

Prooviülesanne

Luu "infosüsteem", mis võtab kasutajalt sisendi, töötleb seda ning kuvab kasutajale tulemuse.

Töövoog

1. Kasutaja sisestab UI kaudu numbri
2. "Infosüsteem" salvestab numbri väärtuse, staatuse "PENDING" ning tekitab asünkroonse tööülesande sõnumi
3. "Infosüsteem" loeb sõnumit ning täidab tööülesande (korrutab numbri viiega ning märgib staatuseks DONE)
4. Kasutaja näeb tööülesandeid ja nende tulemusi (numbri väärtus ja staatus)

"Infosüsteem"

- Backend moodul (salvestab ja töötleb andmeid, pakub UI jaoks REST API otspunkte)
- Frontend moodul (kasutajaliides)
- RabbitMQ liides (asünkroonsete sõnumite platvorm)

Mitte-funktsionaalsed nõuded

- Andmebaas võib olla töömälus, ei pea rakenduse restarte üle elama
- Kasutaja autentimine ei ole vajalik, tööülesandeid saavad näha kõik sama IP-aadressiga kasutajad (ehk siis andmebaasi numbri juurde salvestada ka päringu IP ning lugemisel selle alusel filtreerida)
- Kõik süsteemi osad peavad olema pakendatud OCI standardile vastava tõmmisena ning olema Docker keskkonnas käivitatavad

Tehnoloogiad

Backend

- Java (21 või uuem)
- Spring Boot (3.5.x või uuem)
- Omal valikul töömälu põhine andmebaas (näiteks H2)
- Gradle

Frontend

- Angular (19 või uuem)
- NPM

Asünkroonsed sõnumid

- RabbitMQ (kasuta ametlikke Docker tömmiseid)

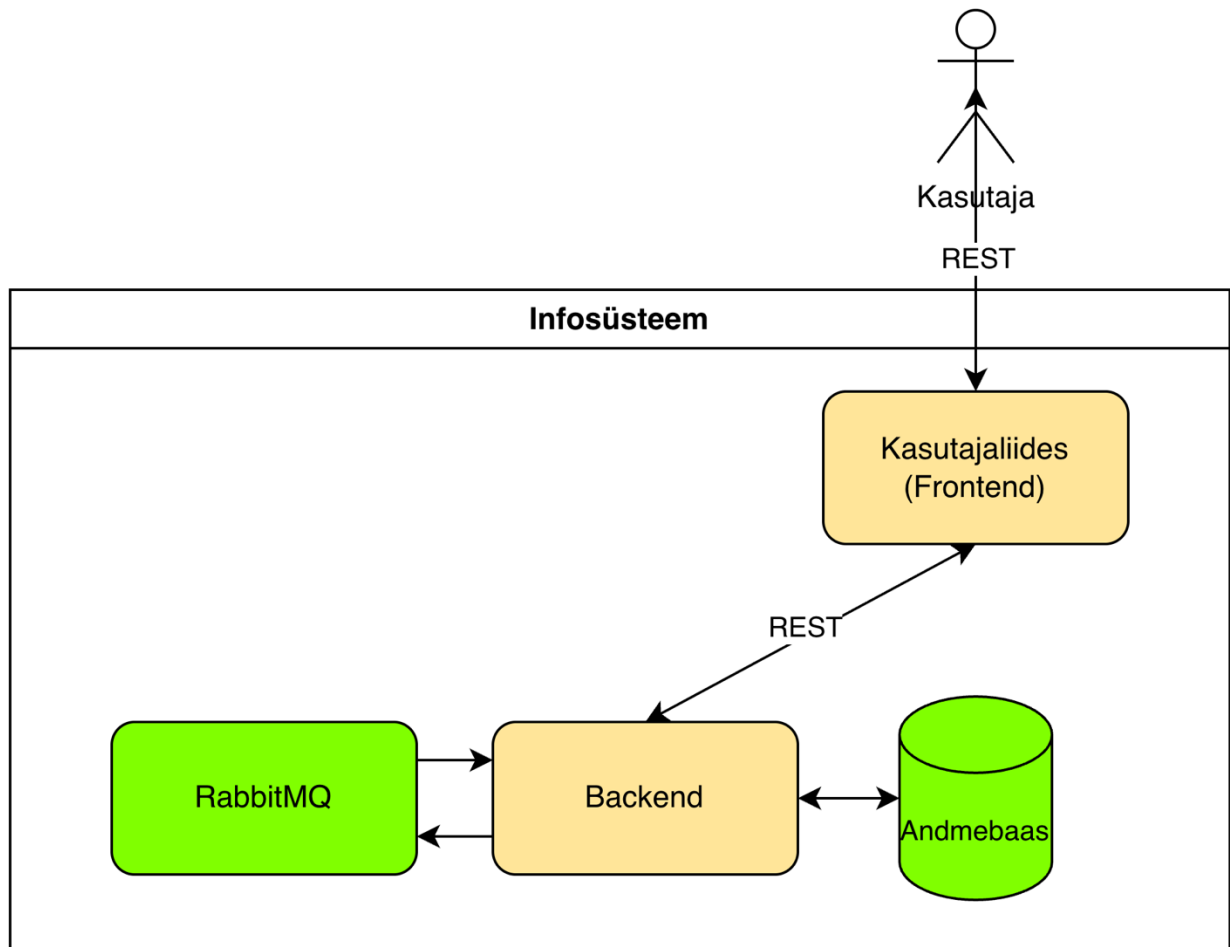
Töö tulemid


- Backend lähtekood + Dockerfile
- Frontend lähtekood + Dockerfile
- Docker compose YAML kogu süsteemi kohalikuks käivitamiseks
- Helm chart Kubernetes klastriks käivitamiseks (vabatahtlik, nende puudumine ei vähenda hinnangut)


Töö puhul ei hinda me kasutajaliidese välimust ega kasutusmugavus. Eeldame vigadeta töötavat rakendust, et hinnata arendaja käekirja.

AI vahendite kasutamine on lubatud. Sel juhul tuleb selgelt dokumenteerida, millised töö osad on AI genereeritud.

Arhitektuuriline skeem



 — Arendaja loodud moodul

 — Olemasolev teek/tõmmis