ДОДАТОК А
КОД ПРОГРАМИ

```
const fastify = require ( 'fastify') ();
const puppeteer = require ( 'puppeteer');
const axios = require ( 'axios'). default;
const pages = require ( './ db.json');
let browser
puppeteer.launch (). then (result => browser = result)

const getGismeteo = async function (city, date) {

   const gismeteo = await browser.newPage ()

   const forecast = {
      name: 'gismeteo.ua'
   },

   userDate = {
      year: date.slice (0, 4),
      month: + (date.slice (5, 7)) - 1,
      day: date.slice (8, 10)
   },

   now = new Date (),

   currentDate = new Date (userDate.year, userDate.month, userDate.day, now.getHours (), now.getMinutes (), now.getSeconds (), now.getMilliseconds ()),

   daysAmount = (currentDate - now) / 86400000

   const cityObject = pages.urls.find (item => item.name === city),
   url = cityObject.GismeteoURL

   await gismeteo.goto ( `https://www.gismeteo.ua/$ {url}`, {timeout: 0})

   const temperatures = (await gismeteo. $$ ( '. chart__temperature> .values> .value')) [daysAmount]
   const maxTempNode = await temperatures. $ ( '. maxt> .unit_temperature_c')

   const maxTemp = await maxTempNode.evaluate (node => node.innerText)
   const minTempNode = await temperatures. $ ( '. mint> .unit_temperature_c')
   const minTemp = await minTempNode.evaluate (node => node.innerText)

   forecast.maxTemperature = maxTemp + '°'
   forecast.minTemperature = minTemp + '°'


   const windNode = (await gismeteo. $$ ( '. w_wind .unit_wind_m_s')) [daysAmount]
   const maxWindSpeed = await windNode.evaluate (node => node.innerText)


   forecast.wind = maxWindSpeed + 'm / s (max)'

   const precsNode = (await gismeteo. $$ ( '. widget__row_precipitation .w_prec__value'))
```

```
[daysAmount]
   const precipitations = await precsNode.evaluate (node => node.innerText)

   forecast.precips = precipitations + 'mm'

   const overcastElems = (await gismeteo. $$ ( '. widget__row_icon .widget__item')) [daysAmount]
   const overcastNode = await overcastElems. $ ( '. tooltip')
   const overcast = await overcastNode.evaluate (node => node.getAttribute ( 'data-text'))

   forecast.overcast = overcast

   gismeteo.close ()

   return forecast

}



const getWeather = async function (city, date) {

   const weather = await browser.newPage ()

   const forecast = {
      name: 'weather.com'
   },

   userDate = {
      year: date.slice (0, 4),
      month: + (date.slice (5, 7)) - 1,
      day: date.slice (8, 10)
   },

   now = new Date (),

   currentDate = new Date (userDate.year, userDate.month, userDate.day, now.getHours (), now.getMinutes (), now.getSeconds (), now.getMilliseconds ()),

   daysAmount = (currentDate - now) / 86400000

   const cityObject = pages.urls.find (item => item.name === city)
   url = cityObject.WeatherURL

   await weather.goto ( `https://weather.com/$ {url}`, {timeout: 0})

   const temperaturesList = (await weather. $$ ( '[data-testid = detailsTemperature]')) [daysAmount]
   const temperatures = await temperaturesList. $$ ( 'span')
   const maxTemp = await temperatures [0] .evaluate (node => node.innerText),
   minTemp = await temperatures [2] .evaluate (node => node.innerText)

   maxTemperatureStr = (Math.round ((parseInt (maxTemp) - 32) * (5/9))) + '°'
   minTemperatureStr = (Math.round ((parseInt (minTemp) - 32) * (5/9))) + '°'
```

```
    if (parseInt (maxTemperatureStr)> 0) {
      forecast.maxTemperature = `+ $ {maxTemperatureStr}`
    }
    else {
      forecast.maxTemperature = maxTemperatureStr
    }

    if (parseInt (minTemperatureStr)> 0) {
      forecast.minTemperature = `+ $ {minTemperatureStr}`
    }
    else {
      forecast.minTemperature = minTemperatureStr
    }


    const windList = (await weather. $$ ( 'div [data-testid = wind]')) [daysAmount]
    const windSpan = await windList. $ ( 'span')
    const wind = await windSpan.evaluate (node => node.innerText)
    const windStr = wind.split ( ") [1]

    forecast.wind = Math.round (windStr / 2.237) + 'm / s (mean)'

    const precipList = (await weather. $$ ( '[data-testid = "Precip"]')) [daysAmount]
    const precipSpan = await precipList. $ ( 'span')
    const precips = await precipSpan.evaluate (node => node.innerText)

    forecast.precips = precips

    const overcastList = (await weather. $$ ( '[data-testid = "wxIcon"]')) [daysAmount]
    const overcastSpan = await overcastList. $ ( 'span')
    const overcast = await overcastSpan.evaluate (node => node.innerText)

    forecast.overcast = overcast

    weather.close ()


    return forecast
}
const getSinoptik = async function (city, date) {

    const sinoptik = await browser.newPage (),
    forecast = {
      name: 'sinoptik.ua'
    }

    const cityObject = pages.urls.find (item => item.name === city)
    url = cityObject.SinoptikURL
    await sinoptik.goto ( `https://sinoptik.ua/$ {url} / $ {date}`)
    sinoptik.screenshot ({path: 'relative', type: 'jpeg'})
```

```
const minTempNode = await sinoptik. $ ( '. loaded .min span')
   const minTemp = await minTempNode.evaluate (node => node.innerText)
   const maxTempNode = await sinoptik. $ ( '. loaded .max span')
   const maxTemp = await maxTempNode.evaluate (node => node.innerText)

   forecast.maxTemperature = maxTemp
   forecast.minTemperature = minTemp

   const winds = await sinoptik. $$ ( 'tr> td> .wind'),
   windsArray = []
   for (let i = 0; i <winds.length; i ++) {
      let el = await winds [i] .evaluate (node => node.innerHTML)
      windsArray.push (el)
   }

   forecast.wind = (Math.max.apply (null, windsArray)) + 'm / s (max)'

   const precipsTr = await sinoptik. $$ ( '. weatherDetails> tbody> tr')
   const precips = await precipsTr [7]. $$ ( 'td'),
   precipsArray = []

   for (let i = 0; i <precips.length; i ++) {
      let el = await precips [i] .evaluate (node => node.innerHTML)
      if (el === '-') {
         precipsArray.push (0)
      } Else {
         precipsArray.push (+ el)
      }
   }

   forecast.precips = (Math.max.apply (null, precipsArray)) + '%'

   const overcastDiv = await sinoptik. $ ( '. main.loaded> .weatherIco')
   const overcast = await overcastDiv.evaluate (node => node.getAttribute ( 'title'))

   forecast.overcast = overcast

   sinoptik.close ()

   return forecast

}

const options = {
   city: '',
   date: ''
}

const methods = {
   'Gismeteo.ua': getGismeteo,
   'Sinoptik.ua': getSinoptik,
   'Weather.com': getWeather
```

```
}

fastify.register (require ( 'fastify-cors'));

fastify.get ( '/', async (request, reply) => {
   const resource = Object.keys (request.query) [0]
   if (! methods [resource]) {
      reply.callNotFound ()
   }

   const forecast = methods [resource] (options.city, options.date)
   return forecast
})



fastify.get ( '/ cities', async (request, reply) => {
   let arr = pages.urls;
   let cities = arr.map (el => el.name)
   return cities
})

fastify.get ( '/ resourses', async (request, reply) => {
   const resourses = Object.keys (methods)

   return resourses
})

fastify.post ( '/', async (request, reply) => {
   const body = request.body

   options.city = body.city
   options.date = body.date

   return options
})



fastify.listen (5000)



<Template>
    <Tr class = "tr">
       <Td class = "name table-bordered">
          {{Forecast.name}}
       </ Td>
       <Td>
          {{Forecast.maxTemperature}}
       </ Td>
       <Td>
          {{Forecast.minTemperature}}
```

```
        </ Td>
        <Td>
          {{Forecast.wind}}
        </ Td>
        <Td>
          {{Forecast.precips}}
        </ Td>
        <Td>
          {{Forecast.overcast}}
        </ Td>
    </ Tr>
</ Template>

<Script>
export default {
    name: 'Forecast',
    props: {
        forecast: {
            type: Object,
            required: true
        }
    }
}
</ Script>

<Style>
.name {
    background-color: #fcfafa
}
</ Style>



<Template>
  <Div class = "container">
    <TheTable: forecasts = "forecasts" />
    <TheModal
      @ Get = "onGetForecasts"
      @ Close = "onClose"
      : Cities = "cities"
      : Resourses = "resourses"
      : Closable = "isModalClosable"
      v-if = "isWaiting"
    />
    <TheLoader v-if = "isLoading" />
  </ Div>
</ Template>

<Script>
import {mapGetters, mapActions} from "vuex";
import TheTable from "../components/TheTable.vue";
import TheModal from "../components/TheModal.vue";
import TheLoader from "../components/TheLoader.vue";
```

```
export default {
  components: {
    TheTable,
    TheModal,
    TheLoader
  },
  computed: {
    ... mapGetters ({
      forecasts: "store / getForecasts",
      isLoading: "store / isLoading",
      isWaiting: "store / isWaiting"
    }),
    isModalClosable: ({forecasts}) => {
      if (forecasts.length === 0) {
        return false;
      } Else {
        return true;
      }
    }
  },
  methods: {
    ... mapActions ({
      setRequest: "store / setRequest",
      setOptions: "store / setOptions",
      loadForecasts: "store / loadForecasts",
      changeWaiting: "store / setWaiting"
    }),
    onClose () {
      this.changeWaiting (false);
    },
    onGetForecasts (city, date, request) {
      const options = {
        city: city,
        date: date
      };

      this.setOptions (options);
      this.setRequest (request);
      this.loadForecasts ();
    }
  },
  async asyncData ({$ axios}) {
    const cities = await $ axios. $ get ( "/ cities");
    const resourses = await $ axios. $ get ( "/ resourses");
    return {cities, resourses};
  }
};
</ Script>

<Style>
</ Style>
```

ДОДАТОК Б
ДЕМОНСТРАЦІЙНИЙ МАТЕРІАЛ У ВИГЛЯДІ ПРЕЗЕНТАЦІЇ

№	Позначення	Найменування	Дод. відомості
		Текстові документи	
1	ГЮИК. 502810.001 ПЗ	Пояснювальна записка	73с., ф. А4
2		Додаток А	8с., ф. А4
		код програми	
3		Додаток Б	12с., ф*.ppt
		демонстраційний	
		матеріал у вигляді	
		презентації	

					ГЮИК. 502810.001 ВД			
Змін.	Арк.	№ документа	Підп.	Дата				
Розроб.		Біленець О.О			Розробка автоматизованої системи обробки даних з розподілених інформаційних джерел	Літ.	Аркуш	Аркушів
Перевір.		Бабак І.М.						73
Н. контр.						ХНУРЕ Кафедра КІТАМ		
Затвер.		Невлюдов І.Ш						