

Adatbázis-alapú entity-resolution szoftvermodul

Fejlesztői dokumentáció

Tartalomjegyzék

Tartalomjegyzék.....	2
1. Bevezető.....	4
1.1. A dokumentum áttekintése.....	4
1.2. Célközönség.....	4
1.3. A dokumentum felépítése.....	4
2. Szótár.....	4
3. Követelménynek való megfelelés.....	5
3.1. Felhasználók kezelés.....	5
3.2. Jogosultság kezelés.....	5
3.3. REST-API.....	5
3.4. Szervezet információk kezelése.....	5
3.5. Szervezet feloldás weboldalak.....	5
3.6. Szervezet feloldás.....	5
4. Architektúráis áttekintő.....	6
4.1. Adatlap.....	6
4.2. Irányelvek.....	6
4.3. Fontosabb osztályok.....	7
Controllerek.....	7
RolesAPI.....	8
Servicek.....	8
4.4. Resolverek.....	10
5. Adatszerkezet.....	11
5.4. Tranzies adatok.....	11
5.5. Perzisztens adatok.....	11
6. Külső interface-ek.....	13
6.4. Rest API inteface.....	13
6.5. Document interface.....	13
7. Üzleti logikát kijánló endpoint-ok.....	15
7.4. AuthenticationController.....	15
7.5. OrganizationAPI.....	18
7.6. ManagementApi.....	21
7.7. HealthAPI.....	22
7.8. LoginApi.....	23

1. Bevezető

1.1. A dokumentum áttekintése

Jelen dokumentum célja az Adatbázis-alapú entity-resolution szoftvermodul alkalmazás bemutatása fejlesztői szemmel.

1.2. Célközönség

Azok a fejlesztők, akik az alkalmazás karbantartásában, fejlesztésében részt vettek, részt fognak venni.

1.3. A dokumentum felépítése

A dokumentum első részében egy áttekintő összefoglalja a projekt célkitűzését. A következő fejezetben egy szótárban összeszedve következik az alkalmazás-specifikus kifejezések gyűjteménye. Ezt követi az alkalmazás követelményeinek felsorolása. A dokumentáció további részeiben az architektúrális felépítésének részleteibe bele menő leírásokkal zárul. Itt kerül bemutatásra az alkalmazott hardveres és szoftveres eszközök, az osztályszintű felépítés, az adatok tárolásának megoldásai és a külső interface-ek.

2. Szótár

A dokumentumban a következő fogalmak kerülnek használatra; a könnyebb áttekinthetőség és érthetőség érdekében az alábbiakban kerülnek bemutatásra, megfogalmazásra:

Elnevezés	Magyarázat
UUID	Egyedi azonosító, ami minden szervezet számára egyedi
REST	A REST (Representational State Transfer) egy szoftver-architektúra típus, elosztott kapcsolat, nagy, internet alapú rendszerek számára, amilyen például a világháló.
API	Az alkalmazásprogramozási felület vagy alkalmazásprogramozási interfész.

3. Követelménynek való megfelelés

3.1. Felhasználók kezelése

Rendszernek biztosítani kell a rendszerbe történő regisztráció lehetőségét. Ehhez a felhasználónak az adatai megadásával kell kitöltenie és a megfelelő REST-API endpointra elküldeni.

3.2. Jogosultság kezelése

A felhasználók jogosultságaik alapján érhetnek el bizonyos oldalakat és szolgáltatásokat. A jogosultságok megszerzésének feltételei általában a felhasználó adatainak és dokumentumainak feltöltése a rendszerben.

3.3. REST-API

Az alkalmazás a REST-es szabványoknak megfelelő API-t implementál – ennek részleteiért, az elérhető endpoint-okért, meghívásuk paraméterezéséért stb. lásd a dokumentum későbbi, releváns fejezetét.

3.4. Szervezet információk kezelése

A rendszerben lévő szervezet adatokat lehet létrehozni, szerkeszteni, törölni és összevonni.

3.5. Szervezet feloldás weboldalak

A szervezetek feloldása során használt cég információs oldalak számára saját feloldó szolgáltatás van.

3.6. Szervezet feloldás

A szervezet feloldások estén EU adószám feloldást is használ cégek összevonása számára.

Az adatokat tisztítás és normalizálás után kerülnek tárolásra.

4. Architektúrális áttekintő

4.1. Adatlap

Aspektus	Választott megoldás(ok)
Szoftveres környezet	Java SE 8, Spring MVC 1.5.2
Architektúra	Kliens / szerver
Felhasználói felület	nincs
Perzisztencia réteg	A rendszer futásához szükséges entitások, potenciális duplikátumok tárolására szolgáló adatbázis; az implementáció adatbázis-agnosztikus, kizárólagos követelmény, hogy támogassa a megszokott indexelés, keresési, standard ANSI SQL-es szabványokat
Alkalmazásszerver	Tomcat 8
Interface-ek, API	Standard Java 8 interface-ek, illetve REST (CRUD) API
Nyelvek	Java

4.2. Irányelvek

Az alkalmazás megvalósítása során az alábbi, legfontosabb irányelvek kerültek figyelembevételre:

- Data Layer -> Controller Layer -> Service Layer kialakítása
- Package by layer kialakítás

- Webes REST alkalmazás. Webserver által kiszolgált, REST API-n keresztül használható alkalmazás

4.3. Fontosabb osztályok

Controllerek

API biztosítása a kérések kiszolgálásához. Jellemzően entitásonként lettek létrehozva az osztályok az alap CRUD műveletekhez és ezt kiegészítő üzleti logikai igényeihez. Az endpointok jogosultság kezeltek, így ezek elérése csak a megfelelő jogkörrel rendelkező felhasználók részére elérhetőek. Controllerek feladata a beérkező input adatok validálása és a megfelelő service meghívása.

Az endpointok REST-en keresztül hívhatóak. Ezt a `@RestController` annotáció biztosítja. Az endpointok elérése a `@RequestMapping` annotáción keresztül lett megadva. Minden endpoint esetében meg lett adva, hogy az milyen `RequestMethod`-ra figyel, milyen bemeneti paraméter(ek)e)t vár és esetlegesen milyen válasszal tér vissza a hívó fél felé. Jellemzően kérésnél az endpoint DTO osztályt vár paraméterül, míg válasznál valamilyen entitás mezőinek részhalmazával (projectionnel) tér vissza. Több kérés is lapozható formában kéri el az adatokat, amit a `Pageable` interface biztosít számunkra. Ennek megadásával a `JpaRepository` képes visszaadni a megadott oldalszámmal, darabszámmal rendezéssel a lekérdezett adathalmazt.

Spring által biztosított validációt használtuk fel a kérések input adatainak ellenőrzéséhez. Validálni a `RequestBody`-ban érkező adatokat (korábban említett DTO-kat) szoktuk, amennyiben a `@Validated` annotációval jelezve lett rá az igény. Minden a rendszerben lévő és a kommunikáció során felhasznált DTO-hoz készült egy a neki megfelelő validáló osztály (`Validator` postfixel ellátva), mely csak az adott típust képes vizsgálni. A kérés beérkezésénél, ha az input adat el lett látva a validálását jelző annotációval, akkor az input adat típusa szerint lefut egy keresés, mely kiválasztja az adat típusának ellenőrzésére képes validátort. Kiválasztás után fut le a megfelelő validációs folyamat és dönti el, hogy tovább engedi a kérést az alkalmazás rétegnek, vagy elutasítja azt, amennyiben nem valid adatokat tartalmazott a kérés. Ezáltal a validálás biztosítja, hogy az alkalmazás logikai rétege már csak a szabályoknak megfelelő adatokkal dolgozzon, növelve a biztonságot és csökkentve az esetleges hibás működés esélyét.

RolesAPI

A /roles Rest API endpointokat kezeli.

UserAPI

A /users Rest API endpointokat kezeli.

LoginApi

A /login Rest API endpointokat kezeli.

OrganizationAPI

A /organizations Rest API endpointokat kezeli.

ManagementApi

A /management Rest API endpointokat kezeli.

HealthAPI

A /health Rest API endpointokat kezeli.

Servicek

Az üzleti logikai részeket implementáló osztályok összessége. Feladatuk az endpointokon keresztül érkező kérések kiszolgálása. Az osztályok @Service annotációval ellátottak, így azok beinjektálhatóak a Controllerekbe, s onnan hívhatóak. Továbbá az osztályok el lettek látva @Transactional annotációval az adatbázis lekérdezések tranzakció menedzsmentjének elvégzéséhez. Ennek megadásával a Spring kezeli a lekérdezések tranzakcióban történő lefuttatását.

CRUD service

Implementálja az általános CRUD műveleteket, s a rendszerben lévő entitások ebből származnak.

PatchConverter

A PatchConverter segíti, hogy a HTTP PATCH művelet megfelelően működjön.

ApplicationConfiguration service

Az e segíti a profilba beolvasható beállításokat, bővebb információ a telepítési útmutatóban található.

Callback service

A szervezet egyedi azonosító (UUID) és a szervezet EU adószám feloldásának POST kérés vissza hívását kezeli.

OrganizationUuidResolver service

A szervezet egyedi azonosító feloldását kezeli.

Health service

A kapcsolódó document szolgáltatás és cég információ feloldók állapotát monitorozza.

Management service

Cég információ feloldókat aktivál és deaktiválását kezeli. A feloldó státusza is lekérdezhető vele.

Organization service

A szervezeteket kezeli.

OrganizationResolutionRequest service

Szervezet feloldás kéréseket kezeli.

Login service

A felhasználói autentikációt kezeli.

MailService

Jelszó emlékeztető email küldést kezeli.

SecurityService

Az aktuális felhasználó esetén vizsgálja be van-e jelentkezve és mi a felhasználó azonosítója.

UserService

Felhasználó kezelést kezeli.

EntityResolutionScheduler

Az entity resolution ütemezést kezeli.

HtmlUnitDriverWebDriverFactory

A HTML webdrivert kezeli.

4.4. Resolverek

A szervezetek feloldását a resolverek segítik.

OrganizationEuTaxNumberResolver

A szervezet EU adószám feloldását kezeli.

OrganizationUuidResolver

A szervezet egyedi azonosítóját oldja fel.

AllamkincstarEntityResolver

A oldalt használja, hogy a szervezet feloldását segítse.

CeginfoEntityResolver

A oldalt használja, hogy a szervezet feloldását segítse.

DummyEntityResolver

Tesztelést segítő feloldó. Valódi feloldást nem végez.

NavEntityResolver

A oldalt használja, hogy a szervezet feloldását segítse.

5. Adatszerkezet

5.4. Tranzies adatok

A rendszer adatfolyamában megjelenő adattároló osztályok, melyek jellemzően a tartós adatok mezőinek egy részhalmazát tárolják. Lekérdezések bemeneteként általában DTO osztályokat használ az alkalmazás, míg kimeneti adatoknál projectionöket.

DTO osztályok a `dto` csomagokban találhatóak. Mező neveken kívül getter/setter metódusokat tartalmaznak csak. Egymásra hivatkozás esetén csak a hivatkozott entitás azonosítóját tartalmazza és nem a teljes perzisztens osztályt. Ezzel elkerülve az egymásra hivatkozás végtelen ciklusát.

A DTO osztályok oda-vissza konvertálásához Mapper osztályokat használ az alkalmazás. Ezek a `hu.organization.model.dto.mapper` csomagban találhatóak. Ezek segítségével képes perzisztens adatból tranziens adatot vagy tranziens adatból perszisztens adatot létrehozni.

Kimenő adatok leképezéséhez projection-eket használ a rendszer. Ezek segítségével egy interface-n keresztül megadható, hogy az adott projection milyen mezőit adja vissza a megadott perzisztens forrás osztálynak. Ez nézetenként más és más projectionöket igényel, ennek megfelelően egy Projekt entitáshoz akár több ilyen projection is meg lett adva. A konvertáláshoz a `ProjectionService`-t kell injektálni a megfelelő service-be, aminek átadva a kívánt projection-t és forrás osztályt, elő fogja állítani a megadott mezőket tartalmazó adat osztályt.

5.5. Perzisztens adatok

A rendszerrel szemben támasztott üzleti igényeknek megfelelően szükséges az adatok perszisztens tárolása. Ehhez ORM használva entitásonként létre lettek hozva a megfelelő osztályok. Az adatbázis generálása ezekből az osztályokból generálódik. Profile fájlban megadott beállítás alapján jön létre vagy frissül az adatbázis szerkezete, tartalma.

A tárolásért felelős osztályok az alábbiak:

- `/dto package`

- BaseEntityDTO: közös adatokat tartalmazza
- OrganizationVO: szervezettel kapcsolatos adatokat tartalmazza
- /user/dto
 - UserDTO: Felhasználói adatokat tartalmaz.
 - UserNameDTO: a felhasználó azonosítóját és nevét tartalmazza.

6. Külső interface-ek

6.4. Rest API interface

A követelményekben megfogalmazottak alapján az alkalmazás REST endpoint-okon keresztül nyújt kommunikációs lehetőséget a külvilággal.

Az endpoint-ok (ugyancsak a követelményeknek megfelelően) védettek autentikációs és authorizációs szempontból is.

Minden, releváns endpoint közösen az api névtér alatt található.

Az endpointok számára TOKEN autentikáció van, ha nincs belépve felhasználó, akihez tartozik token vagy nincs joga erőforrást elérni, akkor nem fér hozzá erőforráshoz.

A document interface által hívható endpointok esetén ApiKey alapú autentikáció van.

6.5. Document interface

A feldolgozandó szervezeteket külső forrásból szerzi be a program, ezt a document modul végzi. A document modult nem tartalmazza a program.

A szoftver hívja a document modult és document modul is hívja a programot.

A document interface kérések esetén ApiKey van használva kérések autentikációjához, ha nincs megfelelő ApiKey kérésekbe, akkor elutasítja a kéréseket.

Kettő interface kell implementálni:

endpoint: procurements

Leírás: felhasználók filterezése

Metódus: POST

URL : /procurements/{id}/{UUID}/organizationUUID

URL paraméterek: String uuid, String id

Adat paraméterek: -

Visszatérési érték: -

Leírás: feloldott szervezet adószámának letárolása

Metódus: POST

URL : /procurements/{id}/{UUID}/ organizationEuTaxNumber

URL paraméterek: String uuid, String id

Adat paraméterek: -

Visszatérési érték: -

7. Üzleti logikát kiajánló endpoint-ok

7.4. AuthenticationController

RolesAPI

endpoint: roles

Név: query

Leírás: jogosultságok lekérdezése

Metódus: GET

URL : /roles

URL paraméterek: -

Adat paraméterek: -

Visszatérési érték: List<Roles>

UserAPI

endpoint: users

Név: query

Leírás: felhasználók filterezése

Metódus: GET

URL : /users

URL paraméterek: @RequestParam Map<String, String> queryParams

Adat paraméterek: -

Visszatérési érték: List<UserDTO>

Név: queryNames
Leírás: felhasználók nevének lekérdezése
Metódus: GET
URL : /users/names
URL paraméterek: -
Adat paraméterek: -
Visszatérési érték: List<UserNameDTO>

Név: count
Leírás: felhasználók nevének lekérdezése
Metódus: GET
URL : /users/count
URL paraméterek: @RequestParam Map<String, String> queryParams
Adat paraméterek: -
Visszatérési érték: ItemCount

Név: get
Leírás: felhasználó lekérdezése azonosító alapján
Metódus: GET
URL : /users/{id}
URL paraméterek: @PathVariable("id") Long id
Adat paraméterek: -
Visszatérési érték: UserDTO

Név: create

Leírás: felhasználó létrehozása

Metódus: POST

URL : /users/{id}

URL paraméterek: -

Adat paraméterek: @RequestBody UserDTO user

Visszatérési érték: long

Név: replace

Leírás: felhasználó módosítása

Metódus: PUT

URL : /users/{id}

URL paraméterek: @PathVariable("id") Long id

Adat paraméterek: @RequestBody UserDTO user

Visszatérési érték: -

Név: remove

Leírás: felhasználó törlése

Metódus: DELETE

URL : /users/{id}

URL paraméterek: @PathVariable("id") Long id

Adat paraméterek: -

Visszatérési érték: -

7.5. OrganizationAPI

endpoint: organizations

Név: query

Leírás: szervezet filterezés, uuid alapján nem használható. Eon api használja.

Metódus: GET

URL : /organizations

URL paraméterek: @RequestParam Map<String, String> queryParams

Adat paraméterek: -

Visszatérési érték: List<OrganizationVO>

Név: query

Leírás: szervezet filterezés uuid alapján. Eon api használja.

Metódus: GET

URL : /organizations/uuid/{uuid}

URL paraméterek: @PathVariable String uuid

Adat paraméterek: -

Visszatérési érték: OrganizationVO

Név: resolve

Leírás: szervezet feloldása. Document api használja.

Metódus: POST

URL : /organizations/resolve

URL paraméterek: -

Adat paraméterek: @RequestBody OrganizationResolveRequest request

Visszatérési érték: OrganizationVO

Név: merge

Leírás: szervezetek összemergelése.

Metódus: GET

URL : /organizations/merge

URL paraméterek: @RequestParam Map<String, String> queryParams

Adat paraméterek: -

Visszatérési érték: -

Név: filterByName

Leírás: szervezet filterezése név/eu adószám alapján

Metódus: GET

URL : /organizations/filterByName

URL paraméterek: @RequestParam Map<String, String> queryParams

Adat paraméterek: -

Visszatérési érték: List<Organization>

Név: count

Leírás: szervezet filterezés név/eu adószám alapján eredményének darabszáma

Metódus: GET

URL : /organizations/count

URL paraméterek: @RequestParam Map<String, String> queryParams

Adat paraméterek: -

Visszatérési érték: itemCount

Név: get

Leírás: szervezet lekérdezése azonosító alapján

Metódus: GET

URL : /organizations/{id}

URL paraméterek: @PathVariable("id") Long id

Adat paraméterek: -

Visszatérési érték: Organization

Név: create

Leírás: szervezet létrehozása

Metódus: POST

URL : /organizations

URL paraméterek: -

Adat paraméterek: @RequestBody OrganizationVO source

Visszatérési érték: long

Név: createOrReplace

Leírás: szervezet létrehozása/módosítása

Metódus: PUT

URL : /organizations/{id}

URL paraméterek: -

Adat paraméterek: @PathVariable("id") Long id, @RequestBody Organization source

Visszatérési érték: -

Név: remove

Leírás: szervezet törlése

Metódus: POST

URL : /organizations/{id}

URL paraméterek: -

Adat paraméterek: @PathVariable("id") Long id

Visszatérési érték: -

7.6. ManagementApi

endpoint: management

Név: getStatus

Leírás: resolver státuszának lekérdezése

Metódus: GET

URL : /organizations

URL paraméterek: -

Adat paraméterek: -

Visszatérési érték: ResolversStatus

Név: enableResolvers

Leírás: resolverek engedélyezése

Metódus: POST

URL : /organizations/enable-resolvers

URL paraméterek: -

Adat paraméterek: -

Visszatérési érték: ResolversStatus

Név: disableResolvers

Leírás: resolverek letiltása

Metódus: POST

URL : /organizations/disable-resolvers

URL paraméterek: -

Adat paraméterek: -

Visszatérési érték: ResolversStatus

7.7. HealthAPI

endpoint: /health

Név: getResolversHealth

Leírás: resolver állapotának lekérdezése

Metódus: GET

URL : /organizations/resolvers

URL paraméterek: -

Adat paraméterek: -

Visszatérési érték: ResolversStatus

Név: isDocumentAccessible

Leírás: document modult elérhetőségének lekérdezése

Metódus: GET

URL : /organizations/document

URL paraméterek: -

Adat paraméterek: -

Visszatérési érték: ResolversStatus

7.8. LoginApi

endpoint: {root}

Név: processLogin

Leírás: bejelentkezés

Metódus: POST

URL : /organizations/login

URL paraméterek: -

Adat paraméterek: @RequestBody LoginRequest authentication

Visszatérési érték: ResolversStatus

Név: generateNewPassword

Leírás: új jelszó kérelem

Metódus: POST

URL : /organizations/new-password

URL paraméterek: -

Adat paraméterek: @RequestBody @Validated GenerateNewPasswordRequest request

Visszatérési érték: ResolversStatus

Az itt fel nem sorolt endpointok a szokásos CRUD műveleteket látják el, vagy ahhoz hasonló, paraméterezésében eltérő műveleteket.