

**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: X 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

Konzultációs lap

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

Tartalom

1	Konzultációs lap	5
2	Munka	9
2.1	Munkamegosztás	10
2.1.1	Pék Gergő	10
2.1.2	Kiss Kolos	10
2.1.3	Garai Nándor	10
2.2	Munka folyamata	10
2.3	Használt fejlesztői eszközök	11
3	API Dokumentáció	13
3.1	API Típusok	14
3.1.1	Unit	14
3.1.2	UnitPutRequest	14
3.1.3	Item	14
3.1.4	ItemPutRequest	14
3.1.5	Warehouse	14
3.1.6	WarehousePutRequest	15
3.1.7	Storage	15
3.1.8	StoragePutRequest	15
3.1.9	User	15
3.1.10	UserPutRequest	15
3.1.11	OperationItem	16
3.1.12	Operation	16
3.1.13	OperationItemRequest	16
3.1.14	OperationRequest	17
3.1.15	OperationCommit	17
3.1.16	MoveRequest	17
3.1.17	LoginRequest	17
3.1.18	LoginResponse	17
3.1.19	RenewResponse	17
3.1.20	UserInfoResponse	18
3.1.21	StorageLimit	18
3.1.22	StorageLimitPutRequest	18
3.1.23	StorageCapacity	18
3.1.24	ItemStack	18
3.1.25	ItemDB	19
3.1.26	ItemStackDB	19
3.1.27	StorageLimitDB	19
3.1.28	WarehouseDB	19
3.1.29	StorageDB	20
3.1.30	LocalAuthorizationDB	20

3.1.31	UserDB	20
3.1.32	DB	20
3.2	Hitelesítés	21
3.2.1	Felhasználó bejelentkeztetése	21
3.2.2	Token megújítása	22
3.2.3	Token törlése	22
3.2.4	A bejelentkezett felhasználó információinak lekérése	23
3.3	Jogosultságok	24
3.3.1	Rendszer engedélyek lekérése	24
3.3.2	Helyi engedélyek lekérése	25
3.3.3	Helyi engedély megadása	26
3.3.4	Helyi engedély törlése	26
3.3.5	Rendszer engedély megadása	27
3.3.6	Rendszer engedély törlése	28
3.4	Cikkek	28
3.4.1	Cikkek lekérése	28
3.4.2	Tárgy azonosítójának változtatása	29
3.4.3	Cikk lekérése	30
3.4.4	Cikk módosítása, létrehozása	31
3.4.5	Cikk törlése	32
3.5	Limit	33
3.5.1	Raktár limit	33
3.5.2	Raktár limitjének módosítása	34
3.5.3	Raktár kapacitás	35
3.6	Műveletek	36
3.6.1	Műveletek lekérése	36
3.6.2	Művelet azonosítójának változtatása	38
3.6.3	Művelet lekérése	39
3.6.4	Művelet létrehozása	40
3.6.5	Művelet visszaigazolása és törlése	41
3.7	Keresés	42
3.7.1	Keresés	42
3.8	Raktárak	43
3.8.1	Raktárak lekérése	43
3.8.2	Raktár azonosítójának változtatása	45
3.8.3	Raktár lekérése	46
3.8.4	Raktárak módosítása, létrehozása	47
3.8.5	Raktár törlése	48
3.8.6	Keresés a raktárban	48
3.9	Egységek	50
3.9.1	Egységek lekérése	50

3.9.2	Egység azonosítójának változtatása	51
3.9.3	Egység lekérése	52
3.9.4	Egységek módosítása, létrehozása	53
3.9.5	Egység törlése.	54
3.10	Felhasználók	54
3.10.1	Felhasználók lekérése	54
3.10.2	Felhasználó azonosítójának változtatása	56
3.10.3	Felhasználó lekérése	56
3.10.4	Felhasználók módosítása, létrehozása	57
3.10.5	Felhasználó törlése	58
3.11	Telephelyek	59
3.11.1	Telephelyek lekérése	59
3.11.2	Telephely azonosítójának változtatása	60
3.11.3	Telephely lekérése	61
3.11.4	Telephely módosítása, létrehozása	62
3.11.5	Telephely törlése	63
3.11.6	Keresés a telephelyen	64
3.12	Biztonsági mentés	65
3.12.1	Mentés készítése	65
3.12.2	Mentés visszaállítása	66
4	API Könyvtár Dokumentáció	67
4.1	Kapcsolat kezelés	68
4.1.1	Kapcsolódás a szerverhez	68
4.1.2	Kijelentkezés	68
4.2	Egységek kezelése	69
4.2.1	Egység osztály	69
4.2.2	Egységek kezelése	69
4.3	Felhasználók kezelése	70
4.3.1	Felhasználó osztály	70
4.3.2	Felhasználók kezelése	70
4.3.3	Felhasználók engedélyeinek kezelése	71
4.4	Telephelyek kezelése	71
4.4.1	Telephely osztály	71
4.4.2	Telephelyek kezelése	72
4.5	Raktárak kezelése	72
4.5.1	Raktár osztály	72
4.5.2	Raktárak kezelése	73
4.6	Cikkek kezelése	73
4.6.1	Cikk osztály	73
4.6.2	Cikkek kezelése	74
4.7	Műveletek kezelése	74

4.7.1	Művelet osztály	74
4.7.2	Műveletek kezelése	75
4.8	Keresés	75
4.8.1	<code>ItemStack</code> osztály	75
4.8.2	Keresés	76
5	Kód Dokumentáció	77
5.1	Felépítés	78
5.2	Adatbázis	79
5.2.1	Áttekintés	79
5.2.2	<code>untis</code>	81
5.2.3	<code>item_types</code>	81
5.2.4	<code>warehouses</code>	81
5.2.5	<code>storages</code>	82
5.2.6	<code>item_stacks</code>	82
5.2.7	<code>storage_limits</code>	83
5.2.8	<code>operations</code>	83
5.2.9	<code>operation_items</code>	84
5.2.10	<code>users</code>	84
5.2.11	<code>authorization</code>	85
5.2.12	<code>global_authorization</code>	85
5.2.13	<code>authentication_token</code>	86
5.2.14	<code>users_view</code>	86
5.2.15	<code>authorization_view</code>	86
5.2.16	<code>authentication_token_view</code>	87
5.2.17	<code>warehouses_view</code>	87
5.2.18	<code>storages_view</code>	87
5.2.19	<code>item_types_view</code>	87
5.2.20	<code>units_view</code>	88
5.2.21	<code>operations_view</code>	88
5.2.22	<code>operation_items_view</code>	89
5.2.23	<code>item_stacks_view</code>	90
5.3	Api szerver	91
5.3.1	Felépítés	91
5.3.2	<code>ApiRouter.php</code>	92
5.3.3	<code>APIUtils.php</code>	93
5.3.4	<code>DB.php</code>	96
5.3.5	<code>BaseTrait.php</code>	96
5.3.6	<code>Logger.php</code>	96
5.3.7	<code>Settings.php</code>	97
5.3.8	<code>Authentication.php</code>	97
5.4	Web frontend	98

5.4.1	Áttekintés	98
5.4.2	Építés	98
5.4.3	Template osztály	98
5.4.4	View osztály	100
5.4.5	ListView osztály	100
5.4.6	ListEditView osztály	101
5.4.7	SelectView osztály	101
5.4.8	DialogView osztály	102
5.4.9	EditDialog osztály	102
5.5	API könyvtár	103
5.5.1	Áttekintés	103
5.5.2	HTTPConnector osztály	103
5.6	Asztali kliens	104
5.6.1	Bevezetés	104
5.6.2	Worker osztály	104
5.6.3	EasyInventoryDesktop osztály	104
5.6.4	BasicView osztály	106
5.6.5	SelectView osztály	106
5.6.6	CardView osztály	106
5.6.7	EditableView osztály	107
5.6.8	LocalEditableView osztály	107
5.6.9	FormDialog osztály	109
5.6.10	EditDialog osztály	109
5.7	Mobil kliens	110
5.7.1	Áttekintés	110
5.7.2	MainActivity osztály	110
5.7.3	LoggedInActivity osztály	110
5.7.4	Worker osztály	111
5.7.5	BaseEditActivity osztály	113
5.7.6	LocalEditActivity osztály	113
5.7.7	PaginatedListActivity osztály	115
5.7.8	BaseListActivity osztály	115
5.7.9	BaseListFragment osztály	117
5.7.10	LocallListFragment osztály	117
5.7.11	SelectableArrayAdapter osztály	119
5.8	Dokumentáció	120
5.8.1	Áttekintés	120
5.8.2	Felhasználói dokumentáció	120
5.8.3	Kód dokumentáció	120
5.8.4	Teszt dokumentáció	120
5.8.5	Kód beillesztő	121

5.8.6	Építés	121
6	Teszt Dokumentáció	123
6.1	API szerver	124
6.1.1	Áttekintés	124
6.1.2	E2E tesztek	124
6.2	Web frontend	125
6.2.1	Áttekintés	125
6.2.2	E2E tesztek	125
6.3	Asztali alkalmazás	126
6.3.1	Áttekintés	126
6.3.2	E2E tesztek	126
6.3.3	NodeMatcher osztály	127
6.4	Mobil alkalmazás	129
6.4.1	Áttekintés	129
6.4.2	E2E tesztek	129
7	Felhasználói Dokumentáció	131
7.1	Bevezetés	132
7.1.1	Fogalmak	132
7.2	Telepítés	134
7.2.1	Server telepítés	134
7.2.2	Asztali kliens telepítés	135
7.2.3	Mobil kliens telepítés	135
7.3	Adminisztráció	136
7.3.1	Arhivált adatok törlése	136
7.3.2	Eseménynapló kezelése	136
7.3.3	Felhasználó kezelés	138
7.3.4	Engedélyek	139
7.4	Használat	141
7.4.1	Bejelentkezés	141
7.4.2	Felhasználó kezelés	142
7.4.3	Egység kezelés	144
7.4.4	Árucikkek kezelése	145
7.4.5	Telephely kezelés	147
7.4.6	Raktár kezelés	148
7.4.7	Műveletek	150
7.4.8	Keresés	151
7.5	Használati példák	152
7.5.1	Fatelep	152
7.5.2	Bevásárló központ	153

8 Továbbfejlesztési lehetőségek	155
8.1 Adminisztráció	156
8.1.1 Integráció monitorozó rendszerekkel	156
8.1.2 Központi beléptető rendszer	156
8.1.3 Webhook-ok	156
8.2 Statisztika	156
8.2.1 Kördiagram telephely/raktár tartalmáról	156
8.2.2 Különböző raktárak/telephelyek összehasonlítása	156
8.3 Használat	157
8.3.1 Térkép a raktárról és benne a tárgyakról	157
8.3.2 Rendszeres művelet order	157
8.3.3 Mértékegység átváltás	157
8.3.4 Raktártípusok	157
8.3.5 Riasztás beállítás	157
8.3.6 QR kód olvasó	158
9 Felhasznált irodalom	159
10 Mellékletek	161
10.1 Felhasznált könyvtárak	162
10.2 Online elérhetőség	162

Munka

2.1 Munkamegosztás

2.1.1 Pék Gergő

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

2.1.2 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.

2.1.3 Garai Nándor

Webes és API tesztek írása, API szerver adatbázis kezelő részének implementálása, web frontend design megtervezése, tesztadatok generálása és bemutató készítése.

2.2 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 teszteken. Végül a dokumentáció elkészítése és a végső simítások következtek.

2.3 Használt fejlesztői eszközök

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

API Dokumentáció

O

3.1 API Típusok

3.1.1 Unit

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

3.1.2 UnitPutRequest

Kulcs	Típus	Tartalom
name	string	Az egység neve

3.1.3 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

3.1.4 ItemPutRequest

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

3.1.5 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

3.1.6 WarehousePutRequest

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

3.1.7 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

3.1.8 StoragePutRequest

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

3.1.9 User

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

3.1.10 UserPutRequest

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

3.1.11 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

3.1.12 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

3.1.13 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

3.1.14 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

3.1.15 OperationCommit

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

3.1.16 MoveRequest

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

3.1.17 LoginRequest

Kulcs	Típus	Tartalom
password	string	A jelszó bejelentkezéshez

3.1.18 LoginResponse

Kulcs	Típus	Tartalom
token	string	A token

3.1.19 RenewResponse

Kulcs	Típus	Tartalom
expiration	int	A token lejárásnak időpontja Unix másodpercekben

3.1.20 UserinfoResponse

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

3.1.21 StorageLimit

Kulcs	Típus	Tartalom
item	Item	A cikk
amount	int	A limit

3.1.22 StorageLimitPutRequest

Kulcs	Típus	Tartalom
amount	int	A limit

3.1.23 StorageCapacity

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

3.1.24 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

3.1.25 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

3.1.26 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

3.1.27 StorageLimitDB

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

3.1.28 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

3.1.29 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

3.1.30 LocalAuthorizationDB

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

3.1.31 UserDB

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

3.1.32 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

3.2 Hitelesítés

3.2.1 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"  
}
```

3.2.2 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  
}
```

3.2.3 Token törlése

Erőforrás

DELETE /tokens

Kérés fejléce

Authorization Bearer token

Válasz kódok

404 Nem létezik

3.2.4 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"  
}
```

3.3 Jogosultságok

3.3.1 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"]`

3.3.2 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" ]
```

3.3.3 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

3.3.4 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

3.3.5 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

3.3.6 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

3.4 Cikkek

3.4.1 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  
  }  
]
```

3.4.2 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

3.4.3 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  
}
```

3.4.4 Cikk módosítása, létrehozása

Erőforrás

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

Paraméterek

item A cikk 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 **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

3.4.5 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** Törölve

3.5 Limit

3.5.1 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  
  }  
]
```

3.5.2 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

3.5.3 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  
  }  
]
```

3.6 Műveletek

3.6.1 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": []  
  }  
]
```

3.6.2 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

3.6.3 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"
    }
  ]
}
```

3.6.4 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

3.6.5 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

3.7 Keresés

3.7.1 Keresés

Erőforrás

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

Paraméterek

- 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

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  
  }  
]
```

3.8 Raktárak

3.8.1 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  
  }  
]
```

3.8.2 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

3.8.3 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
}
```

3.8.4 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

3.8.5 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

3.8.6 Keresés a raktárban

Erőforrás

```
GET /warehouses/{warehouse}/storages/{storage}/search?q={q}&warehouse={warehouse}&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  
  }  
]
```

3.9 Egységek

3.9.1 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  
  }  
]
```

3.9.2 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

3.9.3 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  
}
```

3.9.4 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

3.9.5 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 Törölve

3.10 Felhasználók

3.10.1 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  
  }  
]
```

3.10.2 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

3.10.3 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  
    }  
}
```

3.10.4 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

3.10.5 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

3.11 Telephelyek

3.11.1 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  
  }  
]
```

3.11.2 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

3.11.3 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  
}
```

3.11.4 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

3.11.5 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

3.11.6 Keresés a telephelyen

Erőforrás

```
GET /warehouses/{warehouse}/search?q={q}&warehouse={qwarehouse}&storage={qstorage}&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  
  }  
]
```

3.12 Biztonsági mentés

3.12.1 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": []  
}
```

3.12.2 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

API Könyvtár Dokumentáció

4.1 Kapcsolat kezelés

4.1.1 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 { ... }  
    ...  
}
```

4.1.2 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 { ... }  
    ...  
}
```

4.2 Egységek kezelése

4.2.1 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) { ... }
}
```

4.2.2 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 { ... }
    ...
}
```

4.3 Felhasználók kezelése

4.3.1 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) { ... }
}
```

4.3.2 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
        { ... }
}
```

4.3.3 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 { ... }
    ...
}
```

4.4 Telephelyek kezelése

4.4.1 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() { ... }
}
```

4.4.2 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 { ... }
    ...
}
```

4.5 Raktárak kezelése

4.5.1 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) { ... }
}
```

4.5.2 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 { ... }
    ...
}
```

4.6 Cikkek kezelése

4.6.1 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) { ... }
}
```

4.6.2 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 { ... }
    ...
}
```

4.7 Műveletek kezelése

4.7.1 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) { ... }
}
```

4.7.2 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 { ... }
    ...
}
```

4.8 Keresés

4.8.1 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() { ... }
}
```

4.8.2 Keresés

A kereséshez az API osztályban található funkciókat használhatjuk.

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

Kód Dokumentáció

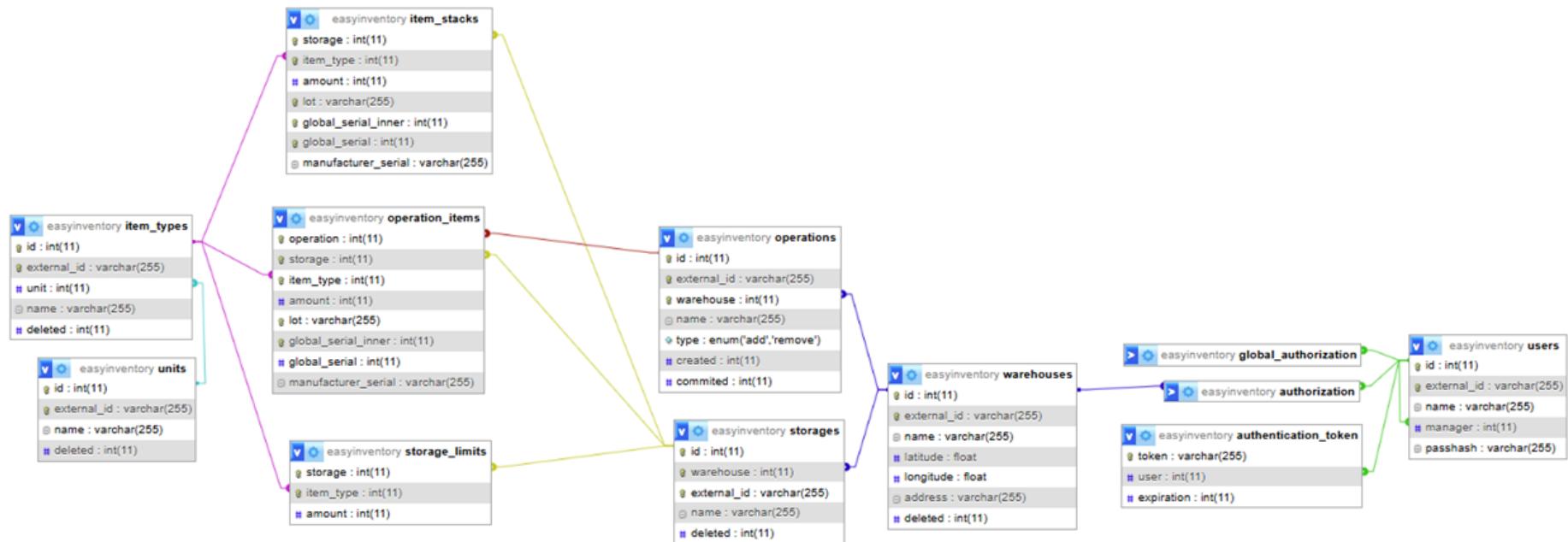
5.1 Felépítés

A gyökérkérrégióban 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.

5.2 Adatbázis

5.2.1 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 zárnai, ami lassítaná a lekéréseket. Az adatbázisban több nézet is található a lekérések egyszerűbbé tételeire. Az alábbi grafikon mutatja a táblák kapcsolatrendszerét:



5.2.2 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)
) $
```

5.2.3 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)
) $
```

5.2.4 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)
) $
```

5.2.5 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)
) $
```

5.2.6 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)
) $
```

5.2.7 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)
)$
```

5.2.8 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)
)$
```

5.2.9 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)
)$
```

5.2.10 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)
)$
```

5.2.11 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)
)$
```

5.2.12 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"),

    primary key(user,authorization),
    foreign key(user) references users(id)
)$
```

5.2.13 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)
)$
```

5.2.14 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$
```

5.2.15 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$
```

5.2.16 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$
```

5.2.17 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$
```

5.2.18 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$
```

5.2.19 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$
```

5.2.20 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$
```

5.2.21 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$
```

5.2.22 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$
```

5.2.23 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$
```

5.3 Api szerver

5.3.1 Felépítés

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 kivitelével 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.

5.3.2 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() { ... }
    ...
}
```

5.3.3 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);
            }
        }
    }
}
```

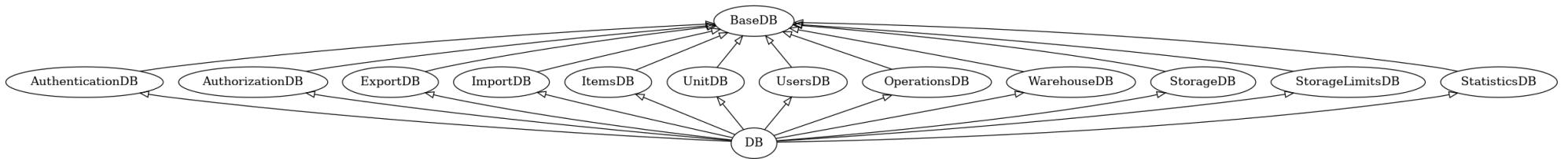


Figure 5.1: Osztálydiagramm

5.3.4 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.

5.3.5 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.

5.3.6 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 { ... }
}
```

5.3.7 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;
    string|null $test_token,int $token_length) { ... }
    public static function getSettings():Settings { ... }
}
```

5.3.8 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 {
        ...
    }
}
```

5.4 Web frontend

5.4.1 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.

5.4.2 Építés

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

5.4.3 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() { ... }  
}
```

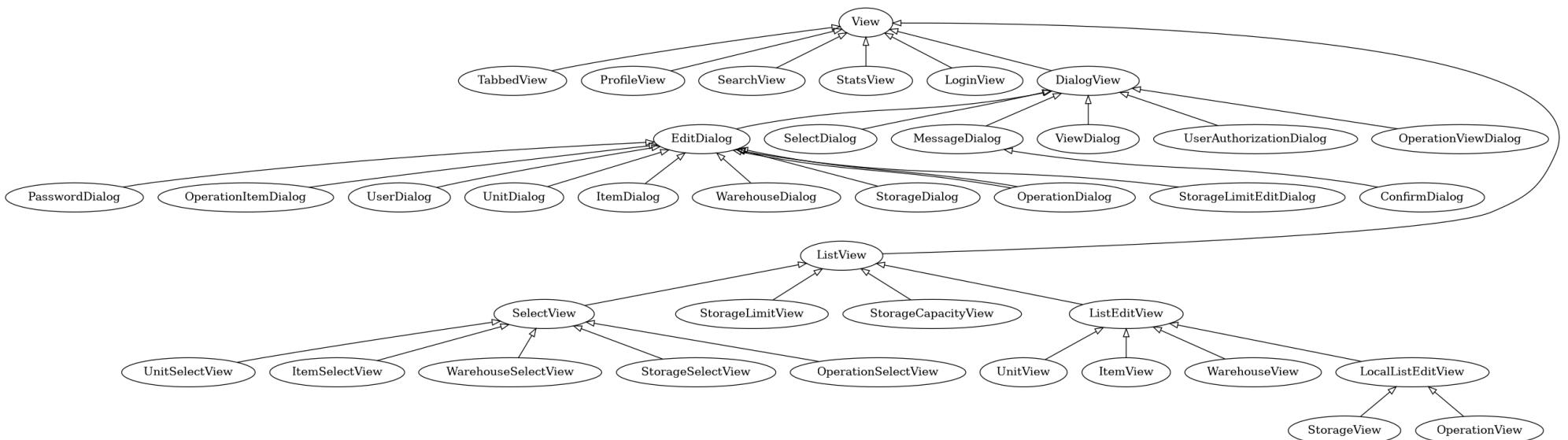


Figure 5.2: Osztálydiagramm

5.4.4 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 részt.

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

5.4.5 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) {}
}
```

5.4.6 ListView 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(tmp, v) { ... }  
}
```

5.4.7 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) { ... }  
}
```

5.4.8 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() { ... }
}
```

5.4.9 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() { ... }
}
```

5.5 API könyvtár

5.5.1 Áttekintés

Az API könyvtárat a mobil és az 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.

5.5.2 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.

5.6 Asztali kliens

5.6.1 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.

5.6.2 Worker osztály

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

5.6.3 EasyInventoryDesktop osztály

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

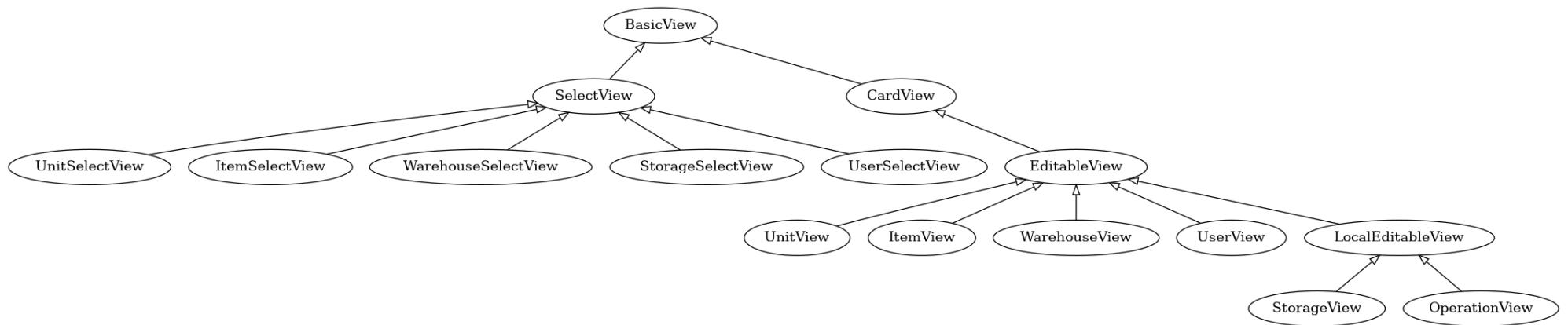


Figure 5.3: Osztálydiagramm

5.6.4 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.

5.6.5 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` megteszeli helyette.

5.6.6 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.

5.6.7 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.

5.6.8 LocalEditableView osztály

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

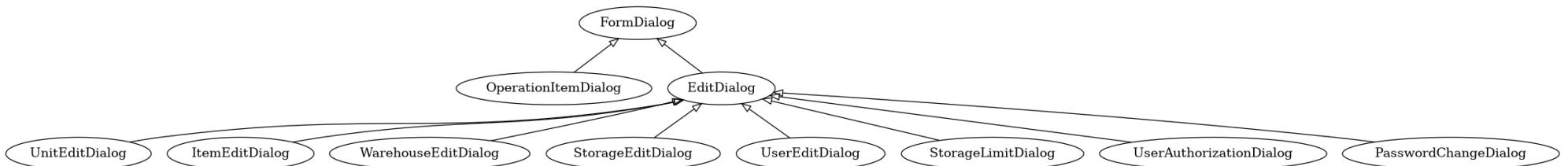


Figure 5.4: Osztálydiagramm

5.6.9 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.

5.6.10 EditDialog osztály

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

5.7 Mobil kliens

5.7.1 Áttekintés

A mobil alkalmazás forrás fájljai a `EasyInventoryMobile` mapában találhatóak, fejlesztésre pedig az Android Studió 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.

5.7.2 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) { ... }
}
```

5.7.3 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 { ... }
}
```

5.7.4 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) { ... }  
}
```

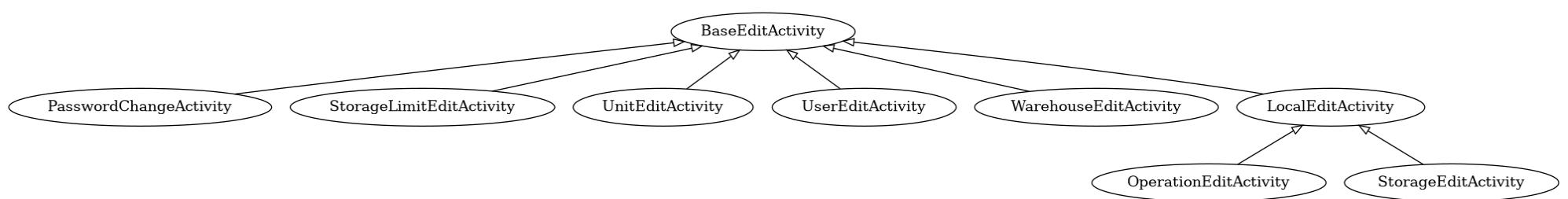


Figure 5.5: Osztálydiagramm

5.7.5 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 kiolvasása 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() { ... }
}
```

5.7.6 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() { ... }
}
```

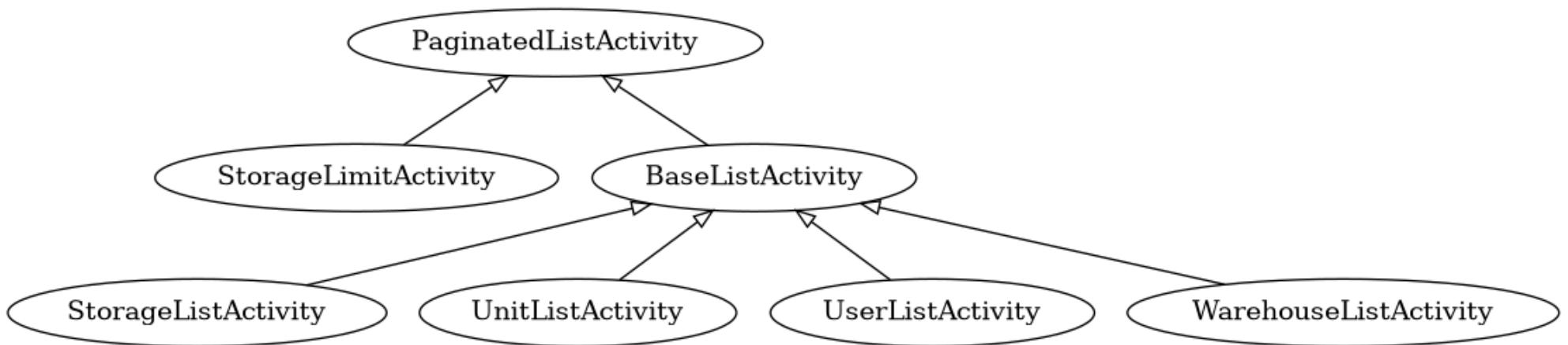


Figure 5.6: Osztálydiagramm

5.7.7 PaginatedListActivity osztály

A `PaginatedListActivity` absztrakt osztály olyan tevékenységek szülőosztálya, amelyek lapozható listát mutatnak. A `load` funkció feladata, hogy a megadott paraméterek (keresési szüveg, kezdőpozíció, mennyiség, 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() { ... }
}
```

5.7.8 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) { ... }
}
```

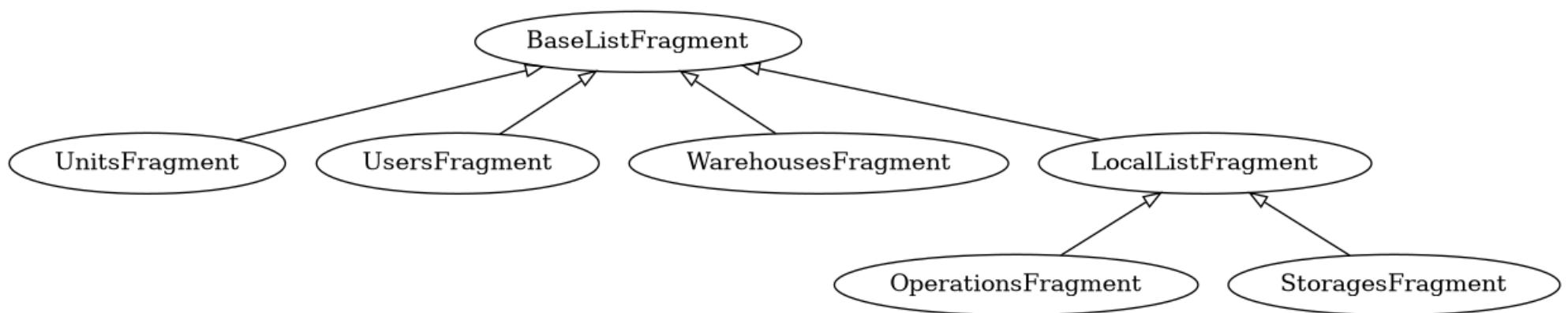


Figure 5.7: Osztálydiagramm

5.7.9 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) {
        ... }
}
```

5.7.10 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() { ... }
}
```

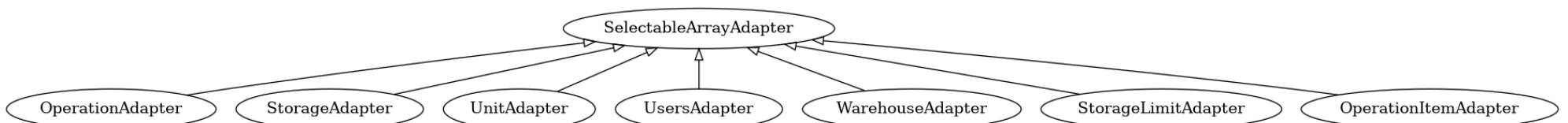


Figure 5.8: Osztálydiagramm

5.7.11 SelectableArrayAdapter osztály

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

5.8 Dokumentáció

5.8.1 Áttekintés

A projekt dokumentációja L^AT_EX 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.

5.8.2 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.

5.8.3 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.

5.8.4 Teszt dokumentáció

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.

5.8.5 Kód beillesztő

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

5.8.6 Építés

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

Teszt Dokumentáció

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 tokent, 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 keretrendszert 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 doesNotExists() { ... }  
    public void acceptDialog() { ... }  
    public void closeDialogs() { ... }  
    public void print() { ... }  
    public void screenshotWindow(String out) { ... }  
}
```

allWindowRoots

A `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 be-meneti adatok is egyaránt tesztelve vannak, illetve a tesztek egy része képernyőképeket készít, amelyeket a dokumentációhoz használunk fel.

Felhasználói Dokumentáció

7.1 Bevezetés

Az alkalmazás célja, hogy lehetővé tegye cégek számára, hogy bármiféle képesítés vagy különösebb hozzáértés nélkül is egyszerűen tudják kezelní a legkomplexebb nyilvántartásokat is. A program lehetővé fogja tenni, hogy különböző jogköröket csatoljunk az egyes felhasználókhöz, melyek befolyásolják, hogy a nyilvántartás mely részeit tekintheti meg és milyen műveleteket hajthat végre a programon belül az adott illető. A rendszer képes lesz nyilvántartani a telephelyeket, raktárakat és cikkeket, 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 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 funkcióit és kezelési lehetőségeit.

7.1.1 Fogalmak

Telephelyek

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

Raktárak

Raktárak a legkisebb tároló egység amikben vannak az árucikkek nyilvántartva. Ezeknek meg lehet adni, hogy cikkenként mennyi fér beléjük.

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 művelet

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 művelet

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 meghatározhatjuk lot-, vagy akár sorozatszám megadásával is.

Lot számok

A lot számok különböző cikk csoportok megkülönböztetésére szolgáló kódok, amelyek nagyobb nyilvántartások kezelésének 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 egyedieknek lenniük, mivel ha több gyártó állítja elő az adott árucikket akkor a sorozatszámaik akár ütközhetnek is.

7.2 Telepítés

7.2.1 Server telepítés

Linux mint

A szerver telepítéséhez a futtatni kell a telepítő programot.

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

Ez a parancs telepíti szükséges csomagokat és bemásolja a frontendet és a szervert a `/var/www/html/` mappába. Ezután ott 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 és a `frontend/index.html` fájlt át kell másolni az apache web mappába. A mysql szerveren létre kell hozni egy felhasználót és be kell állítani. Az apache szerveren engedélyezni kell a `.htaccess` fájlokat és be kell kapcsolni a `rewrite` és a PHP modulokat.

Windows

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

7.2.2 Asztali kliens telepítés

Az asztali kliens fájljai a `EasyInventoryDesktop/dist/bundles/EasyInventoryDesktop` mappában találhatóak. Ezt a mappát akárhova lehet másolni és nem igényel telepítőt, mert mindegyik szükséges dll benne van a mappában. A futtatáshoz a benne található exe fájlt lehet futtatni.

7.2.3 Mobil kliens telepítés

A mobil alkalmazás telepítéséhez Android 14 szükséges.

7.3 Adminisztráció

7.3.1 Arhivált adatok törlése

Fontos!

Az arhivált adatok törlése visszafordíthatatlan! Ezek az adatok adják a statisztikákat. Régi adatok törlésével az akkori 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. Ehez meg kell adni hány évnyi, heti vagy napi adatot akarunk megtartani az időegység jelével.

d nap

w hét

y év

Például: ez a parancs kitöli 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 megjelenítésére van a `log show` parancs. Ha nem adunk meg semmilyen kapcsolót akkor az összes adatot rövid formában írja ki.

```
php log show
```

-u, --user kapcsoló

Ezzel megadhatjuk a felhasználói azonosítót amihez tartozó eseményeket akarjuk megtekinteni.

-e, --error kapcsoló

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

-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
```

Megmutatja az összes szerverhibát részletesen

```
php log -ev show
```

Egy esemény megtekintése

Ha csak egy kérést akarunk megtekinteni akkor a kérés sorszámának megadásával tudjuk. Ehez a kéréshez nincsenek kapcsolók.

```
php log show 100
```

Eseménynapló törlése

Ez a parancs törli a teljes eseménynapló tartalmát:

```
php log clear
```

Általában nem akarjuk az összes eseményt törölni. Ehez meg kell adni hány évnyi, heti vagy napi adatot akarunk megtartani az időegység jelével.

d nap

w hét

y év

Például: ez a parancs kitöli 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 meh kell adni a felhasználó azonosítóját és az új jelszót.

```
php user password admin password123
```

7.3.4 Engedélyek

Két féle engedély típus van: a rendszer és a helyi. A rendszer engedélyek az egész rendszerre vonatkoznak. Azaz ha egy felhasználó egy ilyen engedélyt kap akkor az minden telephelyen érvényes, de a rendszer engedélyek között rendszer adminisztrációval kapcsolatosak engedélyek is vannak. A helyi engedélyek telephelyhez vannak kötve. Azaz ha egy felhasználó egy ilyen engedélyt kap akkor csak 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

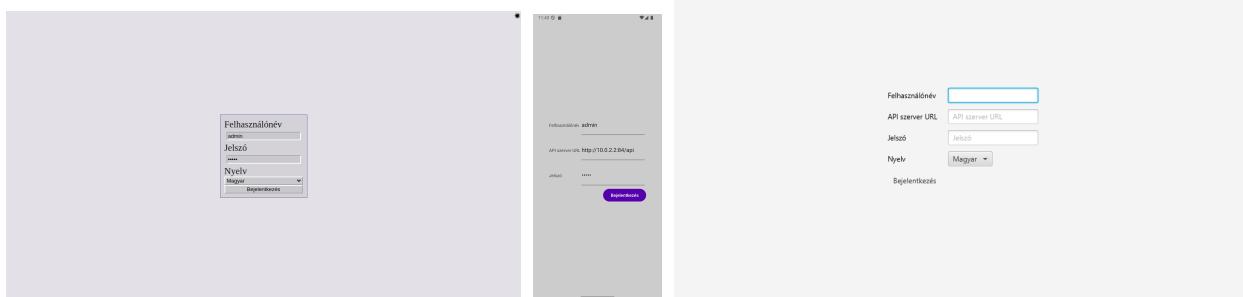
Helyi engedélyek

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

7.4 Használat

7.4.1 Bejelentkezés

A program indításakor először a bejelentkező felületre fogad minket, ahol meg kell adni a felhasználónevét, a API szerver url -jét, a jelszót és nyelvét a bejelentkezéshez. Miután kitöltöttük az adatokat, 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 melyik részét szeretnénk megtekinteni vagy módosítani.



7.4.2 Felhasználó kezelés

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) Törölhetjük a felhasználót A lap alján található gombokkal: Frissíthetjük az oldalt Új felhasználót regisztrálhatunk.

The screenshot shows two views of the user management interface. On the left, a grid view displays user profiles with columns for Name, Surname, and Birthdate. On the right, a detailed view shows a list of users with columns for Name, Surname, Birthdate, and Address. Both views include edit and delete icons for each user entry.

Felhasználó hozzáadás

Felhasználó hozzáadásához a **+** gombra kell kattintani. Ezután megjelenik egy bemeneti ablak, ami kitöltése után a **✓** gombra nyomhatunk a mentéshez.

The screenshot shows the user creation dialog box. It includes fields for 'Felhasználónév' (Username), 'Név' (Name), 'Jelszó' (Password), and 'Menedzser' (Manager). A note at the bottom states 'Nincs kiválasztott felhasználó' (No user selected). There are 'OK' and 'Cancel' buttons at the bottom right.

Felhasználó módosítás

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 engedélyek 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.

Felhasználó törlés

A felhasználót a gombra való kattintással törölhetjük.

7.4.3 Egység kezelés

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.

The screenshot shows two views of the unit management interface. On the left, a grid view lists units like Centiméter, Méter, Kilométer, Gramm, Kilogramm, Tonna, etc. On the right, a detailed list view shows the same units with additional columns for unit type (e.g., length, mass, volume) and unit code (e.g., cm, m, km, g, kg, ton). Both views include edit, delete, and search functions.

Egység hozzáadás

Egy egységet a **+** gombra való kattintással adhatunk hozzá a rendszerhez. Ekkor megjelenik egy bemeneti ablak amit kitöltve **✓** gombra kattintással menthetünk.

The screenshot shows the 'Add Unit' dialog box. It contains fields for 'Azonosító' (Identifier) and 'Név' (Name). Below the dialog are the original unit management screens, showing the new unit has been added to the list.

Egyeség módosítás

Egyeségeket a gombra való kattintással módosíthatunk. Ekkor megjelennek az adatok egy felugró ablak és módosítás után a gombra kattintással menthetünk.

Egyeség törlés

Egyeségeket 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.

Árucikk hozzáadás

Egy árucikkek a gombbal adhatunk hozzá a rendszerhez. Ekkor megjelenik egy bemeneti ablak amit kitöltve gombra kattintással menthetünk.

Árucikk módosítás

Árucikkeket a gombra való kattintással módosíthatunk. Ekkor megjelennek az adatok egy felugró ablak és módosítás után a gombra kattintással menthetünk.

Árucikk törlés

Árucikkeket a gombra való kattintással törölhetünk. Fontos megjegyezni, hogy egy árucikket nem törölhetünk ha van valahol ilyen tárolva a rendszerben.

7.4.5 Telephely kezelés

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.

Telephely hozzáadás

Egy telephelyet a **+** gombbal adhatunk hozzá a rendszerhez. Ekkor megjelenik egy bemeneti ablak amit kitöltve **✓** gombra kattintással menthetünk.

Termékek Egységek Felhasználók Keresés Műveletek		Telephelyek Raktárak Profil Statistikák	admin
Ajki telephely Wifit rendel Ajka Patak utca 70.	Bajai telephely Wifit rendel Baja Körút u. 7.	Békéscsabai telephely Wifit rendel Békéscsaba Csongrád u. 57.	Budapesti telephely Wifit rendel Budapest Szent István körút 48.
Cegládi telephely Wifit rendel Cegládi Károlyfai út 52.	Csongrádi telephely Wifit rendel Csongrád Tisza u. 1.	Dabóczi telephely Wifit rendel Dabóczi Gyula utca 89.	Debreceni telephely Wifit rendel Debrecen Lágymány utca 30.
Dunaujvárosi telephely Wifit rendel Dunaujvárosi Holler Albert utca 20.	Egeri telephely Wifit rendel Eger Bem rakpart 8.	Érdi telephely Wifit rendel Érd Istenhegyi út 99.	Esztergomi telephely Wifit rendel Esztergom Csabai kúpuszta 57.
Fót telephely Wifit rendel Fót Széchenyi utca 55	Gödöllői telephely Wifit rendel Gödöllői Nagyboldogassági utca 15.	Gyáli telephely Wifit rendel Gyáli Műszaki utca 15.	Györi telephely Wifit rendel Győr Rákóczi utca 110.
<input style="float: right;" type="button" value="Archivált adámok listázása"/>			

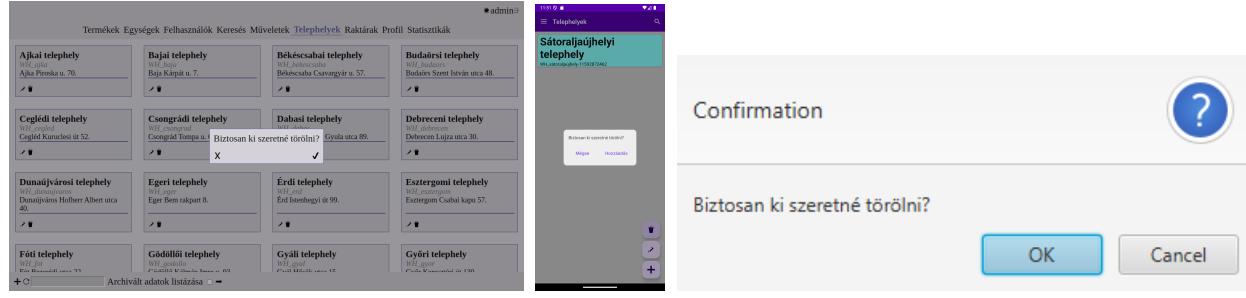
Telephely módosítás

Egy telephelyet a ✎ gombra való kattintással módosíthatunk. Ekkor megjelennek az adatok egy felugró ablak és módosítás után a ✓ gombra kattintással menthetünk.

Termekek Egyesített Fehérhazának Keresés Műveletek		Telephelyek Raktárak Profil Statisztikák	admin@... [Logout]						
Ajka telephely Wifit nem rendeltek meg. Aja Piroks u. 70. Cégeid telephely Wifit nem rendeltek meg. Cegléd Károlyfai út 52. Dunajászvárosi telephely Wifit nem rendeltek meg. Dunajászvárosi Hollófer Ábel utca 26.	Baja telephely Wifit nem rendeltek meg. Baja Kápr u. 7. Csongrádi telephely Wifit nem rendeltek meg. Csongrád Tompa u. 1. Egeri telephely Wifit nem rendeltek meg. Eger Bem rakpart 8.	Békéscsabai telephely Wifit nem rendeltek meg. Békéscsaba Csurgavargy u. 57. Debrecenti telephely Wifit nem rendeltek meg. Debrecenti utca 1. Érdi telephely Wifit nem rendeltek meg. Érd Szenthági út 99.	Budapesti telephely Wifit nem rendeltek meg. Budapest Szent Lőrinc utca 48. Esztergomi telephely Wifit nem rendeltek meg. Esztergom Oszkó kapu 57. Gyáli telephely Wifit nem rendeltek meg. Gyula Gyulai utca 10.						
<input type="checkbox"/> Archivált adatok listázása		<input checked="" type="checkbox"/> Telephely szerekesítése							
<table border="1"> <tr> <td>Konvall</td> <td>WH_sátoraljaújhely1195:</td> </tr> <tr> <td>Nev</td> <td>Sátoraljaújhely telephely</td> </tr> <tr> <td>Cím</td> <td>nekered utca 1.</td> </tr> </table>				Konvall	WH_sátoraljaújhely1195:	Nev	Sátoraljaújhely telephely	Cím	nekered utca 1.
Konvall	WH_sátoraljaújhely1195:								
Nev	Sátoraljaújhely telephely								
Cím	nekered utca 1.								
<p>Azonosító WH_sátoraljaújhely14345</p> <p>Név Sátoraljaújhelyi telephely</p> <p>Cím nekered utca 1.</p>									
OK Cancel									

Telephely törlés

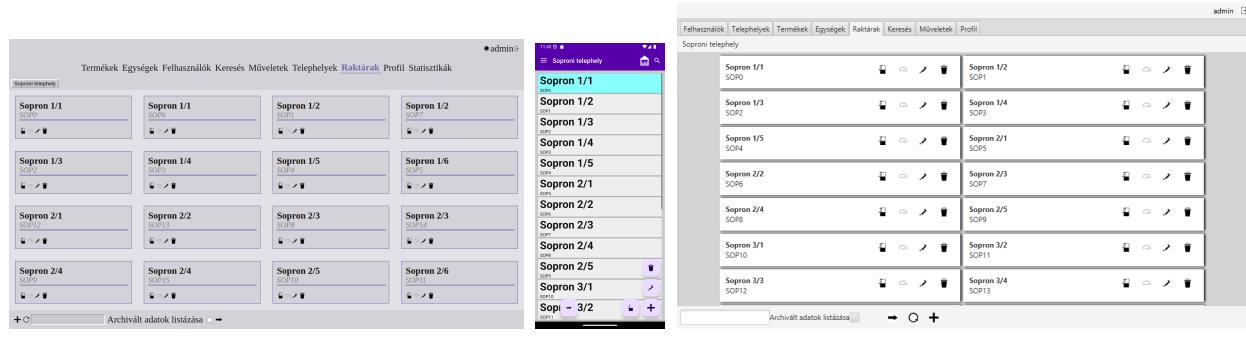
Telephelyet a  gombra való kattintással törölhetünk. Fontos megjegyezni, hogy egy telephelyet nem törölhetünk ha van raktár benne. Azaz először a raktárakat kell törölni.



Telephely	Elérhetőségek	Elhelyezkedése
Ajka telephely	WIFI, mobil	Ajka Patak u. 70.
Bajai telephely	WIFI, mobil	Bajai Kaptár u. 7.
Békéscsabai telephely	WIFI, mobiltelefon	Békéscsaba Csanygyár u. 57.
Budapesti telephely	WIFI, mobil	Budapest Szent István uca 48.
Ceglédi telephely	WIFI, mobil	Cegléd Tempa u. 11.
Csepel telephely	WIFI, mobil	Gyula uca 89.
Dabasi telephely	WIFI, mobil	Dabasen Lajta uca 39.
Dunajivárosi telephely	WIFI, mobil	Dunajivárosi Hoffer Albert utca 20.
Egeri telephely	WIFI, mobil	Eger Ben rakpart 8.
Erdi telephely	WIFI, mobil	Erd Istenhegyi út 99.
Esztergomi telephely	WIFI, mobil	Esztergom Csabai kapu 57.
Füsti telephely	WIFI, mobil	Füsti Gyümölcsök uca 15.
Gödöllői telephely	WIFI, mobil	Gödöllői Völgyi uca 15.
Győri telephely	WIFI, mobil	Győr Városmajor uca 190.

7.4.6 Raktár kezelés

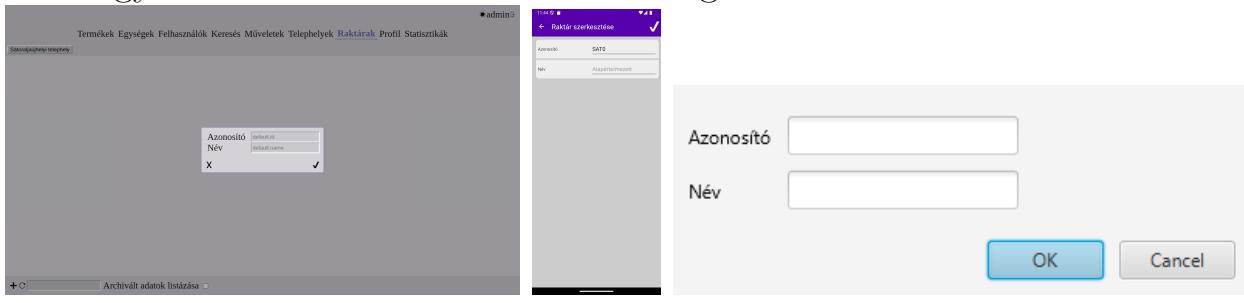
A Raktárok menüpontra kattintva a program mutatja számunkra a raktárokat telephelyenként szétválogatva. A menüsor alatt található lenyíló listával tudjuk kiválasztani, hogy melyik telephelyhez tartozó raktárokat szereznénk megtekinteni, majd a raktárok neve mellett található gombokkal módosíthatjuk és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



Raktár	Sorrend
Sopron 1/1	SOP1
Sopron 1/2	SOP2
Sopron 1/3	SOP3
Sopron 2/1	SOP4
Sopron 2/4	SOP5
Sopron 2/5	SOP6
Sopron 2/6	SOP7
Sopron 2/7	SOP8
Sopron 2/8	SOP9
Sopron 3/1	SOP10
Sopron 3/2	SOP11

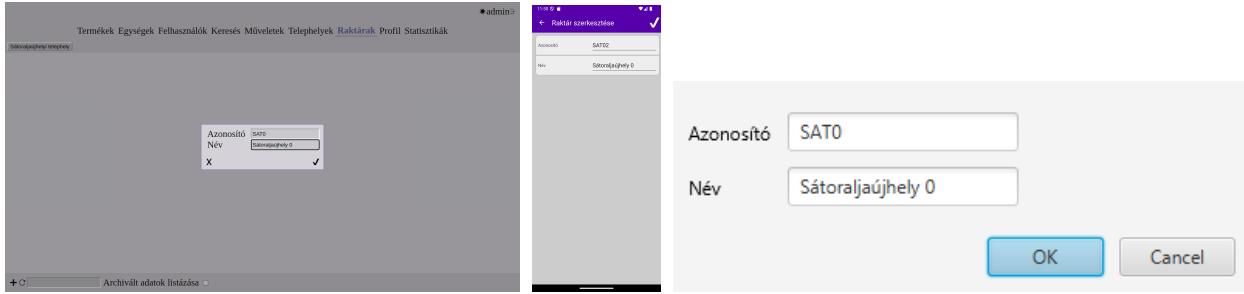
Raktár hozzáadás

Egy telephelyet a **+** gombbal adhatunk hozzá a rendszerhez. Ekkor megjelenik egy bemeneti ablak amit kitölvtve **✓** gombra kattintással menthetünk.



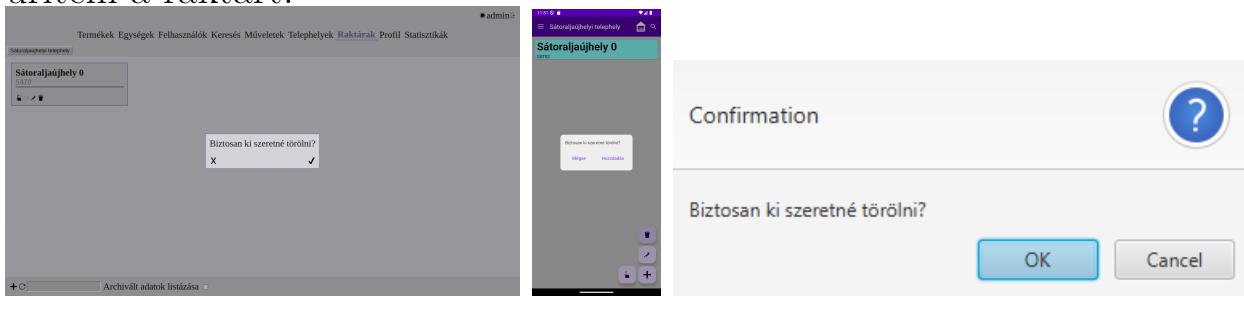
Raktár módosítás

Egy raktárt a **✎** gombra való kattintással módosíthatunk. Ekkor megjelennek az adatok egy felugró ablak és módosítás után a **✓** gombra kattintással menthetünk.



Raktár törlés

Raktárakat a **trash bin** gombra való kattintással törölhetünk. Fontos megjegyezni, hogy egy raktárt nem törölhetünk ha van benne áru. Azaz először ki kell üríteni a raktárt.



Raktár limitek

Raktár limiteket a gombra való kattintással szerkeszthetünk.

Rézkábel	Limit
3mm	0
4mm	0
5mm	0
8mm	0
10mm	0
12mm	0
13mm	0
16mm	0
20mm	0
25mm	0
35mm	0
40mm	0
UTP kábel	0
Acélkábel 3mm	0
Acélkábel 4mm	0
Acélkábel 5mm	0
Acélkábel 8mm	0
Acélkábel 10mm	0
Acélkábel 12mm	0
Acélkábel 13mm	0

7.4.7 Műveletek

Művelet hozzáadás

Egy műveletet a gombbal adhatunk hozzá a rendszerhez. Ekkor megjelenik egy bemeneti ablak amit kitöltve gombra kattintással menthetünk.

Azonosító	Név	Véletlen
default_id	default_name	Alapértelmezett

Művelet törlés

Műveleteket a gombra való kattintással törölhetünk.

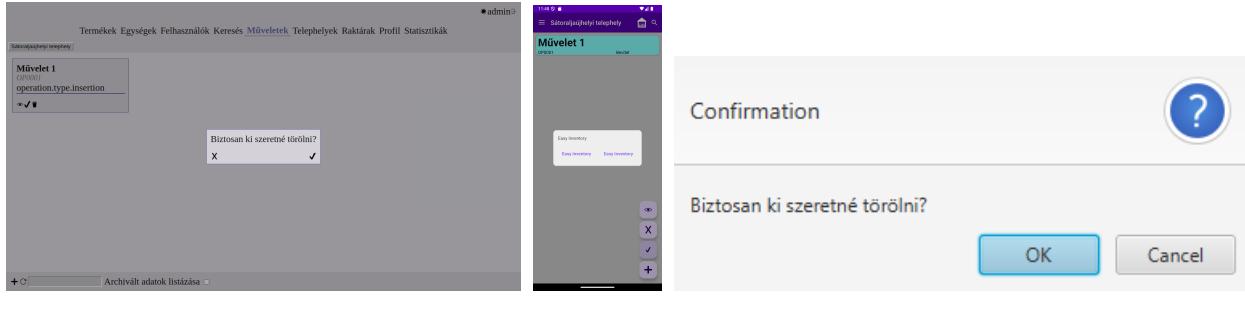
Confirmation

Biztosan ki szeretné törölni?

OK Cancel

Művelet visszaigazolás

Műveleteket a ✓ gombra való kattintással igazolhatunk vissza.



7.4.8 Keresés

A Keresés használatával egyszerűen rá tudunk keresni bármilyen termékre, a program pedig visszaadja, hogy melyik telephelynek melyik raktárában található meg az adott termék, illetve további információkat találunk, mint például a rendelkezésre álló mennyiség vagy a termék sorozatszáma.

Keresés		
Típus	Mennyiségek	Elérhető
Acélkábel 12mm	619	
Plátna teherautó	55	
Plázakábel 3mm	976	
Benzin (95)	1089	
Fűtőszivág	757	
Kukurica	1446	
Piros kockás szőnyeg	1044	
Plázakábel 13mm	774	
Benzin (E95)	1393	
Teherautó	32	
Szedán autó	44	
Acélkábel 16mm	945	
Piros hosszú szálú szőnyeg	889	
Acélkábel 40mm	107	
Vízszivág	1175	
Háromos	1150	
Fehér hosszú szálú szőnyeg	1052	
Zöld políész szőnyeg	1172	
Zöld szálú szőnyeg	812	
Kostol (gr. -gyűrűvel)	1184	
LNG	793	
Termék megtalálásának részletei:		
Raktár megtalálásának részletei:		
Termék megtalálásának részletei:		
Sorozatszámok megtalálásának részletei:		

Telephely megtalálása		
Típus	Mennyiségek	Elérhető
Acélkábel 1...	619 Meter	575 Meter
Plátna teher...	55 Darab	53 Darab
Rézsík 3...	976 Meter	841 Meter
Benzin (95)	1089 Liter	1064 Liter
Föld	757 Köröme...	736 Köröme...
Kukurica	1446 Horde...	1389 Horde...
Piros kocka...	1044 Negye...	995 Negye...
Rézsík 13...	774 Meter	738 Meter
Benzin (E95)	1393 Liter	1352 Liter
Teherautó	32 Darab	29 Darab
Szedán autó	44 Darab	41 Darab
Acélkábel 1...	945 Meter	928 Meter
Piros hossz...	889 Negye...	806 Negye...
Acélkábel 4...	1109 Meter	1027 Meter
Vízszivág	1175 Hordó...	1142 Hordó...

7.5 Használati példák

7.5.1 Fatelep

Tegyük fel, hogy egy favágó vállalatunk több fatelephellyel rendelkezik. Mivel a fakitermelés és annak értékesítése folyamatos folyamat, így a telephelyeken található fa mennyisége is rendszeresen változik. A telephelyek területe véges, ebből kifolyólag elengedhetetlen egy olyan nyilvántartó rendszer alkalmazása, amely pontosan követi, hogy egyes telepeken 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 készletünk.

Amennyiben a fatelep feldolgozással is foglalkozik, és különböző formájú (pl. rönk, hasáb, deszka, gerenda) és méretű faanyagokkal dolgozik, azok megfelelő nyilvántartása és kezelésük egyszerűsítése érdekében alkalmazásunk ideális megoldást kínál. A rendszer lehetővé teszi a különféle faelemek pontos nyomon követését, biztosítva ezzel a hatékony és átlátható működést.

7.5.2 Bevásárló központ

Ebben a példában egy bevásárlóközpont üzemeltetésével foglalkozunk, amely több üzettel 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 valamelyik cikk és utánpótlást kell küldeni.

A nyilvántartás kezelő rendszerünkben munkakörnek megfelelő jogosultságokkal lehet ellátni az egyes felhasználókat, ezzel elkerülve, 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.

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.

Felhasznált irodalom

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

Mellékletek

10.1 Felhasznált könyvtárak

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

10.2 Online elérhetőség

| TODO