Kodeoppgave 2019

Dette dokumentet avslutter med en liten oversikt over tankegangen min da jeg først fikk oppgaven. Jeg opprettet dokumentet og begynte med å notere ned noen tanker jeg hadde om hvordan oppgaven skulle løses. Jeg velger å la dette være med, da det kanskje kan være interessant for dere å se hvordan jeg har tenkt.

# Del 1:

Besvarelsen min består av 2 deler, der den første delen er en lånekalkulator på nett. Denne har noen slidere som brukeren kan bruke til å endre på lånet, og en graf som visualiserer lånet over tid. Her kan man se hvordan avdraget, renter og restbeløp utvikler seg. Det er også lagt inn en graf som viser fordelingen av lånebeløp, renter og gebyrer ved nedbetalt lån. Nettsiden henter data fra en Node.js server jeg har skrevet, som da er den andre delen av oppgaven.

Serveren jeg har lagd er bygget på Node.js og express.js, og har tre funksjoner:

1. Levere en statisk nettside til brukeren.
2. Lage nedbetalingsplan som nettsiden kan bruke
3. Lage filer som brukeren kan laste ned

Nedbetalingsplanen har så og si samme format som Stacc sin API har. Dette er fordi jeg i første omgang fikk nettsiden til å fungere vha. Stacc sin API, og dette var den enkleste måten å løse det på når jeg skulle gå over til å bruke min egen API. Jeg regner også med at API’et til Stacc har den informasjonen som er viktigst, og derfor tenkte jeg det var viktig at denne informasjonen måtte med. Jeg har også lagt til noen andre properties på nedbetalingsplan-objektet som jeg følte det var greit å ha med.

Serveren gjør kun 2 tester på POST-requesten, den sjekker at content-type er json og at laanetype finnes i json-objektet. Den effektive renten som blir regnet ut av serveren er ikke 100% korrekt, da jeg ikke forstår den fullstendige formelen for utregning av effektiv rente.

Serveren skriver og leverer filer i .csv og .json format dersom brukeren ønsker å laste ned nedbetalingsplanen.

Besvarelsen ligger på githuben min: <https://github.com/haardes/Kodeoppgave-Stacc-2019>

For å få «programmet» til å fungere må man starte node.js vha. å navigere seg til oppgave-mappen i terminalen, og utføre kommandoen «node app.js». Serveren vil da begynne å lytte på localhost port 3000 (<http://localhost:3000/>), hvor man finner nettsiden. Det er også mulig å sende POST-requests til serveren på (<http://localhost:3000/api>). Requests må sendes på følgende format:

{

"laanebelop": 2000000,

"nominellRente": 2.5,

"lopetid": 30,

"terminGebyr": 60,

"saldoDato": "YYYY-MM-DD",

"laanetype": "SERIE" || "ANNUITET"

}

# Del 2:

Okei hva skal jeg gjøre?

Først: Lage en nettside som får data fra Stacc API? Sørge for at jeg har noe å levere om jeg ikke får til Node. Postman til å teste API, konsoll til «analyse»

Hvis nettside er oppe og går: Node.js server som lager en nedbetalingsplan basert på en POST-request. Skal returnere et JSON-objekt som blir fremstilt i en interaktiv graf vha. chart.js som brukeren kan se på.

La brukeren laste ned nedbetalingsplanen som fil? JSON og CSV burde gå enkelt, hva med excel-fil?

## Prototype:

Et bilde som inneholder skjermbilde

Automatisk generert beskrivelse

## Respons fra API:

{

"nedbetalingsplan": {

"innbetalinger": [

{

"restgjeld": 2000000.0,

"dato": "2020-01-01",

"innbetaling": 0.0,

"gebyr": 0.0,

"renter": 0.0,

"total": 0.0

},

{

"restgjeld": 1995598.3802136178,

"dato": "2020-02-01",

"innbetaling": 4401.619786382111,

"gebyr": 900.0,

"renter": 5082.071266533372,

"total": 10383.691052915483

},

## Node-server:

Lage betalingene rekursivt? Ha en plan som input og forrige innbetaling, kan hente dager fra forrige og legge innbetalingene til planen vha plan. Trekke fra måneder på «gjenstående tid» eller øke dato på innbetaling og sjekke opp mot lånets varighet og dato for ferdig?

JSON: Finn ut hvordan man returnerer en JSON-fil, objektet bør allerede finnes på serveren

CSV: Finnes det et verktøy for JSON -> CSV? Eller lage filen med hjelp av noen for-løkker?

EXCEL: Går det ann å konvertere en filtype til excel? Hva med all formatering? Sjekk ut

Nedlasting = umulig?? Prøv mandag, spør noen på skolen. NVM, figured it out!