# Класс Messenger

using System;

using System.Collections.Generic;

using System.Text;

namespace ConsoleMessenger

{

[Serializable]

public class Message

{

public string UserName { get; set; }

public string MessageText { get; set; }

public DateTime TimeStamp { get; set; }

public Message()

{

UserName = "System";

MessageText = "Server is running";

TimeStamp = DateTime.UtcNow;

}

public Message(string userName, string messageText, DateTime timeStamp)

{

UserName = userName;

MessageText = messageText;

TimeStamp = timeStamp;

}

public override string ToString()

{

string output = String.Format("{0} {1} {2}", UserName, MessageText, TimeStamp);

return output;

}

}

}

# Контроллер Messanger.cs

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Collections;

using Newtonsoft.Json;

using ConsoleMessenger;

namespace Server.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class Messanger : ControllerBase

{

static List<Message> ListOfMessages = new List<Message>();

// GET api/<Messanger>/5

[HttpGet("{id}")]

public string Get(int id)

{

string OutputString = "Not found";

if ((id < ListOfMessages.Count) && (id >= 0))

{

OutputString = JsonConvert.SerializeObject(ListOfMessages[id]);

}

Console.WriteLine(String.Format("Запрошено сообщение № {0} : {1}", id, OutputString));

return OutputString;

}

// POST api/<Messanger>

[HttpPost]

public IActionResult SendMessage([FromBody] Message msg)

{

if (msg == null)

{

return BadRequest();

}

ListOfMessages.Add(msg);

Console.WriteLine(String.Format("Всего сообщений: {0} Посланное сообщение: {1}", ListOfMessages.Count, msg));

//return new NoContentResult();

return new OkResult();

}

}

}

# Add CORS startup.cs

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddControllers();

services.AddCors(o => o.AddPolicy("MyPolicy", builder =>

{

builder.AllowAnyOrigin()

.AllowAnyMethod()

.AllowAnyHeader();

}));

}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseRouting();

app.UseAuthorization();

app.UseCors("MyPolicy");

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

}

# Go Сервевер

package main  
  
import (  
 "encoding/json"  
 "fmt"  
 "github.com/go-chi/chi"  
 "io/ioutil"  
 "net/http"  
 "strconv"  
)  
  
type Message struct {  
 UserName string  
 MessageText string  
 TimeStamp string  
}  
  
func main() {  
 r := chi.NewRouter()  
 fmt.Print("Service is running")  
 messages := make([]Message, 0)  
  
 r.Post("/api/Messanger", func(w http.ResponseWriter, r \*http.Request) {  
 mes := Message{}  
 raw, \_ := ioutil.ReadAll(r.Body)  
 json.Unmarshal(raw, &mes)  
 fmt.Println(mes)  
 messages = append(messages, mes)  
 })  
  
 r.Get("/api/Messanger/{id}", func(w http.ResponseWriter, r \*http.Request) {  
 id := chi.URLParam(r, "id")  
 id\_num, \_ := strconv.Atoi(id)  
 if id\_num >= 0 && id\_num < len(messages) {  
 msg := messages[id\_num]  
 data, \_ := json.Marshal(msg)  
 w.Write(data)  
 }  
 })  
  
 http.ListenAndServe(":5000", r)

# App.vue

<template>

<v-app>

<v-main>

<ChatPage/>

</v-main>

</v-app>

</template>

<script>

import ChatPage from './components/ChatPage';

export default {

name: 'App',

components: {

ChatPage,

},

data: () => ({}),

}

</script>

# ChatPage1

<template>  
<v-container>  
 <div id="example-1">  
 <button v-on:click="counter += 1">+1</button>  
 <p>Кнопка выше была нажата {{ counter }} раз</p>  
 </div>  
</v-container>  
</template>  
  
<script>  
export default {  
 // Модель данных страницы  
 data: () => {  
 return {  
 counter: 0,  
 username: '',  
 message: '',  
 // messages: [  
 // {"username": "fddffdfdfd", "text": "texttexttexttexttexttext1", "timestamp": "2020-10-15T17:05:12.417897Z"},  
 // { "UserName":"RusAl","MessageText":"Privet na sto let!!!","TimeStamp":"2021-03-05T18:23:10.932973Z"},  
 messages: [],  
 intervalCtx: null,  
 lastMsgID: 0,  
 }  
  
 },  
}  
</script>  
  
<style scoped>  
  
</style>

# ChatPage2

<template>  
 <v-container>  
 <!-- <div id="example-1">-->  
 <!-- <button v-on:click="counter += 1">+1</button>-->  
 <!-- <p>Кнопка выше была нажата {{ counter }} раз</p>-->  
 <!-- </div>-->

<v-card>  
 <v-card-title>MyMessenger</v-card-title>

<!-- Панель сообщений -->  
 <v-card-text>  
 <div class="messages-view">  
 <div class="message" v-for="(message, i) in messages" :key="i">  
 <span class="message-date mr-3">{{ new ***Date***(message.TimeStamp).toDateString() }}</span>  
 <span class="message-username mr-5">{{ message.UserName }}</span>  
 <span class="message-text">{{ message.MessageText }}</span>  
 </div>  
 </div>  
 </v-card-text>

<!-- Панель действий -->  
 <v-card-text>  
 <div>  
 <v-text-field style="width: 40%; min-width: 200px" outlined dense v-model="UserName" label="UserName"/>  
 </div>  
 <div class="d-flex">  
 <v-text-field class="mr-10" outlined dense v-model="MessageText" label="MessageText"/>  
 <v-btn  
 depressed  
 color="primary"  
 @click="onSendClick"  
 :disabled="!MessageText.length"  
 >  
 <v-icon>mdi-send</v-icon>  
 </v-btn>  
 </div>  
 </v-card-text>  
 </v-card>  
 </v-container>  
</template>  
  
<script>  
import API from '../api/api.js';  
  
export default {  
 // Модель данных страницы  
 data: () => {  
 return {  
 // counter: 0,  
 UserName: '',  
 MessageText: '',  
 // messages: [  
 // { "UserName":"RusAl","MessageText":"Privet na sto let!!!","TimeStamp":"2021-03-05T18:23:10.932973Z"},  
 //]  
 messages: [],  
 intervalCtx: null,  
 lastMsgID: 0,  
 }  
  
 },  
  
 // Хук который сработает когда страница создасться  
 mounted() {  
 this.UserName = "Login"  
 this.intervalCtx = setInterval(async () => {  
 try {  
 const msg = await API.*getMessage*(this.lastMsgID)  
 if (typeof msg === typeof {}) {  
 this.messages.push(msg)  
 this.lastMsgID++  
 }  
 } catch (e) {  
 ***console***.error(e)  
 }  
 }, 1000)  
 },  
  
 methods: {  
 // Реакция на кнопку отправки  
 async onSendClick() {  
 try {  
 await API.*sendMessage*(this.UserName, this.MessageText)  
 ***console***.log('cleared')  
 this.message = ''  
 } catch (e) {  
 ***console***.error(e)  
 }  
 },  
 },  
 // После уничтожения компонента  
 destroyed() {  
 clearInterval(this.intervalCtx)  
 }  
}  
</script>  
  
<style lang="sass" scoped>  
.messages-view  
 *//border: 1px solid black* border-radius: 3px  
 overflow-x: hidden  
 overflow-y: scroll  
 height: 40vh  
  
 .message  
 background: #9fc8ea  
 border-radius: 3px  
 padding: 3px  
 margin: 5px  
  
 .message-date  
 color: rgba(255, 255, 255, 0.9)  
  
 .message-username  
 color: rgba(255, 255, 0, 0.8)  
  
 .message-text  
 color: rgba(0, 0, 0, 0.9)  
</style>

# api.js

import ***Axios*** from 'axios'  
  
const axios = ***Axios***.create({  
 baseURL: 'http://localhost:5000',  
})  
  
export default class API {  
 // Получает сообщение по id  
 static async *getMessage*(id) {  
 const resp = await axios.get(`/api/Messanger/${id}`)  
 return resp.data  
 }  
  
 // Отправка сообщения  
 static async *sendMessage*(UserName, MessageText) {  
 await axios.post(`/api/Messanger`, {  
 MessageText, // text: text  
 UserName,  
 })  
 }  
}