# Illegaal2 Arhitektuuridokument

Versioon 1.0

## Dokumendi ajalugu

| Versioon | Kuupäev | Autor | Märkused |
|---|---|---|---|
| 0.1 | 11.09.2015 | Meril Tangsoo | Esialgne mustand. |
| 1.0 | 23.09.2025 | Karme Bärg | Arhitektuur ning arendusjuhend |
| 1.1 | | | |

## Sissejuhatus

Käesolev dokument kajastab kõrgel tasemel Illegaal2 infosüsteemi ja selle liideste ülesehitust ja arhitektuuri. Illegaal2 visioon koosneb mitmest suurest domeenist: menetlused, kinnipidamine, Entry-Exit/ETIAS, arhiiv, haldus ja sissesõidukeelu töölaud. Illegaal2 infosüsteem on läbi erinevate domeenide kasutatav terviklikult või osadena.

## Süsteemi taust

Illegaal2 infosüsteem jõudis kasutusse 1.09.2020 eesmärgiga, et sellest hetkest kõik tagasisaatmise menetlused alustatakse uues süsteemis. Vana Illegaal süsteemi andmeid üle ei kantud.

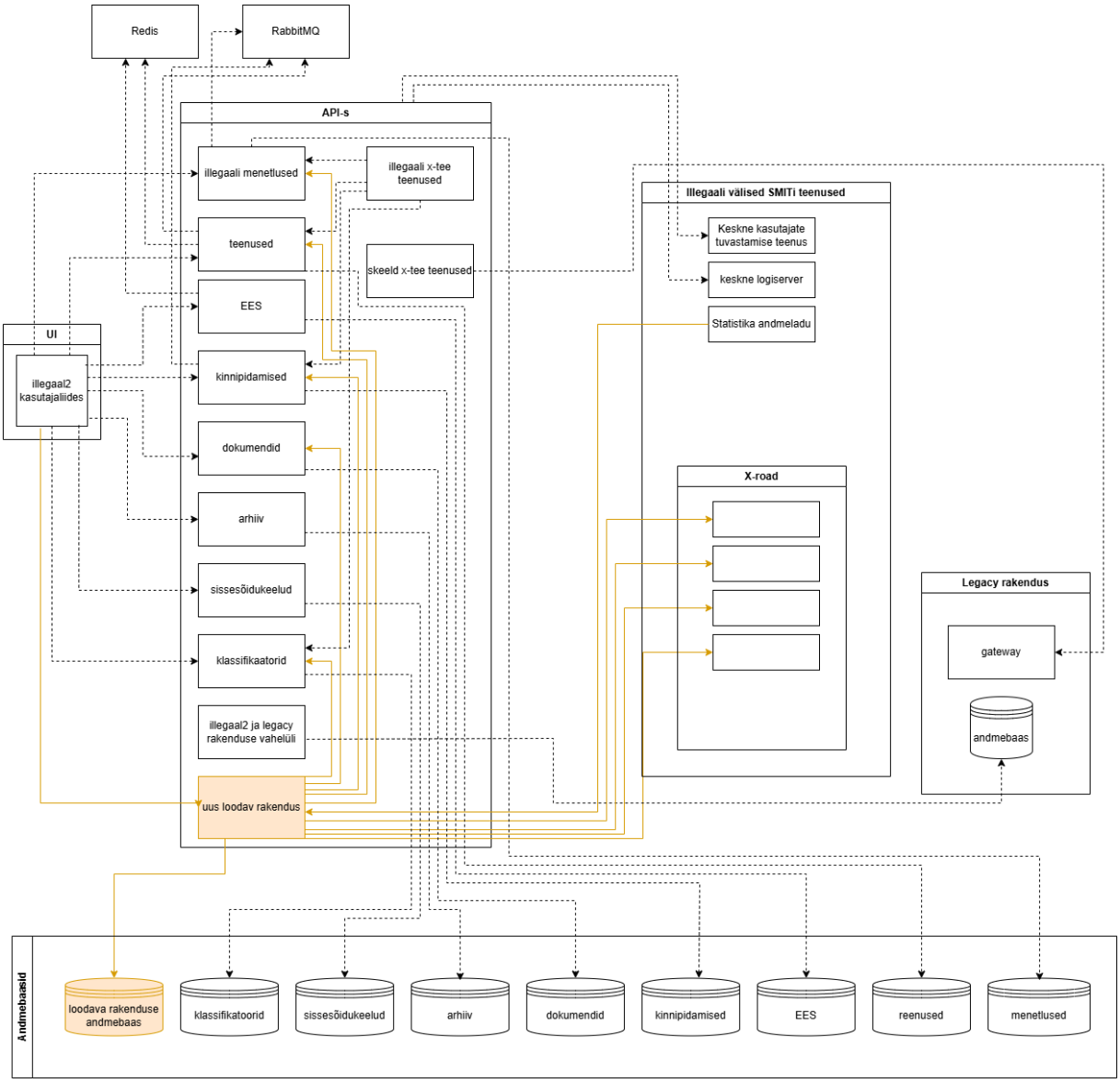
Illegaal2 loomise eesmärk oli kaasajastada ja lihtsustada dokumentide loomist, tööprotsessist tulenevaid funktsionaalsusi ja klassifikaatorite haldust. Infosüsteemi vajadused, funktsionaalsused ja skoop on olnud pidevas muutuses projekti käigus.

Illegaal2 arendus on olnud Siseministeeriumi infotehnoloogia- ja arenduskeskuse vastutada.

## Illegaal2 arhitektuur

Illegaal2 andmekogu koosneb eraldiseisvatest moodulitest, mis on paigaldatavad ja arendatavad üksteisest sõltumatult. Igal moodulil on eraldi andmebaas. Kasutajaliides on realiseeritud ühe eraldiseiseva TypeScript rakendusena.

### Komponentdiagramm




## Kasutatav tarkvara

- Java 17
- Angular 13 (Typescript)
- Spring Boot 2.5.15 (enamus olemasolevad rakendused)
- Spring Boot 2.7.18 (mõned olemasolevad rakendused)
- Junit 4 (pooled rakendused)
- Junit 5 (pooled rakendused)
- Flyway 7.7.3 (8.5.13 API-des mis kasutavad Spring 2.7)
- PostgreSQL 15

## Moodulite vaheline suhtlus

Illegaali omavaheliste moodulite suhtlus toimub üle:

- REST
- RabbitMQ

Kasutajaliides suhtleb backendi moodulitega samuti üle REST-i.

## Suhtlus Illegaali väliste süsteemidega

Illegaali väliste andmekogude ja süsteemidega suheldakse kasutades:

- SOAP üle XTEE
- REST üle XTEE
- REST

## Suhtlus andmebaasiga

Andmebaasiga suhtlemiseks kasutatakse Spring Data JPA raamistikku koos Hibernate'iga. Andmebaasi skeemi versioonihaldus toimub Flyway migratsioonidega.

## Logimine

Rakenduses kasutatakse SLF4j API-t ning eelkonfigureeritud logback implementatsiooni. Loggeri lisamiseks klassile kasutatakse @Slf4j annotatsiooni projektist Lombok.

## Testimine

Rakenduse testimiseks kasutatakse nii integratsiooniteste kui ka ühikteste JUnit ja Mockito abil.

## Koodi kvaliteet

Koodi kvaliteedi formaalseks hindamiseks kasutatakse Sonar-it:

- Sonar-i hoiatused
- Automaattestidega kaetuse protsent (>=80%)

Mitteformaalne kvaliteet:

- Pull-requestide code review'd.
- Koodi loetavus, lihtsus ja mõistetavus.
- Andmete töötluse kiirus ja optimaalsus.

# Lisanõuded uutele moodulitele

Uute moodulite arendamisel tuleb eeskujuks võtta SMIT-i nõutele ning parimatele tavadele vastav Springboot-Java näidisrakendus ning järgida SMIT-i keskseid arendusnõudeid mis on täpsemalt kirjeldatud SMIT-i IDP-s (Internal Developer Portal).

## Tarkvara

- Programmeerimiskeel: Java 21 +
- Raamistik: Spring Boot 3.4.5 +
- Andmebaas: PostgreSQL 15 +
- Andmebaasiga suhtlus: jOOQ 3.19.x +
- Tarkvarapakettide haldamine: Gradle 8.5 + (Kotlin DSL)
- Test raamistik: JUnit 5
- API disain: OpenAPI 3.0 +
- Andmebaasi muudatuste haldus: Flyway 10.18.1 +

## Projekti  struktuur

```
projctName-api /
├── database/
│   └── migration/           # Flyway migration files
├── src/
│   ├── main/
│   │   ├── java/              # Main source code
│   │   │   └── ee/smit/projectName/
│   │   │       ├── config/    # Configuration classes
│   │   │       ├── exception/ # Exception handling
│   │   │       ├── controller/# HTTP layer
│   │   │       ├── service/   # Business logic
│   │   │       └── util/      # Utilities
│   │   └── resources/         # Application resources
│   │       ├── swagger/       # OpenAPI specification (openapi.yml)
│   │       ├── application.yml # Main configuration (+ env specific yml-s)
│   │       └── logback-spring.xml # Logging configuration
│   ├── generated/             # Generated code
│   │   └── java/
│   │       ├── jooq/         # jOOQ generated database code
│   │       └── openapi/      # OpenAPI generated client code
│   └── test/                  # Test source code
│       ├── java/              # Test classes with same package structure
│       └── resources/         # Test configuration (application.yml)
└── build.gradle.kts           # Gradle build configuration with Kotlin DSL
```

## API-keskne arendus

API-first arendus OpenAPI 3.0 spetsifikatsiooni ja koodi genereerimisega.

## Andmebaasiga suhtlus

Tüübikindel andmebaasiga suhtlemine jOOQ raamistiku abil JPA asemel.