Specifikáció

# Követelmények

## Funkcionális követelmények (use-case-ek)

### OAuth2 alapú autentikáció

|  |  |  |
| --- | --- | --- |
| Azonosító | Összefoglaló | Leírás |
| F1.01 |  |  |
| F1.02 |  |  |
| F1.03 |  |  |
| F1.04 |  |  |
| F1.05 |  |  |
| F1.06 |  |  |
| F1.07 |  |  |

### Funkciócsoport1

|  |  |  |
| --- | --- | --- |
| Azonosító | Összefoglaló | Leírás |
| F2.01 |  |  |
| F2.02 |  |  |
| F2.03 |  |  |
| F2.04 |  |  |
| F2.05 |  |  |
| F2.06 |  |  |
| F2.07 |  |  |

### Funkciócsoport2

|  |  |  |
| --- | --- | --- |
| Azonosító | Összefoglaló | Leírás |
| F3.01 |  |  |
| F3.02 |  |  |
| F3.03 |  |  |
| F3.04 |  |  |
| F3.05 |  |  |
| F3.06 |  |  |
| F3.07 |  |  |

## Műszaki követelmények

# Részletes terv

## Vizuális tervek (elérhető oldalak)

## Adatmodell

## Interfészek (REST API)

## OAuth2 – ezt szívesen megcsinálom (Anna)

Az elkészült szoftver OAuth2 autentikációt és autorizációt biztosít, mely lehetővé teszi, külső alkalmazások számára a felhasználók azonosítását, és jogosultságaik elérését. Az OAuth 2.0 specifikációban ismertetett szerepeknek megfelelően dedikált autorizációs szerver (a továbbiakban Authorization Server) lesz megvalósítva, illetve egy erőforrás szerver (a továbbiakban Resource Server) és egy kliens alkalmazás. Az első kettő Spring Security 6 keretrendszer felhasználásával készül el, míg a kliens (…?)

### Authorization Server

A komponens egy külön backendként funkcionál, melynek feladata a felhasználók azonosítása és jogosultságaik kezelése, valamint a kliens alkalmazások számára autorizáció biztosítása. Ez az alkalmazás a terhelés elosztása miatt különválik a projekt során elkészített többi komponenstől, önállóan futtatható.

Az alkalmazáshoz külön adatbázis kapcsolódik, melyben a felhasználók azonosításhoz szükséges adatai lesznek eltárolva (név, id, secret, jogosultságok). Emellett a regisztrált kliensalkalmazások azonosítására alkalmas adatai is innen elérhetők. A jelszóként funkcionáló titkos adatok az adatbázisba hashelt formában íródnak. Az adatbázis relációs sémára épül.

#### Regisztráció

A szerver regisztrációs felületet biztosít új felhasználók számára. A felhasználói adatok megadását követően új entitás kerül az adatbázisba. A felhasználó jogosultságait később az erre jogosult felhasználók állíthatják be.

#### Bejelentkezés

A felhasználót egy általa használt egy kliensalkalmazás átirányítja a bejelentkezési felületre, ahol a felhasználó az egyedi felhasználónevét és jelszavát megadva azonosítja magát. Sikeres bejelentkezést követően a szerver a kliens által megjelölt url-re irányít és elküldi a hozzáférési tokent.

A token egy jwt token, melybe a felhasználó neve, azonosítója és jogosultságai vannak kódolva. A kliens ezt a tokent használhatja a Resource Serverhez intézett kéréseknél azonosításra.

#### Kliensek azonosítása

Az Authorization Servert csak előre meghatározott kliensalkalmazások érhetik el, ezeket előre regisztrálni kell. A regisztrált kliensek azonosítót és titkos adatot kapnak, mellyel azonosíthatják magukat. Emellett a klienshez tartozó átirányítási url-ek is előre definiáltak.

### Resource Server

Ez egy REST API szerver, mely bizonyos privát endpoint-jait védi. Ezekhez csak azonosított és megfelelő jogosultsággal rendelkező felhasználók férhetnek hozzá.

Az alkalmazás a kérések fejlécében jwt-tokent vár, melyet az Authentication Server segítségével validál. A Resource Server nem kezeli a felhasználók jogosultságait, csupán felhasználja az Authentication Server-től kapott információt. Érvényes token esetén kiszolgálja a kérést.

### Client

A kliens alkalmazás az Authentication Server-t felhasználva igényel hozzáférési tokent a felhasználó számára. Ehhez egy kliens ID-val, titkos adattal és átirányítási címmel (ide várja a válasz tokent) felparaméterezett kérést intéz az Authentication Server felé. A válaszban kapott jwt-tokent felhasználja a Resource Server felé intézett kéréseknél.

## Technológiák, keretrendszerek