­МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РАДІОЕЛЕКТРОНІКИ

Звіт

з дисципліни «Аналіз і рефакторинг коду програмного забезпечення»

Виконав:

ст. гр. ПЗПІ-16-1

Черненко М.В.

Перевірила:

Лещинська І.О.

Харків 2018

**Мета роботи**: розробити фронтенд для курсового проекту веб-орієнтованої

системи для аналізу стану води.

Для розробки бекенду був використаний JS (Vue) з Vuex та Vue routers.

Були використовані бібліотеки: Vue maps, axios. Vue-translations.

Для кращого розуміння системи наведу кілька діаграм

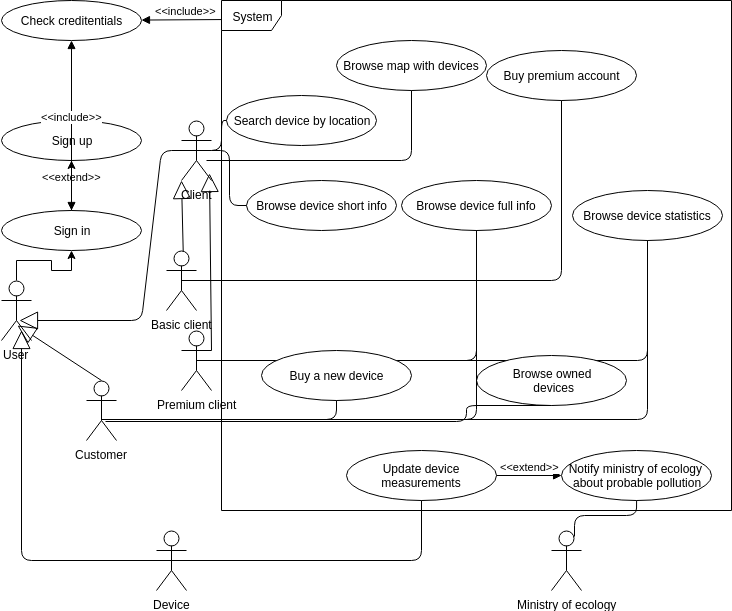


Рис 1.1 Use case діаграма

Наведемо декільна основних ділянок коду

import Vue from 'vue'

import Router from 'vue-router'

import store from './store'

import SignIn from './views/accounts/SignIn.vue'

import SignUp from './views/accounts/SignUp.vue'

import SubscribePremium from './views/clients/SubscribePremium.vue'

import ClientDeviceStatistics from './views/clients/DeviceStatistics'

import Client from './views/clients/Index'

import ClientMap from './views/clients/Map'

import Customer from './views/customer/Index'

import CustomerDashboard from './views/customer/Dashboard'

import CustomerDeviceStatistics from './views/customer/DeviceStatistics'

import CustomerBuyDevice from './views/customer/BuyDevice'

import InvalidUserError from './errors'

Vue.use(Router)

const routes = [

{

path: '/sign-in/',

name: 'sign-in',

component: SignIn

},

{

path: '/sign-up/',

name: 'sign-up',

component: SignUp

},

{

path: '/client',

name: 'client',

component: Client,

redirect: '/client/dashboard/',

meta: { 'requireClient': true },

children: [

{

path: 'dashboard/',

name: 'client-dashboard',

component: ClientMap,

meta: { 'menuItemName': 'dashboard' }

},

{

path: 'subscribe-premium/',

name: 'subscribe-premium',

component: SubscribePremium

},

{

path: '/device/:id/statistics/',

name: 'client-device-statistics',

component: ClientDeviceStatistics,

beforeEnter (to, from, next) {

if (!(isPremium())) {

next({ 'name': 'client' })

}

next()

},

props: true

}

]

},

{

path: '/customer',

name: 'customer',

component: Customer,

redirect: '/customer/dashboard/',

meta: { 'requireCustomer': true },

children: [

{

path: 'dashboard/',

name: 'customer-dashboard',

component: CustomerDashboard,

meta: { 'menuItemName': 'dashboard' }

},

{

path: '/device/:id/statistics/',

name: 'customer-device-statistics',

component: CustomerDeviceStatistics,

props: true

},

{

path: '/buy-device/',

name: 'customer-buy-device',

component: CustomerBuyDevice,

props: true

}

]

}

]

const router = new Router({

routes,

mode: 'history'

})

function isLoggedIn () {

return store.getters.isLoggedIn

}

function isClient () {

return store.getters.isClient

}

function isCustomer () {

return store.getters.isCustomer

}

function isPremium () {

return store.state.isPremium

}

router.beforeEach((to, from, next) => {

if (['sign-in', 'sign-up', 'reset-password', 'set-password'].indexOf(to.name) !== -1) {

next()

} else {

// check that user is logged and is client or customer

if (!isLoggedIn() || !(isClient() || isCustomer())) {

next({ name: 'sign-in' })

} else {

// every router that inherits from /customer/ has customer meta in root matched path

if (to.matched.some(record => record.meta.requireCustomer)) {

// continue if customer goes customer's route

if (isCustomer()) {

next()

} else if (isClient()) {

next({ name: 'client' })

}

} else if (to.matched.some(record => record.meta.requireClient)) {

// continue if client goes client's route

if (isClient()) {

if (to.name === 'subscribe-premium') {

if (!isPremium()) {

next()

} else {

next({ name: 'client' })

}

}

next()

} else if (isCustomer()) {

next({ name: 'customer' })

}

} else if (isCustomer()) {

next({ name: 'customer' })

} else if (isClient()) {

next({ name: 'client' })

} else {

throw new InvalidUserError()

}

}

}

})

export default router

**Код роутерінгу**

<template>

<v-layout fill-height>

<device-history :devices="devices" @deviceClicked="deviceClicked"></device-history>

<v-flex class="fill-height" :class="selectedDevice ? 'xs12 md7' : 'xs12 md10'">

<div class="group xs12">

<GmapAutocomplete @place\_changed="setPlace" placeholder="" required></GmapAutocomplete>

<span class="highlight"></span>

<span class="bar"></span>

<label>Enter location</label>

</div>

<GmapMap

:center="center"

:zoom="zoom"

map-type-id="terrain"

class="map"

@zoom\_changed="\_zoom => this.zoom = \_zoom"

>

<GmapMarker

:key="index"

v-for="(device, index) in devices"

:position="device.position"

:icon="icon"

:title="device.name"

:clickable="true"

@click="deviceClicked(device)"

/>

<GmapMarker v-if="marker" key="marker" :position="marker.position" :title="marker.title"></GmapMarker>

</GmapMap>

</v-flex>

<device-map-detail :device="selectedDevice"></device-map-detail>

</v-layout>

</template>

<script>

import BasicClientApi from './basicApi'

import PremiumClientApi from './premiumApi'

import DeviceMapDetail from '../../components/devices/MapDetail'

import { mapGetters } from 'vuex'

import DeviceHistory from '../../components/devices/DeviceHistory'

export default {

name: 'ClientMap',

components: { DeviceHistory, DeviceMapDetail },

data () {

return {

center: { lat: 10, lng: 10 },

icon: {},

zoom: 5,

devices: [],

selectedDevice: null,

marker: null

}

},

created () {

this.$gmapApiPromiseLazy().then(response => {

this.icon = {

'url': require('../../assets/device.png'),

'scaledSize': new response.maps.Size(30, 30)

}

})

navigator.geolocation.getCurrentPosition(location => {

this.center = { lat: location.coords.latitude, lng: location.coords.longitude }

})

let promise

if (this.$store.state.isPremium) {

promise = PremiumClientApi.getDeviceList({})

} else {

promise = BasicClientApi.getDeviceList({})

}

promise

.then(response => {

for (let device of response.data) {

if (device.measurements) {

const lat = Number(device.measurements.lat)

const lng = Number(device.measurements.lng)

this.devices.push({ ...device, position: { lat, lng } })

}

}

})

},

methods: {

...mapGetters(['isPremium']),

deviceClicked (device) {

this.center = device.position

this.zoom = 7

this.selectedDevice = device

},

setPlace (address) {

const loc = address.geometry.location

const position = { lat: loc.lat(), lng: loc.lng() }

this.marker = { 'title': address.name, position }

this.center = position

}

}

}

</script>

**Відображення карти з пристроями**

<template>

<div v-if="statistics">

<h5 class="text-xs-center mt-3">{{t('Gap')}}</h5>

<div>

<v-radio-group v-model="gap.value" row class="justify-center">

<v-radio v-for="option in gap.options" :key="option.value" :value="option"

:label="option.label"></v-radio>

</v-radio-group>

</div>

<h5 class="text-xs-center">{{t('Measurements')}}</h5>

<div class="d-flex row justify-end">

<v-switch v-for="option in measurement.options" :key="option.value" v-model="measurement.value"

:label="option.label" :value="option" class="pl-5"></v-switch>

</div>

<chart :chartData="prepareChartData" :options="{responsive: true, maintainAspectRatio: true}"

:height="150"></chart>

</div>

</template>

<script>

import Chart from './Chart'

const TEMPERATURE = { value: 'temperature', label: 'Temperature', color: 'rgba(153, 102, 255, 0.2)' }

const LAST\_HOUR = { value: 'last\_hour', label: 'Last hour', timeFormat: 'LTS' }

export default {

name: 'DeviceStatistics',

components: { Chart },

props: ['id', 'statistics'],

data () {

return {

gap: {

value: LAST\_HOUR,

options: [

LAST\_HOUR,

{ value: 'last\_day', label: 'Last day', timeFormat: 'LT' },

{ value: 'last\_month', label: 'Last month', timeFormat: 'L' }

]

},

measurement: {

value: [TEMPERATURE],

options: [

TEMPERATURE,

{ value: 'turbidity', 'label': 'Turbidity', color: 'rgba(255, 99, 132, 0.2)' },

{ value: 'ph\_level', 'label': 'PH level', color: 'rgba(54, 162, 235, 0.2)' },

{ value: 'gyroscope', 'label': 'Gyroscope', color: 'rgba(255, 206, 86, 0.2)' },

{ value: 'accelerator', 'label': 'Accelerator', color: 'rgba(75, 192, 192, 0.2)' },

{ value: 'magnetometer', 'label': 'Magnometer', color: 'rgba(255, 159, 64, 0.2)' }

]

}

}

},

created () {

this.$nextTick(() => {

for (let item of this.gap.options) {

item.label = this.$translate.text(item.label)

}

for (let item of this.measurement.options) {

item.label = this.$translate.text(item.label)

}

this.title = this.$translate.text(this.title)

console.log(this.$translate.locale)

})

},

computed: {

prepareChartData () {

const indicators = this.statistics[this.gap.value.value]

const labels = indicators.map(i => this.$moment(i.timestamp).format(this.gap.value.timeFormat))

let datasets = []

for (let option of this.measurement.value) {

datasets.push({

label: option.label,

backgroundColor: option.color,

data: indicators.map(i => i[option.value])

})

}

return {

labels,

datasets

}

}

}

}

</script>

**Відображення статистики показників для пристроїв**

**Висновок**: у ходi лабораторної роботи було зроблено фронтенд для курсового

проекту веб-орієнтованої системи для отримання та анілізу стану води, наведено кілька діаграм та характерних ділянок коду для кращого розуміння системи.