Avancerad Webbteknologi

Gustav B

1.1 Ansluta till databas

För att kunna hämta datan som ska visas på webbapplikationen krävs det att API är kopplat till databasfilen “weather.db”. Första steget i att göra detta är att använda oss av “sqlite3” som visas på Figur 1

```
var sqlite3 = require("sqlite3").verbose()
```
Figur 1

Detta resulterar i att det går att ansluta oss till den specifika databasfilen “weather.db” som visas i Figur 2.

```
let db = new sqlite3.Database("./weather.db", (err) => {
    if (err) {
        console.log(err.message)
    }
    else {
        console.log("Connected to db")
    }
})
```
Figur 2

1.2 Ansluta till server

För att kunna hämta data med det rest API jag skapat behöver det anslutas till en server vilket sker med hjälp av “express” och “cors”

```
const express = require("express")
const cors = require("cors")
const app = express()
```
Figur 3

När jag försökte att ansluta till servern utan “cors” fick jag error vilket troligtvis har med webbläsarens säkerhet att göra. För att ta mig förbi corse-error behövde jag inkludera “cors” som i Figur 3.

Nu går det att ansluta till servern utan några problem. I detta fallet på port 5000 som i Figur 4.

```
app.get("/", function (req, res) {
    res.send("Connected to server")
})
```

```
app.listen(5000, function () {

})
```
Figur 4

1.3 Hämta/SELECT data från databas

Nu när det går att ansluta sig till servern och databasen så går det att hämta data från databasen och skicka det med hjälp av rest api.

Figur 5 visar ett exempel på “forecast” data som hämtas från databasen och skickas med “status(200)”. Detta exemplet tar in två parametrar “ort” och “days” som sedan används i den sql query som ställs mot databasen (Figur 6).

```
// Get forecast
app.get("/forecast/:ort/:days", function (req, res) {
    var paramDays = parseInt(req.params.days)
    let fromdate = new Date("2020-06-09")
    let todate = new Date("2020-06-09")
    todate.setDate(todate.getDate() + paramDays);

    fromdate = fromdate.toISOString().split('T')[0]
    todate = todate.toISOString().split('T')[0]

    let sql = `SELECT * FROM forecast where fromtime>='${fromdate}' AND
totime<='${todate}' and name='${req.params.ort}'`
    db.all(sql, [], (err, rows) => {
        if (err) {
            throw err
        }
        else {
            res.type("application/json")
            res.status(200).send(rows)
        }
    })
})
```
Figur 5

```
let sql = `SELECT * FROM forecast where fromtime>='${fromdate}' AND
totime<='${todate}' and name='${req.params.ort}'`
```
Figur 6

Figur 7

```
[
  {
    "name": "Grums",
    "fromtime": "2020-06-09 00:00:00",
    "totime": "2020-06-09 06:00:00",
    "periodno": 0,
    "periodname": "Night",
    "auxdata":
"{\"TUNIT\":\"celsius\",\"TVALUE\":\"20.6\",\"ALTUNIT\":\"fahrenheit\",\"ALTVA
LUE\":\"69.08\",\"NUMBER\":\"4\",\"WSYMB3NUMBER\":\"6\",\"FNAME\":\"Cloudy\",\
"RUNIT\":\"mm\",\"RVALUE\":\"0\",\"DEG\":\"203\",\"CODE\":\"SSW\",\"NAME\":\"S
outh-southwest\",\"MPS\":\"0.6\",\"NAME\":\"Light
Air\",\"UNIT\":\"hPa\",\"VALUE\":\"1203\"}"
  },
  {
    "name": "Grums",
    "fromtime": "2020-06-09 06:00:00",
    "totime": "2020-06-09 12:00:00",
    "periodno": 1,
    "periodname": "Morning",
    "auxdata":
"{\"TUNIT\":\"celsius\",\"TVALUE\":\"23.6\",\"ALTUNIT\":\"fahrenheit\",\"ALTVA
LUE\":\"74.48\",\"NUMBER\":\"9\",\"WSYMB3NUMBER\":\"11\",\"FNAME\":\"Thunder\"
,\"RUNIT\":\"mm\",\"RVALUE\":\"0.8\",\"DEG\":\"213\",\"CODE\":\"SSW\",\"NAME\"
:\"South-southwest\",\"MPS\":\"0.3\",\"NAME\":\"Calm\",\"UNIT\":\"hPa\",\"VALU
E\":\"1156\"}"
  },
  {
    "name": "Grums",
    "fromtime": "2020-06-09 12:00:00",
    "totime": "2020-06-09 18:00:00",
    "periodno": 2,
    "periodname": "Day",
    "auxdata":
"{\"TUNIT\":\"celsius\",\"TVALUE\":\"18.6\",\"ALTUNIT\":\"fahrenheit\",\"ALTVA
LUE\":\"65.48\",\"NUMBER\":\"5\",\"WSYMB3NUMBER\":\"7\",\"FNAME\":\"Fog\",\"RU
NIT\":\"mm\",\"RVALUE\":\"0\",\"DEG\":\"210\",\"CODE\":\"S\",\"NAME\":\"South\
",\"MPS\":\"0.1\",\"NAME\":\"Calm\",\"UNIT\":\"hPa\",\"VALUE\":\"1200\"}"
  },
  {
    "name": "Grums",
    "fromtime": "2020-06-09 18:00:00",
    "totime": "2020-06-09 00:00:00",
    "periodno": 3,
    "periodname": "Evening",
    "auxdata":
"{\"TUNIT\":\"celsius\",\"TVALUE\":\"16.6\",\"ALTUNIT\":\"fahrenheit\",\"ALTVA
LUE\":\"61.88\",\"NUMBER\":\"6\",\"WSYMB3NUMBER\":\"8\",\"FNAME\":\"Rain
Showers\",\"RUNIT\":\"mm\",\"RVALUE\":\"0\",\"DEG\":\"210\",\"CODE\":\"SSW\",\
"NAME\":\"South-southwest\",\"MPS\":\"3.2\",\"NAME\":\"Light
Breeze\",\"UNIT\":\"hPa\",\"VALUE\":\"1161\"}"
  }
]
```

Figur 8

På Figur 7 så visas ett exempel där jag tagit en screenshot på min url där jag angav “Grums” som ort och antalet dagar 1. Figur 8 innehåller datan från den “status(200)” som skickas i Figur 5.

Värt att notera från Figur 5 är att jag hade lite problem med data-format på datum och behövde testa mig fram för att konvertera ett javascript date-object så det matchade sql datatypen “datetime”.

Nästan alla andra “GET” från databasen är väldigt lika varandra som exempelvis “user” som visas i Figur 9. Skillnaden i detta fallet är att det inte finns några parametrar och att alla användare hämtas.

```
// Get users
app.get("/user", function (req, res) {
    let sql = `SELECT * FROM 'user'`
    db.all(sql, [], (err, row) => {
        if (err) {
            throw err
        }
        else {
            res.type("application/json")
            res.status(200).send(row)
        }
    })
})
```

Figur 9

1.3 Lägga till/INSERT “user”

För att lägga till en “user” i databasen så används en “INSERT” istället för en “SELECT” som i de tidigare exemplen.

Att lägga till något i databasen är väldigt enkelt. Det är dock viktigt att tänka på vilken data som behövs och hur man ska hantera null värden i fall att alla parametrar inte är ifyllda. I detta fallet så valde jag att sätta “image” till null eftersom den är null på alla andra användarna i databasen också. Just nu så har jag ingen hantering för vad som händer om man inte skriver in exempelvis något i parametern “email” men jag

antar att alla parametrar är viktiga att ha, det vill säga det borde inte finnas någon användare som inte har en email kopplat.

Jag testade koden i Figur 10 genom att försöka lägga till en användare och sedan köra “http://localhost:5000/user” för att hämta alla användare i databasen. Resultatet visade sig att det hade skapats en ny användare vilket har id:5.

```
// Add a user
app.get("/user/:id/:username/:email/:favlocation", function (req, res) {

    let sql = `INSERT INTO user VALUES (${req.params.id},
'${req.params.username}', '${req.params.email}', '${req.params.favlocation}',
null);`
    db.run(sql)
})
```

Figur 10


```
[
  {
    "id": 1,
    "username": "Greger Ohlsson",
    "email": "greger.olsson@net.nu",
    "favlocation": "Grums",
    "image": null
  },
  {
    "id": 2,
    "username": "Sven Grunden",
    "email": "svenne.grunden@swnet.se",
    "favlocation": "Barcelona",
    "image": null
  },
  {
    "id": 3,
    "username": "Nisse Hult",
    "email": "nissehult@net.nu",
    "favlocation": "Grums",
    "image": null
  },
  {
    "id": 4,
    "username": "Glenn Nilsson",
    "email": "glen@swnet.se",
    "favlocation": "Arjeplog",
    "image": null
  },
  {
    "id": 5,
    "username": "name1",
    "email": "email1",
    "favlocation": "location1",
    "image": null
  }
]
```

Figur 11

1.4 Använda API i applikationen

Det sista steget är att applicera det API jag skapat i väderappen genom att använda det för att hämta datan som ska visas. För att göra detta anger jag den url som ska användas för att få ut den data som den tidigare php filen gjorde.

Ett exempel på hur detta ser ut är på Figur 12 där forecast data hämtas. Istället för att länka till en php fil används mitt API istället.

```
      // Promise, forecast
      let pForecast = new Promise(async (resolve, reject) => {
        const url =
`http://localhost:5000/forecast/${params2.ort}/${params2.days}`
        const response = await fetch(url, {
          method: 'GET',
          headers: { 'Content-Type': 'application/json' },
        })
          .then((response) => response.json()).then(data => {
            console.log(url)
            resolve(data)
          });
      })
```

Figur 12