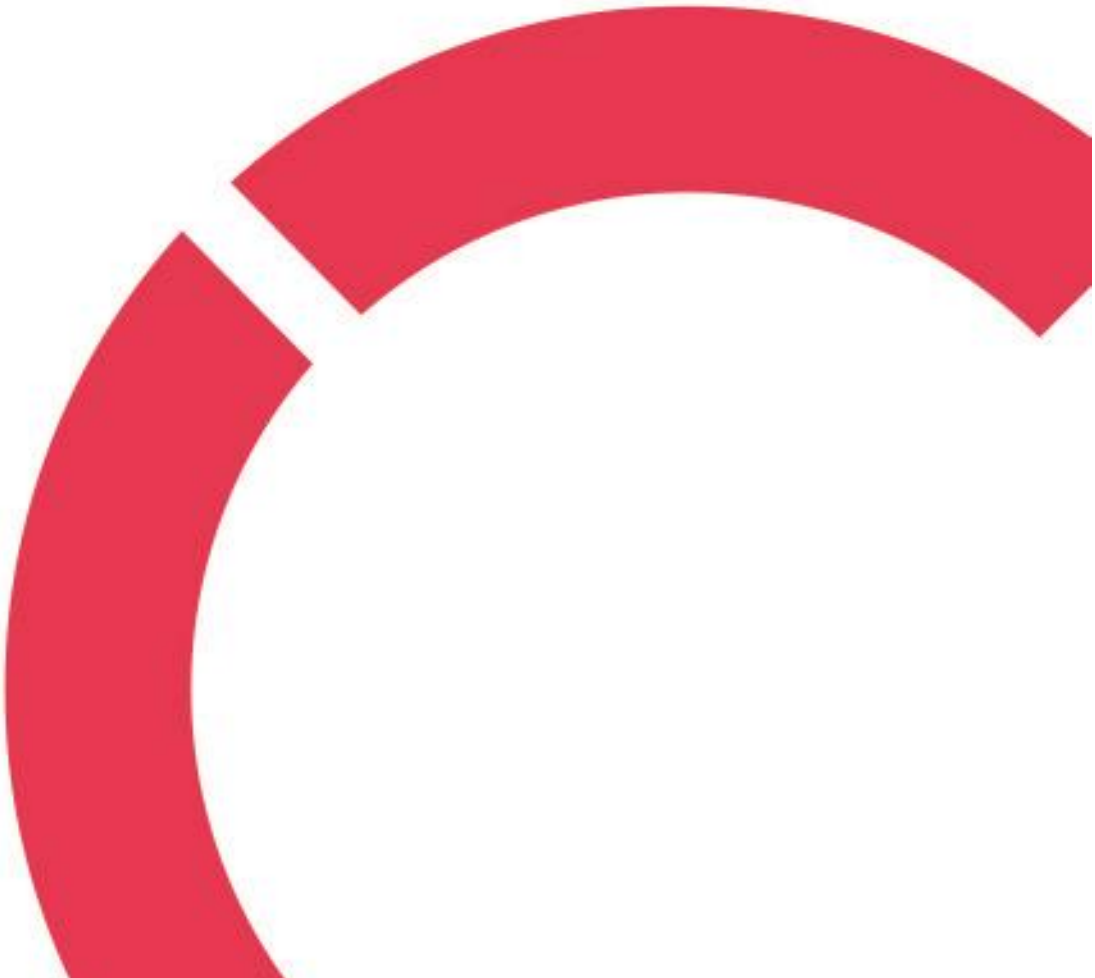
Aigar Voog
Piia Virrankoski
Sakari Sorja

RESEPTILOTTO
Dokumentaatio

CENTRIA-AMMATTIKORKEAKOULU
Insinööri, Tieto- ja viestintätekniikka
Syksy 2023



SISÄLLYS

1 OSAYLEISTÄ

1.1 Linkki repositorioon

- https://github.com/AigarVoog/ReseptiLotto

2 UI DOKUMENTAATIO (REACT)

2.1 Järjestelmä- ja Ohjelmistovaatimukset

- ***Käyttöjärjestelmä***: Mikä tahansa, jossa on tuettu web-selain.
- ***Selain***: Suositellaan uusimpia versioita Chrome, Firefox, Safari tai Edge selaimista.
- ***Node.js***: Versio 10.x tai uudempi (kehitystä varten).
- ***npm***: Viimeisin versio (riippuvuuksien hallintaan).

2.2 Arkkitehtuuri

- **Komponenttihierarkia:** Tässä projektissa käytetään modulaarista komponenttihierarkiaa, joka helpottaa jatkokehitystä ja ylläpitoa

2.3 Teknillinen Dokumentaatio

- **Kehitysympäristö:** Projektia on kehitetty käyttäen Visual Studio Code -editoria.
- **Frameworkit ja kirjastot:**
    - React
    - Material UI (MUI)
    - Styled Components
- **Käyttöliittymäkomponentit:**
    - **App.jsx:** Pääkomponentti, joka hallitsee sovelluksen tilaa ja renderöi muut komponentit.
    - **AppBar.jsx:** Sovelluksen yläpalkki.
    - **RecipeCard.jsx:** Reseptin korttinäkymä, joka näyttää reseptin tiedot.
    - **SearchFilters.jsx:** Hakufilttereiden komponentti, joka mahdollistaa ainesosien valinnan.
    - **PopUp.jsx:** Dialogikomponentti, joka näyttää satunnaisia viestejä käyttäjälle, kun tietyt ehdot täyttyvät (esimerkiksi 'Hae resepti' -napin painallus kolme kertaa).
    - **ThemeWrapper.jsx:** Teeman asettamiseen käytetty komponentti, joka määrittelee sovelluksen visuaalisen ilmeen.

- **Tietoliikenne:** Axios tai natiivi fetch API:n käyttö HTTP-pyyntöjen tekemiseen.
- **Versionhallinta:** Git.

Kehittämisen Ohjeet

- **Projektin kloonaus:** Kloonaa projektirepositorio ja siirry projektin hakemistoon.
- **Riippuvuuksien asennus:** Suorita npm install projektin juuressa.
- **Kehityspalvelimen käynnistys:** Käynnistä kehityspalvelin komennolla ***npm run dev***
- **Ympäristömuuttujat:** Aseta tarvittavat ympäristömuuttujat, kuten API:n päätepisteet, ’***.env’***-tiedostossa.

2.4 Loppukäyttäjän Dokumentaatio

Käyttöohjeet

- Käynnistä sovellus selaimessa.
- Valitse ainesosat SearchFilters-komponentilla.
- Hae reseptiä painamalla "Hae Resepti" -nappia.
- Tarkastele reseptin tietoja RecipeCard-komponentissa.

2.5 Rajapinta

Pyyntöjen esimerkit:

- **POST /getRecipe:**

```
fetch('https://sakarin-endpointti.com/getRecipe', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify({ ingredients: selectedIngredients }),
})
```

2.6 Komponenttien Tarkempi Selostus

2.6.1 App.jsx

Yleiskatsaus

App.jsx on pääkomponentti sovelluksessa, joka hallitsee keskeisiä tiloja, kuten valittua re-septiä (currentRecipe), vanhaa reseptiä (oldRecipe) animaatioita varten (isAnimating), sekä valittuja ainesosia (selectedIngredients). Tämä komponentti huolehtii myös sovelluksen päätoiminnoista: reseptin hakemisesta, uuden reseptin generoinnista ja vanhan reseptin hylkäämisestä.

Tilat (useState)

- **currentRecipe:** Tämänhetkinen näytettävä resepti.
- **oldRecipe:** Edellinen resepti, jota käytetään poistumisanimaatiolle.
- **isAnimating:** Boolean arvo, joka kertoo, onko siirtymäanimaatio meneillään.
- **selectedIngredients:** Taulukko, joka sisältää käyttäjän valitsemat ainesosat.
- **recipeRequestCount:** Seuraa kuinka monta kertaa käyttäjä on pyytänyt uutta reseptiä.

Funktiot

- **fetchRecipeByIngredients:** Asynkroninen funktio, joka hakee reseptin valittujen ainesosien perusteella käyttäen ***fetch***-API:a ja asettaa sen ***currentRecipe***-tilaan.
- **generateRecipe:** Asettaa nykyisen reseptin hylkäämiseen ja kutsuu ***fetchRecipeByIngredients***-funktiota, jolla haetaan uusi resepti.
- **discardRecipe:** Hylkää nykyisen reseptin asettamalla ***isAnimating***-tilan ***true***:ksi, mikä aloittaa poistumisanimaation, ja asettaa ***currentRecipe***-tilan ***null***:ksi.
- **generateNewRecipe:** Generoi satunnaisen reseptin ***sampleRecipes***-taulukosta ja aloittaa syöttöanimaation.
- **generateMockRecipe:** Simuloi uuden reseptin generointiprosessia, ensin hylkäämällä nykyisen ja sitten valitsemalla satunnaisen ***sampleRecipes***-taulukosta.
- **setRecipeRequestCount:** Päivittää ***recipeRequestCount***-tilan arvoa, jota käytetään seuraamaan, kuinka monta kertaa käyttäjä on pyytänyt uutta reseptiä.

Käyttöliittymäelementit

- **ThemeWrapper**: Määrittelee ja soveltaa MUI-teeman koko sovellukseen.
- **AppBar**: Näyttää sovelluksen otsikkopalkin.
- **SearchFilters**: Mahdollistaa ainesosien valinnan ja hallinnan.
- **Button**: Kaksi painiketta, joista toinen hakee reseptin ja toinen generoi mock-reseptin.
- **RecipeCard**: Renderöi reseptin tiedot korttinäkymässä.
- **PopUp.jsx:** Dialogikomponentti, joka näyttää satunnaisia viestejä käyttäjälle.
- **Box** ja **Stack**: MUI-komponentteja, jotka auttavat asettelemaan sisältöä visuaalisesti miellyttävästi ja loogisesti.

Tyylit ja Animaatiot

Komponentissa käytetään MUI:n sx-propia määrittämään tyylejä ja luomaan siirtymäanimaa-tioita slide-in ja slide-out luokkien avulla, jotka määritellään sovelluksen CSS-tiedostossa.

Ympäristö ja Riippuvuudet

- MUI (Material-UI) on valittu käyttöliittymäkomponenttien kirjastoksi, jonka avulla luodaan visuaalisesti miellyttävä ja käytettävyysstandardit täyttävä käyttöliittymä.
- fetch-API:ta käytetään verkkopyyntöjen tekemiseen backend-palvelimeen.

Tärkeät Huomiot Kehittäjille

- Kehittäjän tulee varmistaa, että backend-palvelin, johon ***fetch***-pyynnöt lähetetään, on käynnissä ja sen URL on oikein määritelty ympäristömuuttujissa.
- Animaatioaikoja ja ***setTimeout***-kutsujen viiveitä voidaan säätää käyttöliittymän responsiivisuuden parantamiseksi.
- ***SampleRecipes***-taulukko toimii esimerkkinä ja mock-datan lähteenä kehityksen aikana. Tämän datan avulla voidaan simuloida ja testata käyttöliittymän toimintaa ilman backend-yhteyttä. Kehittäjien tulisi huomata, että tämä data on staattista ja sen tulee korvata dynaamisella datalla tuotantovaiheessa.

2.6.2 AppBar.jsx

Yleiskatsaus

AppBar.jsx on yksinkertainen, mutta tärkeä komponentti, joka vastaa sovelluksen yläpalkin näyttämisestä. Se käyttää Material-UI:n ***AppBar***-komponenttia tarjotakseen visuaalisesti miellyttävän ja selkeän otsikkorivin sovelluksen käyttöliittymään.

Tärkeimmät Elementit

- **AppBar**: Material-UI:n komponentti, joka toimii sovelluksen yläpalkkina.
- **Toolbar**: Työkalurivi, joka sijoittaa sisältönsä keskelle ja toimii säiliönä typografiselle otsikolle.
- **Typography**: Material-UI-komponentti, jota käytetään otsikon tekstiin. Se on asetettu variantille ***h5***, joka määrittää tekstin koon ja ulkoasun.

2.6.3 SearchFilters.jsx

Yleiskatsaus

SearchFilters.jsx on React-komponentti, joka tarjoaa käyttäjille mahdollisuuden valita ainesosia monipuolisesta listasta. Komponentti käyttää Material-UI:n ***Autocomplete***-komponenttia, joka sallii monivalinnat ja tarjoaa käyttäjälle kyvyn kirjoittaa ja valita ehdotuksia dynaamisesti.

Tärkeimmät Elementit

- **Autocomplete**: Material-UI:n komponentti, joka ottaa vastaan ainesosien listan ja mahdollistaa käyttäjän suorittamat haut ja valinnat.
- **Chip**: Pieni komponentti, joka näyttää visuaalisesti valitun ainesosan ja antaa mahdollisuuden poistaa tämän valinnan.
- **TextField**: Tekstikenttä, johon käyttäjä voi kirjoittaa ja jonka kautta Autocomplete-komponentti ottaa vastaan syötettä.

Propsit

- **selectedIngredients**: Prop, joka ottaa vastaan ylemmän tason komponentista App tulevan tilan, joka sisältää käyttäjän valitsemat ainesosat.
- **onIngredientsChange**: Callback-funktio, joka välitetään ***App***-komponentilta ja jota kutsutaan, kun valittujen ainesosien lista muuttuu.

Toiminnallisuus

- Kun käyttäjä kirjoittaa ***TextField***-kenttään, ***Autocomplete*** ehdottaa ainesosia listalta, ja käyttäjä voi valita yhden tai useamman.
- Valitut ainesosat näytetään ***Chip***-komponenteilla, joista kukin on varustettu poistotoiminnolla.
- Käyttäjä voi myös kirjoittaa ainesosan nimen vapaasti (***freeSolo***-ominaisuus), joka ei ole listalla, antaen näin lisää joustavuutta syötteen antamiseen.

2.6.4 RecipeCard.jsx

Yleiskatsaus

RecipeCard.jsx on visuaalisesti miellyttävä komponentti, joka esittää yksittäisen reseptin tiedot korttimuodossa. Se hyödyntää Material-UI:n ***Card***-komponenttien kirjastoa esittääkseen reseptin otsikon, kuvan, kuvauksen sekä ainesosat ja valmistusohjeet taitettavassa alueessa.

Tärkeimmät Elementit

- ***Card***: Pohjakomponentti, joka toimii kontainerina muille kortin elementeille.
- ***CardHeader***: Näyttää reseptin otsikon ja alaotsikon (valmistusaika).
- ***CardMedia***: Näyttää reseptin kuvan.
- ***CardContent***: Sisältää reseptin kuvauksen.
- ***CardActions*** ja ***IconButton***: Toteuttaa toiminnon, jolla voidaan laajentaa korttia näyttämään ainesosalista ja valmistusohjeet.
- ***Collapse***: Piilottaa tai näyttää lisätietoalueen, joka sisältää ainesosat ja ohjeet.

- ***List*** ja ***ListItem***: Esittää ainesosat ja valmistusohjeet listamuodossa.

Tilat (useState)

- ***expanded***: Boolean-tila, joka määrittää, onko taitettava osa avattu vai suljettu.

Funktiot

- ***handleExpandClick***: Tämä funktio kääntää ***expanded***-tilan arvon. Kun käyttäjä klikkaa laajennusnuolta (***ExpandMoreIcon***), tämä funktio aktivoituu ja vaihtaa näkymän avatuksi tai suljetuksi

Tärkeät Huomiot Kehittäjille

- ***Reseptidatan Odotusarvo***: Kehittäjien on huomioitava, että komponentti odottaa ***recipe***-propin, joka sisältää oletuksena tietyt kentät (***title***, ***image***, ***description***, ***ingredients***, ***steps***). Jos rakenteeseen tulee muutoksia, ne tulee päivittää vastaavasti tässä komponentissa.

- ***Kuvan Varakäsittely***: Jos kuvan URL on virheellinen tai kuva puuttuu, tulisi kehittäjän toteuttaa varakäsittely esimerkiksi oletuskuvan näyttämiseksi.

2.6.5 PopUp.jsx

Yleiskatsaus

PopUp.jsx on React-komponentti, joka toimii hälytysdialogina näyttäen käyttäjälle satunnaisesti valitun viestin. Viestin tarkoituksena on huomauttaa käyttäjää, kun hän on tehnyt tietyn määrän toimintoja — tässä tapauksessa, kun ***recipeRequestCount*** on saavuttanut arvon neljä. Komponentti käyttää Material-UI:n ***Dialog***-komponenttia näyttääkseen modal-tyylisen ilmoituksen.

Tilat (useState)

- ***open:*** Boolean-tila, joka määrittää, onko dialogi auki vai kiinni.
- ***randomMessage:*** Merkkijono-tila, joka pitää sisällään näytettävän satunnaisen viestin.

Efektit (useEffect)

- Tämä sivuvaikutus (side effect) tarkkailee ***recipeRequestCount***-tilaa. Kun ***recipeRequestCount*** saavuttaa arvon neljä tai suurempi, dialogi avataan ***(setOpen(true))*** ja valitaan satunnainen viesti ***messages***-taulukosta asettamalla se ***randomMessage***-tilaan.

Funktiot

- ***handleClose***: Kutsuttaessa asettaa open-tilan arvoksi false, jolloin dialogi sulkeutuu.
- ***getRandomMessage***: Palauttaa satunnaisen viestin messages-taulukosta. Se valitsee ensin satunnaisen indeksin ja palauttaa kyseisen indeksin viestin.

Käyttöliittymäelementit

- ***Dialog***: Pohjakomponentti, joka näyttää modal-dialogin.
- ***DialogTitle***: Näyttää otsikon dialogille, joka on tässä tapauksessa "Hohhoijaa!".
- ***DialogContent*** ja DialogContentText: Näyttää sisällön dialogissa, joka on tässä tapauksessa satunnainen viesti randomMessage-tilasta.
- ***DialogActions*** ja ***Button***: Näyttää toimintapainikkeen dialogissa, jonka kautta käyttäjä voi sulkea dialogin.

Tärkeät Huomiot Kehittäjille

- Dialogi on suunniteltu näyttämään, kun ***recipeRequestCount*** saavuttaa neljä. Jos tämän logiikan halutaan olevan muuttuvainen tai konfiguroitavissa, sitä tulisi säätää vastaavasti.
- ***messages***-taulukon viestit on kovakoodattu komponenttiin. Jos viestien pitää päivittyä dynaamisesti tai niiden lähteeksi on esimerkiksi ulkoinen data, tämä rakenne tulisi muuttaa sopivaksi.

2.6.6 ThemeWrapper.jsx

Yleiskatsaus

ThemeWrapper.jsx on komponentti, joka luo ja tarjoaa yhtenäisen teeman koko React-sovelluksen käyttöliittymälle käyttäen Material-UI:n teematoimintoa. Se käärii sovelluksen juurikomponentin (<***App*** />) tai vastaava) ja välittää theme-objektin kaikille lapsikomponenteille ***ThemeProvider***-komponentin avulla. Tämä mahdollistaa määriteltyjen tyylisääntöjen yhtenäisen käytön kaikissa käyttöliittymän osissa.

Tärkeimmät Elementit

- ***ThemeProvider***: Material-UI:n komponentti, joka ottaa theme-objektin ja välittää sen kontekstin kautta kaikille lapsikomponenteille.
- ***createTheme***: Material-UI:n funktio, joka luo mukautetun teeman annetuista asetuksista.

Teeman Määrittely

- ***palette***: Määrittää värimaailman, jota käytetään eri tarkoituksiin (esimerkiksi ***primary***, ***secondary***, ***error***, ***warning***, ***info***, ***success*** jne.) sovelluksessa.
- ***primary ja secondary***: Värit, jotka ovat pääasiallisesti käytössä läpi sovelluksen.
- ***error***, ***warning***, ***info***, ***success***: Värit, joita käytetään spesifisten tilanteiden viestimiseen käyttäjälle, esimerkiksi virheilmoitukset ja varoitukset.
- ***background***: Taustan väriasetukset.
- ***text***: Tekstin väriasetukset.

Käyttö

Komponentti käärii yleensä sovelluksen pääkomponentin (<***App*** />) ja antaa kaikille lapsikomponenteille pääsyn määriteltyihin teema-asetuksiin. Tämän avulla voidaan esimerkiksi varmistaa, että kaikkialla sovelluksessa käytetään yhtenäisiä värimääritelmiä ja muita tyyliasetuksia.

3 BACKEND DOKUMENTAATIO (C#, ASP.NET CORE)

3.1 Järjestelmä- ja Ohjelmistovaatimukset

- **Käyttöjärjestelmä**: Mikä tahansa, jossa on tuettu web-selain.
- **Selain**: Suositellaan uusimpia versioita Chrome, Firefox, Safari tai Edge selaimista.
- **Visual Studio 2022**

3.2 Arkkitehtuuri

- ASP.NET Core tarjoaa helpon mahdollisuuden hyödyntää MVC-arkkitehtuuria ja tässä projektissa se otettiin käyttöön vaikkakin tässä tapauksessa sitä hyödynnetään vain Backend-puolella Controllerin ominaisuudessa (Koska UI toteutettiin tässä projektissa erillisenä kokonaisuutenaan, joka kommunikoi Backendin kanssa erillisen rajapinnan kautta), lukuun ottamatta yhtä pientä mallia (Resepti). MVC = Model (Mallit), Views (Näkymät) ja Controllers (Ohjaimet).

3.3 Teknillinen Dokumentaatio

- **Kehitysympäristö:** Projektia on kehitetty käyttäen Visual Studio 2022 -editoria.
- **Frameworkit ja kirjastot:**
    - Micorosoft.AspNetCore.All (2.0.0)
    - Microsoft.EntityFrameworkCore.SqlServer (2.0.0)
    - Microsoft.AspNetCore.Cors (2.2.0)
    - Microsoft.NETCore.App (2.0.0)
    - Microsoft.VisualStudio.Web.CodeGeneration.Design (2.0.0)
- **Tietoliikenne:** Microsoft.EntityFrameworkCore.SqlServer (2.0.0) tietokanta-hakujen tekemiseen Azuren tietokannasta.
- **Versionhallinta:** Git.

Kehittämisen Ohjeet

- **Projektin kloonaus:** Kloonaa projektirepositorio GitHubista ja käynnistä projekti Visual Studio 2022-editorissa.
- **Riippuvuuksien asennus:**
    - Lisää Microsoft.NETCore.App (2.0.0) asentamalla .NET Core 2.0 SDK osoitteesta: https://dotnet.microsoft.com/en-us/download/dotnet/thank-you/sdk-2.0.0-windows-x86-installer
    - Lisää Microsoft.Data.SqlClient Nuget Package Managerissa kommennolla: NuGet\Install-Package Microsoft.Data.SqlClient -Version 5.1.2. (voi asentaa myös koodiriviltä klikkaamalla hiiren oikealla näppäimellä ja valita Microsoft.Data.SqlClient)
    - Lisää Microsoft.AspNetCore.Cors Nuget Package Managerissa kommennolla: NuGet\Install-Package Microsoft.AspNetCore.Cors -Version 2.2.0

3.4 Loppukäyttäjän Dokumentaatio

Käyttöohjeet

- Käynnistä sovellus selaimessa painamalla Visual Studiossa IIS Express-nappia, jossa vihreä play-ikoni edessä. Backend käynnistyy osoitteeseen: http://localhost:53569/. Backend lienee hyvä käynnistää ennen Frontendiä.

3.5 Rajapinta

Rajapintaa hallinoi ReseptiController, joka luo http://localhost:53569/api/Resepti/id -osoitteeseen jokaista Frontendistä tulevaa reseptipyyntöä kohtaan yhden JSON-object muotoisen reseptin id-numerolla alkaen numerosta 1 eteenpäin, niin monta kuin pyyntöjä tulee.

3.6 Backendin toiminnallisuudet

1. Reseptien haku tietokannasta (Select-query)
2. Viimeksi käytetyn reseptin päivämäärän tarkistus -> Otetaanko mukaan reseptiarvontaan, vai onko reseptiä käytetty liian hiljattain. (2 päivään reseptiä ei oteta mukaan)
3. Reseptin konversio XML -> JSON
4. Haettujen ja hyväksyttyjen reseptien tallennus tietorakenteeseen (2-ulotteinen array)
5. Yhden reseptin arvonta hyväksytyistä resepteistä, välillä 1-Arrayn pituus (Reseptit) ja tekee siitä JSON-objektin.
6. Last_used_daten päivittäminen tietokantaan (Update-query)
7. Arvotusta Reseptistä tehdään item API-rajapintaan, josta Frontend noutaa reseptin.

Olen kommentoinut koodiin hieman mitä mikäkin kohta tekee, joten ReseptiController.cs-tiedostosta löytyy vielä tarkempaa tietoa ohjelman toiminnasta.

3.7 Tärkeimmät elementit ja CORS

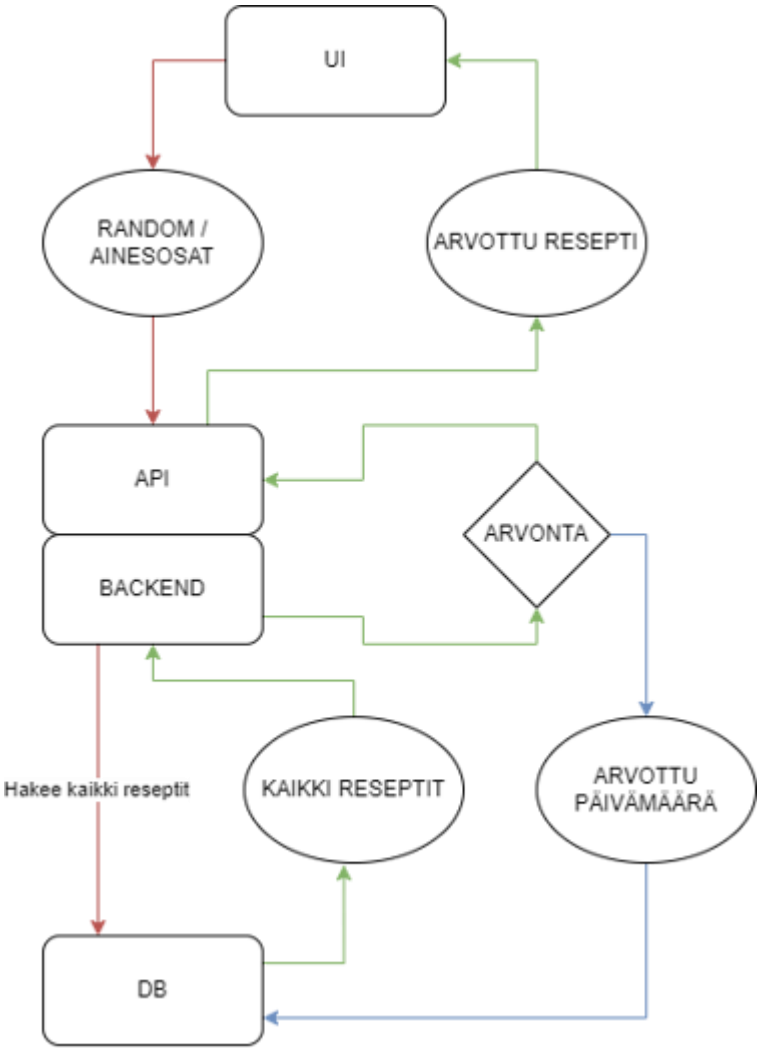
Tärkeimpänä ja oikeastaan ainoana elementtinä toimii backendin puolella ReseptiController, joka sisältää niin rajapinnan hallinnan, kuin myös kaikki backendin toiminnallisuudet. Tämän lisäksi on HomeController, mitä käytin esimerkiksi kehitysvaiheessa tyhjentämään/nollaamaan kaikki tietokantaan last_used_date-kenttiin laitetut arvot, mutta ohjelman toiminnan kannalta se ei ole olennainen. CORSia varten piti ensin lisätä NuGet Package: (Microsoft.AspNetCore.Cors) ja sitten lisätä koodia Startup.cs-tiedostoon CORSin toimimista varten. CORS siis tarkoittaa:

CORS (Cross-origin Resource Sharing): Cross-origin resurssien jakaminen (CORS) on mekanismi, jonka avulla web-sivun rajoitettuja resursseja voidaan käyttää toisesta toimialueesta sen toimialueen ulkopuolelta, josta ensimmäinen resurssi toimitettiin. Ja koska Frontend toimii eri toimialueella, niin CORS piti kytkeä päälle, että tiedon sai kulkemaan API-rajapinnan kautta UI:hin.

3.8 Tärkeät Huomiot Kehittäjille

- Kehittäjän tulee varmistaa, että backend-palvelin käynnistyy ilman virheitä, koska Backend toimii Databasen(Azure) ja Frontendin välillä halutun tiedon muodostajana ja välittäjänä. Jos backend ei toimi oikein tai ei ole käynnissä, niin käyttöliittymä ei saa haluttuja tietoja ja ohjelma ei toimi oikein.
- Backend on varmasti hyvä käynnistää ennen frontendin käynnistämistä.

3.9 Projektikaavio



Projektikaavion on tarkoitus havainnollistaa käyttäjälle ohjelman toiminnan kokonaisuutta, niin että olisi helpompi ymmärtää mitä kaikkea ohjelma tekee, ja tukee siten tekstimuotoista kerrontaa ohjelman toiminnoista.

1. Frontend laittaa postilla pyynnön backendille reseptiä varten rajapintaan.
2. Rajapinnasta pyyntö menee backendille käsittelyyn.
3. Backend hakee kaikki vapaana olevat reseptit tietokannasta eli kaikki reseptit, joita ei ole kahteen päivään arvottu.
4. Backend saa reseptit tietokannasta ja suorittaa niistä arvonnan.
5. Backend merkitsee arvotun reseptin last_used_daten uuden arvon tietokantaan.
6. Backend siirtää arvotun reseptin rajapintaan.
7. Frontend hakee arvotun reseptin rajapinnasta.

4 DATABASE DOKUMENTAAIO (AZURE)

4.1 Tietokannan valinta ja sen tuomat vaatimukset

Tietokanta päädyttiin rakentamaan Azuren SQL Databaseen, joka on Microsoftin tarjoama tietokantaohjelmisto palveluna (DBaas-PaaS). Tämä poisti tarpeen käyttää omaa palvelinta, jonne pääsyn mahdollistaminen kaikille projektin tekijöille olisi ollut hankala toteuttaa. Lisäksi Azure tarjoaa skaalautuvuuden, virheensiedon jatkuvalla varmuuskopioinnilla, automaattisen ylläpidon päivityksillä ja korjauksilla ja vankan tietoturvan.

Microsoft huolehtii siis tietokantaohjelmiston hallinnoimisesta, joten projektissa voitiin keskittyä itse tietokannan suunnitteluun ja rakentamiseen. Lisäksi halusimme projektissa käyttää teknologiaa, joka on laajasti käytössä yritysten IT-infrastruktuurissa.

Azure asettaa omat haasteensa, koska sen tietoturva on niin vankka. Jokaisella projektin jäsenellä tuli olla pääsy Azureen, jotta tietokantaan voi lähettää kyselyjä. Lisäksi back endille piti erikseen taata pääsy tietokantaan avaamalla käyttössäoleville IP-osoitteille pääsy Azuren palomuurin läpi.

4.2 Tietokannan rakentaminen

Tietokanta rakennettiin Microsoftin ohjeilla suoraan Azureen:

https://learn.microsoft.com/en-us/azure/azure-sql/database/single-database-create-quickstart?view=azuresql&tabs=azure-portal

4.2.1 Taulun suunnittelu

Koska sovellus ei vaadi kuin yhden taulun tietokantaan, ei relaatioita tarvinnut tässä vaiheessa ottaa huomioon.

Taulu koostuu yksilöivästä ID-numerosta, last used-tietueesta ja content-tietueesta.

Tietokannan mahdollista skaalautuvuutta ajatellen jokaiselle reseptille luotiin yksilöivä ID-numero, jota voidaan myöhemmin hyödyntää mahdollisen relaatiotietokannan pääavaimena. Lisäksi ID-numeron perusteella suoritetaan reseptien arvonta.

last used-tietue osallistuu reseptin arvonnan ajalliseen rajaamiseen siten, ettei ohjelma pääse arpomaa samaa reseptiä useita kertoja peräkkäin.

content-tietue sisältää ruokareseptin ainesosat ja valmistusohjeen XML-muodossa, jotta parsetus front endissä on mahdollisimman helppo toteuttaa

4.2.2 Taulun luominen

Taulu luotiin suoraan Azuren SQL Databasessa käyttämällä Query editoria. Taulu luotiin koodilla:

```
CREATE TABLE reseptit (
id INT AUTO_INCREMENT PRIMARY KEY,
last_used_date DATE,
content TEXT
);
```

4.3 Tietokannan hallinta

Kaikki muu hallinta tietokantaan toteutettiin käyttämällä Azure Data Studio -sovellusta.

Muiden projektiin osallistuneiden pääsyä tietokantaan hallinnoitiin antamalla heidän käyttämälleen IP-osoitteelle pääsy tietokannan palomuurin läpi:

EXECUTE sp_set_database_firewall_rule N'Example DB Rule','IP-osoite','IP-osoite';

Reseptit vietiin tietokantaan Azure Data Studiossa tämän esimerkin mukaisesti:

```
INSERT INTO dbo.reseptit
VALUES (NULL,'<recipe>
   <title>Pyttipannu</title>
   <description>Perinteinen pyttipannu herkullisilla yrteillä</description>
   <ingredients>
      <ingredient>6 – 7 keitettyä perunaa</ingredient>
      <ingredient>1 – 2 sipulia</ingredient>
      <ingredient>200 – 300 g makkaraa tai tähteeksi jäänyttä lihaa</ingredient>
      <ingredient>öljyä</ingredient>
      <ingredient>suolaa, pippuria</ingredient>
      <ingredient>paprikajauhetta</ingredient>
      <ingredient>1 kananmuna/henk.</ingredient>
      <ingredient>pinnalle persiljaa, tilliä tai ruohosipulia</ingredient>
   </ingredients>
   <timeToCook>45 min</timeToCook>
   <instructions>
      <step>Kuori perunat ja leikkaa kuutioiksi.</step>
      <step>Hienonna sipuli ja kuutioi makkara/liha.</step>
      <step>Paista perunoita pannulla öljyssä hetki.</step>
      <step>Lisää hienonnettu sipuli- ja makkara-/lihakuutiot ja jatka ruskistamista.</step>
      <step>Maista ja mausta suolalla, pippurilla ja paprikajauheella.</step>
      <step>Ripottele pinnalle persiljaa, tilliä tai ruohosipulia.</step>
   </instructions>
</recipe>');
```