Projektrapport - Långpendlaren

Om tjänsten
Långpendlaren som vi valt att skapa har syfte att kombinera trafikverkets och spotifys APIer för att skapa spellistor som varierar utefter hur lång olika resor är. Målet är att spellistorna ska skapas utifrån preferenser så som genre, favoritartister eller annat som gör att användaren kan påverka vilka typer av spellistor som skapas.

Applikationen som levererats gör detta genom att reslängden hämtas genom att användaren väljer stationer för både avresa och ankomst, sedan beräknas tidsskillnaden mellan dessa stationer. Denna tidsskillnaden används sedan för att skapa en spellista som är ungefär lika lång som tiden, låtar hämtas från spotify och läggs till i användarens spellista tills reslängden är uppnådd.

Spellistorna som skapas kan användaren sedan välja att spara eller ta bort beroende på om hen är nöjd med förslaget eller inte. Vidare så görs allt detta genom att användaren loggas in på spotify genom vår tjänst. Gällande preferenser för spellistorna som skapades så har det ännu inte implementerats val som genre och artister ännu.

Den tekniska lösningen
Vilka tekniker ni har använt för att implementera applikationen. Detta gäller både frontend och eventuell backend
Sett till Frontend använder tjänsten HTML, CSS och Javascript. Presentationen av detta sker genom Electron som paketerar detta i form av en applikation likt en desktopapplikation men använder webbtekniker. Sett till Backend använder tjänsten Java och webbramverket Javalin för konstruktionen av vårt egna API samt för att skicka XML-förfrågningar för att hämta data från Trafikverkets och Spotifys API.

En beskrivning av de datakällor/externa API:er ni använder.
Tjänsten använder externa API:er i Trafikverkets API för att hämta information om stationer, avgångar, ankomster för tåg som sedan bearbetas i applikationen. Spotifys API innehåller funktionalitet för att logga in till sitt spotifykonto, detta bidrar till att användare av vår tjänst kan skapa spellistor innehållande låtar som sedan sparas till deras spotifykonto.

Om ni behandlar den data ni använder, ska även detta beskrivas.
Tjänsten sparar enbart datan lokalt för varje session vilket innebär att ingenting förutom spellistorna sparas permanent. Datan för varje resa sparas lokalt tills dess att applikationen startas om. Datan som används i tjänsten omfattar alla stationer, avgångstider och ankomsttider för ett specifikt tåg. Utöver detta hanteras också data från Spotify genom att hämta låtar och sedan lägga till dessa i användarens konto. Samtlig data hämtas först, behandlas genom att skrivas om och presenteras i ett annat Json format för att förenkla, sedan skicka tillbaka detta från APIt som används i frontend.

Tjänsten behandlar datan omfattande tider, detta genom att hämta avgångstiden från en station och ankomsttiden till en station för ett valt tåg. Reslängden mellan stationerna används sedan för att bestämma längden på spellistan som på ett ungefär skall matcha reslängden. Samma förenkling på datan som hämtas från trafikverket görs i vår API gällande Json formateringen.

Användarmanual

Alternativ 1
Installationsinstruktioner

1. Öppna den zippade mappen.
2. Kör filen install.cmd om det är windows, alternativt install.sh för mac/linux.

Körinstruktioner

1. Bara att köra!

Alternativ 2
Installationsinstruktioner

1. Installera Node.js och NPM genom att ladda ner på https://nodejs.org/en/ alternativt om det redan är installerat stämma av att det finns genom att skriva node -v och sedan npm -v i terminalen.
2. Installera electron och dependencies. Detta görs genom att öppna terminalen i frontendmappen och sedan skriva npm i

Körinstruktioner

1. Starta servern genom att köra backend.java
2. Kör frontend.exe
3. Starta sedan electron genom att öppna terminalen i frontendmappen och sedan skriva kommandot npm start

API-dokumentation
Det vi tänkte när vi designade vår API var att vi ville samla både Trafikverkets och Spotify APIer under ett API där vi sedan behandlade datan. Därav försökte vi att designa något som gjorde att all den funktionalitet som vi behövde implementeras i vårt eget API. Vårt egna API blir alltså som en samling för funktionaliteten som vi behöver från andra APIer.

Vi funderade över några best practices som vi försökte implementera i vårt API. Exempel på detta var att resursernas namn var substantiv och i plural för att göra det enklare att förstå. Även att namngivning representerar vilken del som funktionaliteten berör, som /trafikverket för tåg och /spotify för musik. I vår APIs design så använder vi GET, PUT och POST och följer detta så att exempelvis GET inte ändrar resurser.

Utefter detta så hämtade skapade vi olika endpoints beroende på det vi behövde från de externa APIerna. Denna funktionalitet blev endpoints som vi sedan använder för att exempelvis logga in användaren på Spotify, hämta resor och behandla datan som vi får. Designen blev därför olika endpoints som namngavs utifrån om det var från Spotify eller Trafikverket. Några exempel är följande:

- localhost/trafikverket/stations

- localhost/trafikverket/trains/stops/{trainIdent}
- localhost/spotify/login
- localhost/spotify/playlist/add/{pid}/{tids}

Den fulla dokumentation om långpendlarens API finns tillgänglig via följande länk:
https://langpendlaren.docs.apiary.io/#reference/0/data