:

_____ (Говоров А. И.)
(подпись)

_____ (Добряков Д. И.)
(подпись)

Дата ____

Санкт-Петербург
2022

ВВЕДЕНИЕ

Актуальность

Электронные сервисы для оказания услуг показания погод весьма популярны. Погода весьма непредсказуема и приходится пользоваться услугами интернет-сервисов. Данное приложения помогает узнать погоду на ближайшие 5 дней. Сервис легко масштабируемый.

Цели и задачи

1. Определение средств разработки
2. Определение функциональных требований
3. Проектирование и реализация серверной части
4. Проектирование и реализация клиентской части

ГЛАВА 1. СРЕДСТВА РАЗРАБОТКИ И ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

1 Средства разработки

ITMO_ICT_WebDevelopment_2021-2022 курс, в рамках которого мы разрабатывали сервис для выбранного нами приложения, позволил мне реализовать серверную часть приложения. Клиентская часть написана на Vue, как и требовалось по заданию.

2 Функциональные требования

Функциональные требования по этому проекту заключались в наличие 9 компонентов в

1) Разработка одностраничного веб-приложения (SPA) с использованием фреймворка Vue.JS
2) Использование миксинов (необязательно, но желательно)
3) Использование Vuex (необязательно, но желательно)
4) В проекте должно быть, как минимум, 10 страниц (обязательно)

3 Проектирование и реализация серверной части

Серверная часть реализована с помощью Django rest framework для основных запросов, Djoser - для регистрации и авторизации посредством jwt токенов.

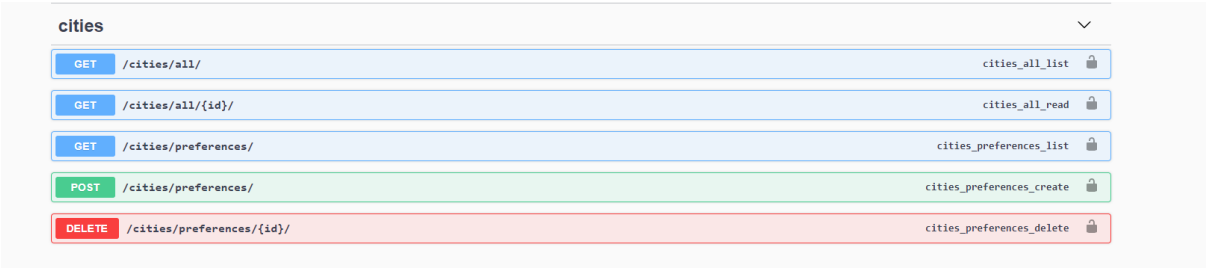
```
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'corsheaders',
    'rest_framework',
    'djoser',
    'rest_framework_simplejwt',
    'drf_yasg',
    'cities'
]
```

Реализованные модели:

```python
class City(models.Model):
    id = models.IntegerField("Unique id of city", primary_key=True)
    name = models.CharField("City name", max_length=100)
    country = models.CharField("Coutry code", max_length=100)
    country_verbose = models.CharField("Country name", max_length=100)
    longitude = models.FloatField("Longitude")
    latitude = models.FloatField("Latitude")

    def __str__(self):
        return str(self.name)


class CityPreference(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE, related_name="preferences")
    city = models.ForeignKey(City, on_delete=models.CASCADE, related_name="preferences")

    def __str__(self):
        return f"{self.user.username} - {self.city.name}"
```

Модель города, и модель города к юзеру, для реализации возможности привязывать город к пользователю.

Реализованные пути:



Сериалайзер и вьюшка для их обработки:

Views.py

```python
class CityViewSet(ReadOnlyModelViewSet):
    serializer_class = CitySeriazlier
    queryset = City.objects.all()


class CityPreferenceViewSet(ListModelMixin, CreateModelMixin, DestroyModelMixin, GenericViewSet):
    permission_classes = [IsAuthenticated]

    def get_serializer_class(self):
        if self.action == 'list':
            return CityPreferenceReadSeriazlier
        return CityPreferenceWriteSeriazlier

    def get_queryset(self):
        return CityPreference.objects.filter(user=self.request.user)
```

Serializers.py:

```python
from rest_framework import serializers
from cities.models import City, CityPreference


class CitySeriazlier(serializers.ModelSerializer):
    class Meta:
        model = City
        fields = '__all__'


class CityPreferenceReadSeriazlier(serializers.ModelSerializer):
    city = CitySeriazlier()

    class Meta:
        model = CityPreference
        fields = ['id', 'city']


class CityPreferenceWriteSeriazlier(serializers.ModelSerializer):
    user = serializers.HiddenField(default=serializers.CurrentUserDefault())

    class Meta:
        model = CityPreference
        fields = ['id', 'user', 'city']
```

А также реализован CORS метод для предоставления доступа ко всем страницам:

```python
CORS_ALLOW_ALL_ORIGINS = True
```

Итого, все пути:

```python
urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/auth/', include('djoser.urls')),
    path('api/auth/', include('djoser.urls.jwt')),
    path('api/cities/', include('cities.urls')),
    re_path(r'^swagger/$', schema_view.with_ui('swagger', cache_timeout=0), name='schema-swagger-ui'),
    re_path(r'^redoc/$', schema_view.with_ui('redoc', cache_timeout=0), name='schema-redoc')
]
```

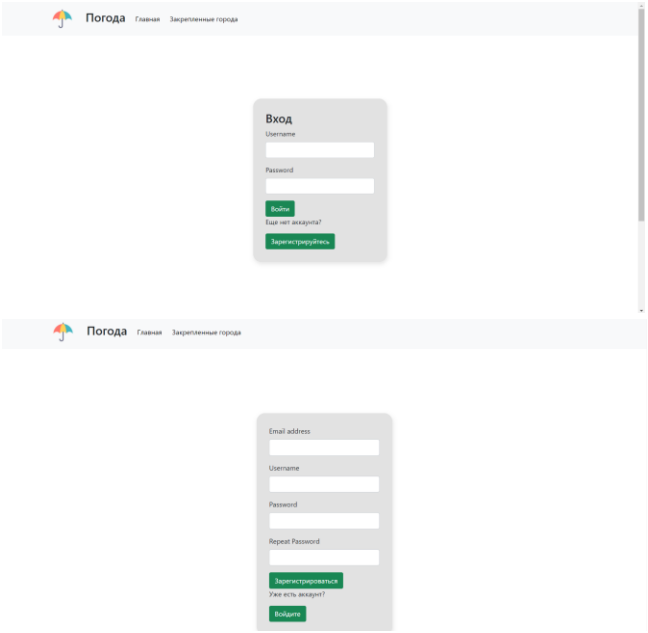
Реализован максимально простой сервер для клиентской части, так как тема курсовой работы фронтенд разработка.

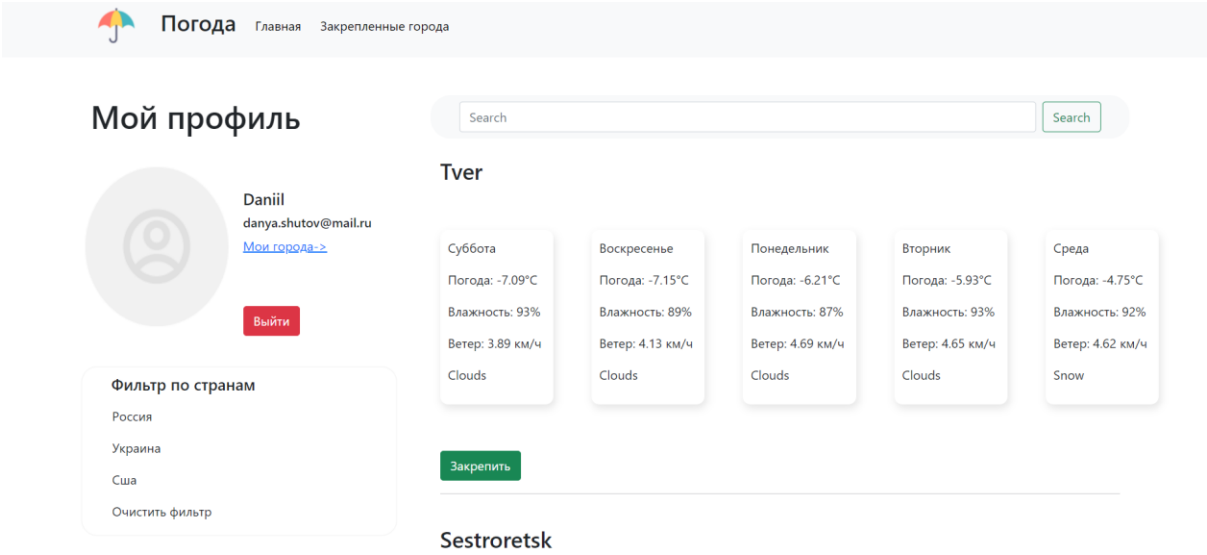
4. Проектирование и реализация клиентской части

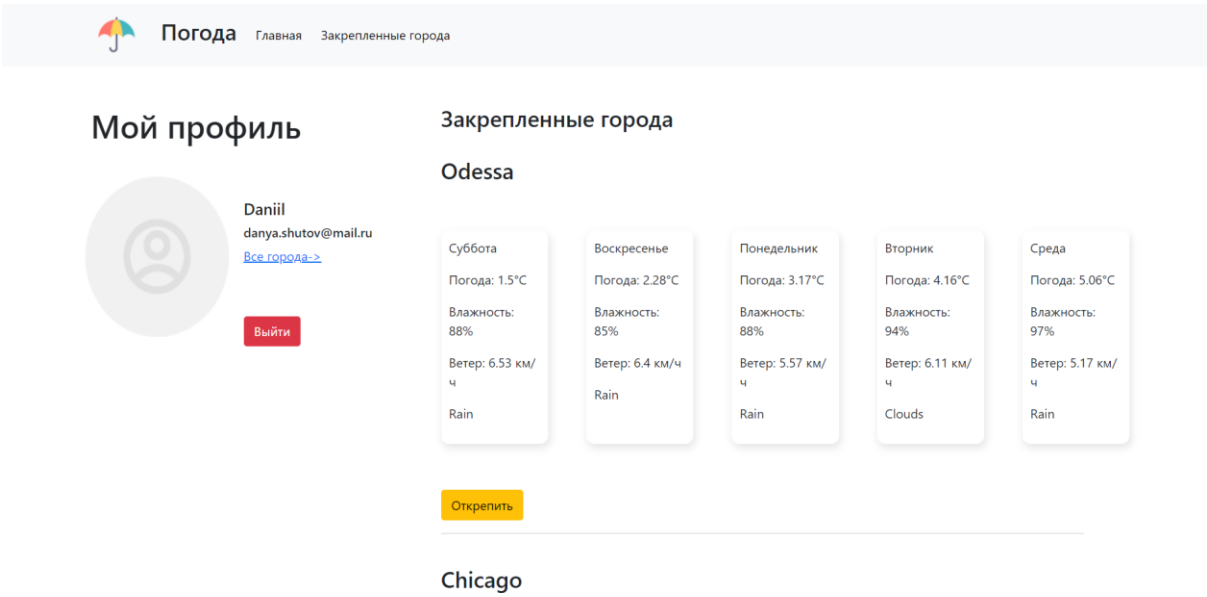
Используемый стек технологий: vuex, router-vue, axios, vue
Подключен стор и роутер, для хранения данных и роутинга по страницам.

```vue
<template>
  <div id="app">
    <Header/>
      <router-view/>
    <Footer/>
  </div>
</template>

<style>
  .link{
    width: 100%;
    height: 100%;
  }
</style>
<script>
  import Header from "@/components/Header";
  import Footer from "@/components/Footer";
  export default {
    components: {Footer, Header}
  }
</script>
```

В данном приложение 4 страницы, страница логина, регистрации, главная и города юзера.




Погода
Главная
Закрепленные города
Мой профиль
Search
Search
Daniil
danya.shutov@mail.ru
Мои города->
Выйти
Фильтр по странам
Россия
Украина
Сша
Очистить фильтр
Tver
Суббота
Погода: -7.09°C
Влажность: 93%
Ветер: 3.89 км/ч
Clouds
Воскресенье
Погода: -7.15°C
Влажность: 89%
Ветер: 4.13 км/ч
Clouds
Понедельник
Погода: -6.21°C
Влажность: 87%
Ветер: 4.69 км/ч
Clouds
Вторник
Погода: -5.93°C
Влажность: 93%
Ветер: 4.65 км/ч
Clouds
Среда
Погода: -4.75°C
Влажность: 92%
Ветер: 4.62 км/ч
Snow
Закрепить
Sestroretsk


Погода
Главная
Закрепленные города
Мой профиль
Daniil
danya.shutov@mail.ru
Все города->
Выйти
Закрепленные города
Odessa
Суббота
Погода: 1.5°C
Влажность: 88%
Ветер: 6.53 км/ч
Rain
Воскресенье
Погода: 2.28°C
Влажность: 85%
Ветер: 6.4 км/ч
Rain
Понедельник
Погода: 3.17°C
Влажность: 88%
Ветер: 5.57 км/ч
Rain
Вторник
Погода: 4.16°C
Влажность: 94%
Ветер: 6.11 км/ч
Clouds
Среда
Погода: 5.06°C
Влажность: 97%
Ветер: 5.17 км/ч
Rain
Открепить
Chicago

```
import Vue from 'vue'
import VueRouter from 'vue-router'
import Personal from '../views/Personal.vue'
import SignIn from '../views/SignIn.vue'
import SignUp from "../views/SignUp";
import Preferences from "@/views/Preferences";

Vue.use(VueRouter)

const routes = [
  {
    path: '/',
    name: 'Personal',
    component: Personal
  },
  {
    path: '/signin',
    name: 'SignIn',
    component: SignIn
  },
  {
    path: '/signup',
    name: 'SignUp',
    component: SignUp
  },
  {
    path: '/preferences',
    name: 'Preferences',
    component: Preferences
  },

]

const router = new VueRouter( options: {
  mode: 'history',
  base: process.env.BASE_URL,
  routes
})

export default router
```

Рассмотрим страницу логина:

```vue
<template>
    <div class="body">
        <form @submit.prevent="onSubmit(form)" class="container form">
            <h3>Вход</h3>
            <div class="mb-3">
                <label for="exampleInputUsername" class="form-label">Username</label>
                <input v-model="form.username" type="Text" class="form-control" id="exampleInputUsername">
            </div>
            <div class="mb-3">
                <label for="exampleInputPassword1" class="form-label">Password</label>
                <input v-model="form.password" type="password" class="form-control" id="exampleInputPassword1">
            </div>
            <button type="submit" class="btn btn-success">Войти</button>
            <p>Еще нет аккаунта?</p>
            <router-link to="/signup">
                <button type="submit" class="btn btn-success">Зарегестрируйтесь</button>
            </router-link>

        </form>
    </div>
</template>

<script>
    export default {
        name: 'Login',
        data() {
            return {
                form: {
                    username: "",
                    password: "",
                }
            };
        },
        methods: {
            onSubmit(form) {
                this.$store.dispatch( type: 'onSubmitSignIn', form)
            }
        },
    }
</script>
```

Используется v-model и метод отправки формы, которая вызывает вызов action vuex.

```vue
onSubmitSignIn(store, payload) {
    instance.post( url: `/auth/jwt/create/`, JSON.stringify(payload)) ...
        .then(function (data : AxiosResponse<any> ) {
            if (200 <= data.status < 300) {
                return data.data;
            }
            return Promise.reject(data.status);
        }) ...
        .then(token => {
            localStorage.setItem('token', JSON.stringify(token));
            router.push('/')
        })
},
refreshToken() {
```

Отправляется запрос на сервер по дефолтному пути предоставляемого djoser’ом. Создается токен, и возвращается пара access refresh, которая хранится в локал store.

При успешном ответе, также пользователя перебрасывает на главную страницу, где при монтировании компоненты отправляется запрос на данные юзера:

```vue
<template>
    <div class="windowMe row">
        <img class="col" alt="" src="../../public/Me.png">
        <div class="col pb-2">
            <h5 class="mt-5 city"> {{ username }}</h5>
            <h6 class="mt-2">{{ email }}</h6>
            <router-link to="/preferences" v-if="this.$route.path!=='/preferences'"><p>Мои города-></p></router-link>
            <router-link to="/" v-if="this.$route.path!=='/'"><p>Все города-></p></router-link>
            <button @click="logOut" type="submit" class="mt-5 pt- btn btn-danger">Выйти</button>
        </div>
    </div>
</template>

<script>
    import ...

    export default {
        name: "Profile",
        mounted() {
            this.$store.dispatch( type: 'mountedProfile')
        },
        computed: {
            ...mapGetters({
                username: 'username',
                email: 'email',
            })
        },
        methods: {
            logOut() {
                localStorage.removeItem( key: 'token')
                router.push('/signin')
            }
        }
    }
</script>
```

Рассмотрим axios запросы детальнее: есть два типа запросов – для авторизированных юзеров и нет.

Общая часть всех запросов: дефолтная часть:

```
Vue.use(Vuex)

const instance = axios.create({
    baseURL: 'http://127.0.0.1:8000/api',
    timeout: 1000,
    headers: {'Content-type': 'application/json; charset=UTF-8',}
});
```

Для авторизированных пользователей, добавляется токен

```
mountedProfile({commit}) {
    instance
        .get( url: `/auth/users/me/`, config: {
            headers: {
                Authorization: `Bearer ${JSON.parse(localStorage.getItem( key: 'token'))?.access}`
            }
        }) ...
        .then(function (data : AxiosResponse<any> ) {
            if (200 <= data.status < 300) {
                return data.data;
            }
            return Promise.reject(data.status);
        }) ...
        .then(user => {
            commit('setUser', {username: user.username, email: user.email})
        })
```

Который парсится с локал стора, по окончанию запроса вызывается коммит данных (мутации), где переписываются данные, сам стор выглядит так:

```
const store = new Vuex.Store( options: {
    state: {
        id: null,
        username: '',
        email: '',
        filterCountry: '',
        filterSearch: '',
        citypreference: [],
        citylist: []
    },
    getters: {
        filterCountry(state) {
            return state.filterCountry
        },
        filterSearch(state) {
            return state.filterSearch
        },
        citylist(state) {
            return state.citylist
        },
        citypreference(state) {
            return state.citypreference
        },
        email(state) {
            return state.email
        },
        username(state) {
            return state.username
        },
    },
    mutations: {
        setUser(state, payload) {
            state.id = payload.id
            state.username = payload.username
            state.email = payload.email
        },
        setCities(state, payload) {
            state.citylist = payload
        }
```

Реализована получение данных и перезапись данных стора.

Теперь рассмотрим главную страницу:

```
<main class="body">
  <div class="bg-color">
    <div class="container" id="body">
      <div class="row">
        <div class="col-lg">
          <h1 id="title">Мой профиль</h1>
          <Profile />
          <CountryFilter />
        </div>
        <CityList />
      </div>
    </div>
  </div>
</main>
```

Она состоит из 3 компонент, 1 – профиль, окно юзера, компонент настройки фильтров по странам, и сам список городов с поиском в нем.

Есть страница закрепленных городов, выглядит она аналогично:

```
<main class="body">
    <div class="container" >
        <div class="row">
            <div class="col-lg">
                <h1 id="title">Мой профиль</h1>
                <Profile />
            </div>
            <div class="col-lg-8 mt-5">
                <h3>Закрепленные города</h3>
                <div class="d-flex flex-column me-5" v-for="(city) in citypreference" :key="city.id">
                    <City :city="city.city" :apiKey="'6e9d6b613a503e6f7f3f03473587c4c9'" :id="city.id" :add="false"/>
                </div>
            </div>
        </div>
    </div>
</main>
template>
```

С одним исключением, больше нет промежуточной компоненты citylist, и сразу строится список городов исходя из данных полученных с сервера, также в City есть переменная add, которая показывает, какое действие нужно делать с городом на определенной странице, теперь рассмотрим сам город: циклом по всему списку городов с сервера, мы обрисовываем каждый.

Citylist:

```vue
<template>
    <div class="col-lg-8 mt-5">
        <div class="row">
            <nav class="navbar navbar-light bg-light">
                <div class="container-fluid">
                    <form @submit.prevent="setSearch(search)" class="search d-flex">
                        <input v-model="search" class="form-control me-2" type="search" placeholder="Search"
                               aria-label="Search">
                        <button class="btn btn-outline-success" type="submit">Search</button>
                    </form>
                </div>
            </nav>
            <div class="d-flex flex-column me-5" v-for="(city, index) in citylist.filter(el => {
                    let flag = el.country === filterCountry || filterCountry=== '';
                    if (filterSearch!=='') {
                        flag = flag && (el.name.toLowerCase().indexOf(filterSearch.toLowerCase()) > -1)
                    }
                    return flag})" :key="index">
                <City :city="city" :apiKey="'6e9d6b613a503e6f7f3f03473587c4c9'" :id="city.id" :add="true"/>
            </div>
        </div>
    </div>
</template>
```

Реализован фильтр на значение в поиске, и значению выбранной страны.
Данные погоды конкретного города приходят с открытого апи openweathermap.

```vue
<template>
    <div class="mt-4">
        <h3>{{ city.name }}</h3>
        <div class="d-flex flex-row">
            <div class="col-lg-2 me-5 cardWeather " v-for="(weatherDay, index) in week" :key="index">
                <div class="row justify-content-center">
                    <p><span class="date">{{date[index]}}</span></p>
                    <p>Погода: <span class="temp"></span>{{weatherDay.main.temp}}°C</p>
                    <p>Влажность: <span class="humidity"></span>{{weatherDay.main.humidity}}%</p>
                    <p>Ветер: <span class="wind"></span>{{ weatherDay.wind.speed }} км/ч</p>
                    <p><span class="wearther">{{ weatherDay.weather[0].main }}</span></p>
                </div>
            </div>
        </div>
        <button v-if="add===true" @click="addCity(id)" type="submit" class="mt-5 pt- btn btn-success">Закрепить</button>
        <button v-else @click="removeCity(id)" type="submit" class="mt-5 pt- btn btn-warning">Открепить</button>
        <hr/>
    </div>
</template>

<script>
    import axios from 'axios'
    const a = new Date()

    export default {
        name: "City",
        props: ['city', 'apiKey', 'id', 'add'],
        data: () => ({...}),
        methods: {
            addCity(id) {
                this.$store.dispatch( type: 'addCity', id)
            },
            removeCity(id) {
                this.$store.dispatch( type: 'removeCity', id)
            }
        },
        mounted () {
            axios
                .get( url: `http://api.openweathermap.org/data/2.5/forecast?id=${this.city.id}&appid=${this.apiKey}&units=metric`)
                .then(response => {
                        for (let i = 0; i < 5; i++) {
                            let weatherDay = (response.data.list[i])
```

В темплейтах можем увидеть использование v-if для выбора действия над городом.
При монтировке компоненты отправляется запрос к серверу прогноза погоды, которые

записываются в локальные переменные. (ограниченно только первыми 5 днями, для удобства отображения).

ЗАКЛЮЧЕНИЕ

Выводы по работе

Благодаря данной работе я научился интересным методам работы с клиентской частью приложений, изучил vue, vuex, vue-router, а также узнал много нового о axios. Также впервые пришлось самостоятельно реализовывать всю архитектуру приложения. С полным осознанием того, что нужно сделать для работоспособности.

СПИСОК ЛИТЕРАТУРЫ

1. Документация Django [Электронный ресурс] https://docs.djangoproject.com/en/3.0/.
2. Документация Django REST Framework [Электронный ресурс] https://www.django-rest-framework.org.
3. Документация Vuex [Электронный ресурс] https://vuex.vuejs.org/ru/
4. Документация Vue-router [Электронный ресурс] https://router.vuejs.org/ru/api/
5. Документация VueJs [Электронный ресурс] https://vuejs.org/