CHEF AT HANDS

Naslov projekta: Chef at hands
Člani skupine: Anej Tomplak, Matej Kristan, Jan Napast
Številka projektne skupine: 7
Ime skupine: Chef at hands
Github: https://github.com/ChefAtHands

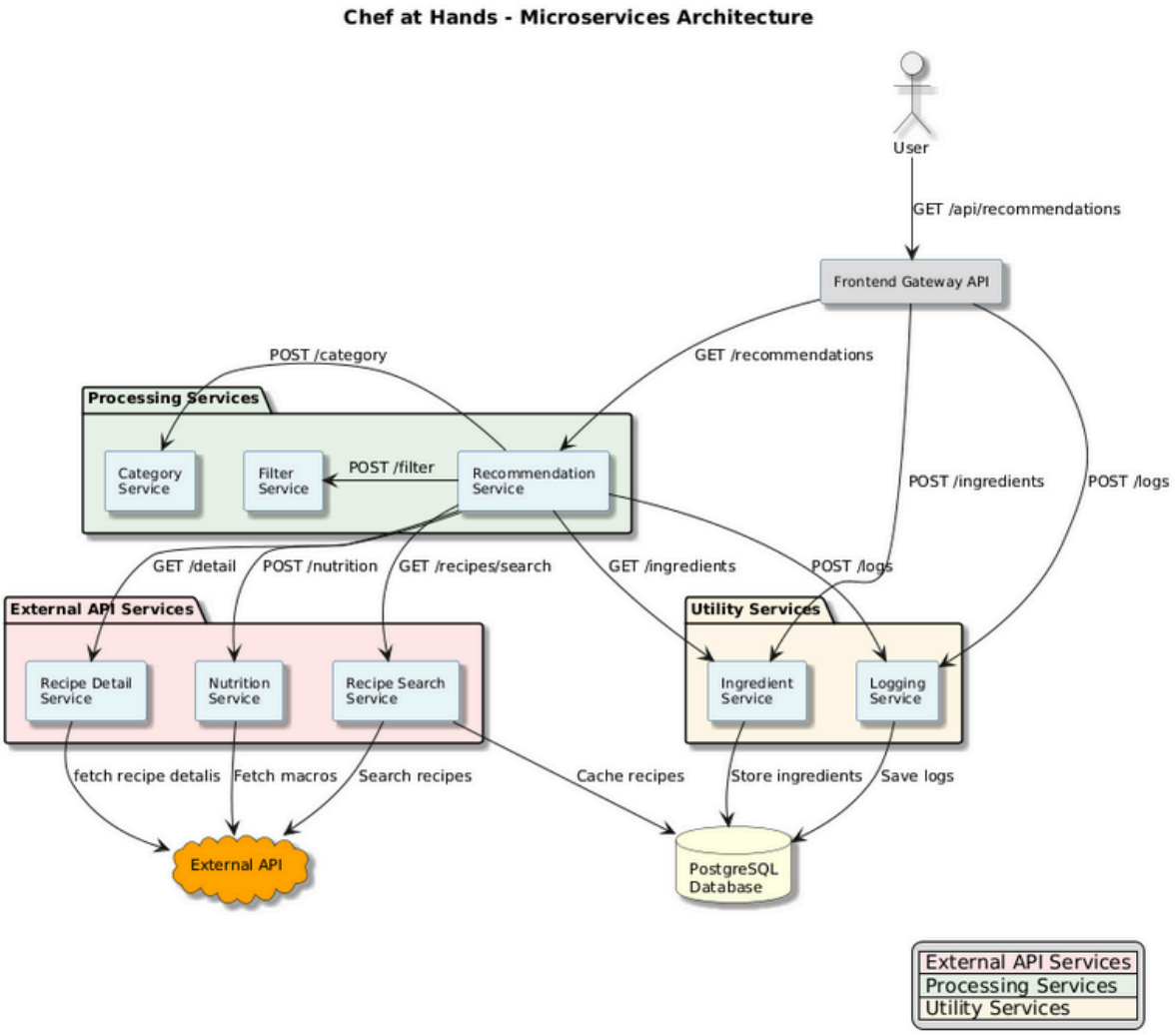
1. Kratek opis projekta

Chef at hands je aplikacija za pomoč pri iskanju pravega obroka za vsakega posameznika. Chef at hands vam predlaga jedi, ki jih lahko pripravite iz sestavin ,ki jih že imate doma. Vsem se je že zgodilo da ima doma ogromno sestavin, vendar ideje za jed pa nikakor ni. Lahko bi iskali po vseh različnih knjigah in spletnih straneh po receptih, ki pa ponavadi zahtevajo še kakšne sestavine, ki jih nimate ali pa ne poznate kako se jed imenuje. Zato poskrbi CAH, saj vam predlaga recepte na podlagi sestavin, ki jih imate doma. Te le vpišete v aplikacijo in CAH vam vrne recepte po vaših željah. Hitro in preprosto do ideje in hrane.

2. Ogrodje in razvojno okolje

Na podlagi zasnove arhitekture aplikacije in preteklih izkušenj smo se odločili, da bomo uporabljali mikorstoritveno arhitekturo Maven, ki je po našem mnenju najbolj primerna za naš projekt. Posamezne mikrostoritve bodo razvite z uporabo Spring boot ogrodja v programskem jeziku Java, saj omogoča enostavno vzpostavitev REST API-jev. Za podatkovno bazo bomo uporabili PostgreSQL, kamor bomo shranjevali sestavine, ki jih uporabnik vnese in recepte, ter morda še kaj, kar je povezano z zunanjim API-jem, kar bomo videli kasneje ob implementaciji. Za razvoj uporabniškega vmesnika bomo uporabili React (ali Angular). V kasnejših fazah projekta načrtujemo uporabo Dockerja za kontejnerizacijo posameznih mikrostoritev in lažje upravljanje z odvisnostmi ter morebitno vključitev Kubernetes za orkestracijo, če se bo to izkazalo kot smiselno za razvojno ali produkcijsko okolje.

3. Shema arhitekture



4. Seznam funkcionalnosti mikrostoritev

Aplikacija *Chef at Hands* temelji na mikrostoritveni arhitekturi, razdeljeni v pet glavnih sklopov:

1. Frontend Gateway API

- Predstavlja vhodno točko za uporabnika.
- Sprejema zahteve (npr. *GET /api/recommendations*) in jih posreduje ustreznim zalednim storitvam.
- Upravlja z usmerjanjem prometa in avtentikacijo uporabnikov.

2. Processing Services (procesne storitve)

Te storitve izvajajo glavno poslovno logiko aplikacije.

- Category Service – razvršča recepte po kategorijah (npr. predjed, glavna jed, sladica, mesno/vegetarijansko, …).

- Filter Service – filtrira rezultate iskanja glede na uporabnikove kriterije (npr. kalorije, beljakovine, maščobe, …).

- Recommendation Service – združuje podatke iz več storitev (kategorije, filtri, sestavine, zunanje API-je) in uporabniku ponudi personalizirane priporočene recepte.

    - Komunicira z Ingredient Service, Recipe Search Service, Nutrition Service in Logging Service.

3. External API Services (zunanje storitve)

Skrbijo za povezovanje z zunanjimi API-ji (npr. Spoonacular).

- Recipe Detail Service – pridobi podrobnosti receptov (slika, postopek, sestavine, …).

- Recipe Search Service – omogoča iskanje receptov po kriterijih.

- Nutrition Service – pridobi hranilne podatke o receptih.
  → Vse storitve uporabljajo *External API* in po potrebi hranijo rezultate v lokalni bazi.

4. Utility Services (pomožne storitve)

- Ingredient Service – upravlja s podatki o sestavinah, shrani jih v bazo (PostgreSQL).

- Logging Service – beleži vse dogodke, napake in zahteve, ter jih shranjuje v bazo.

5. Podatkovna plast

- PostgreSQL Database – centralna baza za shranjevanje podatkov (sestavine, logi, predpomnjeni recepti).

5. Primeri uporabe

1. **Iskanje recepta po sestavinah**

   Uporabnik vnese seznam sestavin, ki jih ima doma.

   Sistem predlaga recepte, ki jih je mogoče pripraviti z vnesenimi sestavinami.

   Uporabnik lahko pregleduje predlagane recepte.

2. **Filtriranje receptov po prehranskih vrednostih**

   Uporabnik želi recepte z določenim številom kalorij ali vsebnostjo proteinov.

   Filter Service filtrira recepte glede na te kriterije.

3. **Iskanje receptov po kategorijah**

   Uporabnik izbere kategorijo (npr. sladica, glavna jed).

   Category Service predlaga recepte iz izbrane kategorije.

4. **Pregled podrobnosti recepta**

   Uporabnik klikne na recept.

   Recipe Detail Service prikaže seznam sestavin, postopek, slike in hranilne vrednosti.

5. **Sledenje aktivnosti**

   Logging Service beleži, katere recepte je uporabnik iskal ali si jih ogledal, za morebitno izboljšanje priporočil.

Primer: Personalizirano priporočilo receptov za uporabnika z omejitvami prehrane

Scenarij:

- Uporabnik v aplikacijo vnese svoje sestavine, prehranske preference (npr. vegetarijansko, nizko vsebnost maščob) in alergije.

- Sistem mora združiti podatke iz več storitev, da pripravi personalizirane priporočene recepte.

Potek:

1. **Frontend Gateway API** prejme zahtevo uporabnika.

2. **Ingredient Service** preveri, katere recepte je mogoče pripraviti z vnesenimi sestavinami.

3. **Recipe Search Service** poišče recepte, ki ustrezajo kriterijem sestavin.

4. **Category Service** doda kategorijo vsakemu receptu.

5. **Filter Service** filtrira recepte glede na prehranske preference in alergije.

6. **Nutrition Service** preveri hranilne vrednosti in potrdi, da recepti ustrezajo kriterijem.

7. **Recommendation Service** združi podatke in izbere najbolj primerne recepte.

8. **Logging Service** beleži vse korake za analizo in izboljšanje priporočil.

9. **Frontend** prikaže personalizirane priporočene recepte uporabniku.

Rezultat:
Uporabnik dobi seznam receptov, ki so:

- pripravljivi z domačimi sestavinami,
- ustrezajo prehranskim omejitvam,
- razvrščeni po kategorijah in hranilnih vrednostih,
- z možnostjo pogleda podrobnosti recepta.