Szoftverarchitektúrák

Raktárkezelő rendszer

*Készítette: Kamrás Márton (NEPTUN)*

*Kiss Rebeka Enikő (EVLIR3)*

Tartalom

[Összefoglaló 3](#_Toc436503192)

[Rendszerterv 4](#_Toc436503193)

[Architektúra 4](#_Toc436503194)

[Adatbázis terv 5](#_Toc436503195)

[Biztonság 6](#_Toc436503196)

[Felhasználói felület 6](#_Toc436503197)

[Funkciók 6](#_Toc436503198)

[Általános funkciók 6](#_Toc436503199)

[Alkalmazotti funkciók 6](#_Toc436503200)

[Tulajdonosi funkciók 7](#_Toc436503201)

[Menedzseri funkciók 7](#_Toc436503202)

[Telepítési útmutató 8](#_Toc436503203)

[Üzemeltetői 8](#_Toc436503204)

[Felhasználói 8](#_Toc436503205)

[Továbbfejlesztési lehetőségek 9](#_Toc436503206)

[Verziókövetés 10](#_Toc436503207)

# Összefoglaló

A Raktárkezelő rendszer egy böngészőből elérhető alkalmazás, melynek segítségével a cégek a figyelemmel kísérhetik raktárkészleteiket. Az alkalmazás web szerverre való telepítés után bárhonnan elérhető, így nem igényel speciális felszerelést.

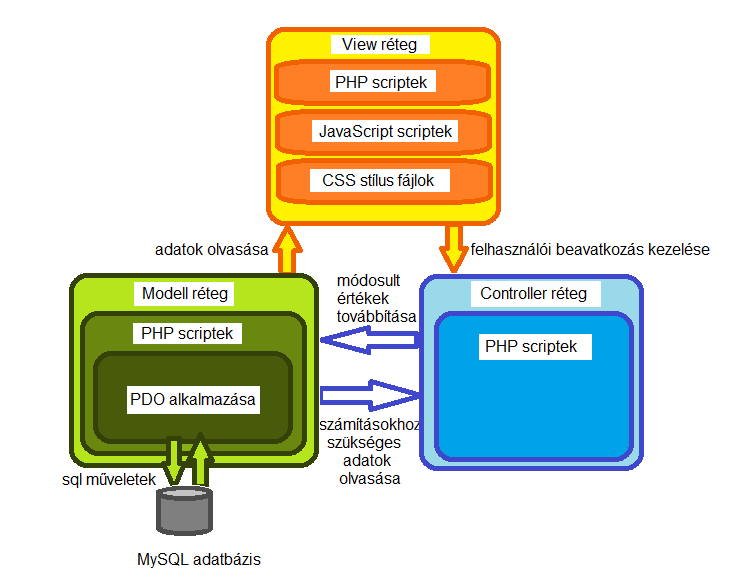
A raktárakat egyszerre, felsorolás szinten is megtekinthetik, vagy kiválasztva az adott raktárt (esetleg egy minta alapján újat hozzáadva) annak készleteit, a raktárak közti készlet mozgásokat monitorozhatják. Az egész napos számítógép előtt ülést elkerülendő, az alkalmazás bizonyos eseményekről értesítést küldhet. Továbbá az örökmozgók kedvéért egy mobilra optimalizált honlapot is biztosítunk.

A biztonságért a jelszavaknál egy egyszerű titkosítás felel, míg a kommunikáció https felett történik.

A felhasználói felület kialakításánál az elsődleges szempont a funkciók könnyű elérése volt, hiszen az alkalmazást használóknak sok adatot kell egyszerre átlátni.

# Rendszerterv

## Architektúra



. ábra

A program a *Model-View-Controller* architektúrát követi, kiegészítve az adatbázis réteggel, melyet a *Wrapper* mintával kapcsoltunk az alkalmazáshoz. A *Wrapper* mintával az SQL utasításokat akartuk elfedni, így elkerülendő az esetleges elrontott vagy épp rosszindulatú parancsok végrehajtását.

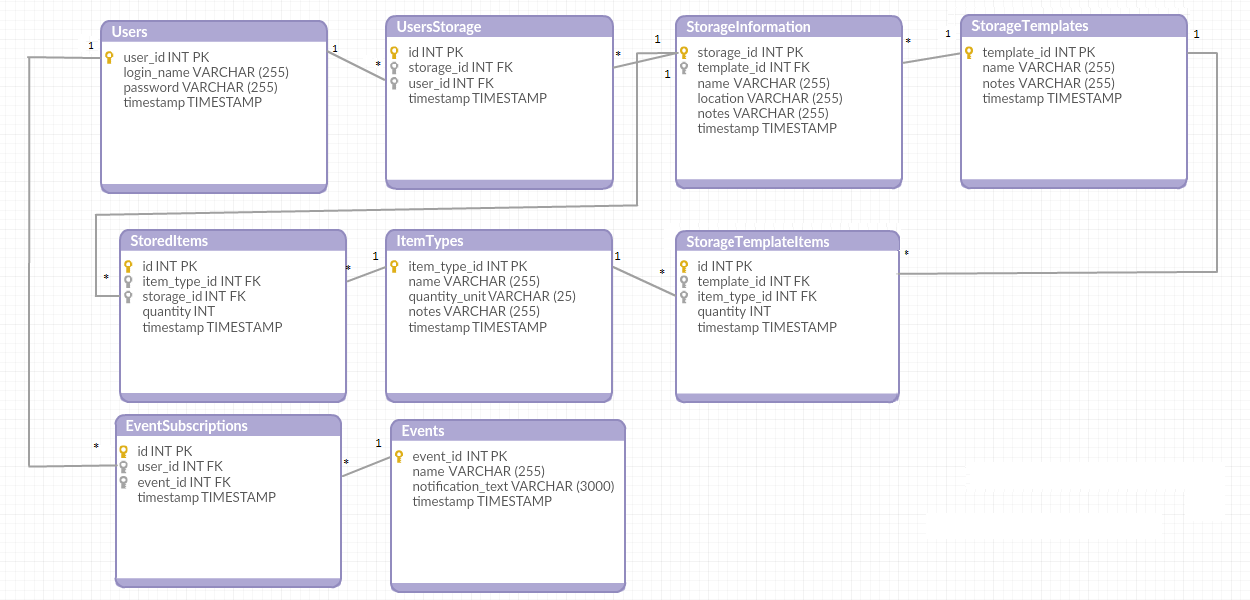
A *Model* réteg kommunikál a MySQL szerverrel, ahol az adatbázis tárolódik. A *Model* réteg osztályai felelősek az adatok tárolásáért.

A *Controller* réteg tartalmazza az üzleti logikát, a *Model* réteg osztályainak adatai alapján. Közvetlenül már nem fér hozzá az SQL utasításokhoz, csak a *Model* által publikált metódusokat ismeri.

A *View* réteg pedig csak a megjelenítésért felel, a megjelenítendő adatokat a modellektől kéri el. A felhasználó által módosított mezőket a *Controllernek* adja át, aki elvégzi a szükséges műveleteket.

A megjelenítő rétegben további mintaként megjelenik a *Component configurator*. Ez az egyes felhasználók által látott modulok betöltésekor kap szerepet. A betöltendő modulokat a program egy konfigurációs fájlból olvassa fel, s ebben a fájlban van leírva az egyes felhasználói szerepek által látható modulok listája is. Ez futás időben is változtatható. Az egyes modulok betöltésért egy script felel, amely az oldal frissítésekor lefut.

## Adatbázis terv



. ábra

Az adatbázis 4 fő részre bontható:

A felhasználókra (*Users* tábla). Ebben a táblában mentődnek el a felhasználó belépési adatai. A felhasználó név egyszerű karaktersorozatként tárolódik, de a jelszónál, csak a jelszóhoz tartozó hash-t mentjük le.

A raktárak (*UsersStorage, StorageInformation, StorageTemplateItems, StorageTemplates* táblák). Ezekben a táblákban találhatóak meg a raktárak leírásai (*StorageInformation*), a felhasználók által használt raktárak (*UsersStorage*), illetve az új raktárak létrehozásához szükséges raktárminták (*StorageTemplates*). A *StorageTemplateItems* tartalmazza a raktárkészlet leírását, raktárminta – termék típus – mennyiség hármasokból felépítve.

A raktárak leírásainál meg kell adnunk, hogy milyen minta alapján készült a raktár, mi a neve, hol található. Emellett egyéb jegyzetet is megadhatunk. A különböző raktármintáknak nevet adhatunk, s megjegyzést csatolhatunk hozzájuk.

Az egyes felhasználókhoz tartozó raktárakhoz nem csatolunk más információt.

A tárolandó termékek/készletek (*StoredItems, ItemTypes*). Az *ItemTypes* a lehetséges termékek leírását adja meg (név, mértékegység párosokkal, és esetleges megjegyzésekkel), míg a *StoredItems* a konkrét készleteket raktáranként (miből, melyik raktárban, mennyi van).

Végül pedig az események (*Events, EventSubscriptions*). A felhasználók különböző eseményekre iratkozhatnak fel, s ezekről értesítést kaphatnak. A lehetséges eseményeket az *Events* tábla, míg az egyes feliratkozásokat az *EventSubsriptions* tábla tartalmazza. Egy esemény névből és egy értesítési szövegből áll.

Emellett minden tábla tartalmaz egy *timestamp* mezőt, melyben az adott rekord hozzáadásának vagy módosításának időpontja mentődik.

## Biztonság

A felhasználók jelszavainak védelme „*Secure Salted Password Hashing*”-gel történik, ami a következőképpen működik: a felhasználó megadja a jelszavát, a rendszer hozzáad egy előre meghatározott sót, majd ennek a karaktersorozatnak veszi a hash leképezését, s ezt a leképezést hasonlítja össze az adatbázisban tárolttal. Ha egyezik, az authentikáció sikeres és a felhasználó bejelentkeztetése megtörténik.

A hálózati forgalom lehallgatása ellen *HTTPS* protokollt használ a program, melyet a konfigurációban határoztunk meg. A *https* széles körben elterjedt protokoll, ezért részletesen most nem ismertetjük[[1]](#endnote-1).

Az adatbázissal rendelkező rendszerek esetén szükség van még *SQL injection*[[2]](#endnote-2) elleni védekezésre is. A program ezt *PDO* (*PHP Data Objects*)[[3]](#endnote-3) alkalmazásával védi ki. A *PDO* egy objektum az adatbázis kapcsolatok, lekérdezések, stb. kényelmes, hatékony, átlátható kezelésére. A *PDO*-val ellátott modellek már nem publikálnak SQL parancsokat a fentebbi rétegeknek, emellett *prepared statement*-eket használ az alkalmazás, ami az elsődleges védelmi vonal az *SQL injection* ellen.

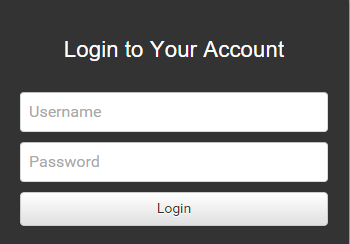
A felhasználó jogosultságai szerint változik az elérhető modulok, gombok láthatósága a felhasználói felületet, így a jogosulatlan hozzáférést igyekszik a program megakadályozni. Az egyes modulokat hozzárendelhetjük szerepkörökhöz, így meghatározva, hogy ki láthatja. Ez konfigurációs fájlból, illetve *on-the-fly* módon is meghatározható a *Component Configurator* mintának köszönhetően.

## Felhasználói felület

## Funkciók

### Általános funkciók

Bejelentkezés: a felhasználó belép a felületre, megadja felhasználó nevét és jelszavát, a rendszer az authentikáció után tovább engedi.



3. ábra – bejelentkezési felület (host.cim/sm/Login)

Kijelentkezés: a felhasználó már korábban belépett a felületre és 10 percig inaktív. A rendszer automatikusan kijelentkezteti.

A felhasználó korábban belépett a felületre, még nem telt el 10 percnyi inaktivitás, s a „Logout” gombra kattint. A rendszer kijelentkezteti.

### Alkalmazotti funkciók

*A következő funkcióknál feltételezzük, hogy a felhasználó bejelentkezett és alkalmazott (employee) szintű jogosultságai vannak.*

Összes raktár lekérdezése:

Adott raktárhoz tartozó információk lekérése (raktárban lévő tételek neve, készlet)

Új tételek felvétele

Tételek törlése

Tételek mennyiségének módosítása

Be- és kiszállítások adatai (melyik raktárba, melyik raktárból, mit, mennyit)

Be- és kiszállítások felvétele

Be- és kiszállítások törlése

Statisztikák lekérdezése (üres helyek száma, tételfogyás)

Statisztikák létrehozása

### Tulajdonosi funkciók

*A következő funkcióknál feltételezzük, hogy a felhasználó bejelentkezett és tulajdonos (owner) szintű jogosultságai vannak.*

Új raktár felvétele a rendszerbe

Raktár törlése

Raktár adatainak módosítása

Sablonok létrehozása

Sablonok módosítása

Sablonok törlése

Értesítési események konfigurálása

### Menedzseri funkciók

*A következő funkcióknál feltételezzük, hogy a felhasználó bejelentkezett és menedzser (manager) szintű jogosultságai vannak.*

Statisztikák lekérdezése

Egyéni lekérdezések létrehozása

Regisztráció értesítési eseményekre

# Telepítési útmutató

### Üzemeltetői

Előfeltételek:

* egy működő szerver, szervergép, -park, (főként apache szerveren tesztelve)
* működőképes MySQL szerver

Ha az előfeltételek teljesültek, akkor:

* az *apache szerver* adminisztrátori felületén deployolja az alkalmazást (pontos utasítások a Tomcat dokumentációjában[[4]](#endnote-4)),
* hozzon létre egy új adatbázist a *MySQL* szerveren „*StorageManager*” néven[[5]](#endnote-5),
* importálja az adatbázisba az alkalmazás db mappájában található *StorageManager.sql* –t,
* vegyen fel új felhasználókat, s töltse fel az adatbázist a szükséges adatokkal

### Felhasználói

Előfeltételek:

* egy már működő számítógép vagy mobiltelefon, operációs rendszerrel, a felhasználó számára hozzáférhető felhasználói fiókkal,
* internet hozzáférés,
* weblapok megtekintésére alkalmas böngésző
* az üzemeltető publikálta az alkalmazást
* az üzemeltető létrehozott az alkalmazásban egy felhasználót

Ha az előfeltételek teljesültek, akkor:

* Nyisson meg egy új ablakot a böngészőben
* Navigáljon az üzemeltető által megadott honlapra
* Jelentkezzen be az üzemeltetőtől kapott felhasználónévvel és jelszóval
* használja a rendszert.

# Továbbfejlesztési lehetőségek

A programot több szempont szerint megvizsgáltuk, ezen szempontok szerint a következő lehetőségeket találtuk.

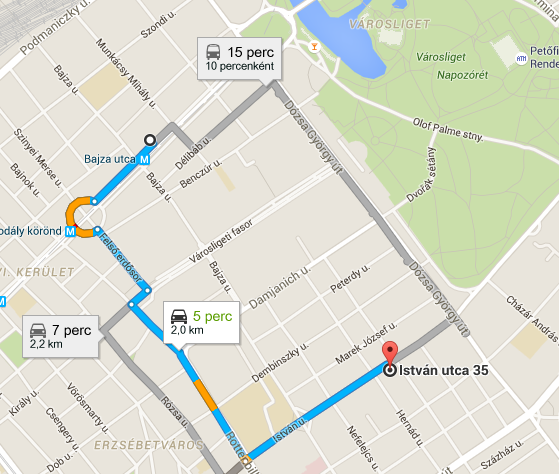
*Biztonsággal kapcsolatos fejlesztés:*

Az authentikációhoz tartozó jelszó tárolását egyszerű hasheléssel oldottuk meg („salt” hozzáadásával). Ez egy alapszintű titkosítás, az egyik lehetőség ennek a titkosításnak a lecserélése egy bonyolultabbra.

*Felhasználói felület fejlesztései:*

A raktárak helyszíneit lehetséges lenne térképen is megjeleníteni (pl. Google Maps), ezzel segítve a felhasználót a raktárak helyzeteinek áttekintését.

A raktárak közötti szállítási idő nem szerepel az adatbázisban, s nem is számítható jelenleg. A Google Maps útvonaltervező szolgáltatását integrálva, viszont ez lehetségessé válik, bár a kötelező pihenő időt nem adja a szolgáltatás hozzá, ezzel ki kell egészíteni a pontosabb számításokhoz.



4. ábra

Ezzel a fejlesztéssel tervezhetőbb lenne a kamionok lepakolása, illetve a raktár telítettségének korlátozása.

A raktárak telítettségét a felhasználói felületen egy dedikált ábra jelezhetné dinamikusan. Az ábra egy lehetséges megvalósítása:

* 0-40% 
* 40-80% 
* 80-100% 

# Verziókövetés

|  |  |  |  |
| --- | --- | --- | --- |
| Szerkesztő neve | Szerkesztés ideje | Módosítások | Új verziószám |
| Kiss Rebeka | 2015. 10. 01. | Dokumentum létrehozása | v0.1 |
| Kiss Rebeka | 2015. 10. 27. | Összefoglaló hozzáadása | v1.0 |
| Kamrás Márton | 2015. 10. 28. | Előzetes architektúra tervek | v1.1 |
| Kiss Rebeka | 2015. 11. 03. | Előzetes adatbázis tervek | v1.2 |
| Kiss Rebeka | 2015. 11. 04. | Funkciók leírása | v1.3 |
| Kamrás Márton | 2015. 11. 05. | Telepítési útmutató kitöltése | v1.4 |
| Kamrás Márton | 2015. 11. 10. | Felhasználói felület leírása | v1.5 |
| Kiss Rebeka | 2015. 11. 10. | Dokumentáció hiányosságainak pótlása | v1.6 |
| Kiss Rebeka | 2015. 11. 28. | Dokumentum formázás, véglegesítés | v2.0 |

# Irodalomjegyzék

1. <https://en.wikipedia.org/wiki/HTTPS> [↑](#endnote-ref-1)
2. <https://en.wikipedia.org/wiki/SQL_injection> [↑](#endnote-ref-2)
3. <http://php.net/manual/en/book.pdo.php> [↑](#endnote-ref-3)
4. <http://tomcat.apache.org/tomcat-7.0-doc/> [↑](#endnote-ref-4)
5. <http://dev.mysql.com/doc/refman/5.7/en/creating-database.html> [↑](#endnote-ref-5)