Vasúti információszolgáltató és jegykiadó rendszer Programfejlesztési dokumentáció 2011

# Project összefoglaló

A cél egy olyan online vasúti információs rendszer kifejlesztése, amely teljes körűen kezelni tudja az egyes környéken közlekedő vonatokat és azok menetrendjét. A járatok meghatározott útvonalak között közlekednek, így mindig ugyanazokat a megállókat érintik. A járatokhoz egy megállósorozat tartozik, amely az adott vonat útját írja le. A járatokhoz a különböző indulási időpontok tartoznak. A megállókat külön is lehet kezelni. A felhasználónak lehetősége nyílik az on-line vasúti menetrend böngészésére és jegyvásárlásra, amelyről egy nyomtatható formátumú elismervényt kap. A rendszert egy admin felületen keresztül tarthatjuk karban.

**Verziók**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Verzió | Szerző | Dátum | Státusz | Megjegyzés |
| 0.1 | Makra Tibor | 2011-11-08 | Tervezet | Legelső verzió |
|  |  |  |  |  |

Státusz osztályozás:

Tervezet Befejezetlen dokumentum

Előterjesztés A projekt menedzser bírálatával

Tartalom

[1 Project összefoglaló 1](#_Toc308981018)

[Tartalom 2](#_Toc308981019)

[2 Követelmények 3](#_Toc308981020)

[2.1 Funkcionális követelmények 3](#_Toc308981021)

[2.2 Nem funkcionális követelmények 4](#_Toc308981022)

[2.3 A rendszer működésének szemléltetése 5](#_Toc308981023)

[2.4 Osztályok 6](#_Toc308981024)

[3 Programterv 10](#_Toc308981025)

[3.1 Csomagok 10](#_Toc308981026)

[3.2 Alkalmazás logika 12](#_Toc308981027)

[3.3 Adatbázis 16](#_Toc308981028)

[3.4 Felhasználói felület 16](#_Toc308981029)

# Követelmények

## Funkcionális követelmények

A felhasználó az alábbi funkciókat érheti el:

* két város között közlekedő vonatok menetrendjének listázása indulási és érkezési idővel
* megállóhelyen átmenő vonatok listázása
* jegyvásárlás, nyomtatható formátumú elismervény
* adott dátumra vonatkozó járatok keresése
* átszállás megjelenítése
* díjszabás lekérdezése, kedvezmény beállítása
* az utazás időtartamának lekérdezése
* a két város közötti távolság lekérdezése (km-ben)
* adott útvonalon gyorsvonatok listázása
* kerékpárszállító járatok lekérdezése
* állat szállítást vállaló járatok lekérdezése

Az adminisztrátor az alábbi funkciókat érheti el bejelentkezés után:

* összes felhasználó által elérhető funkció
* járat hozzáadása
* járat attribútumainak módosítása
* járat törlése
* megálló hozzáadása
* megálló törlése
* megálló attribútumainak módosítása

## Nem funkcionális követelmények

Fejlesztési környezet:

* Windows vagy Linux operációs rendszer
* Apache Tomcat 7.0
* Eclipse Helios
* PostgreSql
* Eclipse Subclips (SVN)
* JDK 1.7

Futtatási környezet:

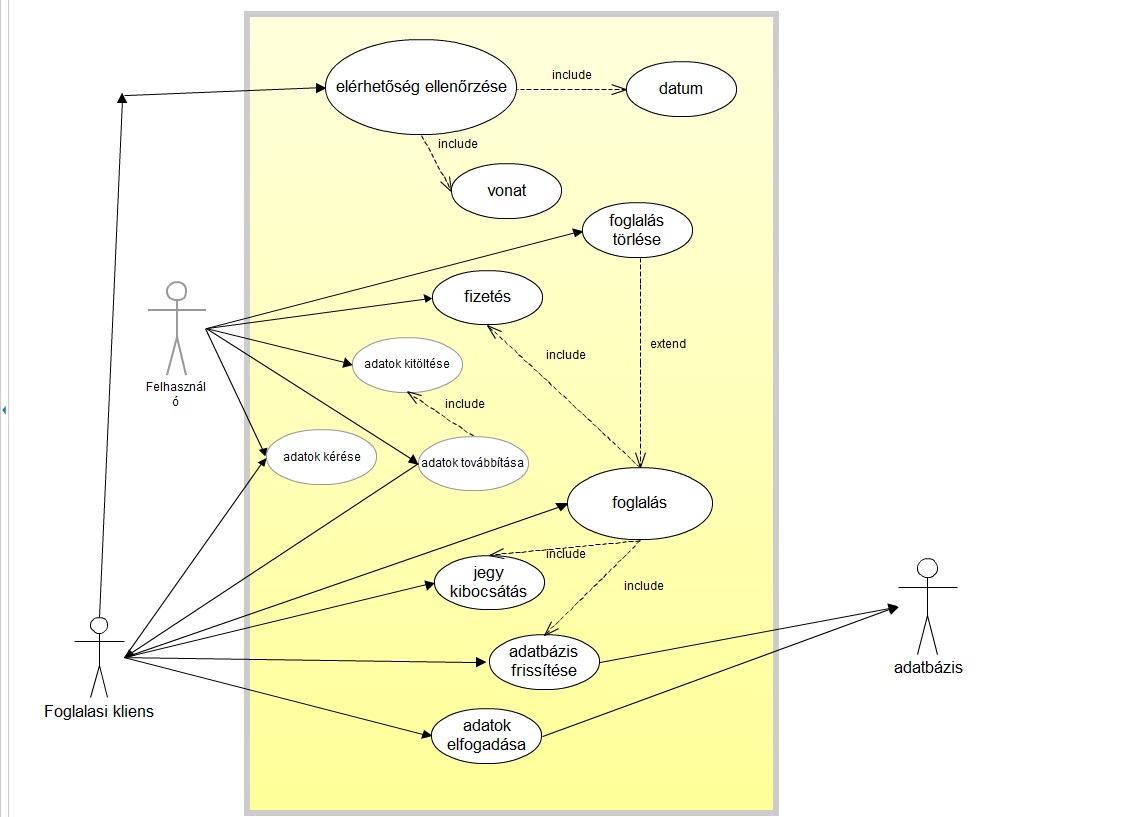
* Tetszőleges operációs rendszer
* Tetszőleges böngésző

Tesztelési környezet

* Eclipse JUnit
* SourceInventory
* Test Link
* Apache JMeter

## A rendszer működésének szemléltetése

A rendszer működését a következő használati eset diagramon keresztül szemléltetjük:

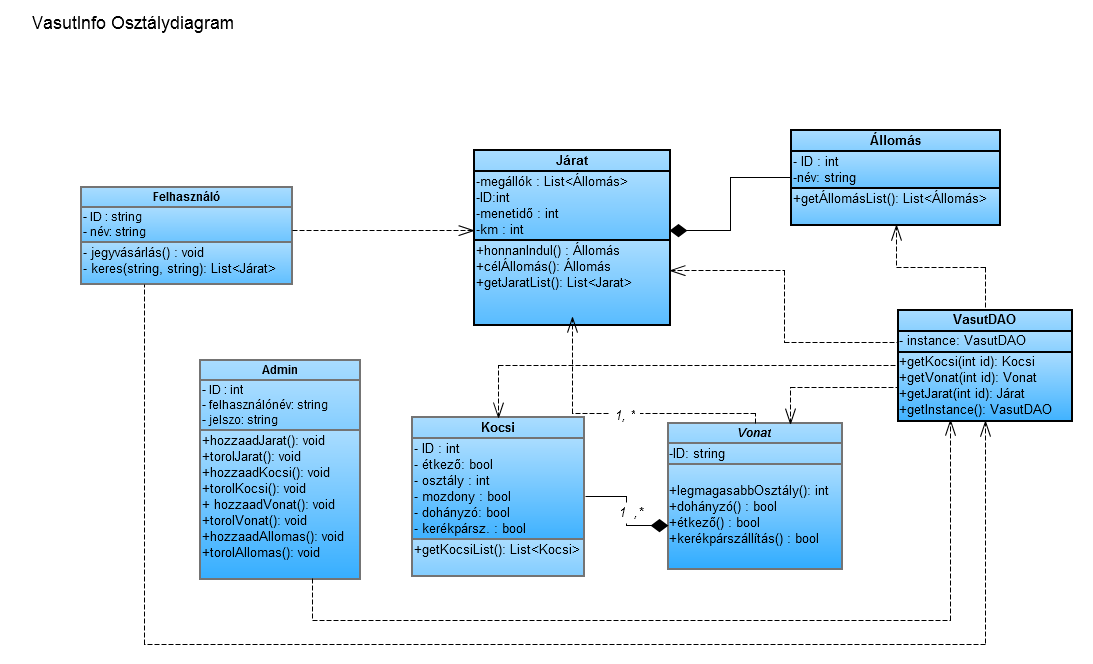


Ez a diagram a jegyvásárlás folyamatát szemlélteti.

* A felhasználó lekérdezi az adatokat az adott vonatokról, ezt továbbítja a foglalás kliensnek.
* A foglalás kliens ellenőrzi az elérhetőségeket, küldi az információkat a felhasználónak.
* Foglalás esetén a felhasználó továbbítja a kérést a foglalási kliensnek, aki kibocsájtja a jegyet.
* A foglalási kliens további feladatai hogy kommunikáljon az adatbázissal, frissítse az adatbázist.

## Osztályok

Az osztályok felépítését az alábbi diagram szemlélteti:



Felhasználó osztály:

Változók:

* ID: szöveg típusú változó a felhasználó azonosító tárolására
* név: szöveg típusú változó a felhasználó nevének tárolására

Metódusok:

* jegyvásárlás(): Jegyvásárlás metódus
* keres(string, string): List<Járat>: Járat keresése metódus, két szöveg változót kap és egy listával tér vissza

Admin osztály:

Változók:

* ID: egész típusú változó az admin azonosító tárolására
* név: szöveg típusú változó az admin nevének tárolására
* jelszó: egész típusú változó az admin jelszavát tárolja

Metódusok:

* hozzaadJarat(): Járat létrehozása metódus, nincs visszatérési érték
* törölJarat(): Járat törlő metódus, nincs visszatérési érték
* hozzaadKocsit(): Kocsi létrehozása metódus, nincs visszatérési érték
* törölKocsi(): Kocsi törlő metódus, nincs visszatérési érték
* hozzaadVonat(): Vonat létrehozása metódus, nincs visszatérési érték
* törölVonat(): Vonat törlő metódus, nincs visszatérési érték
* hozzaadAllomas(): Állomás létrehozása metódus, nincs visszatérési érték
* törölJarAllomas(): Állomástörlő metódus, nincs visszatérési érték

Járat osztály:

Változók:

* megállók: állomások listáját tárolja
* ID: egész típusú változó, azonosítót tárol
* menetidő: a menetidőt tárolja egy egész típusú változóban
* km: a megtett utat tárolja egy egész típusú változóban

Metódusok:

* honnanIndul(): Egy állomás típusú változó ami az állomást tárolja ahonnan indul a vonat
* célÁllomás(): A cél állomást tárolja egy állomás típusú változóban
* getJaratList(): Állomások listáját adja vissza egy listában ami járat típusú változókat tárol

Kocsi osztály:

Változók:

* ID: szöveg típusú változó a kocsi azonosítót tárolására
* étkező: egy bool típusú változó, amely értéke 1, ha étkező kocsi, 0 ha nem étkező kocsi
* osztály: egy egész típusú változóban tárolja a kocsi osztályát
* mozdony: egy bool típusú változó, amely értéke 1, ha a kocsi mozdony, 0 ha nem mozdony
* dohányzó: egy bool típusú változó, amely értéke 1, ha a kocsi dohányzó, 0 ha nem dohányzó
* kerékpársz: egy bool típusú változó, amely értéke 1, ha a kocsi kerékpárszállító 0, ha nem kerékpár szállító

Metódusok:

* getKocsiList(): Egy listát ad vissza ami kocsi típusú változókat tartalmaz, ezzel lehet lekérdezni a vonat Kocsijait

Vonat osztály:

Változók:

* ID: szöveg típusú változó a vonat azonosítót tárolására

Metódusok:

* legmagasabbOsztály(): egy egész típusú értéket ad vissza, a vonat legmagasabb kocsiosztályát
* dohányzó(): egy bool típusú változót ad vissza, 1-et, ha a kocsi dohányzó, 0-t, ha nem
* étkező(): egy bool típusú változót ad vissza, 1-et, ha a kocsi étkező, 0-t, ha nem
* kerékpárszállítás():egy bool típusú változót ad vissza, 1-et, ha a kocsi kerékpárszállító, 0-t, ha nem

Állomás osztály:

Változók:

* ID: szöveg típusú változó az állomás azonosítót tárolására
* név: szöveg típusú változó a állomás nevének tárolására

Metódusok:

* getÁllomásList(): egy listát ad vissza ami az állomásokat tárolja

VasutDAO osztály:

Változók:

* instance: adatbázis beállítására szolgál

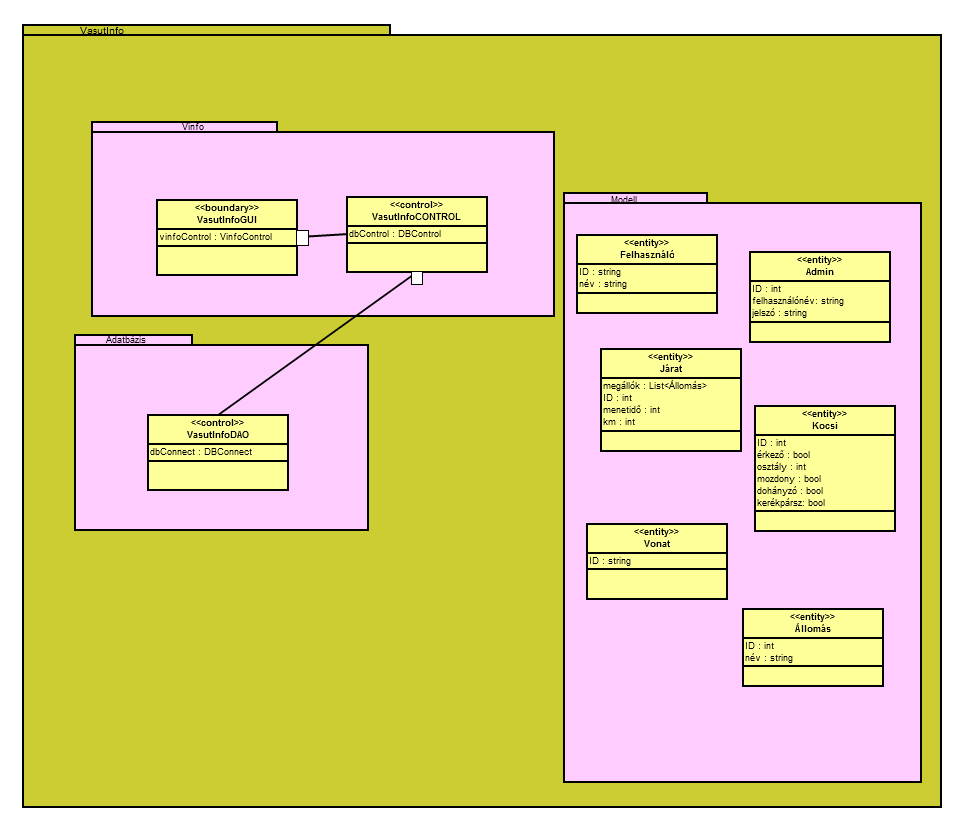
Metódusok:

* getKocsi(int id): egy kocsit kérdez le az adatbázisból
* getVonat(int id): egy vonatot kérdez le az adatbázisból
* getJarat(int id): egy járatot kérdez le az adatbázisból
* getInstance(): VasutDAO-t kérdezi le

# Programterv

## Csomagok

A csomagokat a következő diagram szemlélteti:



A Modell csomag

A Modell csomag tartalmazza az osztály leírások során már leírt osztályokat:

* Felhasználó osztály
* Admin osztály
* Járat osztály
* Kocsi osztály
* Vonat osztály
* Állomás osztály

A Vinfo csomag

A Vinfo csomag két osztályt tartalmaz:

* VasutInfoGUI
* DBControl

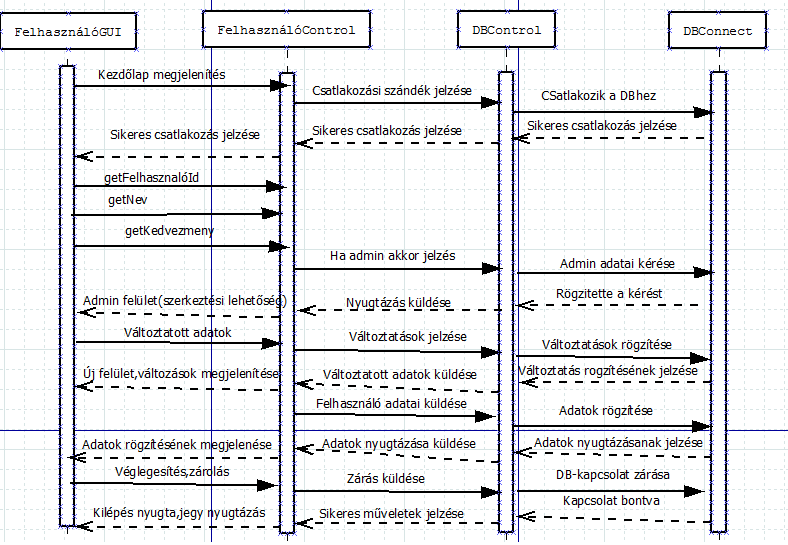
A VasutInfoGUI az felhasználói felület megjelenítéséért, a DBControl osztály az adatbázis csomaggal való kommunikációért felel.

Az Adatbázis csomag

Az adatbázis csomag tartalma a VasutInfoDAO-t, amely az adatbázissal való kommunikációért felel.

## Alkalmazás logika

Felhasználó szekvencia diagram:



* A FelhasználóGUI megjeleníti a felhasználói felületet, a FelhasználóControl jelzi a csatlakozási szándékát a DBControlnak amely csatlakozik az adatbázishoz.
* Sikeres csatlakozás esetén egy jezés továbbítódik egészen a FelhasználóGUI-ig.
* A FelhasználóGUI bekéri a felhasználó id-jét, nevét, ezeket küldi a FelhasználóControl-nak.

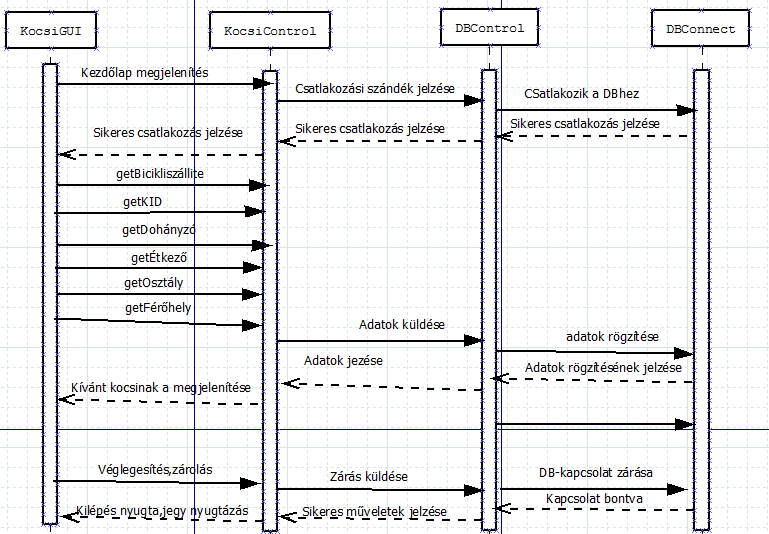
Admin bejelentkezése esetén:

* FelhasználóControl küld egy jezést a DBControl-nak, ami lekérdezi az Admin adatait az Adatbázistól a DBConnect-en keresztül.
* Az adatbázis rögzíti a kérést, üzenetet küld a DBControl-nak, amely nyugtázást küld a FelhasználóControl-nak, amely üzenetet küld a FelhasználóGUI-nak hogy megjelenítheti az Admin felületet.
* A felhasználóGUI a megáldoztatott adatokat küldi tovább teljesen az Adatbázisig ami rögzíti azokat.
* A DBConect jelzi a DBControl-nak hogy az adatok megváltoztak, a DBControl elküldi a megváltozott adatokat a FelhasználóControl-nak, amely továbbküldi a FelhasználóGUI-nak ami megjeleníti azokat.

Felhasználó bejelentkezése esetén:

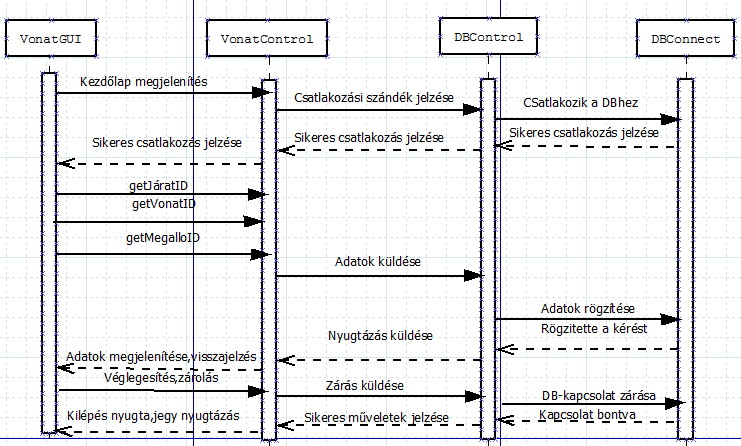
* FelhasználóControl küld egy jezést a DBControl-nak, ami lekérdezi az Felhasználó adatait az Adatbázistól a DBConnect-en keresztül.
* Az adatbázis rögzíti a kérést, üzenetet küld a DBControl-nak, amely nyugtázást küld a FelhasználóControl-nak, amely üzenetet küld a FelhasználóGUI-nak hogy jelenítse meg a felhasználói felületet.
* A FelhasználóGUI üzenetet küld a FelhasználóControl-nak, a véglegesítésről, zárolásról, az zárást küld a DBControl-nak amely üzenetet küld a DBConnectnek hogy zárja a kapcsolatot az adatbázissal.
* Sikeres művelet esetén a DBConnect üzenetet küld teljesen a FelhaszálóGUI-ig hogy a zárolás sikeres.

Kocsi szekvencia diagram:



* A KocsiGUI megjeleníti a kezdőlapot, jelzést küld a KocsiControl-nak, az jelzi csatlakozási szándékát a DBControl-nak, amely csatlakozik az adatbázishoz.
* A DBConnect jezést küld teljesen a KocsiGUI-ig hogy a csatlakozás sikeres.
* A KocsiGUI adatokat küld a KocsiControl-nak , KocsiControl tovább küldi a DBControlnak amely továbbküldi az adatbázisnak a kérést.
* A DBConnectől eljutnak egészen a KocsiGUI-ig a kért adatok, amely megjeleníti azokat.
* A KocsiGUI üzenetet küld a KocsiControl-nak, a véglegesítésről, zárolásról, az zárást küld a DBControl-nak amely üzenetet küld a DBConnectnek hogy zárja a kapcsolatot az adatbázissal.
* Sikeres művelet esetén a DBConnect üzenetet küld teljesen a KocsiGUI-ig hogy a zárolás sikeres.

Vonat szekvencia diagram:



* A VonatGUI megjeleníti a kezdőlapot, jelzést küld a VonatControl-nak, az jelzi csatlakozási szándékát a DBControl-nak, amely csatlakozik az adatbázishoz.
* A DBConnect jezést küld teljesen a VonatGUI-ig hogy a csatlakozás sikeres.
* A VonatGUI adatokat küld a VonatControl-nak , VonatControl tovább küldi a DBControlnak amely továbbküldi az adatbázisnak a kérést.
* A DBConnectől eljutnak egészen a VonatGUI-ig a kért adatok, amely megjeleníti azokat.
* A VonatGUI üzenetet küld a VonatControl-nak, a véglegesítésről, zárolásról, az zárást küld a DBControl-nak amely üzenetet küld a DBConnectnek hogy zárja a kapcsolatot az adatbázissal.
* Sikeres művelet esetén a DBConnect üzenetet küld teljesen a VonatGUI-ig hogy a zárolás sikeres.

## Adatbázis

Az adatbázis felépítését a következő egyed-kapcsolat diagram szemlélteti:

