

VÁCI SZAKKÉPZÉSI CENTRUM BORONKAY GYÖRGY MŰSZAKI  
TECHNIKUM ÉS GIMNÁZIUM

## VIZSGAREMEK

Pék Gergő

Kiss Kolos

Garai Nándor

2025

# VÁCI SZAKKÉPZÉSI CENTRUM BORONKAY GYÖRGY MŰSZAKI TECHNIKUM ÉS GIMNÁZIUM



# VIZSGAREMEK

# EasyInventory

Konzulens: Gyombolainé Cséry Zsuzsanna Készítette: Pék Gergő  
Kiss Kolos  
Garai Nándor

---

## **Hallgatói nyilatkozat**

Alulírottak, ezúton kijelentjük, hogy a szakdolgozat saját, önálló munkánk, és korábban még sehol nem került publikálásra. Szakdolgozatunk a Váci Szakképzési Centrum Boronkay György Műszaki Technikum és Gimnázium Szoftverfejlesztő és tesztelő technikus képzésén készítettük. Tudomásul vesszük, hogy szakdolgozatunkat a Váci Szakképzési Centrum Boronkay György Műszaki Technikum és Gimnázium tárolja.

---

Pék Gergő

---

Kis Kolos

---

Garai Nándor

---

Vizsgázók neve: Pék Gergő

Kiss Kolos

Garai Nándor

Szakdolgozat címe: EasyInventory

Program által nyújtott szolgáltatások:

- Raktárak és telephelyek kezelése
- Mértékegységek és terméktípusok kezelése
- Raktárak tartalmának módosítása műveletek segítségével
- Felhasználók kezelése
- Lokális és rendszerszintű jogosultságok
- Raktárak tartalmáról statisztikák kimutatása
- Keresés raktárakban és telephelyeken
- Raktárak kapacitásának beállítása
- Felhasználói felület asztali ,webes, mobilos platformon

Sorszám	A konzultáció időpontja	A konzulens aláírása
1.	2024. október 11.	
2.	2024. november 15.	
3.	2024. december 13.	
4.	2025. január 17.	
5.	2025. február 14.	
6.	2025. március 14.	

A szakdolgozat beadható:  
Vác, 2025.

.....  
Konzulens

A szakdolgozatot átvettetem:  
Vác, 2025.

.....  
A szakképzést folytató intézmény felelőse

# Tartalomjegyzék

<b>1 Projekt leírás</b>	<b>11</b>
1.1 Bevezetés . . . . .	11
1.2 Munkamegosztás . . . . .	12
1.2.1 Pék Gergő . . . . .	12
1.2.2 Kiss Kolos . . . . .	12
1.2.3 Garai Nándor . . . . .	12
1.3 Munka folyamata . . . . .	12
1.4 Használt fejlesztői eszközök . . . . .	13
<b>I Fejlesztői dokumentáció</b>	<b>14</b>
<b>2 API dokumentáció</b>	<b>15</b>
2.1 API Típusok . . . . .	15
2.1.1 Unit . . . . .	15
2.1.2 UnitPutRequest . . . . .	15
2.1.3 Item . . . . .	15
2.1.4 ItemPutRequest . . . . .	15
2.1.5 Warehouse . . . . .	16
2.1.6 WarehousePutRequest . . . . .	16
2.1.7 Storage . . . . .	16
2.1.8 StoragePutRequest . . . . .	16
2.1.9 User . . . . .	16
2.1.10 UserPutRequest . . . . .	17
2.1.11 OperationItem . . . . .	17
2.1.12 Operation . . . . .	17
2.1.13 OperationItemRequest . . . . .	18
2.1.14 OperationRequest . . . . .	18
2.1.15 OperationCommit . . . . .	18
2.1.16 MoveRequest . . . . .	18
2.1.17 LoginRequest . . . . .	18
2.1.18 LoginResponse . . . . .	18
2.1.19 RenewResponse . . . . .	19
2.1.20 UserinfoResponse . . . . .	19
2.1.21 StorageLimit . . . . .	19
2.1.22 StorageLimitPutRequest . . . . .	19

2.1.23	StorageCapacity	19
2.1.24	ItemStack	19
2.1.25	ItemDB	20
2.1.26	ItemStackDB	20
2.1.27	StorageLimitDB	20
2.1.28	WarehouseDB	20
2.1.29	StorageDB	21
2.1.30	LocalAuthorizationDB	21
2.1.31	UserDB	21
2.1.32	DB	21
2.2	Hitelesítés	22
2.2.1	Felhasználó bejelentkeztetése	22
2.2.2	Token megújítása	23
2.2.3	Token törlése	23
2.2.4	A bejelentkezett felhasználó információinak lekérése	24
2.3	Jogosultságok	25
2.3.1	Rendszer engedélyek lekérése	25
2.3.2	Helyi engedélyek lekérése	26
2.3.3	Helyi engedély megadása	27
2.3.4	Helyi engedély törlése	27
2.3.5	Rendszer engedély megadása	28
2.3.6	Rendszer engedély törlése	29
2.4	Cikkek	29
2.4.1	Cikkek lekérése	29
2.4.2	Tárgy azonosítójának változtatása	30
2.4.3	Cikk lekérése	31
2.4.4	Cikk módosítása, létrehozása	32
2.4.5	Cikk törlése	33
2.5	Limit	34
2.5.1	Raktár limit	34
2.5.2	Raktár limitjének módosítása	35
2.5.3	Raktár kapacitás	36
2.6	Műveletek	37
2.6.1	Műveletek lekérése	37
2.6.2	Művelet azonosítójának változtatása	38
2.6.3	Művelet lekérése	39
2.6.4	Művelet létrehozása	41
2.6.5	Művelet visszaigazolása és törlése	42
2.7	Keresés	43
2.7.1	Keresés	43
2.8	Raktárak	44

2.8.1	Raktárak lekérése . . . . .	44
2.8.2	Raktár azonosítójának változtatása . . . . .	45
2.8.3	Raktár lekérése . . . . .	46
2.8.4	Raktárak módosítása, létrehozása . . . . .	47
2.8.5	Raktár törlése . . . . .	48
2.8.6	Keresés a raktárban . . . . .	49
2.9	Egységek . . . . .	50
2.9.1	Egységek lekérése . . . . .	50
2.9.2	Egység azonosítójának változtatása . . . . .	51
2.9.3	Egység lekérése . . . . .	52
2.9.4	Egységek módosítása, létrehozása . . . . .	53
2.9.5	Egység törlése . . . . .	54
2.10	Felhasználók . . . . .	55
2.10.1	Felhasználók lekérése . . . . .	55
2.10.2	Felhasználó azonosítójának változtatása . . . . .	56
2.10.3	Felhasználó lekérése . . . . .	57
2.10.4	Felhasználók módosítása, létrehozása . . . . .	58
2.10.5	Felhasználó törlése . . . . .	59
2.11	Telephelyek . . . . .	60
2.11.1	Telephelyek lekérése . . . . .	60
2.11.2	Telephely azonosítójának változtatása . . . . .	61
2.11.3	Telephely lekérése . . . . .	62
2.11.4	Telephely módosítása, létrehozása . . . . .	63
2.11.5	Telephely törlése . . . . .	64
2.11.6	Keresés a telephelyen . . . . .	64
2.12	Biztonsági mentés . . . . .	66
2.12.1	Mentés készítése . . . . .	66
2.12.2	Mentés visszaállítása . . . . .	66
<b>3</b>	<b>API könyvtár dokumentáció</b>	<b>68</b>
3.1	Kapcsolat kezelés . . . . .	68
3.1.1	Kapcsolódás a szerverhez . . . . .	68
3.1.2	Kijelentkezés . . . . .	68
3.2	Egységek kezelése . . . . .	69
3.2.1	Egység osztály . . . . .	69
3.2.2	Egységek kezelése . . . . .	69
3.3	Felhasználók kezelése . . . . .	70
3.3.1	Felhasználó osztály . . . . .	70
3.3.2	Felhasználók kezelése . . . . .	70
3.3.3	Felhasználók engedélyeinek kezelése . . . . .	71
3.4	Telephelyek kezelése . . . . .	71
3.4.1	Telephely osztály . . . . .	71

3.4.2	Telephelyek kezelése	72
3.5	Raktárak kezelése	72
3.5.1	Raktár osztály	72
3.5.2	Raktárak kezelése	73
3.6	Cikkek kezelése	73
3.6.1	Cikk osztály	73
3.6.2	Cikkek kezelése	74
3.7	Műveletek kezelése	74
3.7.1	Művelet osztály	74
3.7.2	Műveletek kezelése	75
3.8	Keresés	75
3.8.1	ItemStack osztály	75
3.8.2	Keresés	76
<b>4</b>	<b>Kóddokumentáció</b>	<b>77</b>
4.1	Felépítés	77
4.2	Adatbázis	78
4.2.1	Áttekintés	78
4.2.2	Táblaleírások	80
4.2.3	Nézetek	84
4.3	API szerver	89
4.3.1	Áttekintés	89
4.3.2	Osztályleírások	90
4.4	Frontend	96
4.4.1	Áttekintés	96
4.4.2	Építés	96
4.4.3	Osztályleírások	96
4.5	API könyvtár	101
4.5.1	Áttekintés	101
4.5.2	HTTPConnector osztály	101
4.6	Asztali kliens	102
4.6.1	Bevezetés	102
4.6.2	Asztali kliens osztályai	102
4.7	Mobil kliens	108
4.7.1	Áttekintés	108
4.7.2	Osztályleírások	108
<b>5</b>	<b>Dokumentáció generálása</b>	<b>118</b>
5.1	Áttekintés	118
5.2	Felhasználói dokumentáció	118
5.3	Fejlesztői dokumentáció	118
5.3.1	Kóddokumentáció	118

5.3.2	Tesztdokumentáció	119
5.3.3	Kód beillesztő	119
5.4	Építés	119
<b>6</b>	<b>Tesztdokumentáció</b>	<b>120</b>
6.1	API szerver	120
6.1.1	Áttekintés	120
6.1.2	E2E tesztek	120
6.2	Web frontend	121
6.2.1	Áttekintés	121
6.2.2	E2E tesztek	121
6.3	Asztali alkalmazás	122
6.3.1	Áttekintés	122
6.3.2	E2E tesztek	122
6.3.3	NodeMatcher osztály	122
6.4	Mobil alkalmazás	125
6.4.1	Áttekintés	125
6.4.2	E2E tesztek	125
<b>II</b>	<b>Felhasználói dokumentáció</b>	<b>126</b>
<b>7</b>	<b>Felhasználói dokumentáció</b>	<b>127</b>
7.1	Bevezetés	127
7.1.1	Fogalmak	127
7.2	Telepítés	129
7.2.1	Szerver telepítés	129
7.2.2	Asztali kliens telepítés	130
7.2.3	Mobil kliens telepítés	130
7.3	Adminisztráció	131
7.3.1	Arhivált adatok törlése	131
7.3.2	Eseménynapló kezelése	131
7.3.3	Felhasználó kezelés	133
7.3.4	Engedélyek	134
7.3.5	Beállítások	135
7.4	Használat	138
7.4.1	Bejelentkezés	138
7.4.2	Felhasználói fiókok kezelése	139
7.4.3	Egységek kezelése	141
7.4.4	Árucikkek kezelése	143
7.4.5	Telephelyek kezelése	145
7.4.6	Raktárak kezelése	147
7.4.7	Műveletek	149

7.4.8	Keresés . . . . .	150
7.5	Használati példák . . . . .	151
7.5.1	Fatelep . . . . .	151
7.5.2	Bevásárló központ . . . . .	152
<b>III</b>	<b>Utószó</b>	<b>153</b>
<b>8</b>	<b>Továbbfejlesztési lehetőségek</b>	<b>154</b>
8.1	Adminisztráció . . . . .	154
8.1.1	Integráció monitorozó rendszerekkel . . . . .	154
8.1.2	Központi beléptető rendszer . . . . .	154
8.1.3	Webhook-ok . . . . .	154
8.2	Statisztika . . . . .	154
8.2.1	Kördiagram telephely/raktár tartalmáról . . . . .	154
8.2.2	Különböző raktárak/telephelyek összehasonlítása . . . . .	155
8.3	Használat . . . . .	155
8.3.1	Térkép a raktárról és benne a tárgyakról . . . . .	155
8.3.2	Rendszeres művelet order . . . . .	155
8.3.3	Mértékegység átváltás . . . . .	155
8.3.4	Raktártípusok . . . . .	155
8.3.5	Riasztás beállítás . . . . .	156
8.3.6	QR kód olvasó . . . . .	156
<b>9</b>	<b>Felhasznált irodalom</b>	<b>157</b>
<b>10</b>	<b>Munkakönyvtárak</b>	<b>158</b>
<b>11</b>	<b>Mellékletek</b>	<b>159</b>
11.1	Online elérhetőség . . . . .	159

# 1. Projekt leírás

## 1.1 Bevezetés

A rendszer célja, lehetővé tenni cégek számára, hogy bármiféle képesítés vagy különösebb hozzáértés nélkül is egyszerűen tudják kezelní a legkomplexebb nyilvántartásokat is. A rendszer lehetővé teszi különböző jogkörök felhasználókhöz való csatolását, melyek befolyásolják, hogy a nyilvántartás mely részeit tekintheti meg és milyen műveleteket hajthat végre a programon belül az adott felhasználó. A rendszer képes telephelyek, raktárak és cikkek nyilvántartására, miközben figyelembe veszi a különböző egységek és tárolóhelyek közötti kapcsolatok kezelését is. Mindezek mellett, egyik fő szempontunk, hogy a rendszer a lehető leg helytakarékosabb módon rendezze el az adott cikkeket a raktárokon és a telephelyeken belül, figyelembe véve azt is, hogy ne kerülhessenek egymás mellé olyan anyagok, amelyek reakcióba lépése károsodásokhoz vezethet. Az alábbiakban részletesen ismertetjük a program főbb funkciót és kezelési lehetőségeit. Ezt a feladatot azért választottuk, mert kellő kihívásnak találtuk.

## 1.2 Munkamegosztás

### 1.2.1 Pék Gergő

Csoportvezetés, API megtervezése és elkészítése, mobil és asztali alkalmazás létrehozása, webes frontend és API szerver architektúrájának tervezése, adatbázis megtervezése és elkészítése, dokumentáció megírása, stílusának tervezése és kód minőségének ellenőrzése.

### 1.2.2 Kiss Kolos

API tesztek írása, webes frontend elkészítése, felhasználói dokumentáció megírása és nyelvi helyességének ellenőrzése, design elemek (ikonok, logo) megrajzolása, nyelvi fájlok (angol, magyar) elkészítése és asztali alkalmazás felhasználói felülete.

### 1.2.3 Garai Nándor

Webes és API tesztek írása, API szerver implementálása, web frontend design megtervezése, tesztadatok generálása, adatbázis tervezése, bemutató készítése és asztali alkalmazás felhasználói felülete.

## 1.3 Munka folyamata

A fejlesztés teszt alapú megközelítéssel zajlott. Először az API készült el, majd párhuzamosan dolgoztunk a három felületen (mobil, asztali, web) és a hozzájuk tartozó tesztekken. Végül a dokumentáció elkészítése és a végső simítások következtek.

## 1.4 Használt fejlesztői eszközök

- Netbeans
- VSCode
- Android Studio
- PHPMyAdmin
- Inkscape
- XeLatex

# I. FEJLESZTŐI DOKUMENTÁCIÓ

---

# 2. API dokumentáció

## 2.1 API Típusok

### 2.1.1 Unit

Kulcs	Típus	Tartalom
id	string	Az egység azonosítója
name	string	Az egység neve
deleted	int / null	A törlés ideje UNIX másodpercben

### 2.1.2 UnitPutRequest

Kulcs	Típus	Tartalom
name	string	Az egység neve

### 2.1.3 Item

Kulcs	Típus	Tartalom
id	string	A cikk azonosítója
name	string	A cikk neve
unit	Unit	A cikk mértékegysége
deleted	int / null	A törlés ideje UNIX másodpercben

### 2.1.4 ItemPutRequest

Kulcs	Típus	Tartalom
name	string	A cikk neve
unit	string	A cikk mértékegységének az azonosítója

## 2.1.5 Warehouse

Kulcs	Típus	Tartalom
id	string	A telephely azonosítója
name	string	A telephely neve
address	string	A telephely címe
deleted	int / null	A törlés ideje UNIX másodpercben

## 2.1.6 WarehousePutRequest

Kulcs	Típus	Tartalom
name	string	A telephely neve
address	string	A telephely címe

## 2.1.7 Storage

Kulcs	Típus	Tartalom
id	string	A raktár azonosítója
name	string	A raktár neve
warehouse	Warehouse	A telephely amiben van a raktár
deleted	int / null	A törlés ideje UNIX másodpercben

## 2.1.8 StoragePutRequest

Kulcs	Típus	Tartalom
id	string	A raktár azonosítója
name	string	A raktár neve

## 2.1.9 User

Kulcs	Típus	Tartalom
id	string	A felhasználó azonosítója
name	string	A felhasználó neve
manager	User / null	A felhasználó főnöke

## 2.1.10 UserPutRequest

Kulcs	Típus	Tartalom
name	string	A felhasználó neve
password	string / null	Az új jelszó
manager	string / null	A felhasználó főnökének az azonosítója

## 2.1.11 OperationItem

Kulcs	Típus	Tartalom
type	Item	Cél cikk
storage	Storage	Cél raktár
global_serial	int	A cél cikk sorozatszáma (0 ha nincs)
manufacturer_serial	string / null	A cél cikk gyártó sorozatszáma
amount	int	Mennyiség

## 2.1.12 Operation

Kulcs	Típus	Tartalom
id	string	A művelet azonosítója
name	string	A művelet neve
is_add	boolean	Igaz ha ez a művelet hozzáad, hamis ha elvesz
items	OperationItem lista	A művelet részei
created	int	A létrehozás ideje UNIX másodpercben
committed	int / null	A visszaigazolás ideje UNIX másodpercben

## 2.1.13 OperationItemRequest

Kulcs	Típus	Tartalom
type	Item	Cél cikk
storage	Storage / null	Cél raktár
global_serial	int	A cél cikk sorozatszáma (0 ha nincs)
manufacturer_serial	string / null	A cél cikk gyártó sorozatszáma
amount	int	Mennyiség

## 2.1.14 OperationRequest

Kulcs	Típus	Tartalom
name	string	A művelet neve
is_add	boolean	Igaz ha ez a művelet hozzáad, hamis ha elvesz
items	OperationItem Request lista	A művelet részei

## 2.1.15 OperationCommit

Kulcs	Típus	Tartalom
cancel	boolean	Igaz ha törl, hamis ha visszaigazol

## 2.1.16 MoveRequest

Kulcs	Típus	Tartalom
from	string	A erőforrás azonosítója
to	string	A erőforrás új azonosítója

## 2.1.17 LoginRequest

Kulcs	Típus	Tartalom
password	string	A jelszó bejelentkezéshez

## 2.1.18 LoginResponse

Kulcs	Típus	Tartalom
token	string	A token

## 2.1.19 RenewResponse

Kulcs	Típus	Tartalom
expiration	int	A token lejárásának időpontja Unix másodpercekben

## 2.1.20 UserinfoResponse

Kulcs	Típus	Tartalom
user	string	A bejelentkezett felhasználó azonosítója
username	string	A bejelentkezett felhasználó neve

## 2.1.21 StorageLimit

Kulcs	Típus	Tartalom
item	Item	A cikk
amount	int	A limit

## 2.1.22 StorageLimitPutRequest

Kulcs	Típus	Tartalom
amount	int	A limit

## 2.1.23 StorageCapacity

Kulcs	Típus	Tartalom
item	Item	A cikk
limit	int	A limit
stored_amount	int	A tárolt mennyiség

## 2.1.24 ItemStack

Kulcs	Típus	Tartalom
item	Item	A cikk
amount	int	A tárolt mennyiség
available_amount	int	Az elérhető mennyiség
global_serial	int	A sorozatszám
manufacturer_serial	string / null	A gyártói sorozatszám
lot	string / null	A lot szám

## 2.1.25 ItemDB

Kulcs	Típus	Tartalom
id	string	A cikk azonosítója
name	string	A cikk neve
unit	string	A cikk mértékegységének azonosítója
deleted	int / null	A törlés ideje UNIX másodpercben

## 2.1.26 ItemStackDB

Kulcs	Típus	Tartalom
item	string	A cikk azonosítója
amount	int	A tárolt mennyiség
global_serial	int	A sorozatszám
manufacturer_serial	string / null	A gyártói sorozatszám
lot	string / null	A lot szám

## 2.1.27 StorageLimitDB

Kulcs	Típus	Tartalom
item	string	A cikk azonosítója
amount	int	A limit

## 2.1.28 WarehouseDB

Kulcs	Típus	Tartalom
id	string	A telephely azonosítója
name	string	A telephely neve
address	string	A telephely címe
deleted	int / null	A törlés ideje UNIX másodpercben
storages	StorageDB lista	A telephely raktárai
operations	Operation lista	A műveletek

## 2.1.29 StorageDB

Kulcs	Típus	Tartalom
id	string	A raktár azonosítója
name	string	A raktár neve
deleted	int / null	A törlés ideje UNIX másodpercben
limits	StorageLimitDB lista	A raktár limitjei
item_stacks	ItemStackDB lista	A tárolt cikkkek

## 2.1.30 LocalAuthorizationDB

Kulcs	Típus	Tartalom
warehouse	string	A telephely azonosítója
authorizations	string list	A megadott helyi engedélyek

## 2.1.31 UserDB

Kulcs	Típus	Tartalom
id	string	A felhasználó azonosítója
name	string	A felhasználó neve
manager	string	A felhasználó főnökének azonosítója
system_authorizations	string list	A felhasználó rendszer szintű engedélyei
local_authorizations	LocalAuthorizationDB lista	A felhasználó rendszer szintű engedélyei

## 2.1.32 DB

Kulcs	Típus	Tartalom
units	Unit lista	Az egységek
items	ItemDB lista	A árucikk fajták
warehouses	WarehouseDB lista	A telephelyek
user	UserDB lista	A felhasználók

## 2.2 Hitelesítés

### 2.2.1 Felhasználó bejelentkeztetése

Erőforrás

POST /users/`user`/auth

#### Paraméterek

**user** Felhasználó azonosítója

#### Kérés fejléce

**Authorization** Bearer token

**Content-Type** application/json

#### Kérés

Egy LoginRequest objektum.

```
{  
    "password": "admin123"  
}
```

#### Válasz kódok

**404** Nem létezik

**200** OK

#### Válasz

Egy LoginResponse objektum.

```
{  
    "token": "rLQalTg3WnQW9EYgB9XSbtEzifc1bcnC"  
}
```

## 2.2.2 Token megújítása

Erőforrás

**POST /tokens**

### Kérés fejléce

**Authorization** Bearer token

### Válasz kódok

**200** OK

**404** Nem létezik

### Válasz fejléce

**Content-Type** application/json

### Válasz

Egy RenewResponse objektum.

```
{  
    "expiration":900  
}
```

## 2.2.3 Token törlése

Erőforrás

**DELETE /tokens**

### Kérés fejléce

**Authorization** Bearer token

## Válasz kódok

**404** Nem létezik

### 2.2.4 A bejelentkezett felhasználó információinak lekérése

Erőforrás

GET /userinfo

## Kérés fejléce

**Authorization** Bearer token

## Válasz kódok

**401** Nincs bejelentkezve

**200** OK

## Válasz fejléce

**Content-Type** application/json

## Válasz

Egy **UserinfoResponse** objektum.

```
{  
    "user": "janos1990",  
    "username": "János"  
}
```

## 2.3 Jogosultságok

### 2.3.1 Rendszer engedélyek lekérése

Erőforrás

```
GET /users/user/authorizations/system
```

#### Paraméterek

**user** Felhasználó azonosítója

#### Kérés fejléce

**Authorization** Bearer token

#### Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**404** Nem létezik

**200** OK

#### Válasz fejléce

**Content-Type** application/json

#### Válasz

A megadott rendszer engedélyek azonosítójainak listája.

```
["view_types"]
```

## 2.3.2 Helyi engedélyek lekérése

Erőforrás

```
GET /users/user/authorizations/local/warehouse
```

### Paraméterek

**user** Felhasználó azonosítója

**warehouse** Telephely azonosítója

### Kérés fejléce

**Authorization** Bearer token

### Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**404** Nem létezik

**200** OK

### Válasz fejléce

**Content-Type** application/json

### Válasz

A megadott helyi engedélyek azonosítójainak listája.

```
[ "add" , "remove" ]
```

### 2.3.3 Helyi engedély megadása

Erőforrás

```
PUT /users/user/authorizations/local/warehouse/authorization
```

#### Paraméterek

**user** Felhasználó azonosítója

**warehouse** Telephely azonosítója

**authorization** Az engedély azonosítója

#### Kérés fejléce

**Authorization** Bearer token

#### Válasz kódok

**401** Nincs bejelentkezve

**403** Nincs megfelelő engedély

**404** Nem létezik

**204** Engedély megadva

### 2.3.4 Helyi engedély törlése

Erőforrás

```
DELETE /users/user/authorizations/local/warehouse/authorization
```

#### Paraméterek

**user** Felhasználó azonosítója

**warehouse** Telephely azonosítója

**authorization** Az engedély azonosítója

## Kérés fejléce

**Authorization** Bearer token

## Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**404** Nem létezik

**204** Törölve

## 2.3.5 Rendszer engedély megadása

Erőforrás

```
PUT /users/user/authorizations/system/authorization
```

## Paraméterek

**user** Felhasználó azonosítója

**authorization** Az engedély azonosítója

## Kérés fejléce

**Authorization** Bearer token

## Válasz kódok

**401** Nincs bejelentkezve

**403** Nincs megfelelő engedély

**404** Nem létezik

**204** Engedély megadva

## 2.3.6 Rendszer engedély törlése

Erőforrás

```
DELETE /users/user/authorizations/system/authorization
```

### Paraméterek

**user** Felhasználó azonosítója

**authorization** Az engedély azonosítója

### Kérés fejléce

**Authorization** Bearer token

### Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**404** Nem létezik

**204** Törölve

## 2.4 Cikkek

### 2.4.1 Cikkek lekérése

Erőforrás

```
GET /items?q=q&archived=archived&offset=offset&limit=limit
```

## Paraméterek

**q** Keresési szöveg

**archived** Ha `true` akkor arhivált értékeket is visszaad

**offset** Visszaadott elemek kezdő pozíciója, 0 ha hiányzik

**limit** Maximum visszaadott elemek száma, 20 ha hiányzik

## Kérés fejléce

**Authorization** Bearer token

## Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**200** OK

## Válasz fejléce

**Content-Type** application/json

## Válasz

Egy **Item** lista.

```
[  
  {  
    "id": "carrot_box",  
    "name": "Répa",  
    "unit": {  
      "id": "small_box",  
      "name": "Kis Doboz",  
      "deleted": null  
    },  
    "deleted": null  
  }  
]
```

## 2.4.2 Tárgy azonosítójának változtatása

Erőforrás

POST /items

## Kérés fejléce

**Authorization** Bearer token

**Content-Type** application/json

## Kérés

Egy MoveRequest objektum.

```
{  
  "from": "carrot_box"  
  "to": "repa_box"  
}
```

## Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**400** Hibás kérés

**404** Nem létezik

**409** A cél azonosító már létezik.

**204** Átnevezve

## 2.4.3 Cikk lekérése

Erőforrás

```
GET /items/item
```

## Paraméterek

**item** A cikk azonosítója

## Kérés fejléce

**Authorization** Bearer token

## Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**404** Nem létezik

**200** OK

## Válasz fejléce

**Content-Type** application/json

## Válasz

Egy **Item** objektum.

```
{  
  "id": "carrot_box",  
  "name": "Répa",  
  "unit": {  
    "id": "small_box",  
    "name": "Kis Doboz",  
    "deleted": null  
  },  
  "deleted": null  
}
```

## 2.4.4 Cikk módosítása, létrehozása

### Erőforrás

```
PUT /items/item?update=update&create=create
```

## Paraméterek

**item** A cikk azonosítója

**update** Ha `true` akkor meglévő erőforrást frissíthet

**create** Ha `true` akkor új erőforrást létrehozhat

## Kérés fejléce

**Authorization** Bearer token

**Content-Type** application/json

## Kérés

Egy ItemPutRequest objektum.

```
{  
    "name": "Krumpli",  
    "unit": "small_box"  
}
```

## Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**400** Hibás kérés

**404** Nem létezik

**204** Módosítva

**201** Létrehozva

## 2.4.5 Cikk törlése

Erőforrás

```
DELETE /items/{items}
```

## Paraméterek

**item** A cikk azonosítója

## Kérés fejléce

**Authorization** Bearer token

## Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**404** Nem létezik

**204** Törölte

# 2.5 Limit

## 2.5.1 Raktár limit

Erőforrás

```
GET /warehouses/{warehouse}/storages/{storage}/limits?q={q}&offset={offset}&limit={limit}
```

## Paraméterek

**warehouse** Telephely azonosítója

**storage** Raktár azonosítója

**q** Keresési szöveg

**offset** Visszaadott elemek kezdő pozíciója, 0 ha hiányzik

**limit** Maximum visszaadott elemek száma, 20 ha hiányzik

## Kérés fejléce

**Authorization** Bearer token

## Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**200** OK

## Válasz fejléce

**Content-Type** application/json

## Válasz

Egy StorageLimit objektum lista.

```
[  
  {  
    "item":{  
      "id":"carrot_box",  
      "name":"Répa",  
      "unit":{  
        "id":"small_box",  
        "name":"Kis Doboz",  
        "deleted":null  
      },  
      "deleted":null  
    },  
    "amount":10  
  }  
]
```

## 2.5.2 Raktár limitjének módosítása

Erőforrás

```
PUT /warehouses/{warehouse}/storages/{storage}/item
```

### Paraméterek

**warehouse** Telephely azonosítója

**storage** Raktár azonosítója

**item** A cikk azonosítója

### Kérés fejléce

**Authorization** Bearer token

**Content-Type** application/json

### Kérés

Egy StorageLimitPutRequest objektum.

```
{  
  "amount":10  
}
```

## Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**400** Hibás kérés

**404** Nem létezik

**204** Módosítva

**201** Létrehozva

### 2.5.3 Raktár kapacitás

Erőforrás

```
GET /warehouses/{warehouse}/storages/{storage}/capacity?  
q={q}&offset={offset}&limit={limit}
```

## Paraméterek

**warehouse** Telephely azonosítója

**storage** Raktár azonosítója

**q** Keresési szöveg

**offset** Visszaadott elemek kezdő pozíciója, 0 ha hiányzik

**limit** Maximum visszaadott elemek száma, 20 ha hiányzik

## Kérés fejléce

**Authorization** Bearer token

## Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**404** Nem létezik

**200** OK

## Válasz fejléce

**Content-Type** application/json

## Válasz

Egy StorageCapacity objektum lista.

```
[
  {
    "item": {
      "id": "carrot_box",
      "name": "Répa",
      "unit": {
        "id": "small_box",
        "name": "Kis Doboz",
        "deleted": null
      },
      "deleted": null
    },
    "stored_amount": 3,
    "limit": 10
  }
]
```

# 2.6 Műveletek

## 2.6.1 Műveletek lekérése

Erőforrás

```
GET /warehouses/{warehouse}/operations?q={q}&archived={archived}&offset={offset}&limit={limit}
```

## Paraméterek

**warehouse** Telephely azonosítója

**q** Keresési szöveg

**archived** Ha `true` akkor arhivált értékeket is visszaad

**offset** Visszaadott elemek kezdő pozíciója, 0 ha hiányzik

**limit** Maximum visszaadott elemek száma, 20 ha hiányzik

## Kérés fejléce

**Authorization** Bearer token

## Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**200** OK

## Válasz fejléce

**Content-Type** application/json

## Válasz

Egy Operation objektum lista.

```
[  
  {  
    "id": "OP0001",  
    "is_add": false,  
    "name": "Test Operation",  
    "created": 1234567,  
    "committed": null,  
    "items": []  
  }  
]
```

## 2.6.2 Művelet azonosítójának változtatása

Erőforrás

```
POST /warehouses/warehouse/operations
```

## Paraméterek

**warehouse** Telephely azonosítója

## Kérés fejléce

**Authorization** Bearer token

**Content-Type** application/json

## Kérés

Egy MoveRequest objektum.

```
{  
  "from": "OP0001"  
  "to": "OP000"  
}
```

## Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**400** Hibás kérés

**404** Nem létezik

**409** A cél azonosító már létezik.

**204** Átnevezve

## 2.6.3 Művelet lekérése

Erőforrás

```
GET /warehouses/{warehouse}/operations/{operation}
```

## Paraméterek

**warehouse** Telephely azonosítója

**operation** Tárgy azonosítója

## Kérés fejléce

**Authorization** Bearer token

## Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**404** Nem létezik

**200** OK

## Válasz fejléce

**Content-Type** application/json

## Válasz

Egy Operation objektum.

```
{  
    "id": "OP0001",  
    "is_add": false,  
    "name": "Test Operation",  
    "created": 1234567,  
    "committed": null  
    "items": [  
        {  
            "item": {  
                "id": "carrot_box",  
                "name": "Répa",  
                "unit": {  
                    "id": "small_box",  
                    "name": "Kis Doboz",  
                    "deleted": null  
                },  
                "deleted": null  
            },  
            "amount": 20,  
            "storage": {  
                "id": "ST1",  
                "name": "Storage 1",  
                "warehouse": {  
                    "id": "WH1",  
                    "name": "Warehouse 1",  
                    "deleted": null  
                },  
                "deleted": null  
            },  
            "global_serial": null,  
            "manufacturer_serial": null,  
            "lot": "L00001"  
        }  
    ]  
}
```

## 2.6.4 Művelet létrehozása

Erőforrás

```
PUT /warehouses/{warehouse}/operations/{operation}
```

### Paraméterek

**warehouse** Telephely azonosítója

**operation** Tárgy azonosítója

### Kérés fejléce

**Authorization** Bearer token

**Content-Type** application/json

### Kérés

Egy OperationRequest objektum.

```
{
  "is_add":false,
  "name":"New Test Operation",
  "items": []
}
```

### Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**400** Hibás kérés

**404** Nem létezik

**409** A cél azonosító már létezik, vagy nem lehet végrehajtani a műveletet.

**201** Létrehozva

## 2.6.5 Művelet visszaigazolása és törlése

Erőforrás

```
DELETE /warehouses/warehouse/operations/operation
```

### Paraméterek

**warehouse** Telephely azonosítója

**operation** Tárgy azonosítója

### Kérés fejléce

**Authorization** Bearer token

### Kérés

Egy **OperationCommit** objektum.

```
{  
    "cancel": false  
}
```

### Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**404** Nem létezik

**204** Törölve vagy visszaigazolva

## 2.7 Keresés

### 2.7.1 Keresés

Erőforrás

```
GET /search?q=q&warehouse=qwarehouse&storage=qstorage&lot=lot&serial=serial
```

#### Paraméterek

**q** Keresési szöveg

**arehouse** Ha `true` akkor telephely szerint is csoportosít

**storage** Ha `true` akkor raktár szerint is csoportosít

**lot** Ha `true` akkor lot szám szerint is csoportosít

**serial** Ha `true` akkor sorozatszám szerint is csoportosít

**offset** Visszaadott elemek kezdő pozíciója, 0 ha hiányzik

**limit** Maximum visszaadott elemek száma, 20 ha hiányzik

#### Kérés fejléce

**Authorization** Bearer token

#### Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**200** OK

#### Válasz fejléce

**Content-Type** application/json

## Válasz

Egy `ItemStack` objektum lista.

```
[  
  {  
    "amount":170,  
    "available_amount":170,  
    "global_serial":null,  
    "type":"carrot_box",  
    "lot":"L00001",  
    "manufacturer_serial":null,  
    "warehouse":null,  
    "storage":null  
  }  
]
```

# 2.8 Raktárak

## 2.8.1 Raktárak lekérése

### Erőforrás

```
GET /warehouses/{warehouse}/storages?q={q}&archived={archived}&offset={offset}&  
limit={limit}
```

### Paraméterek

**warehouse** Telephely azonosítója

**q** Keresési szöveg

**archived** Ha `true` akkor arhivált értékeket is visszaad

**offset** Visszaadott elemek kezdő pozíciója, 0 ha hiányzik

**limit** Maximum visszaadott elemek száma, 20 ha hiányzik

### Kérés fejléce

**Authorization** Bearer token

## Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**200** OK

## Válasz fejléce

**Content-Type** application/json

## Válasz

Egy Storage objektum lista.

```
[  
  {  
    "id": "ST1",  
    "name": "Storage 1",  
    "warehouse": {  
      "id": "WH1",  
      "name": "Warehouse 1",  
      "address": "fake street",  
      "deleted": null  
    },  
    "deleted": null  
  }  
]
```

## 2.8.2 Raktár azonosítójának változtatása

Erőforrás

**POST /warehouses/*warehouse*/storages**

## Paraméterek

**warehouse** Telephely azonosítója

## Kérés fejléce

**Authorization** Bearer token

**Content-Type** application/json

## Kérés

Egy MoveRequest objektum.

```
{  
    "from": "ST1"  
    "to": "storage1"  
}
```

## Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**400** Hibás kérés

**404** Nem létezik

**409** A cél azonosító már létezik.

**204** Átnevezve

## 2.8.3 Raktár lekérése

Erőforrás

```
GET /warehouses/{warehouse}/storages/{storage}
```

## Paraméterek

**warehouse** Telephely azonosítója

**storage** Raktár azonosítója

## Kérés fejléce

**Authorization** Bearer token

## Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**404** Nem létezik

**200** OK

## Válasz fejléce

**Content-Type** application/json

## Válasz

Egy Storage objektum.

```
{  
  "id": "ST1",  
  "name": "Storage 1",  
  "warehouse": {  
    "id": "WH1",  
    "name": "Warehouse 1",  
    "address": "fake street",  
    "deleted": null  
  },  
  "deleted": null  
}
```

## 2.8.4 Raktárak módosítása, létrehozása

Erőforrás

```
PUT /warehouses/{warehouse}/storages/{storage}?update={update}&create={create}
```

## Paraméterek

**warehouse** Telephely azonosítója

**storage** Raktár azonosítója

**update** Ha true akkor meglévő erőforrást frisíthet

**create** Ha true akkor új erőforrást létrehozhat

## Kérés fejléce

**Authorization** Bearer token

**Content-Type** application/json

## Kérés

Egy StoragePutRequest objektum.

```
{  
    "name": "New Name"  
}
```

## Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**400** Hibás kérés

**404** Nem létezik

**204** Módosítva

**201** Létrehozva

## 2.8.5 Raktár törlése

Erőforrás

```
DELETE /warehouses/{warehouse}/storages/{storage}
```

## Paraméterek

**warehouse** Telephely azonosítója

**storage** Raktár azonosítója

## Kérés fejléce

**Authorization** Bearer token

## Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**404** Nem létezik

**204** Törölve

## 2.8.6 Keresés a raktárban

Erőforrás

```
GET /warehouses/{warehouse}/storages/{storage}/search?q={q}&warehouse={arehouse}&storage={storage}&lot={lot}&serial={serial}
```

### Paraméterek

**warehouse** Telephely azonosítója

**storage** Raktár azonosítója

**q** Keresési szöveg

**arehouse** Ha `true` akkor telephely szerint is csoportosít

**storage** Ha `true` akkor raktár szerint is csoportosít

**lot** Ha `true` akkor lot szám szerint is csoportosít

**serial** Ha `true` akkor sorozatszám szerint is csoportosít

**offset** Visszaadott elemek kezdő pozíciója, 0 ha hiányzik

**limit** Maximum visszaadott elemek száma, 20 ha hiányzik

### Kérés fejléce

**Authorization** Bearer token

### Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**404** Nem létezik

**200** OK

### Válasz fejléce

**Content-Type** application/json

## Válasz

Egy `ItemStack` objektum lista.

```
[  
  {  
    "amount":170,  
    "available_amount":170,  
    "global_serial":null,  
    "type":"carrot_box",  
    "lot":"L00001",  
    "manufacturer_serial":null,  
    "warehouse":null,  
    "storage":null  
  }  
]
```

## 2.9 Egységek

### 2.9.1 Egységek lekérése

Erőforrás

```
GET /units?q=q&archived=archived&offset=offset&limit=limit
```

## Paraméterek

**q** Keresési szöveg

**archived** Ha `true` akkor arhivált értékeket is visszaad

**offset** Visszaadott elemek kezdő pozíciója, 0 ha hiányzik

**limit** Maximum visszaadott elemek száma, 20 ha hiányzik

## Kérés fejléce

**Authorization** Bearer token

## Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**200** OK

## Válasz fejléce

**Content-Type** application/json

## Válasz

Egy Unit objektum.

```
[  
  {  
    "id": "box",  
    "name": "Box",  
    "deleted": null  
  }  
]
```

## 2.9.2 Egység azonosítójának változtatása

Erőforrás

**POST /units**

## Kérés fejléce

**Authorization** Bearer token

**Content-Type** application/json

## Kérés

Egy MoveRequest objektum.

```
{  
  "from": "pellet"  
  "to": "pellet2"  
}
```

## Válasz kódok

- 403** Nincs megfelelő engedély
- 401** Nincs bejelentkezve
- 400** Hibás kérés
- 404** Nem létezik
- 409** A cél azonosító már létezik.
- 204** Átnevezve

### 2.9.3 Egység lekérése

Erőforrás

`GET /units/unit`

## Paraméterek

- unit** Egység azonosítója

## Kérés fejléce

**Authorization** Bearer token

## Válasz kódok

- 403** Nincs megfelelő engedély
- 401** Nincs bejelentkezve
- 404** Nem létezik
- 200** OK

## Válasz fejléce

**Content-Type** application/json

## Válasz

Egy Unit objektum.

```
{  
  "id": "box",  
  "name": "Box",  
  "deleted": null  
}
```

### 2.9.4 Egységek módosítása, létrehozása

Erőforrás

```
PUT /units/unit?update=update&create=create
```

## Paraméterek

**unit** Egység azonosítója

**update** Ha true akkor meglévő erőforrást frisíthet

**create** Ha true akkor új erőforrást létrehozhat

## Kérés fejléce

**Authorization** Bearer token

**Content-Type** application/json

## Kérés

Egy UnitPutRequest objektum.

```
{  
  "name": "New Name"  
}
```

## Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**400** Hibás kérés

**404** Nem létezik

**204** Módosítva

**201** Létrehozva

## 2.9.5 Egység törlése.

Erőforrás

```
DELETE /units/unit
```

## Paraméterek

**unit** Egység azonosítója

## Kérés fejléce

**Authorization** Bearer token

## Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**404** Nem létezik

**204** Törölte

## 2.10 Felhasználók

### 2.10.1 Felhasználók lekérése

Erőforrás

```
GET /users?q=q&offset=offset&limit=limit
```

#### Paraméterek

**q** Keresési szöveg

**offset** Visszaadott elemek kezdő pozíciója, 0 ha hiányzik

**limit** Maximum visszaadott elemek száma, 20 ha hiányzik

#### Kérés fejléce

**Authorization** Bearer token

#### Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**200** OK

#### Válasz fejléce

**Content-Type** application/json

## Válasz

Egy `User` objektum lista.

```
[  
  {  
    "name": "János",  
    "manager": {  
      "id": "admin",  
      "name": "Admin",  
      "manager": null  
    }  
  },  
  {  
    "id": "admin",  
    "name": "Admin",  
    "manager": null  
  }  
]
```

## 2.10.2 Felhasználó azonosítójának változtatása

Erőforrás

```
POST /users
```

### Kérés fejléce

**Authorization** Bearer token

**Content-Type** application/json

### Kérés

Egy `MoveRequest` objektum.

```
{  
  "from": "janos1990"  
  "to": "jani1990"  
}
```

## Válasz kódok

- 403** Nincs megfelelő engedély
- 401** Nincs bejelentkezve
- 400** Hibás kérés
- 404** Nem létezik
- 409** A cél azonosító már létezik.
- 204** Átnevezve

### 2.10.3 Felhasználó lekérése

Erőforrás

**GET /users/*user***

## Paraméterek

**user** Felhasználó azonosítója

## Kérés fejléce

**Authorization** Bearer token

## Válasz kódok

- 403** Nincs megfelelő engedély
- 401** Nincs bejelentkezve
- 404** Nem létezik
- 200** OK

## Válasz fejléce

**Content-Type** application/json

## Válasz

Egy User objektum.

```
{  
  "name": "János",  
  "manager": {  
    "id": "admin",  
    "name": "Admin",  
    "manager": null  
  }  
}
```

## 2.10.4 Felhasználók módosítása, létrehozása

### Erőforrás

```
PUT /users/{user}
```

## Paraméterek

**user** Felhasználó azonosítója

## Kérés fejléce

**Authorization** Bearer token

**Content-Type** application/json

## Kérés

Egy UserPutRequest objektum.

```
{  
  "name": "Test User",  
  "password": "pw123",  
  "manager": "admin"  
}
```

## Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**400** Hibás kérés

**404** Nem létezik

**204** Módosítva

**201** Létrehozva

## 2.10.5 Felhasználó törlése

Erőforrás

```
DELETE /users/user
```

## Paraméterek

**user** Felhasználó azonosítója

## Kérés fejléce

**Authorization** Bearer token

## Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**404** Nem létezik

**204** Törölve

## 2.11 Telephelyek

### 2.11.1 Telephelyek lekérése

Erőforrás

```
GET /warehouses?q=q&archived=archived&offset=offset&limit=limit
```

#### Paraméterek

**q** Keresési szöveg

**archived** Ha `true` akkor arhivált értékeket is visszaad

**offset** Visszaadott elemek kezdő pozíciója, 0 ha hiányzik

**limit** Maximum visszaadott elemek száma, 20 ha hiányzik

#### Kérés fejléce

**Authorization** Bearer token

#### Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**404** Nem létezik

**200** OK

#### Válasz fejléce

**Content-Type** application/json

## Válasz

Egy **Warehouse** objektum lista.

```
[  
  {  
    "id": "WH1",  
    "name": "Warehouse 1",  
    "address": "fake street",  
    "deleted": null  
  }  
]
```

## 2.11.2 Telephely azonosítójának változtatása

Erőforrás

```
POST /warehouses
```

### Kérés fejléce

**Authorization** Bearer token

**Content-Type** application/json

### Kérés

Egy **MoveRequest** objektum.

```
{  
  "from": "WH1"  
  "to": "WHBP"  
}
```

### Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**400** Hibás kérés

**404** Nem létezik

**409** A cél azonosító már létezik.

**204** Átnevezve

### 2.11.3 Telephely lekérése

Erőforrás

GET /units/*warehouse*

#### Paraméterek

**warehouse** Telephely azonosítója

#### Kérés fejléce

**Authorization** Bearer token

#### Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**404** Nem létezik

**200** OK

#### Válasz fejléce

**Content-Type** application/json

#### Válasz

Egy **Warehouse** objektum.

```
{  
    "id": "WH1",  
    "name": "Warehouse 1",  
    "address": "fake street",  
    "deleted": null  
}
```

## 2.11.4 Telephely módosítása, létrehozása

Erőforrás

```
PUT /warehouses/{warehouse?update=update&create=create}
```

### Paraméterek

**warehouse** Telephely azonosítója

**update** Ha `true` akkor meglévő erőforrást frisíthet

**create** Ha `true` akkor új erőforrást létrehozhat

### Kérés fejléce

**Authorization** Bearer token

**Content-Type** application/json

### Kérés

Egy `WarehousePutRequest` objektum.

```
{  
    "name": "Warehouse 1 (modified)",  
    "address": "fake street2"  
}
```

### Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**400** Hibás kérés

**404** Nem létezik

**204** Módosítva

**201** Létrehozva

## 2.11.5 Telephely törlése

Erőforrás

```
DELETE /warehouses/warehouse
```

### Paraméterek

**warehouse** Telephely azonosítója

### Kérés fejléce

**Authorization** Bearer token

### Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**404** Nem létezik

**204** Törölve

## 2.11.6 Keresés a telephelyen

Erőforrás

```
GET /warehouses/warehouse/search?q=q&warehouse=arehouse&storage=storage&  
lot=lot&serial=serial
```

## Paraméterek

**warehouse** Telephely azonosítója

**q** Keresési szöveg

**qwarehouse** Ha `true` akkor telephely szerint is csoportosít

**qstorage** Ha `true` akkor raktár szerint is csoportosít

**lot** Ha `true` akkor lot szám szerint is csoportosít

**serial** Ha `true` akkor sorozatszám szerint is csoportosít

**offset** Visszaadott elemek kezdő pozíciója, 0 ha hiányzik

**limit** Maximum visszaadott elemek száma, 20 ha hiányzik

## Kérés fejléce

**Authorization** Bearer token

## Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**404** Nem létezik

**200** OK

## Válasz fejléce

**Content-Type** application/json

## Válasz

Egy **ItemStack** objektum lista.

```
[  
  {  
    "amount":170,  
    "available_amount":170,  
    "global_serial":null,  
    "type":"carrot_box",  
    "lot":"L00001",  
    "manufacturer_serial":null,  
    "warehouse":null,  
    "storage":null  
  }  
]
```

## 2.12 Biztonsági mentés

### 2.12.1 Mentés készítése

Erőforrás

GET /db

#### Kérés fejléce

**Authorization** Bearer token

#### Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**200** OK

#### Válasz fejléce

**Content-Type** application/json

#### Válasz

Egy DB objektum.

```
{  
  "units": [] ,  
  "items": [] ,  
  "users": [] ,  
  "warehouses": []  
}
```

### 2.12.2 Mentés visszaállítása

Erőforrás

PUT /db

## Kérés fejléce

**Authorization** Bearer token

**Content-Type** application/json

## Kérés

Egy DB objektum.

```
{  
  "units": [] ,  
  "items": [] ,  
  "users": [] ,  
  "warehouses": []  
}
```

## Válasz kódok

**403** Nincs megfelelő engedély

**401** Nincs bejelentkezve

**400** Hibás kérés

**204** Módosítva

# 3. API könyvtár dokumentáció

## 3.1 Kapcsolat kezelés

### 3.1.1 Kapcsolódás a szerverhez

Az API szerverhez való kapcsolódást a `connect` funkció hajtja végre. A funkció egy `API` objektumot ad vissza, aminek a segítségével kommunikálhatunk a szerverrel.

```
public class API {  
    public static API connect(HTTPConnector connector, String url,  
        String username, String password) throws IOException,  
        APIException, JSONException { ... }  
    ...  
}
```

### 3.1.2 Kijelentkezés

A kijelentkezéshez a `logout` metódust kell meghívni. Ezt követően nem lehetséges az `API` objektum további használata.

```
public class API {  
    public void logout() throws IOException, APIException,  
        JSONException { ... }  
    ...  
}
```

## 3.2 Egységek kezelése

### 3.2.1 Egység osztály

Az egység adatainak tárolásához a `Unit` osztály használható.

```
public class Unit implements ToJSON {
    public final String id;
    public final String name;
    public final long deleted;
    public Unit(String id, String name, long deleted) { ... }
    public Unit(String id, String name) { ... }
    public Unit(JSONObject o) throws JSONException { ... }
    @Override
    public JSONObject toJSON() { ... }
    @Override
    public int hashCode() { ... }
    @Override
    public boolean equals(Object obj) { ... }
}
```

### 3.2.2 Egységek kezelése

A cikkek kezeléséhez az `API` osztályban található funkciókat használhatjuk.

```
public class API {
    public Unit[] getUnits(String query, int offset, int len, boolean
        archived) throws IOException, APIException, JSONException {
        ...
    }
    public Unit getUnit(String id) throws IOException, APIException,
        JSONException { ... }
    public boolean putUnit(String id, String name, boolean create,
        boolean update) throws IOException, APIException,
        JSONException { ... }
    public boolean deleteUnit(String id) throws IOException,
        APIException { ... }
    public boolean moveUnit(String id, String new_id) throws
        IOException, APIException, JSONException { ... }
    ...
}
```

## 3.3 Felhasználók kezelése

### 3.3.1 Felhasználó osztály

Az felhasználók adatainak tárolásához a `User` osztály használható.

```
public class User implements ToJSON {
    public final String id;
    public final String name;
    public final String password;
    public final User manager;
    public User(String id, String name, String password, User manager)
        { ... }
    public User(JSONObject o) throws JSONException { ... }
    @Override
    public JSONObject toJSON() { ... }
    @Override
    public int hashCode() { ... }
    @Override
    public boolean equals(Object obj) { ... }
}
```

### 3.3.2 Felhasználók kezelése

A felhasználók kezeléséhez az `API` osztályban található funkciókat használhatjuk.

```
public class API {
    public UserInfo getUserinfo() throws IOException, APIException,
        JSONException { ... }
    public User[] getUsers(String query, int offset, int len) throws
        IOException, APIException, JSONException { ... }
    public User getUser(String id) throws IOException, APIException,
        JSONException { ... }
    public boolean putUser(String id, String name, String password,
        String manager_id, boolean create, boolean update) throws
        IOException, APIException, JSONException { ... }
    public boolean deleteUser(String id) throws IOException,
        APIException { ... }
    public boolean moveUser(String id, String new_id) throws
        IOException, APIException, JSONException { ... }
    public void changePassword(String old_password, String
        new_password) throws IOException, APIException, JSONException
        { ... }
}
```

### 3.3.3 Felhasználók engedélyeinek kezelése

A felhasználók engedélyeinek kezeléséhez az API osztályban található funkciókat használhatjuk.

```
public class API {
    public SystemAuthorization getSystemAuthorization(String user)
        throws IOException, APIException, JSONException { ... }
    public LocalAuthorization getLocalAuthorization(String user,
        String warehouse) throws IOException, APIException,
        JSONException { ... }
    public boolean grantSystemAuthorization(String user, String
        authorization) throws IOException, APIException { ... }
    public boolean revokeSystemAuthorization(String user, String
        authorization) throws IOException, APIException { ... }
    public boolean grantLocalAuthorization(String user, String
        warehouse, String authorization) throws IOException,
        APIException { ... }
    public boolean revokeLocalAuthorization(String user, String
        warehouse, String authorization) throws IOException,
        APIException { ... }
    ...
}
```

## 3.4 Telephelyek kezelése

### 3.4.1 Telephely osztály

Az telephelyek adatainak tárolásához a `Warehouse` osztály használható.

```
public class Warehouse implements ToJSON {
    public final String id;
    public final String name;
    public final String address;
    public final long deleted;
    public Warehouse(String id, String name, String address, long
        deleted) { ... }
    public Warehouse(String id, String name, String address) { ... }
    public Warehouse(JSONObject o) throws JSONException { ... }
    @Override
    public JSONObject toJSON() { ... }
    @Override
    public boolean equals(Object obj) { ... }
    @Override
    public int hashCode() { ... }
}
```

### 3.4.2 Telephelyek kezelése

A telephelyek kezeléséhez az API osztályban található funkciókat használhatjuk.

```
public class API {
    public Warehouse[] getWarehouses(String query,int offset,int
        len,boolean archived) throws IOException,APIException,
        JSONException { ... }
    public Warehouse getWarehouse(String id) throws IOException,
        APIException,JSONException { ... }
    public boolean putWarehouse(String id,String name,String
        address,boolean create,boolean update)
    public boolean deleteWarehouse(String id) throws IOException,
        APIException { ... }
    public boolean moveWarehouse(String id,String new_id) throws
        IOException,APIException,JSONException { ... }
    ...
}
```

## 3.5 Raktárak kezelése

### 3.5.1 Raktár osztály

Az raktárak adatainak tárolásához a `Storage` osztály használható.

```
public class Storage implements ToJSON {
    public final String id;
    public final String name;
    public final Warehouse warehouse;
    public final long deleted;
    public Storage(Warehouse warehouse,String id,String name,long
        deleted) { ... }
    public Storage(Warehouse warehouse,String id,String name) { ... }
    public Storage(JSONObject o) throws JSONException { ... }
    @Override
    public JSONObject toJSON() { ... }
    @Override
    public int hashCode() { ... }
    @Override
    public boolean equals(Object obj) { ... }
}
```

## 3.5.2 Raktárak kezelése

A raktárak kezeléséhez az API osztályban található funkciókat használhatjuk.

```
public class API {
    public Storage getStorage(String warehouse, String id) throws
        IOException, APIException, JSONException { ... }
    public boolean putStorage(String warehouse, String id, String
        name, boolean create, boolean update) throws IOException,
        APIException, JSONException { ... }
    public boolean deleteStorage(String warehouse, String id) throws
        IOException, APIException { ... }
    public boolean moveStorage(String warehouse, String id, String
        new_id) throws IOException, APIException, JSONException { ... }
    public StorageLimit[] getStorageLimits(String warehouse, String
        id, String query, int offset, int len) throws IOException,
        APIException, JSONException { ... }
    public boolean setStorageLimit(String warehouse, String id,
        String item, int amount) throws IOException, APIException,
        JSONException { ... }
    public StorageCapacity[] getStorageCapacity(String warehouse,
        String id, String query, int offset, int len) throws
        IOException, APIException, JSONException { ... }
    ...
}
```

## 3.6 Cikkek kezelése

### 3.6.1 Cikk osztály

A cikkek adatainak tárolásához a Item osztály használható.

```
public class Item implements ToJSON {
    public final String id;
    public final String name;
    public final Unit unit;
    public final long deleted;
    public Item(String id, String name, Unit unit, long deleted) { ... }
    public Item(String id, String name, Unit unit) { ... }
    public Item(JSONObject o) throws JSONException { ... }
    @Override
    public JSONObject toJSON() { ... }
    @Override
    public int hashCode() { ... }
    @Override
    public boolean equals(Object obj) { ... }
}
```

## 3.6.2 Cikkek kezelése

A cikkek kezeléséhez az `API` osztályban található funkciókat használhatjuk.

```
public class API {
    public Item[] getItems(String query, int offset, int len, boolean
        archived) throws IOException, APIException, JSONException {
        ...
    }
    public Item getItem(String id) throws IOException, APIException,
        JSONException { ... }
    public boolean putItem(String id, String name, String unit_id,
        boolean create, boolean update) throws IOException,
        APIException, JSONException { ... }
    public boolean deleteItem(String id) throws IOException,
        APIException { ... }
    public boolean moveItem(String id, String new_id) throws
        IOException, APIException, JSONException { ... }
    ...
}
```

# 3.7 Műveletek kezelése

## 3.7.1 Művelet osztály

Az műveletek adatainak tárolásához a `Operation` osztály használható.

```
public class Operation implements ToJSON {
    public final String id;
    public final String name;
    public final boolean is_add;
    public final OperationItem[] items;
    public final long deleted;
    public Operation(String id, String name, boolean is_add,
        OperationItem[] items, long deleted) { ... }
    public Operation(String id, String name, boolean is_add,
        OperationItem[] items) { ... }
    public Operation(JSONObject o) throws JSONException { ... }
    @Override
    public JSONObject toJSON() { ... }
    @Override
    public int hashCode() { ... }
    @Override
    public boolean equals(Object obj) { ... }
}
```

## 3.7.2 Műveletek kezelése

A műveletek kezeléséhez az API osztályban található funkciókat használhatjuk.

```
public class API {
    public Operation[] getOperations(String warehouse, String query,
        int offset, int len, boolean archived) throws IOException,
        APIException, JSONException { ... }
    public boolean putOperation(String warehouse, String id, String
        name, boolean is_add, OperationItem[] items) throws
        IOException, APIException, JSONException { ... }
    public boolean cancelOperation(String warehouse, String id)
        throws IOException, APIException { ... }
    public boolean finishOperation(String warehouse, String id)
        throws IOException, APIException { ... }
    ...
}
```

## 3.8 Keresés

### 3.8.1 ItemStack osztály

A keresés eredményeinek tárolásához az ItemStack osztályt használhatjuk.

```
public class ItemStack implements ToJSON {
    public final Warehouse warehouse;
    public final Storage storage;
    public final Item item;
    public final String lot;
    public final String manufacturer_serial;
    public final int amount;
    public final int available_amount;
    public final int global_serial;
    public ItemStack(Warehouse warehouse, Storage storage, Item item,
        String lot,
        String manufacturer_serial, int amount, int
        available_amount, int global_serial) { ... }
    public ItemStack(JSONObject o) throws JSONException { ... }
    @Override
    public int hashCode() { ... }
    @Override
    public boolean equals(Object obj) { ... }
    @Override
    public JSONObject toJSON() { ... }
}
```

## 3.8.2 Keresés

A kereséshez az API osztályban található funkciókat használhatjuk.

```
public class API {  
    public ItemStack[] getCurrentItems(String warehouse, String  
        storage, String query,  
        ...  
}
```

# 4. Kódokumentáció

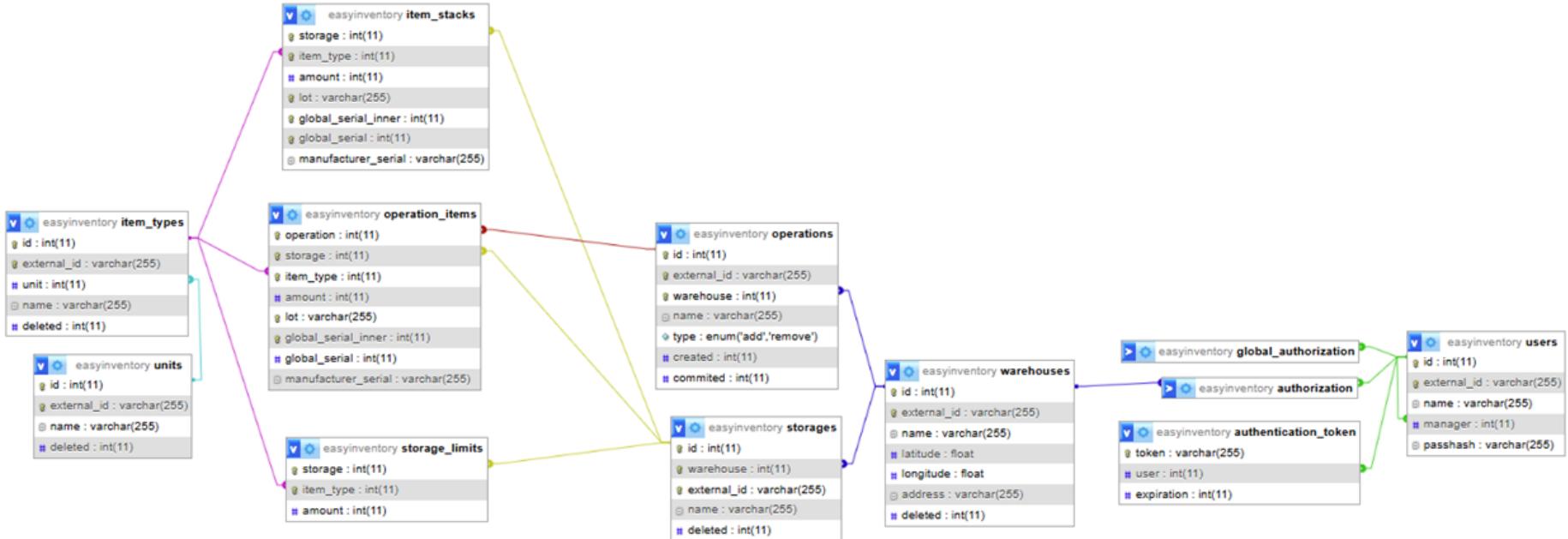
## 4.1 Felépítés

A gyökérkennelben találhatók a `api`, `apitest`, `EasyInventoryAPI`, `EasyInventoryDesktop`, `EasyInventoryMobile`, `doc`, `frontend` és a `lang` mappák. Továbbá ez tartalmazza a `startphp.bat` scriptet, ami elindítja a teszt PHP szervert és a `vscode.bat` fájlt, ami elindítja a VS-Code fejlesztői környezetet. A `lang` mappa tartalmazza a nyelvfájlokat és egy `update.py` scriptet, ami ellenőrzi a nyelvfájlokat és frissíti a programok nyelvfájljait.

## 4.2 Adatbázis

### 4.2.1 Áttekintés

A rendszer Mysql relációs adatbázist használ. Az adatbázis létrehozásához szükséges SQL kód a `api/db.sql` fájlban található. A táblákban sokszor megtalálható az `external_id` attributum, amely az API által használt azonosítót tárolja. Erre azért van szükség, mert a valódi azonosító módosításához a táblák nagy részét le kéne zárni, ami lassítaná a lekéréseket. Az adatbázisban több nézet is található a lekérések egyszerűbbé tételeire. Az alábbi grafikon mutatja a táblák kapcsolatrendszerét:



## 4.2.2 Táblaleírások

### units

A `units` tábla a mértékegységek tárolásáért felelős.

```
create table if not exists units (
    id int auto_increment not null,
    external_id varchar(255) not null unique,
    name varchar(255) not null,
    deleted int default null,
    primary key(id)
) $
```

### item\_types

Az `item_types` tábla a különböző cikk típusok tárolásáért felelős.

```
create table if not exists item_types (
    id int auto_increment not null,
    external_id varchar(255) not null unique,
    unit int not null,
    name varchar(255) not null,
    deleted int default null,
    primary key(id),
    foreign key(unit) references units(id)
) $
```

### warehouses

A `warehouses` tábla a különböző telephelyek tárolásáért felelős.

```
create table if not exists warehouses (
    id int not null auto_increment,
    external_id varchar(255) not null unique,
    name varchar(255) not null,
    latitude float,
    longitude float,
    address varchar(255),
    deleted int default null,
    primary key(id)
) $
```

## storages

A `storages` tábla a különböző raktárak tárolásáért felelős.

```
create table if not exists storages (
    id int not null auto_increment,
    warehouse int not null,
    external_id varchar(255) not null,

    name varchar(255) not null,
    deleted int default null,

    foreign key(warehouse) references warehouses(id),
    primary key(id),
    unique (warehouse,external_id)
) $
```

## item\_stacks

Az `item_stacks` tábla az éppen tárolt különböző árucikkek tárolásáért felelős.

```
create table if not exists item_stacks (
    storage int not null,
    item_type int not null,

    amount int not null,
    lot varchar(255),
    global_serial_inner int not null default 0,
    global_serial int as (if(global_serial_inner=0,NULL,
        global_serial_inner)) virtual,
    manufacturer_serial varchar(255),

    unique(global_serial),
    foreign key(storage) references storages(id),
    foreign key(item_type) references item_types(id),
    primary key(storage,item_type,lot,global_serial_inner)
) $
```

## storage\_limits

A `storage_limits` tábla a raktár limiteket tárolásáért felelős.

```
create table if not exists storage_limits (
    storage int not null,
    item_type int not null,
    amount int not null,

    primary key(storage,item_type),
    foreign key(storage) references storages(id),
    foreign key(item_type) references item_types(id)
) $
```

## operations

Az operations tábla a műveletek tárolásáért felelős.

```
create table if not exists operations (
    id int auto_increment not null,
    external_id varchar(255) not null,
    warehouse int not null,
    name varchar(255),
    type enum ('add','remove'),
    created int not null,
    committed int,
    primary key(id),
    unique(external_id,warehouse),
    foreign key(warehouse) references warehouses(id)
)$
```

## operation\_items

Az operation\_items tábla a művelet részek tárolásáért felelős.

```
create table if not exists operation_items (
    operation int not null,
    storage int not null,
    item_type int not null,
    amount int not null,
    lot varchar(255),
    global_serial_inner int not null default 0,
    global_serial int as (if(global_serial_inner=0,NULL,
        global_serial_inner)) virtual,
    manufacturer_serial varchar(255),
    primary key(operation,storage,item_type,lot,global_serial_inner),
    foreign key(operation) references operations(id) on delete cascade,
    foreign key(storage) references storages(id),
    foreign key(item_type) references item_types(id)
)$
```

## users

A `users` tábla a felhasználók tárolásáért felelős.

```
create table if not exists users (
    id int auto_increment not null,
    external_id varchar(255) not null unique,
    name varchar(255),
    manager int,
    passhash varchar(255) not null,
    primary key(id),
    foreign key(manager) references users(id)
) $
```

## authorization

Az `authorization` tábla a helyi engedélyek tárolásáért felelős.

```
create table if not exists authorization (
    user int not null,
    warehouse int not null,
    authorization enum ('view', 'create_add_operation',
        'create_remove_operation', 'handle_operation', 'configure'),
    primary key(user,warehouse,authorization),
    foreign key(user) references users(id),
    foreign key(warehouse) references warehouses(id)
) $
```

## global\_authorization

A `global_authorization` tábla a rendszer engedélyek tárolásáért felelős.

```
create table if not exists global_authorization (
    user int not null,
    authorization enum (
        "view_warehouses",
        "delete_warehouses",
        "create_warehouses",
        "modify_warehouses",
        "delete_types",
        "create_types",
        "modify_types",
        "view_users",
        "delete_users",
        "create_users",
        "modify_users",
        "view_statistics"),
    primary key(user,authorization),
    foreign key(user) references users(id)
)$
```

## authentication\_token

Az `authentication_token` tábla a bejelentkezési tokenek tárolásáért felelős.

```
create table if not exists authentication_token (
    token varchar(255) not null,
    user int not null,
    expiration int,
    primary key(token),
    foreign key(user) references users(id)
)$
```

## 4.2.3 Nézetek

### users\_view

A `users_view` a felhasználók adatainak lekérését egyszerűsíti.

```
create or replace view users_view as
    select a.external_id id,a.name `name`,b.external_id manager,b.
        name manager_name,a.passhash passhash from users a
    left join users b on a.manager=b.id$
```

## authorization\_view

Az `authorization_view` tábla a helyi és rendszer engedélyek lekérését egyszerűsíti.

```
create or replace view authorization_view as
    select external_id id,authorization,null warehouse from users
        inner join global_authorization on users.id=
            global_authorization.user
    union
    select users.external_id id,authorization,warehouses.
        external_id warehouse from users
        inner join authorization on users.id=authorization.user
        inner join warehouses on warehouses.id=authorization.
            warehouse$
```

## authentication\_token\_view

Az `authentication_token_view` a bejelentkezési tokenek lekérését egyszerűsíti.

```
create or replace view authentication_token_view as
    select token,external_id user,expiration from
        authentication_token
        inner join users on users.id=authentication_token.user
        where expiration>UNIX_TIMESTAMP()$
```

## warehouses\_view

A `warehouses_view` tábla a telephelyek lekérését egyszerűsíti.

```
create or replace view warehouses_view as
    select external_id id,name,latitude,longitude,address,deleted
        from warehouses$
```

## storages\_view

A `storages_view` tábla a raktárak lekérését egyszerűsíti.

```
create or replace view storages_view as
    select warehouses.external_id warehouse,warehouses.name
        warehouse_name,warehouses.address warehouse_address,
        warehouses.deleted warehouse_deleted,storages.external_id id
        ,storages.name,storages.deleted deleted from storages
        inner join warehouses on storages.warehouse=warehouses.id$
```

## item\_types\_view

A item\_types\_view tábla a cikk típusok lekérését egyszerűsíti.

```
create or replace view item_types_view as
  select item_types.external_id id, item_types.name, units.
    external_id unit, units.name unit_name, units.deleted
    unit_deleted, item_types.deleted deleted from item_types
      inner join units on units.id=item_types.unit$
```

## units\_view

A units\_view tábla a mérték egységek lekérését egyszerűsíti.

```
create or replace view units_view as
  select external_id id, name, deleted from units$
```

## operations\_view

A operations\_view tábla a műveletek lekérését egyszerűsíti.

```
create or replace view operations_view as
  select operations.external_id id, warehouses.external_id
    warehouse, warehouses.name warehouse_name, warehouses.address
    warehouse_address, warehouses.deleted warehouse_deleted,
    operations.name, type, created, committed from operations
      inner join warehouses on operations.warehouse=warehouses.
        id$
```

## operation\_items\_view

A operation\_items\_view tábla a művelet részek lekérését egyszerűsíti.

```
create or replace view operation_items_view as
  select
    operations.external_id operation,
    operations.type `type`,
    operations.committed committed,
    storages.external_id storage,
    storages.name storage_name,
    storages.deleted storage_deleted,
    warehouses.external_id warehouse,
    warehouses.name warehouse_name,
    warehouses.address warehouse_address,
    warehouses.deleted warehouse_deleted,
    item_types.external_id item,
    item_types.name item_name,
    item_types.deleted item_deleted,
    units.external_id unit,
    units.name unit_name,
    units.deleted unit_deleted,
    amount,
    lot,
    global_serial,
    manufacturer_serial
  from operation_items
    inner join operations on operations.id=operation_items.
      operation
    inner join storages on storages.id=operation_items.storage
    inner join item_types on item_types.id=operation_items.
      item_type
    inner join units on units.id=item_types.unit
    left join warehouses on warehouses.id=operations.warehouse$
```

## item\_stacks\_view

A item\_stacks\_view tábla a tárolt áru lekérését egyszerűsíti.

```

create or replace view item_stacks_view as
select
    warehouses.external_id warehouse,
    warehouses.name warehouse_name,
    warehouses.address warehouse_address,
    storages.external_id `storage`,
    storages.name storage_name,
    item_types.external_id item,
    item_types.name item_name,
    units.external_id unit,
    units.name unit_name,
    item_stacks.amount amount,
    item_stacks.lot lot,
    item_stacks.global_serial global_serial,
    item_stacks.manufacturer_serial manufacturer_serial,
    (item_stacks.amount - COALESCE(sum(oi.amount), 0))
        available_amount
from item_stacks
inner join storages on storages.id=item_stacks.storage
inner join warehouses on storages.warehouse=warehouses.id
inner join item_types on item_types.id=item_stacks.
    item_type
inner join units on item_types.unit=units.id
left join (
    select * from operation_items inner join operations on
        operation_items.operation=operations.id where
        operations.type='remove' and committed is null
) oi on (
    oi.storage=item_stacks.storage and
    oi.item_type=item_stacks.item_type and
    oi.lot=item_stacks.lot and
    oi.global_serial_inner=item_stacks.global_serial_inner
)
group by
    item_stacks.storage,
    item_stacks.item_type,
    item_stacks.lot,
    item_stacks.global_serial_inner$
```

## 4.3 API szerver

### 4.3.1 Áttekintés

---

#### Átirányítási réteg

A `.htaccess` -nek köszönhetően minden beérkező kérés az `index.php` fájlba érkezik, amely ezt követően a `src/Routing/Router.php` fájlba továbbítja azokat. A `Router.php` fájl megvizsgálja a kérések URL -jét és metódusát, majd a query kivételével dekódolja és továbbítja az URL paramétereket a `src/API` mappában található fájlokba.

#### API kezelő réteg

Az API kérések értelmezése és az engedélyek ellenörzése, illetve az azokra való válasz küldése a `src/API` mappában lévő fájlok feladata. A kérések formátumának ellenörzéséért például a `src/API/APIUtils.php` fájl felelős.

#### Adatbázis kapcsolat réteg

A `src/Database` mappában található fájlok feladata az adatbázissal való kapcsolat megteremtése és az adatbázis műveletek kezelése.

## 4.3.2 Osztályleírások

### ApiRouter.php

Amennyiben a kérés metódusa megegyezik a `$method` -al, illetve a kérés URL -je hasonlít a `$pattern` -ben található mintával, a route funkció feladata, hogy lefuttassa a `$function` funkciót az URL -ből kiszedett és dekódolt paraméterekkel.

```
private static function route($method, $pattern, $function): void {
    if($method !== $_SERVER['REQUEST_METHOD'])
        return;
    $url=explode("?",$_SERVER['REQUEST_URI'])[0];
    $url = explode("/",trim($url,"/"));
    $pattern = explode("/",trim($pattern,"/"));
    $params = [];
    if(count($pattern) !== count($url))
        return;

    for($i = 0; $i < count($pattern); $i++)
        if($pattern[$i]==":")
            $params []= urldecode($url[$i]);
        else if($url[$i] != $pattern[$i])
            return;
    $function(...$params);
}
```

Az API -ra érkező kérés hatására a `handle` nevű funkció hívódik meg, amely a `route` segítségével átirányítja a dekódolt kéréseket az api kezelő rétegbe.

```
class ApiRouter {
    public static function handle() { ... }
    ...
}
```

## APIUtils.php

A validate funkció feladata megvizsgálni, hogy a `$data` értéke a `$template` szerint van-e formázva. Amennyiben a `$template` egy szöveg, akkor a funkció a `$data` típusát ellenőrzi. Ilyen értékek például:

- "string"
- "integer"
- "double"

(Ha a `$template` szerint tört szám kell, akkoraz egész szám is elfogadott, mivel egyszerűen átalakítható.)

Amennyiben a `$template` egy lista, ellenőrzi, hogy a `$data` is lista-e. Ezt követően a `$template` az első elemét használja mintaként és rekurzívan ellenőrzi a `$data` lista elemeit. Ilyen értékek például:

[**"string"**] Olyan listákat fogad el amiben csak szövegek vannak.

[**"integer"**] Olyan listákat fogad el amiben csak egészek vannak.

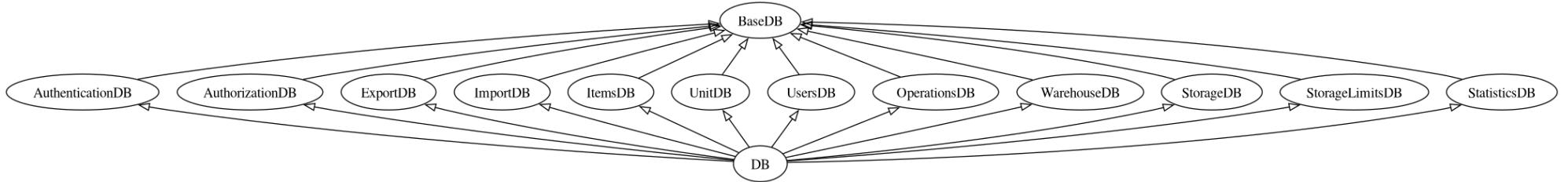
[**"double"**] Olyan listákat fogad el amiben csak törtszám listák vannak.

Abban az esetben, ha a `$template` egy asszociatív lista, a validate funkció ellenőrzi, hogy a `$data` is asszociatív lista e. Ezt követően a `$template`, saját értékeit mintaként használva, rekurzívanellenőrzi a `$data` értékeit. Amennyiben a `$template` kulcsa "?" -el végződik, nincs szükség rá, hogy a `$data` -nak is meglegyen. Ilyen értékek például:

[**"key"=>"integer"**] Ez elfogadja a [`"key"=>10`] értéket.

[**"key?"=>"string"**] Ez elfogadja a [`"key"=>"szöveg"`] és [] értékeket.

```
static function validate(array|string $template,mixed $data) {
    $datatype=gettype($data);
    if(gettype($template)==="string") {
        if(!self::isCompatible($template,$datatype))
            Response::badRequest();
    }else{
        if($datatype!="array")
            Response::badRequest();
        if(isset($template[0])) {
            foreach($data as $d) {
                self::validate($template[0],$d);
            }
        }else{
            foreach($template as $key=>$templ) {
                if(str_ends_with($key,"?")){
                    $d=$data[substr($key,0,strlen($key)-1)]??null;
                    if($d!==null)
                        self::validate($templ,$d);
                }else
                    self::validate($templ,$data[$key]??null);
            }
        }
    }
}
```



Kép 4.1: Adatbázis kapcsolati réteg osztálydiagrammja

## DB.php

A DB osztály felelős az adatbázishoz való kapcsolódásért. A legtöbb metódus külön `trait`-ekben van implementálva, melyeket ez az osztály használ. Ez a megoldás nagyban javítja az olvashatóságat, mivel a hasonló feladatokat ellátó funkciók ugyanabban a fájlban vannak és nem minden a `DB.php`-ban.

## BaseTrait.php

Sok hasznos SQL lekérdezéssel foglalkozó metódus van implementálva a `BaseTrait`-ben, így az összes többi `trait` használja a `BaseTrait`-et ezeknek a bizonyos funkcióknak az eléréséhez.

## Logger.php

A `Logger` osztály az eseménynapló kezelésével foglalkozik.

```
class Logger {
    function __construct($src_ip,$src_port,$useragent,$http,$method
        ,$url,$req_body) { ... }
    function __destruct() { ... }
    public static function set_logger(self $l):void { ... }
    public static function set_userid(string|null $userid):void {
        ...
    }
    public static function set_response(int $code,string $body):
        void { ... }
    public static function log(string $message):void { ... }
    public static function error($e):void { ... }
}
```

## Settings.php

A `Settings` osztály a `config.json` olvasásáért felelős.

```
class Settings {
    public readonly DBConnection $db_connection;
    public readonly LogSettings $success_log;
    public readonly LogSettings $bad_log;
    public readonly LogSettings $rejected_log;
    public readonly LogSettings $error_log;
    public readonly string|null $test_token;
    public readonly int $token_length;
    public readonly int $token_expiration;
        string|null $test_token,int $token_length,int
        $token_expiration) { ... }
    public static function getSettings():Settings { ... }
}
```

## Authentication.php

A `Authentication` osztály a hiteletisért felelős.

```
class Authentication {
    public static function getToken():string|null { ... }
    public static function isTestUser():bool { ... }
    public static function getCurrentUserId():string|null { ... }
    public static function isValidPassword(string $password):bool {
        ...
    }
    public static function verifyPassword(string $user_id,string
        $password):bool { ... }
    public static function authenticate(string $user_id,string
        $password):string|null { ... }
    public static function createHash(string $password): string {
        ...
    }
}
```

## 4.4 Frontend

### 4.4.1 Áttekintés

A webes frontend html része több fájlba van tagolva az olvashatóság érdekében. A html nagy része sablon amit a javascript használ az oldal megjelenítéséhez. Ez lehetővé teszi az újratöltés mentes navigálást.

### 4.4.2 Építés

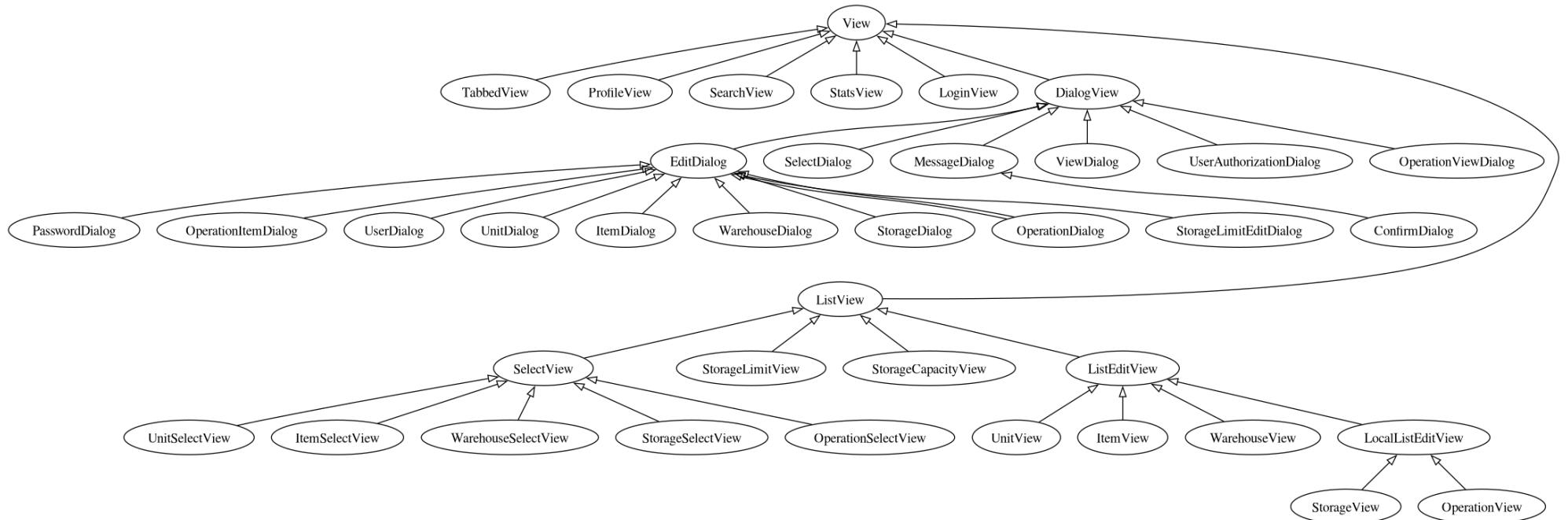
A frontend építéséhez a Linux alapú rendszeren `build` vagy Windowson `build.bat` fájlt kell futtatni. Ez létrehozza az `index.html` fájlt a széttagolt html fájlokból.

### 4.4.3 Osztályleírások

#### Template osztály

A `Template` osztály segíti a html minták kezelését. A htmlben osztályokkal vannak megjelölve azok az elemek amiket JavaScript kód módosít. A `get` funkció ezek az elemek lekérését segíti. A `getNode` funkció visszaadja az új template gyökér elemét.

```
class Template {
    template;
    constructor(template) { ... }
    get(classname) { ... }
    getNode() { ... }
}
```



Kép 4.2: Osztálydiagramm

## View osztály

A `View` osztály a felhasználói felület részek szülőosztálya. Ebben az osztályban az `initTemplate` metódus egyszer hívódik meg és a feladata az, hogy inicializálja a felhasználói felület részét.

```
class View {  
    view_template=null;  
    template_id;  
    constructor(template_id) { ... }  
    getNode() { ... }  
    initTemplate() {}  
}
```

## ListView osztály

A `ListView` osztály a listák szülőosztálya. Ebben a listában lehet lapozni, keresni és újratölteni. A `load` funkció feladata, hogy a megadott paraméterekre (keresési szöveg, archivált-e, kezdő pozíció és mennyiség) vissza adjon egy `Promise` objektumot, amely később vissza adja majd a betöltött értékeket. A töltés animáció egész addig látható a felületen, amíg a Promise nem ad vissza értéket vagy hibát. A `createItemCard` funkció feladata, hogy a betöltött értékeket vizualizálja.

```
class ListView extends View {  
    offset=0;  
    content;  
    spinner;  
    query;  
    prev;  
    next;  
    archived;  
    reloading=false;  
    constructor() { ... }  
    initTemplate() { ... }  
    reload() { ... }  
    load(q,archived,offset,length) {}  
    createItemCard(v) {}  
}
```

## ListEditView osztály

A `ListEditView` osztály a `ListView` osztályt szerkesztési menüvel egészíti ki. A `createEditView` funkció feladata az, hogy létre hozza a szerkesztési menüt. A `item` a szerkeszteni kívánt érték. Ez `null` ha újat akarunk létrehozni. Az `item_callback` funkciót kell meghívnia a menünek a mentéshez. Ennek a paraméterei az eredeti érték és az új érték. A `save` funkció feladata az új vagy módosított érték mentése.

```
class ListEditView extends ListView {  
    initTemplate() { ... }  
    createItemCard(v) { ... }  
    createItemNode(item) {}  
    createEditView(item, item_callback) {}  
    delete(item) {}  
    save(original, modified) {}  
    initEditDelete(tmpl, v) { ... }  
}
```

## SelectView osztály

A `SelectView` osztály a kiválasztási listák szülőosztálya. Ez az osztály a `ListView` osztályt egészíti ki és ezért ebben a listában is lehet lapozni, keresni és újratölteni.

```
class SelectView extends ListView {  
    selected=null;  
    selectedNode=null;  
    getSelected() { ... }  
    initTemplate() { ... }  
    createItemCard(v) { ... }  
}
```

## DialogView osztály

A `DialogView` osztály a felugró menük szülőosztálya. A `hasOk` funkció igazat ad, amennyiben ennek a menünek van OK gombja. Ebben az esetben implementálni kell az `ok` funkciót. Ez a funkció az OK gomb lenyomásakor hívódik meg és egy `Promise` objektumot ad vissza, majd a töltés animáció egész addig látható lesz, amíg ez az objektum kész nem lesz.

```
class DialogView extends View {  
    ok_button=null;  
    constructor(template_id) { ... }  
    ok() {}  
    hasOk() { ... }  
}
```

## EditDialog osztály

A `EditDialog` osztály a szerkesztési menük szülőosztálya. A `getItem` funkció visszaadja a módosított értéket vagy hibát ha rosszul van kitöltve a menü.

```
class EditDialog extends DialogView {  
    item_callback;  
    item;  
    constructor(template_id,item_callback,item) { ... }  
    validate() { ... }  
    getItem() {}  
    ok() { ... }  
    hasOk() { ... }  
}
```

## 4.5 API könyvtár

### 4.5.1 Áttekintés

Az API könyvtárat a mobil és asztali alkalmazás használja ahoz, hogy kommunikáljon az API szerverrel. A könyvtárban az `API` osztály implementálja az összes funkciót amikkel tudunk kommunikálni az API-val. A használatáról található egy részletesebb leírás a saját dokumentációjában (68. oldal).

### 4.5.2 HTTPConnector osztály

A `HTTPConnector` egy absztrakt osztály, amelynek feladata, hogy az adott paraméterek alapján létrehozzon egy kérést, majd vissza adja annak eredményét. Annak érdekében, hogy az osztályt használó programok hatékonyabban irányíthassák a kéréseket a könyvtár használata közben, ez az osztály nincs beépítve.

```
public interface HTTPConnector {  
    APIResponse execute(String method, String url, String token,  
                        String body, boolean readResponseBody) throws IOException;  
}
```

## 4.6 Asztali kliens

### 4.6.1 Bevezetés

Az asztali kliens az API könyvtár segítségével kommunikál az API szerverrel. A kliens a JavaFX felhasználói felület platformot használja. A fejlesztéshez a Netbeans IDE használható.

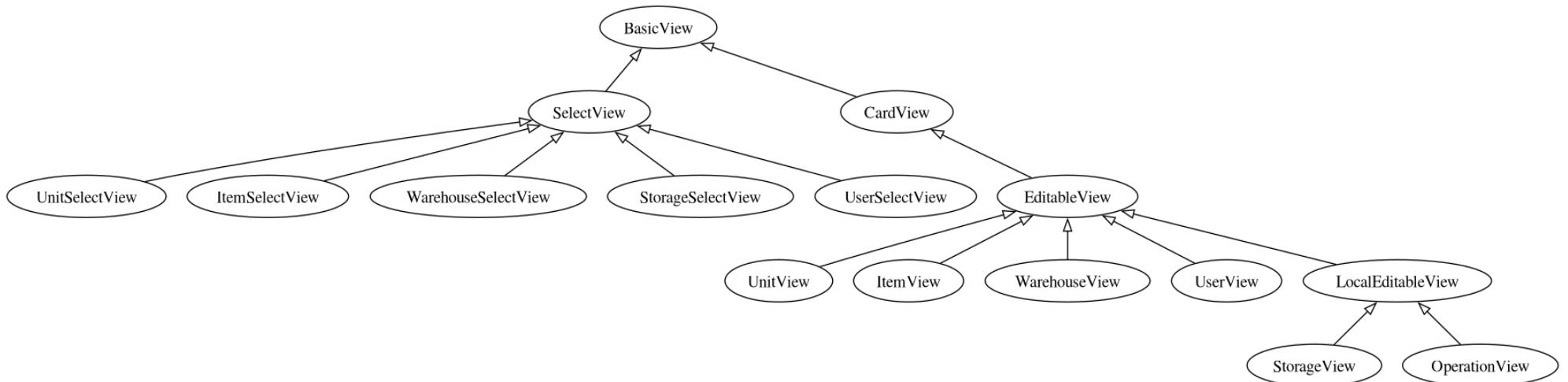
### 4.6.2 Asztali kliens osztályai

#### Worker osztály

A `Worker` osztály a háttérben futó folyamatok kezeléséért felelős.

#### EasyInventoryDesktop osztály

A program az `EasyInventoryDesktop` osztályban kezdődik, ahol a kezdő metódus betölti a nyelvfájlokat, beállítja az ablakot és elindítja a bejelentkezési felület létrehozását. Emellett ez az osztály állítja be az eseménykezelőket és hozza létre a bejelentkezés utáni felületet is.



Kép 4.3: Listázó osztályok diagramma

## BasicView osztály

A **BasicView** absztrakt osztály a különböző listázási felületek szülőosztálya. Ez kezeli a listázáshoz tartozó újratöltés, keresés és pagináció menüelemeket. A **reload** funkció betölti a lista elemeit a megadott paraméterek alapján (keresési szöveg, kezdő pozíció, mennyiség és archivált-e). Ezt a funkciót a **Worker** szála hívja meg. A **createEntry** funkció feladata, hogy vizualizálja a betöltött adatokat.

## SelectView osztály

A **SelectView** absztrakt osztály a különböző kiválasztási felületek szülőosztálya. Ez az osztály implementálja a **createEntry** funkciót, amely a **showInfo** meghívásáért felel, amely vizualizálja a betöltött adatot. A **showInfo**-nak nem kell kiválasztás gombot hozzáadnia, mivel azt a **createEntry** megteszi helyette.

## CardView osztály

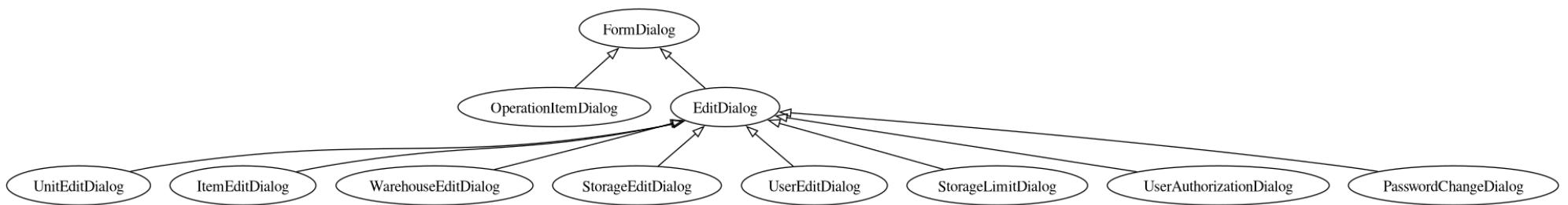
A **CardView** absztrakt osztály olyan listázási felületek szülőosztálya amelyek káryákban jelenítik meg az adatokat. A **createActionButtons** funkció feladata a akció gombot létrehozása. A **getArchivalUnixtime** funkció feladata az archiválás idejének visszaadása, **Long.MAX\_VALUE** ha nem archivált. A **showInfo** funkció feladata, hogy vizualizálja az adatot a kártyán belül.

## EditableView osztály

A **EditableView** absztrakt osztály a **CardView** osztályt szerkesztési gombokkal egészíti ki. Ez implementálja a **createActionButtons** funkciót szerkesztés és törlés gomb létrehozásával. A **showDialog** funkció feladata a szerkesztési és létrehozási dialógus létrehozása. A **delete** funkció feladata az adat törlése.

## LocalEditableView osztály

A LocalEditableView absztrakt osztály a EditableView osztályt telephely kiválasztási menüvel egészíti ki.



Kép 4.4: Dalókusztályok diagramma

## FormDialog osztály

A `FormDialog` absztrakt osztály a bemeneti felügrő ablakok szülőosztálya.

A `createEditFields` funkció a bemenetek lérhehozásáért felelős. Az `applyEdits` funkció a bemenet értelmezéséért felelős.

## EditDialog osztály

Az `EditDialog` absztrakt osztály a `FormDialog` osztályt mentéssel egészíti ki.

## 4.7 Mobil kliens

### 4.7.1 Áttekintés

A mobil alkalmazás forrás fájljai a `EasyInventoryMobile` mapában találhatóak, fejlesztésre pedig az Android Studio használható. A nyelvfájlokat a `langconvert.py` python program generálja. Az ikonok megegyeznek a webes frontend ikonjaival, viszont importálva lettek, így ikonmódosítás esetén érdemes újra importálni.

### 4.7.2 Osztályleírások

#### MainActivity osztály

A program a `MainActivity` osztályban kezdődik, ami megjeleníti a bejelentkezési felületet, illetve amennyiben még nincs, engedélyt kér a hálózat használatához.

```
public class MainActivity extends AppCompatActivity {
    public static easyinventoryapi.API api;
    public static String user_id, user_name;
    public static boolean isStopped() { ... }
    @Override
    protected void onCreate(Bundle savedInstanceState) { ... }
}
```

#### LoggedInActivity osztály

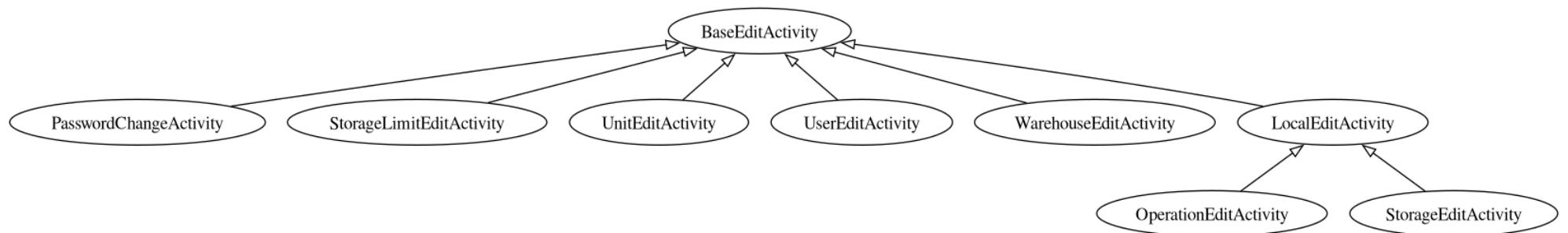
A `LoggedInActivity` osztály megjeleníti a bejelentkezett felületet. A különböző fülek kódjai a saját `Fragment` osztályaikban vannak.

```
public class LoggedInActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) { ... }
    public void setToolbarController(ToolbarController t) { ... }
    @Override
    public boolean onSupportNavigateUp() { ... }
    public interface ToolbarController { ... }
}
```

## Worker osztály

A **Worker** osztály a háttérben futó folyamatok kezeléséért felelős.

```
public class Worker extends Thread {  
    public static final Worker GLOBAL=new Worker();  
    @Override  
    public void run() { ... }  
    public void addTask(Runnable task) { ... }  
}
```



Kép 4.5: Szerkesztő osztályok diagramja

## BaseEditActivity osztály

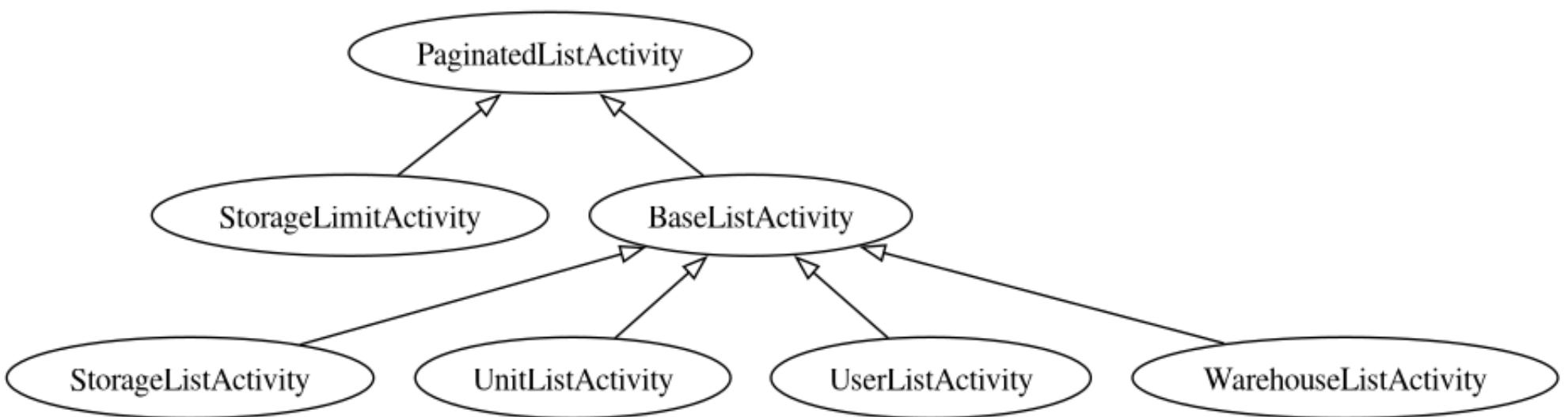
A `BaseEditActivity` absztrakt osztály a szerkesztési vagy hozzáadási tevékenységeknek a szülőosztálya. A `initUI` metódus feladata a felhasználói felület inicializálása. A `getItem` funkció feladata a bemeneti adat értelmezése. A `save` metódus feladata az adat mentése. A `readContext` funkció feladata a módosítandó adat kiolvassása a tevékenység adatokból.

```
public abstract class BaseEditActivity<T> extends AppCompatActivity
{
    public BaseEditActivity(int layout) { ... }
    protected abstract void initUI(@Nullable T item);
    protected abstract @NonNull T getItem();
    protected abstract void save(@Nullable T original, @NonNull T
        item) throws FormattedException;
    protected abstract T readContext(Intent i);
    protected void readExtraContext(Intent i) {}
    @Override
    protected void onCreate(Bundle savedInstanceState) { ... }
    @Override
    public boolean onSupportNavigateUp() { ... }
}
```

## LocalEditActivity osztály

A `LocalEditActivity` absztrakt osztály a `BaseEditActivity` osztályt telephely bekéréssel egészíti ki. Ilyen tevékenység indításánál meg kell adni a kiválasztott telephely azonosítóját a `warehouse_id` kulcsal.

```
public abstract class LocalEditActivity<T> extends BaseEditActivity
<T> {
    public LocalEditActivity(int layout) { ... }
    @Override
    protected void readExtraContext(Intent i) { ... }
    public String getSelectedWarehouseId() { ... }
}
```



Kép 4.6: Listázó activity osztályok diagramma

## PaginatedListActivity osztály

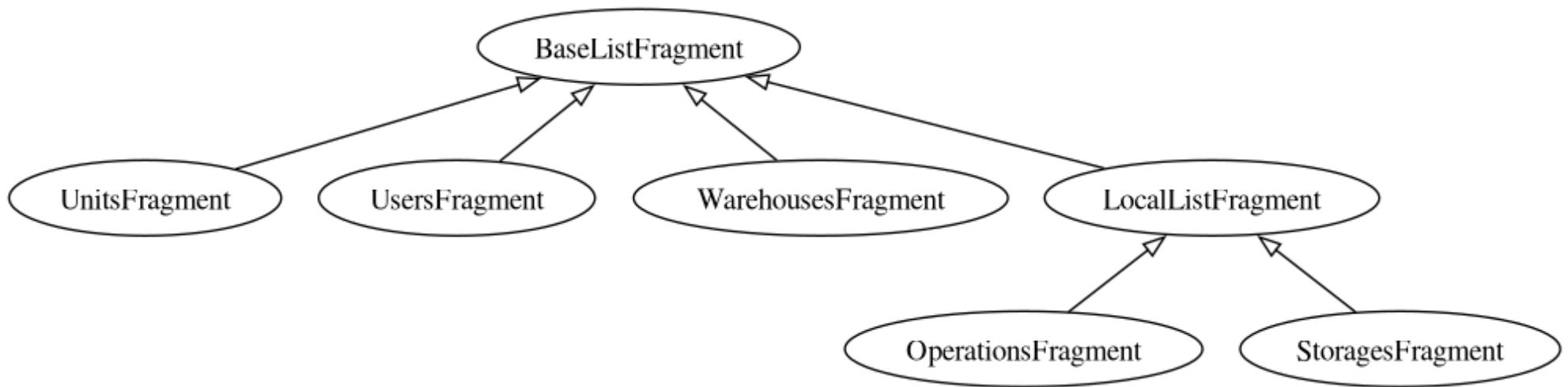
A `PaginatedListActivity` absztrakt osztály olyan tevékenységek szülőosztálya, amelyek lapozható listát mutatnak. A `load` funkció feladata, hogy a megadott paraméterek (keresési szüveg, kezdőpozíció, mennyiségek, archivált-e) alapján betöltsse az adatokat.

```
public abstract class PaginatedListActivity<T> extends
AppCompatActivity {
    public static final int PAGE_SIZE=20;
    protected int offset=0;
    protected SearchView search;
    protected SelectableArrayAdapter<T> adapter;
    protected SwipeRefreshLayout sw;
    protected PaginatedListActivity(int activity,Function<Activity,
        SelectableArrayAdapter<T>> adapter_factory) { ... }
    protected void reload() { ... }
    protected abstract T[] load(String query,int offset,int length,
        boolean archived) throws FormattedException;
    protected void readExtraContext(Intent i){}
    @Override
    protected void onCreate(Bundle savedInstanceState) { ... }
    @Override
    public boolean onSupportNavigateUp() { ... }
}
```

## BaseListActivity osztály

A `BaseListActivity` absztrakt osztály a kiválasztási felületek szülőosztálya. A `getNullDisplay` funkció feladata, hogy amennyiben a `load` funkció által visszaadott értékek között `null` érték is található, kicserélje azt a funkció visszatérési értékére.

```
public abstract class BaseListActivity<T extends ToJSON> extends
PaginatedListActivity<T> {
    protected BaseListActivity(Function<Activity,
        SelectableArrayAdapter<T>> adapter_factory) { ... }
    protected abstract T getNullDisplay();
    @Override
    protected void reload() { ... }
    protected abstract T[] load(String query,int offset,int length,
        boolean archived) throws FormattedException;
    @Override
    protected void onCreate(Bundle savedInstanceState) { ... }
}
```



Kép 4.7: Listázó fragment osztályok diagrammja

## BaseListFragment osztály

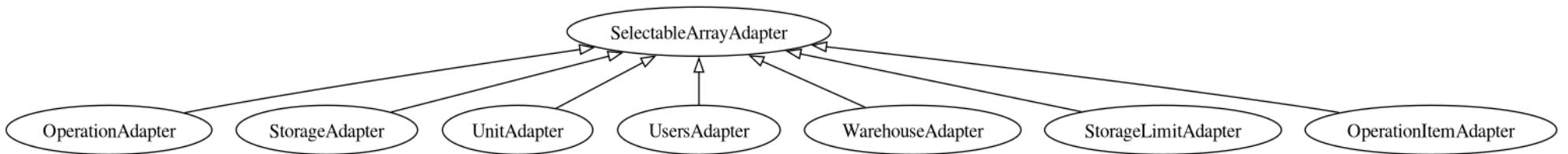
A `BaseListFragment` absztrakt osztály a lista `Fragment`-ek szülőosztálya, melynek feladata a lapozás, a keresés, az újratöltés, a törlés és a szerkesztés indítása. A `PaginatedListActivity` osztályhoz hasonlóan itt is megtalálható a `load` funkció. Az adat törlése a `delete` metódus feladata.

```
public abstract class BaseListFragment<T extends ToJSON> extends
    Fragment implements LoggedInActivity.ToolbarController {
    protected abstract T[] load(String query, int offset, int length,
        boolean archived) throws FormattedException;
    protected abstract void delete(String id) throws
        FormattedException;
    protected void reload() { ... }
    protected void addExtraContext(Intent i) {}
    protected boolean canAdd() { ... }
    protected void initRoot(View root, SelectableArrayAdapter<T>
        adapter, Function<T, String> id_getter, Class<?> edit_activity
    ) { ... }
    @Override
    public void controlToolbar(TextView title, ImageButton
        warehouse_button, SearchView search_bar) { ... }
    protected void startActivity(Class<?> activity) { ... }
    protected void startActivityForSelected(Class<?> activity) {
        ... }
}
```

## LocalListFragment osztály

A `LocalListFragment` absztrakt osztály a `BaseListFragment` osztályt telephely bekéréssel egészíti ki.

```
public abstract class LocalListFragment<T extends ToJSON> extends
    BaseListFragment<T> {
    @Override
    protected void addExtraContext(Intent i) { ... }
    @Override
    protected boolean canAdd() { ... }
    @Override
    protected void initRoot(View root, SelectableArrayAdapter<T>
        adapter, Function<T, String> id_getter, Class<?>
        edit_activity) { ... }
    @Override
    public void controlToolbar(TextView title, ImageButton
        warehouse_button, SearchView search_bar) { ... }
    public @Nullable Warehouse getSelectedWarehouse() { ... }
}
```



Kép 4.8: Osztálydiagramm

## SelectableArrayAdapter osztály

A `SelectableArrayAdapter` absztrakt osztály az  `ArrayAdapter` osztály implementálását segíti kijelölhető listákra.

# 5. Dokumentáció generálása

## 5.1 Áttekintés

A projekt dokumentációja L<sup>A</sup>T<sub>E</sub>X dokumentum leíró nyelvet használja. A forrás fájlok a `doc/src` mappában találhatók. A legtöbb stílus elem és a csomagok importálása a `shared.tex` fájllban van beállítva.

## 5.2 Felhasználói dokumentáció

A felhasználói dokumentáció a `user` mappában található. A dokumentum által használt ikonok a `user/resources/icon` mappában találhatóak. Az ikonok megváltoztatása esetén az itt található ikonokat is frissíteni kell. A `user/resources/web`, `user/resources/mobile` és `user/resources/desktop` mappákban található képeket a programok E2E tesztjei készítik. A programok kinézetének változtatásakor az itteni képeket érdemes frissíteni.

## 5.3 Fejlesztői dokumentáció

### 5.3.1 Kód dokumentáció

A kód dokumentáció a `code` mappában találhatóak. Az dokumentum által használt kód részek a `code/generated` mappában találhatóak. Ezeket a kód részeket az építő program autómatikusan generálja.

## 5.3.2 Tesztdokumentáció

A teszt dokumentáció a `test` mappában, a dokumentum által használt kód részletek pedig a `test/generated` mappában találhatóak. Ezeket a kód részleteket az építő program autómatikusan generálja.

## 5.3.3 Kód beillesztő

A kód beillesztő program a `codeextractor.py` python program.

# 5.4 Építés

A dokumentáció építéséhez a `compile` fájlt kell futtatni. Ez generált pdf fájlokat az `out` mappába helyezi.

# 6. Tesztdokumentáció

Fontos!

A tesztelő rendszerek teljesen átírják az adatbázist.

## 6.1 API szerver

### 6.1.1 Áttekintés

Az API szerver teszteléséért az `apitest` mappában található python fájlok felelősek. A tesztek lefuttatása előtt fel kell állítani a teszt API szervert. Első lépésként az API szerver konfigurációs fájljában (`config.json`) kell beállítani a teszt tokenet, majd ugyan ezt el kell végezni a teszt konfigurációs fájlban is. A sikeres beállítás után a `main.py` futtatásával indíthatók el a tesztek.

### 6.1.2 E2E tesztek

Az E2E tesztek az API-t hívják és ellenőrzik a válasz kódot és adatot. A tesztelés megkezdése előtt az rendszer visszatölt egy megadott biztonsági mentést, ami tartalmazza a tesztadatokat.

## 6.2 Web frontend

### 6.2.1 Áttekintés

A webes frontend tesztelésére csupán E2E autómata teszteket használunk, mivel ezek kellően letesztelek azt, így nincs szükség további tesztekre.

### 6.2.2 E2E tesztek

Az E2E tesztekhez python alapú playwright keretrendszert használunk. Az E2E tesztek minden műveletet végrehajtanak kattintás és gépelés szimulációval. Helyes és hibás bemeneti adatok egyaránt tesztelve vannak, illetve a tesztek egy része képernyő képeket készít a program működéséről, amiket a dokumentációhoz használunk fel.

## 6.3 Asztali alkalmazás

### 6.3.1 Áttekintés

Az asztali alkalmazás a TestNG tesztelési keretrendszer használja.

### 6.3.2 E2E tesztek

Az asztali alkalmazás tesztelése olyan E2E teszteket történik, amelyek felhasználói interakciókat szimulálnak és megjelenített szövegeket ellenőriznek. A felhasználói felület elemeinek tesztelhetősége érdekében az elemek CSS osztályokkal vannak ellátva. A tesztek egy része képernyőképeket készít a dokumentációhoz.

### 6.3.3 NodeMatcher osztály

A `NodeMatcher` osztály feladata a felhasználói felület elemeinek kiválasztása és az interakció szimulálása a kiválasztott elemekkel.

```
public class NodeMatcher {
    public static NodeMatcher allWindowRoots() { ... }
    public NodeMatcher descendants() { ... }
    public NodeMatcher withClass(String classname) { ... }
    public NodeMatcher labels(String text) { ... }
    public NodeMatcher textFields() { ... }
    public NodeMatcher tabPanes() { ... }
    public NodeMatcher buttons() { ... }
    public NodeMatcher with(Function<NodeMatcher,NodeMatcher> m) {
        ...
    }
    public static void sync(Runnable r) { ... }
    public Node get() { ... }
    public int count() { ... }
    public void replaceText(String text) { ... }
    public void replaceNumber(int value) { ... }
    public void click() { ... }
    public void clickASync() { ... }
    public void selectTab(int index) { ... }
    public void exists() { ... }
    public void doesNotExist() { ... }
    public void acceptDialog() { ... }
    public void closeDialogs() { ... }
    public void print() { ... }
    public void screenshotWindow(String out) { ... }
}
```

## allWindowRoots

Az `allWindowRoots` funkció létrehoz egy `NodeMatcher` objektumot amiben minden ablak gyökér eleme ki van választva.

## descendants

A `descendants` metódus létrehoz egy `NodeMatcher` objektumot amiben azok az elemek vannak kijelölve amik legalább egy kijelölt elem leszármazottja.

## with

A `with` metódus bekér egy lambda függvényt, amellyel egy `NodeMatcher` objektumot egy másik `NodeMatcher` objektummá alakít át. Erre a folyamatra azért van szükség, hogy a metódus le tudja szűrni a kijelölt elemeket és elérni, hogy csak olyan kiválasztást adjon vissza, amelyben legalább egy elem található. A függvény bemenete minden esetben egy olyan kijelölés, amiben megtalálható az az elem, amit éppen szűr. A leszűrés végén a metódus a kijelölést egy új `NodeMatcher` objektum formájában kiadja. Ez például arra használható ha csak azokat az elemeket akarjuk amikben van egy bizonyos címke.

## withClass

A `withClass` metódus egy olyan `NodeMatcher` objektumot hoz létre, amelyben csak azok az elemek vannak kiválasztva, amelyek el vannak látva a megadott CSS stílussal.

## click

A `click` metódus kattintást szimulál a kiválasztott gombon. Ha nem gomb van kiválasztva vagy nem egy elem van kiválasztva akkor hibát dob.

## replaceText

A `replaceText` metódus szöveg beírását szimulálja a kiválasztott bemeneti mezőn. Ha nem bemeneti mező van kiválasztva vagy nem egy elem van kiválasztva akkor hibát dob.

## screenshotWindow

A `screenshotWindow` metódus a kijelölt elem ablakáról készít egy képernyőképet és a megadott névvel elmenti. Ha egy elem van kiválasztva akkor hibát dob.

## 6.4 Mobil alkalmazás

### 6.4.1 Áttekintés

A mobil alkalmazás tesztelése J4Unit teszt keretrendszerrel történik.

### 6.4.2 E2E tesztek

A mobil alkalmazás tesztelése olyan felhasználói felület tesztekkel történik, amellyek kattintásokat és szövegek beírását szimulálják. Ezek a tesztek ugyanazt tesztelik, amit az asztali alkalmazás E2E tesztjei, annyi különbösséggel, hogy ezek az **Espresso** UI tesztelési keretrendszerét használják a felhasználói interakciók szimulálására. A tesztekkel helyes és hibás beviteli adatok is egyaránt tesztelve vannak, illetve a tesztek egy része képernyőképeket készít, amelyeket a dokumentációhoz használunk fel.

## II. FELHASZNÁLÓI DOKUMENTÁCIÓ

---

# 7. Felhasználói dokumentáció

## 7.1 Bevezetés

### 7.1.1 Fogalmak

#### Telephelyek

A telephelyek különböző telkek vagy raktárépületek, amelyek területén különböző raktárak találhatóak.

#### Raktárak

A raktárakon belül találhatóak a különböző termékek, így a raktár számít a legkissem tároló egységnek. A raktárak számára meg lehet adni, hogy mennyi fér beléjük az egyes cikkekből.

#### Műveletek

A műveletek segítségével tudunk cikkeket hozzáadni vagy éppen eltávolítani a raktárakból. Mint arra a korábbi mondat is utalt, két féle művelet létezik: a hozzáadási és az eltávolítási.

##### Hozzáadási műveletek

A hozzáadási művelet használatával különböző cikkeket tudunk felvinni egy adott raktár nyilvántartásába. Létrehozásakor lehetőségünk van különböző követelmények megadására is, mellyekkel szabályozhatjuk, hogy hová kerülnek a hozzáadásra szánt cikkek.

### Eltávolítási műveletek

Az eltávolítási művelet segítségével lehetőségünk van cikkek kivételére a hozzájuk tartozó raktárból. Létrehozásakor a hozzáadási művelethez hasonló képpen itt is lehetőségünk van szabályozni, hogy melyik raktárból és melyik cikkek kerüljenek eltávolításra, amit meghatározhatunk lot-, vagy akár sorozatszám megadásával is.

### Lot számok

A lot számok különböző cikk csoportok megkülönböztetésére szolgáló kódok, amelyek nagyobb nyílvántartások kezelése esetén nélkülözhetetlenek.

### Globális sorozat szám

A globális sorozatszám egy olyan szám, amelynek az egész rendszeren belül egyedinek kell lennie, még az azonos árucikkek között is.

### Gyártói sorozat szám

Minden hozzáadott árucikkhez hozzárendelhetünk egy gyártói sorozatszámot. Ezeknek a számoknak nem feltétlenül kell egyedinek lenniük, mivel több gyártó esetén, az árucikkek sorozatszámai akár ütközhetnek is.

## 7.2 Telepítés

### 7.2.1 Szerver telepítés

#### Linux mint

A szerver telepítéséhez le kell futtatni a telepítő programot:

```
chmod +x install.sh && ./install.sh
```

Ez a parancs telepíti a szükséges csomagokat, illetve bemásolja a frontendet és a szervert a `/var/www/html/` mappába. Ezt követően be tudjuk állítani a szervert az `api/config.json` fájlban.

#### Más linux alapú rendszerek

A szerver működéséhez a következő programok szükségesek:

- php
- php-mysqli
- apache
- libapache2-mod-php
- mariadb-server

A szerver telepítéséhez az `api` és a `frontend/assets` mappát, illetve a `frontend/index.html` fájlt át kell másolni az apache web mappába. Ezt követően a mysql szerveren létre kell hozni egy felhasználót, majd beállítani azt. Végezetül pedig, az apache szerveren engedélyezni kell a `.htaccess` fájlokat, majd be kell kapcsolni a `rewrite` és a `PHP` modulokat.

## Windows

A szerver Windowsra való telepítéséhez először a XAMPP-ot kell telepíteni, majd át kell másolni az `api` és a `frontend/assets` mappát, illetve a `frontend/index.html` fájlt a htdocs mappába.

### 7.2.2 Asztali kliens telepítés

Az asztali kliens fájljai az `EasyInventoryDesktop/dist/bundles/EasyInventoryDesktop` mappában találhatóak. Ezt a mappát bárhová át lehet másolni és nem igényel telepítőt, mivel mined szükséges dll -t tartalmaz a mappában. Indításhoz a benne található exe fájlt kell futtatni.

### 7.2.3 Mobil kliens telepítés

A mobil kliens az Android Studió alátal létrehozott apk fájl futtatásával telepíthető Android 14-re.

## 7.3 Adminisztráció

### 7.3.1 Arhivált adatok törlése

Fontos!

Az arhivált adatok törlése visszafordíthatatlan! Ezek az adatok adják a statisztikákat. Régi adatok törlésével a velük egyidejű statisztika is törlődik.

Az arhivált adatok törlése hasznos lehet ha az adatbázis mérete már túl nagy.

Az összes adat törlésére az alábbi parancsot kell futtatni:

```
php archive delete all
```

Általában nem akarjuk az összes arhivált adatot törölni. Ebben az esetben meg kell adnunk, hogy hány évnyi, heti vagy napi adatot szeretnénk megtartani. Ezt az adott időegység jelével tudjuk megtenni.

**d** nap

**w** hét

**y** év

Például: ez a parancs kitörli az összes arhivált adatot, ami 5 évnél régebben lett törölve.

```
php archive delete 5y
```

### 7.3.2 Eseménynapló kezelése

Az eseménynapló sok hasznos információt tárol a szerver működéséről. Ezekhez az információk segíthetnek a hibás beállítások javításán. Az eseménynaplót a parancssoron keresztül kezelhetjük.

## Eseménynapló megtekintése

Az eseménynapló megjelenítését a `log show` parancs használatával érhetjük el. Abban az esetben, ha nem adunk meg semmilyen kapcsolót, az összes adat rövid formátumban lesz kiírva.

```
php log show
```

### **-u, --user kapcsoló**

Ennek a segítségével megadhatjuk azt a felhasználói azonosítót, amelyhez tartozó eseményeket kívánjuk megtekinteni.

### **-e, --error kapcsoló**

Szerverhibát okozó kérésekre szűri le az eseményeket. Ezek lehetnek adatbázis kapcsolódási hibák is.

### **-r, --rejected kapcsoló**

Elutasított kérésekre szűri le az eseményeket.

### **-b, --bad kapcsoló**

Hibás kérésekre szűri le az eseményeket.

### **-s, --success kapcsoló**

Sikeres kérésekre szűri le az eseményeket.

### **-v, --verbose kapcsoló**

Az eredményeket hosszú formában jeleníti meg.

## Pédák

Megmutatja az összes kérést, amit az `admin` felhasználó végzett:

```
php log -u admin show
```

Megmutatja az összes szerverhibát és elutasított kérést:

```
php log -er show
```

Részletesen megmutatja az összes szerverhibát:

```
php log -ev show
```

## Egy esemény megtekintése

Ha csak egy kérést akarunk megtekinteni akkor a kérés sorszámának megadásával tudjuk. Abban az esetben, ha csak egy kérést akarunk megtekinteni, akkor a kérés sorszámának megadásával tudjuk azt megtenni. Ehhez a kéréshez nem tartoznak kapcsolók.

```
php log show 100
```

## Eseménynapló törlése

Ez a parancs törli a teljes eseménynapló tartalmát:

```
php log clear
```

Legtöbbször nem akarjuk az összes eseményt törölni. Ebben az esetben meg kell adnunk, hogy hány évnyi, heti vagy napi adatot szeretnénk megtartani. Ezt az adott időegység jelével tudjuk megtenni.

**d** nap

**w** hét

**y** év

Például: ez a parancs kitörli az összes eseményt, ami 5 napnál régebbi.

```
php log clear 5d
```

## 7.3.3 Felhasználó kezelés

A parancssorról a felhasználóknak új jelszót tudunk adni, felhasználókat tudunk törölni és új adminisztrátorokat létrehozni.

### Adminisztrátor létrehozása

Új adminisztrátor létrehozásához meg kell adnunk az azonosítóját, a nevét és a jelszavát

```
php user create admin Admin password123
```

## Felhasználó törlése

Felhasználó törléséhez meg kell adnunk az azonosítóját. Ez a művelet sikertelen ha az adott felhasználó más felhasználók főnöke.

```
php user delete admin
```

## Új jelszó adása

Az új jelszó adáshoz meg kell adni a felhasználó azonosítóját és az új jelszót.

```
php user password admin password123
```

### 7.3.4 Engedélyek

Két féle engedély típus létezik: a rendszer és a helyi. A rendszer engedélyek az egész rendszerre vonatkoznak, tehát, amennyiben egy felhasználó egy ilyen engedélyt kap, akkor az minden telephelyen érvényes. Ezzel ellentétben a rendszerengedélyek között a rendszer adminisztrációjával kapcsolatos engedélyek is találhatóak. A helyi engedélyek telephelyhez vannak kötve, tehát, amennyiben egy felhasználó egy ilyen engedélyt kap, abban az esetben az csupán az adott telephelyen érvényes.

## Rendszer engedélyek

- Telephelyek megtekintése
- Telephelyek hozzáadása
- Telephelyek módosítása
- Telephelyek törlése
- Típusok hozzáadása
- Típusok módosítása
- Típusok törlése
- Felhasználók megtekintése
- Felhasználók hozzáadása
- Felhasználók módosítása
- Felhasználók törlése

## Helyi engedélyek

- Raktárak kezelése
- Hozzáadó művelet létrehozása
- Eltávolító művelet létrehozása
- Műveletek visszaigazolása

### 7.3.5 Beállítások

A `config.json` konfigurációs fájl a szerver beállításait tartalmazza.

## Adatbázis beállítások

A **database** szekció az alkalmazás adatbázis-kapcsolatához szükséges paramétereket tartalmazza.

**host** Az adatbázis szerver címe.

**user** Az adatbázis felhasználó neve.

**password** Az adatbázis felhasználó jelszava.

**database\_name** Az adatbázis neve, amelyhez csatlakozik az alkalmazás (például: easyinventory).

**port** Az adatbázis portja, amelyen a kapcsolatot létesíteni kell.

## Token lejárati idő

A **token\_expiration** kulcs határozza meg, hogy hány másodpercig érvényes egy generált token. Ez az érték alapértelmezetten 120 másodperc, tehát 2 perc.

## Teszt token

A **test\_token** kulcs egy fix token-t ad meg, amelyet a rendszer teszteléséhez használhatunk. Ezt normál futtatáskor **null**-ra kell állítani.

## Naplózási beállítások

A **logging** szekció az alkalmazás különböző naplózási beállításait tartalmazza. Négy fő típusú naplózás van: **success**, **bad**, **rejected**, és **error**. Mindegyik típusnak van négy különböző beállítása a következő mezők szerint: **request\_body**, **response\_body**, **debug**, és **error**.

**success** Sikeres kérések naplózása

**bad** Hibás kérések naplózása

**rejected** Elutasított kérések naplózása

**error** Hiba naplózás

**request\_body** A kérés törzsének naplózása

**response\_body** A válasz törzsének naplózása

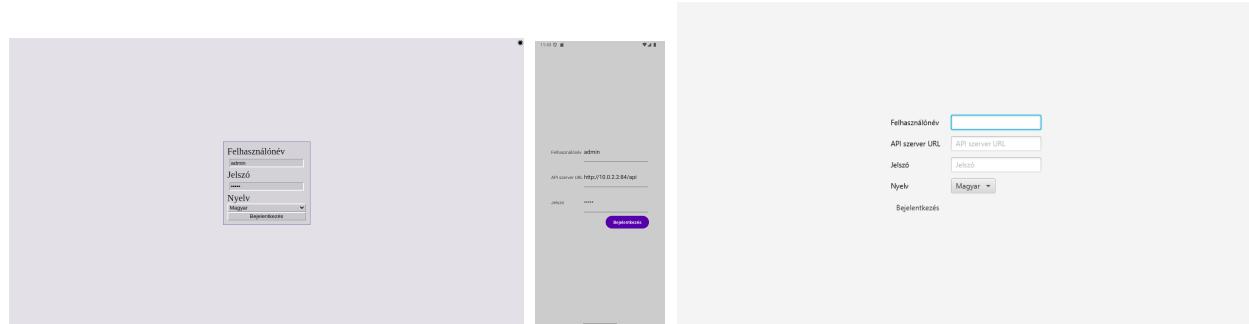
**debug** A hibakeresési információk naplózása

**error** A hibák naplózása

## 7.4 Használat

### 7.4.1 Bejelentkezés

A program indításakor először a bejelentkező felületre fogad minket, ahol a bejelentkezéshez meg kell adnunk a felhasználónevét, a API szerver url -jét, a jelszót és a használni kívánt nyelvet (angol/magyar). A megfelelő adatok megadása után, a Bejelentkezés vagy Login gombra kattintva, amennyiben a megadott adatok helyesek, továbbjutunk a kezelőfelületre. A lap bal felső sarkában láthatjuk az alkalmazást használó személy nevét, illetve a mellette található gombal ki is jelentkezhetünk a programból. A lap tetején elhelyezkedő menüelemek segítségével választhatjuk ki, hogy az adatbázis mely részét szeretnénk megttekinteni vagy módosítani.



#### 7.4.2 Felhasználói fiókok kezelése

A Felhasználók menüpont segítségével megtekinthetjük az összes felhasználót, aki hozzáféréssel rendelkezik az alkalmazáshoz. minden felhasználó mellett két gomb található, amelyek segítségével: Módosíthatjuk a felhasználó adatait (név, felhasználónév, jogosultságok, jelszó) vagy törölhetjük a felhasználót. A lap alján található gombokkal: Frissíthetjük az oldalt vagy új felhasználót regisztrálhatunk.

## Felhasználói fiókok hozzáadása

Felhasználói fiók hozzáadásához a **+** gombra kell kattintanunk, majd a megjelenő bemeneti ablak kitöltése után, a **✓**-ra kattintva tudjuk elmenteni azt.

Termékek Egyések Felhasználók Keresés Műveletek Telephelyek Raktárak Profil Statisztikák		• admin	
<b>admin</b> admin  = ☰ ♀	<b>Alfonso Sloan</b> alfonso_sloan1973 Patty Bewser  = ☰ ♀	<b>Alice Ford</b> alice_ford1979 Jenny Dillon  = ☰ ♀	<b>Alison Franklin</b> alison_franklin1978 Mario Rivera  = ☰ ♀
<b>Alonzo Nicholson</b> alonzo_nicholson Morton Hunter  = ☰ ♀	<b>Alphonse M</b> alphons_m1974 Luanne Melan Nev Menedzser Jelző  X	Felhasználók Felhasználónév Név Menedzser Jelző  X	<b>Amelia Valentine</b> amelia_valentine1977 Lacy French  = ☰ ♀
<b>Andrea Ward</b> andrea_ward1976 Brinice Odsmell  = ☰ ♀	<b>Andrew Finney</b> andrew_finney2004 Les Howze  = ☰ ♀	<b>Angel Roman</b> angel_roman1961 Lizette Singh  = ☰ ♀	<b>Annabelle Levy</b> annabelle_levy1971 Chris Myers  = ☰ ♀
<b>Antone Juarez</b> antone_juarez1965 Davis Hunt  = ☰ ♀	<b>Arron Silva</b> arron_silva1980  = ☰ ♀	<b>Barbara McConnell</b> barbara_mcconnell1977 Pearlie Frey  = ☰ ♀	<b>Barney Adkins</b> barney_adkins1973 Liliana Levine  = ☰ ♀

E - Felhasználat szerkesztése ✓

Felhasználónév: anna-1155287246

Név: Anna

Jelző:

Menedzser: Működőkörök felhasználó

**Felhasználónév**

**Név**

**Jelszó**

**Menedzser**

Nincs kiválasztott felhasználó

OK
Cancel

## Felhasználói fiókok módosítása

A felhasználó adatait a gombra való kattintással módosíthatjuk. Ezután a megjelent adatlap ki lesz töltve a mentett értékekkel. A módosítás után a gombra kattintással lehet menteni.

## Felhasználó engedélyeinek módosítása

A felhasználó engedélyeit a gombra való kattintással módosíthatjuk. Ilyenkor feljön egy ablak amiben az összes engedély ki van listázva. Itt a gombbal vehetünk el és a gombbal adhatunk engedélyt.

## Felhasználói fiókok törlése

A felhasználót a gombra való kattintással törölhetjük.

## 7.4.3 Egységek kezelése

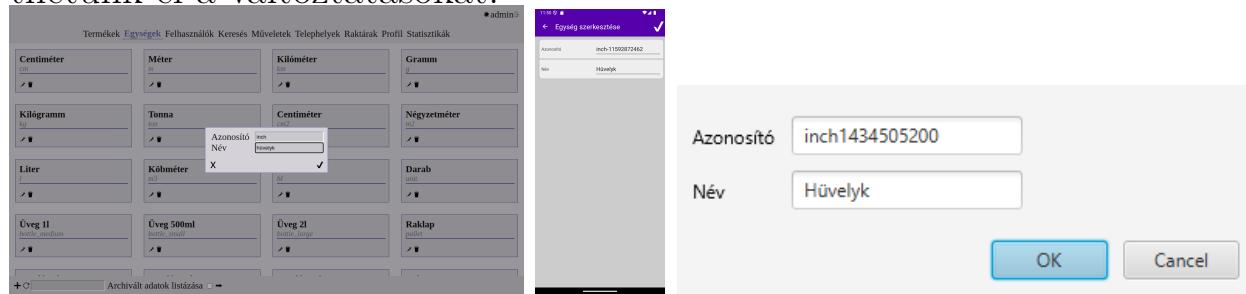
Az Egységek menüpont segítségével megtekinthetjük, módosíthatjuk, illetve törölhetjük a nyilvántartásban szereplő, mérésre szolgáló egységeket. A lap alján itt is megtalálhatóak a frissítés és a hozzáadás gombok, amelyekkel frissíthetjük az oldalt, illetve a megfelelő adatok megadásával újabb egységgel bővíthetjük az adatbázist.

## Egységek hozzáadása

Egy egységet a **+** gombra való kattintás után megjelenő bemeneti ablak kitöltésével, majd a **✓** gombra való kattintással hozhatunk létre.

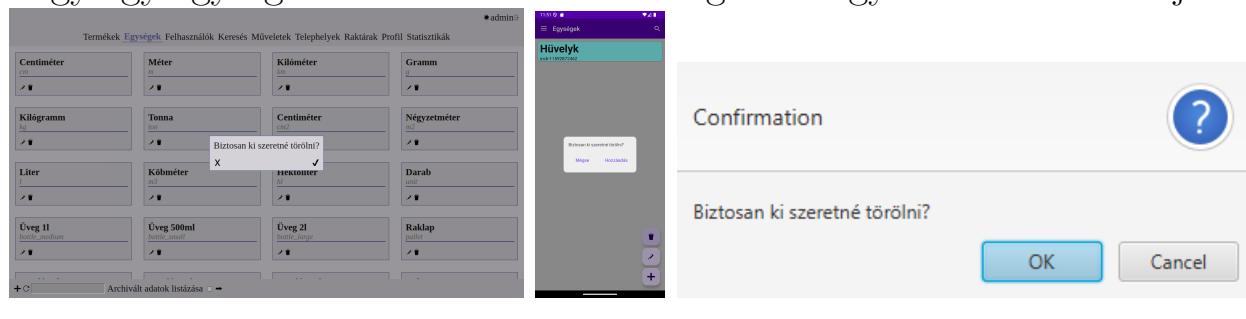
## Egységek módosítása

Egységeket a gombra való kattintással módosíthatunk. A kattintást követően megjelennek az adatok egy felugró ablakban, ahol könnyedén módosíthatjuk is azokat. A módosítás után a gombra való kattintással menthetünk el a változtatásokat.



## Egységek törlése

Egységeket a gombra való kattintással törölhetünk. Fontos megjegyezni, hogy egy egységet nem törölhetünk ha legalább egy cikk azt használja.



## 7.4.4 Árucikkek kezelése

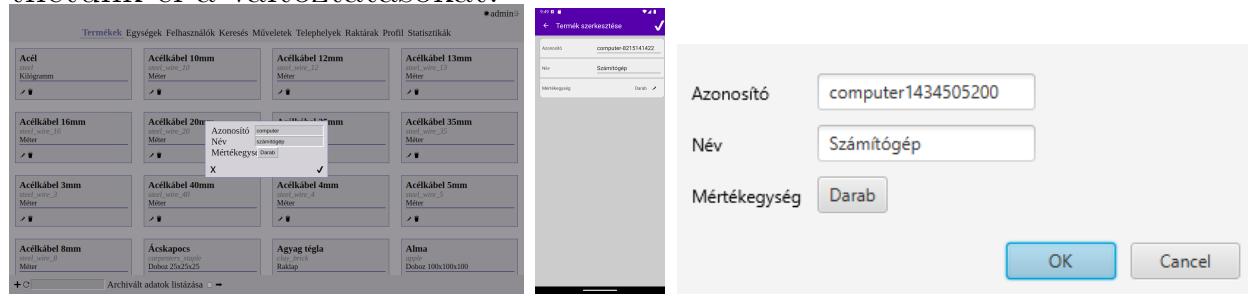
Az árucikkek menüpont segítségével megtekinthetjük, módosíthatjuk, illetve törölhetjük a nyilvántartásban szereplő, árucikkeket. A lap alján itt is megtalálhatóak a frissítés és a hozzáadás gombok, amelyekkel frissíthetjük az oldalt, illetve a megfelelő adatok megadásával újabb cikkekkel bővíthetjük a rendszert.

## Árucikkek hozzáadása

Egy árucikket a **+** gombra való kattintás után megjelenő bemeneti ablak kitöltésével, majd a **✓** gombra való kattintással hozhatunk létre.

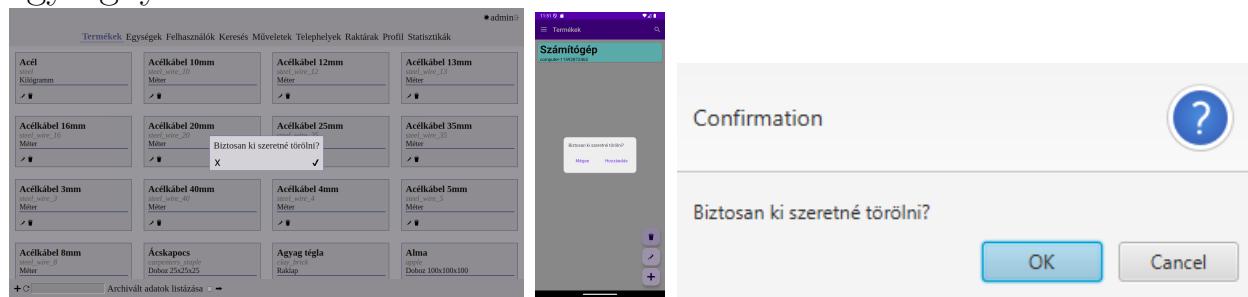
## Árucikkek módosítása

Árucikkeket a gombra való kattintással módosíthatunk. A kattintást követően megjelennek az adatok egy felugró ablakban, ahol könnyedén módosíthatjuk is azokat. A módosítás után a gombra való kattintással menthetünk el a változtatásokat.



## Árucikkek törlése

Árucikkeket a gombra való kattintással törölhetünk. Fontos megjegyezni, hogy egy árucikket nem törölhetünk, ha az adott cikkből akár csak egy egységnyi is tárolva van a rendszerben.



## 7.4.5 Telephelyek kezelése

Az telephelyek menüpont segítségével megtekinthetjük, módosíthatjuk, illetve törölhetjük a nyilvántartásban szereplő, telephelyeket. A lap alján itt is megtalálhatóak a frissítés és a hozzáadás gombok, amelyekkel frissíthetjük az oldalt, illetve a megfelelő adatok megadásával újabb telephellyel bővíthetjük a rendszert.

The screenshot shows two parts of the application interface. On the left, a grid of telephone entries with columns for name, address, and phone number. On the right, a detailed view of a specific entry ('Ajkai telephely') with fields for address and phone number, and a 'Műveletek' (Actions) section containing 'Frissítés' (Update), 'Töröl' (Delete), and 'Hozzáadás' (Add). Below these are tabs for 'Feltárolások', 'Telephelyek', 'Termékek', 'Egyégek', 'Raktárak', 'Keresés', 'Műveletek', and 'Profil'. On the far right, there is a separate window titled 'Telephelyek' showing a list of telephone entries with columns for name, address, and phone number, along with edit and delete icons.

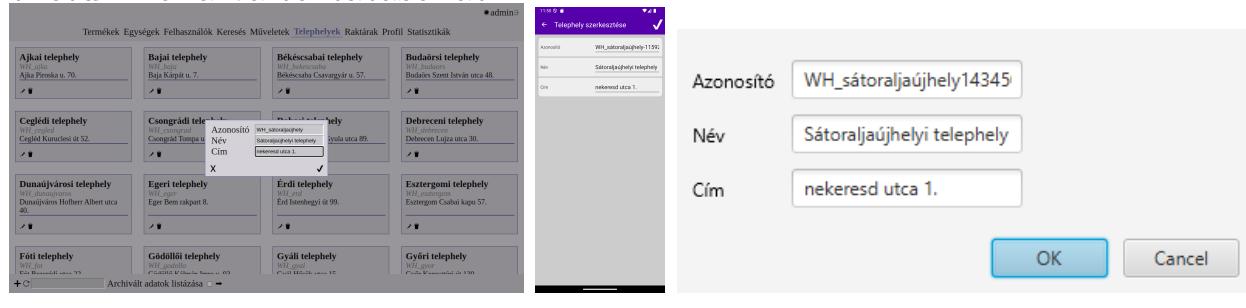
## Telephelyek hozzáadása

Egy telephelyet a **+** gombra való kattintás után megjelenő bemeneti ablak kitöltésével, majd a **✓** gombra való kattintással hozhatunk létre.

The screenshot shows the 'Telephelyek' section with a new entry creation dialog open. The dialog has fields for 'Azonosító' (Identifier), 'Név' (Name), and 'Cím' (Address). The 'Azonosító' field contains 'WH\_toronytelephely-1192'. The 'Név' field contains 'Sátortársi telephely'. The 'Cím' field is empty. At the bottom are 'OK' and 'Cancel' buttons. To the left of the dialog, the main 'Telephelyek' list is visible.

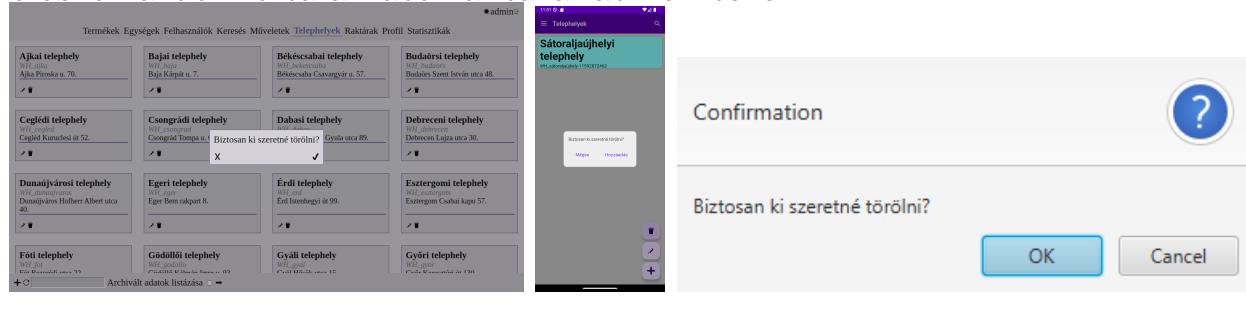
## Telephelyek módosítása

Telephelyeket a  gombra való kattintással módosíthatunk. A kattintást követően megjelennek az adatok egy felugró ablakban, ahol könnyedén módosíthatjuk is azokat. A módosítás után a  gombra való kattintással menthetünk el a változtatásokat.



## Telephelyek törlése

Telephelyet a  gombra való kattintással törölhetünk. Fontos megjegyezni, hogy amennyiben található benne raktár, a telephelyet nem törölhetjük, így először a benne található raktárakat kell törölni.



## 7.4.6 Raktárak kezelése

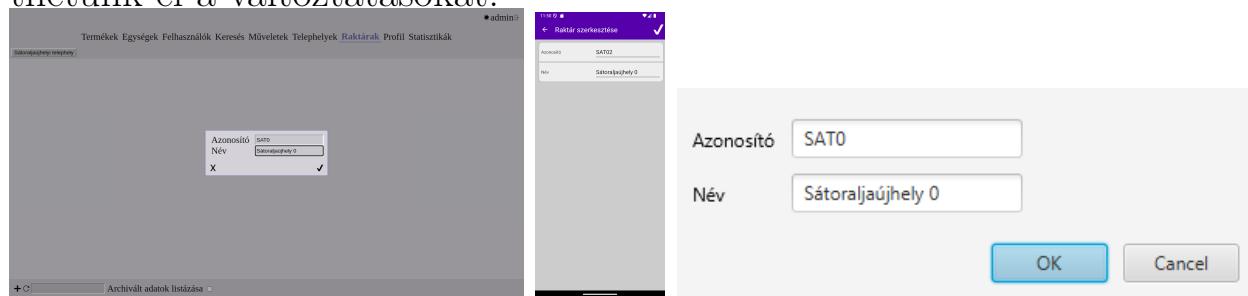
A Raktárak menüpontra kattintva a program megmutatja számunkra a raktárakat telephelyenként szétválogatva. A menüsor alatt található lenyíló listával tudjuk kiválasztani, hogy melyik telephelyhez tartozó raktárakat szeretnénk megtekinteni, majd a raktárak neve mellett található gombokkal módosíthatjuk és törölhetjük azokat. A lap alján elhelyezkedő, hozzáadás gombbal pedig új raktárt adhatunk hozzá a kiválasztott telephelyhez.

### Raktárak hozzáadása

Egy raktárt a **+** gombra való kattintás után megjelenő bemeneti ablak kitöltésével, majd a **✓** gombra való kattintással hozhatunk létre.

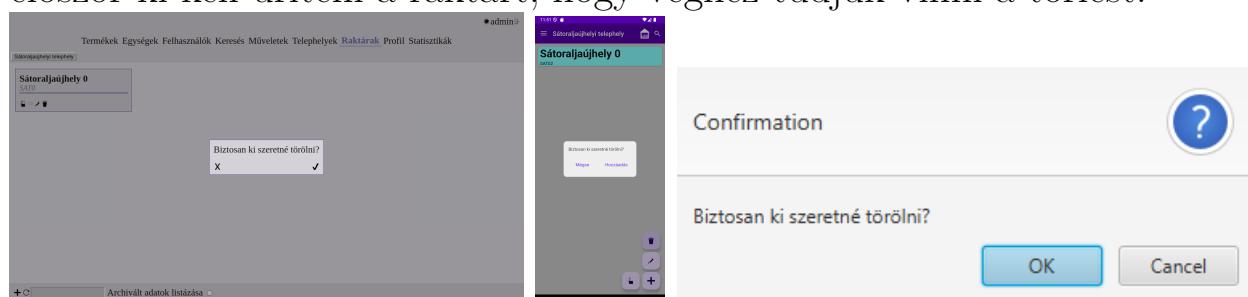
## Raktárak módosítása

Raktárakat a gombra való kattintással módosíthatunk. A kattintást követően megjelennek az adatok egy felugró ablakban, ahol könnyedén módosíthatjuk is azokat. A módosítás után a gombra való kattintással menthetünk el a változtatásokat.



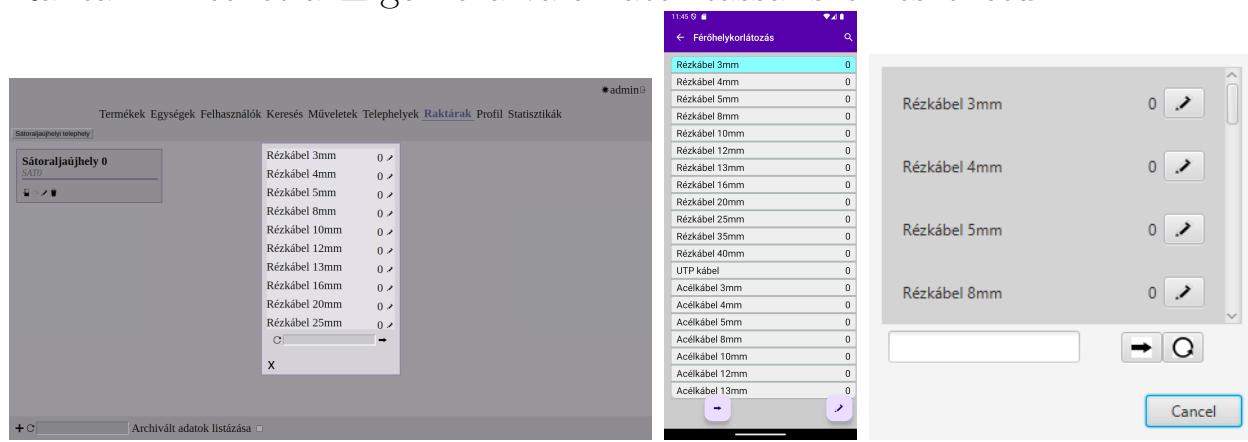
## Raktárak törlése

Raktárakat a gombra való kattintással törölhetünk. Fontos megjegyezni, hogy egy raktárt nem törölhetünk addig, amíg található benne áru, tehát először ki kell üríteni a raktárt, hogy véghez tudjuk vinni a törlést.



## Raktár limitek

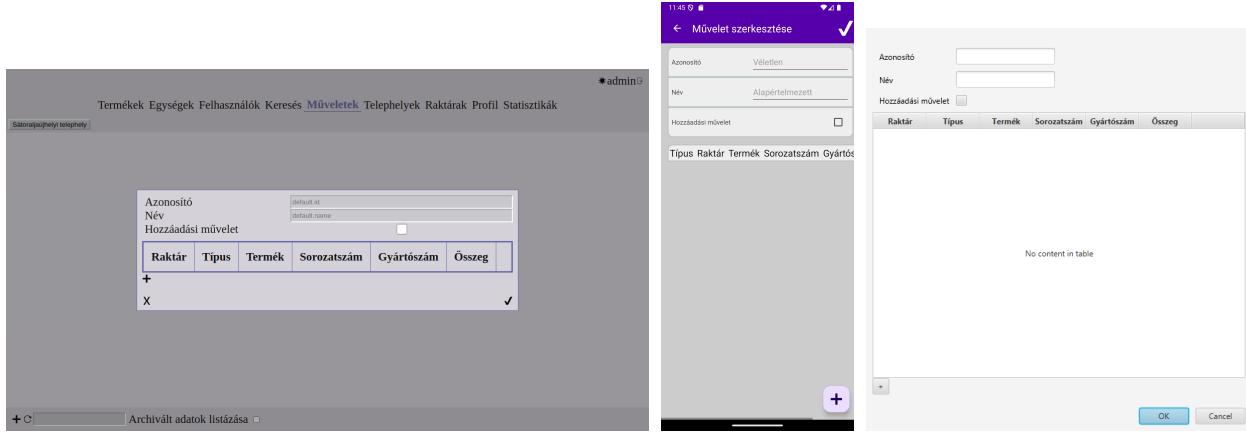
Raktár limiteket a gombra való kattintással szerkeszthetünk.



## 7.4.7 Műveletek

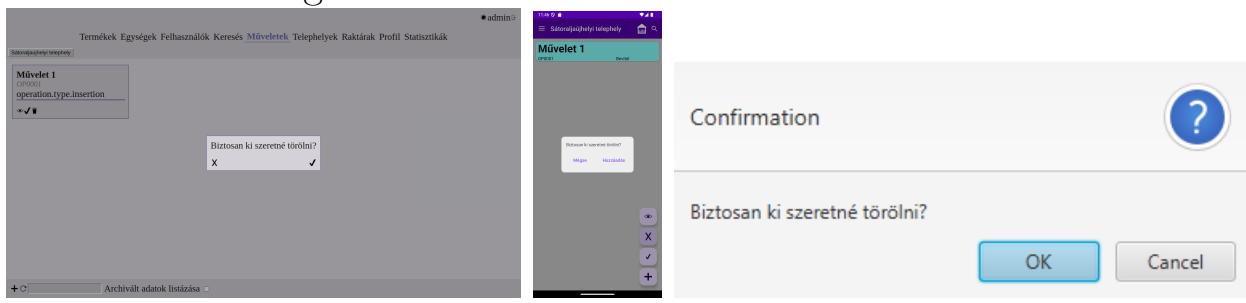
### Műveletek hozzáadása

Egy műveletet a **+** gombra való kattintás után megjelenő bemeneti ablak kitöltésével, majd a **✓** gombra való kattintással hozhatunk létre.



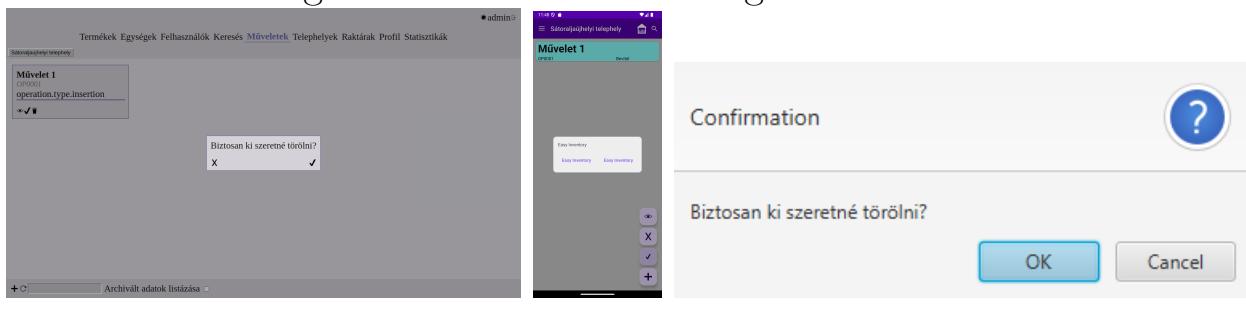
### Műveletek törlése

Műveleteket a **─** gombra való kattintással törölhetünk.



### Műveletek visszaigazolása

Műveleteket a **✓** gombra való kattintással igazolhatunk vissza.



## 7.4.8 Keresés

A Keresés használatával egyszerűen rá tudunk keresni bármilyen termékre. A termék nevének beírása után a program visszaadja, hogy melyik telephelynek, melyik raktárában található meg az adott cikk. Emellett további információkat is találhatunk az adott cikkről, mint például a rendelkezésre álló mennyiség vagy a termék sorozatszáma.

The screenshot shows two windows side-by-side. The left window is titled 'Keresés' and displays a table of search results. The right window is also titled 'Keresés' and shows a detailed list of items found.

Típus	Mennyiség	Elérhető
Acelkábel 16mm	789 Méter	706 Méter
Cement	888 Kilogramm	802 Kilogramm
Ferdeháti autó	36 Darab	36 Darab
huzalszeg	1417 Doboz 25x25x25	1323 Doboz 25x25x25
Rézkábel 12mm	772 Méter	721 Méter
Zöld köralfalú szönyeg	1261 Négyzetméter	1222 Négyzetméter
Bab	990 Konzervdoboz	873 Konzervdoboz
Barna csíkos szönyeg	1092 Négyzetméter	1040 Négyzetméter

**Right Window (Detailed Search Results):**

Típus	Mennyiség	Elérhető
Acelkábel 12mm	619	55
Platis lehengő	976	976
Rézkábel 3mm	1089	707
Kukorica	1446	1446
Piros kockás szönyeg	1046	774
Rézkábel 13mm	774	757
Benzin (E85)	1393	736
Teherszeg	32	44
Szedan autó	945	889
Acelkábel 16mm	945	889
Acelkábel 40mm	1109	1175
Vörösberzsenyi	1150	1150
Narancs	1175	1052
Fehér hosszú szálú szönyeg	1052	1132
Zöld pöttyös szönyeg	1052	914
Acelkábel 10mm	914	1194
Kotol (pr. színre)	1194	793
LNG	793	793

## 7.5 Használati példák

### 7.5.1 Fatelep

Tegyük fel, hogy egy olyan favágó vállalatot vezetünk, amely több fatelephellyel is rendelkezik. A fakitermelés és annak értékesítése folyamatos folyamat, így a telephelyeken található fa mennyisége is rendszeresen változik, a telephelyek kapacitása pedig véges. Ebből kiindulva elengedhetetlen egy olyan nyilvántartó rendszer alkalmazása, amely pontosan követi, hogy az egyes telepeken mennyi fa található, illetve mennyi fa tárolására van még lehetőség. Emellett, mivel a fa folyamatos értékesítése zajlik, fontos, hogy nagyobb megrendelések esetén ne fogadjunk el olyan rendelést, amelyhez nincs elegendő fa a készletünkben. Amennyiben a fatelep feldolgozással is foglalkozik, és különböző formájú (pl. rönk, hasáb, deszka, gerenda) és méretű faanyagokkal dolgozik, azok megfelelő nyilvántartása és kezelésének egyszerűsítése érdekében alkalmazásunk ideális megoldást kínál. A rendszer lehetővé teszi a különféle faelemek pontos nyomonkövetését, biztosítva ezzel a hatékony és átlátható működést.

## 7.5.2 Bevásárló központ

Ebben a példában egy bevásárlóközpont üzemeltetésével foglalkozunk, amely több üzlettel és különböző területeken található szolgáltatásokkal rendelkezik. A központ forgalma és a rendelkezésre álló árukészletek folyamatosan változnak, így szükség van egy olyan nyilvántartó rendszerre, amely pontosan követi a termékek és készletek állapotát, valamint a raktárkapacitásokat is. A vásárlók folyamatosan fogyasztják a boltok kínálatát, így fontos tudni róla, ha elfogy, vagy fogyóban van valamelyik cikk és utánpótlást kell küldeni az adott termékből. A nyilvántartás kezelő rendszerünkben munkakörnek megfelelő jogosultságokkal lehet ellátni az egyes felhasználókat, ezzel elkerülve azt, hogy valaki olyat tegyen, amire nincs felhatalmazása. Az alkalmazásunk ideális választás a központ napi működésének zökkenőmentes irányításához, segítve a készletek, üzletek és szolgáltatások hatékony kezelését és nyilvántartását.

# III. UTÓSZÓ

---

# 8. Továbbfejlesztési lehetőségek

## 8.1 Adminisztráció

### 8.1.1 Integráció monitorozó rendszerekkel

Integráció monitorozó rendszerekkel mint például Grafana. Ezen keresztül lehetne nézni a logokat.

### 8.1.2 Központi beléptető rendszer

A központi beléptető rendszer megkönnyítené a felhasználókezelést és segítene beintegrálni a rendszert nagyobb vállalatok rendszerébe.

### 8.1.3 Webhook-ok

Webhook-okat lehetne létrehozni, hogy különböző folyamatokat elindítson ha történek egy bizonyos esemény.

## 8.2 Statisztika

### 8.2.1 Kördiagram telephely/raktár tartalmáról

Egy kördiagram ami mutatná a raktár tartalmát és miből milyen arányban van. Lehetne darabszám vagy elfoglalt hely alapján is elkészítve a diagram

## 8.2.2 Különböző raktárak/telephelyek összehasonlítása

Lenne egy táblázat ami összehasonlítja a raktárak vagy telephelyek tartalmát árucikkek szerint Könnyebb lenne átlátni és rendszerezni a raktárat

## 8.3 Használat

### 8.3.1 Térkép a raktárról és benne a tárgyakról

Amikor egy konkrét tárgyra rákeresünk akkor egy térkép megjelenne és lehetne látni hogy a raktáron belül hol találhatóak a tárgyak

### 8.3.2 Rendszeres művelet order

Létre lehetne hozni rendszeresített műveletet ami a megadott időközönként létrehoz egy műveletet. A hozzá tartozó tárgyak polc számát automatikusan kiszámolná.

### 8.3.3 Mértékegység átváltás

Mértékegységeket át lehessen váltani, hogy a raktár telítettségét jobban meg lehessen mondani, és könnyebb legyen kezelní a raktár tárolókapacitását.

### 8.3.4 Raktártípusok

Raktártípusokat lehessen létrehozni és beállítani. Ennek köszönhetően amikor új raktárat hozunk létre akkor az összes korlátozás és egyéb dolog automatikusan be lenne állítva.

### **8.3.5 Riasztás beállítás**

Riasztást lehetne beállítani ha hamarosan elfogy valami.

### **8.3.6 QR kód olvasó**

QR kód olvasó a telefonos alkalmazáshoz: A mobilalkalmazásban QR kódok olvasásának beépítése, ami egyszerűsítheti a felhasználók számára az információk gyors elérését, illetve felvitelét.

# 9. Felhasznált irodalom

- <https://developer.mozilla.org>
- <https://www.php.net/docs.php>
- <https://playwright.dev/docs/intro>
- <https://developer.android.com/training/testing/espresso>

# 10. Munkakönyvtárak

- Playwright
- <https://github.com/stleary/JSON-java>

# 11. Mellékletek

## 11.1 Online elérhetőség

<https://github.com/KissKolos/Easy-Inventory.git>