Fejlesztői Dokumentáció

7. Feladat: Harry Potter

Készítette: Kosztik Anett
Neptun : ZSWVRW

Készítette: Kosztik Anett
neptun Kód: ZSWVRW

Fejlesztői dokumentáció (1. rész)

1.) **Bevezetés**

- A projekt:
  a **mágiaügyi nyilvántartó rendszer** megvalósítása.

- Ez egy olyan alkalmazás, amelyben nyilvántartható a diákok elhelyezése és továbbá feljegyezhetőek a környéken élő ismert lények is.
- Cél közösség a Roxfort (Albus Dumbledor) és a Mágiaügyi Minisztérium.
- Ebben a nyilvántartó rendszerben minden varázslótanonc és ismert lény számontartható, kereshető. Megkönnyítheti az adminisztrációt a Minisztériumban.

2.) **Feladat leírása**

Harry Potter elvarázsolt világába is belopózott a számítástechnika.
Roxfortban a felvett varázsló ifjoncokat a Teszlek Süveg osztotta be az iskola 4 házának (Griffendél, Hugrabug, Hollóhát, Mardekár) egyikébe, azonban a hosszú évek során szegény süveg elkopott. Albus Dumbledor-nak új megoldást kellett találnia a diákok elhelyezésének és számontartásának a biztosítására, továbbá fel szeretné jegyezni a környéken élő ismert lényeket is. 2 nap tanakodás után felkérte Pomona Bimba professzort, hogy termesszen mágikus babot (NetBeans), melyet felhasználva létrejöhet a mágiaügyi nyilvántartó rendszer. Dumbledor a történtek után felkért téged, hogy készítsd el az alkalmazást. Minimum elvárásai a programmal kapcsolatban:

• Legyenek megtekinthetőek a házak, tanulók, a lények és a jellemek.
• Legyen lehetősége új ház felvitelére a ház nevének és címerének megadásával. (Az iskola bővülhet új házakkal.)
• Legyen lehetősége új tanulók felvételére a tanuló nevének és életkorának megadásával. Fontos, hogy egy tanuló tartozhat valamelyik házhoz!
• Legyen lehetőségünk új lény felvitelére a lény nevének megadásával.
• Minden lényről és tanulóról számon tartjuk, hogy milyen a jelleme. A jellem lehet törvényes jó, semleges vagy kaotikus gonosz. Új lények és tanulók esetén tegyük fel, hogy mindenki semleges. A jellemet lehessen bővíteni.
• A tanulókról tudjuk, hogy a nevük nem változhat meg. A jellemek elnevezése sem változhat. A házak esetén csak a házhoz tartozó tanulók száma változhat, azonban ez a mező sem szerkeszthető felhasználói interakcióval. A lények esetében számon tartunk egy első találkozási dátumot is, melyet csak abban az esetben lehet szerkeszteni, ha még nem volt töltött.
• Legyen lehetőségünk szűrni tanulókra és lényekre a nevük alapján. A felületen nem szükséges megjeleníteni az azonosítókat.

Az adatbázisban az alábbi adatokat tároljuk el (ezek még nem feltétlenül a fizikai adattáblák):

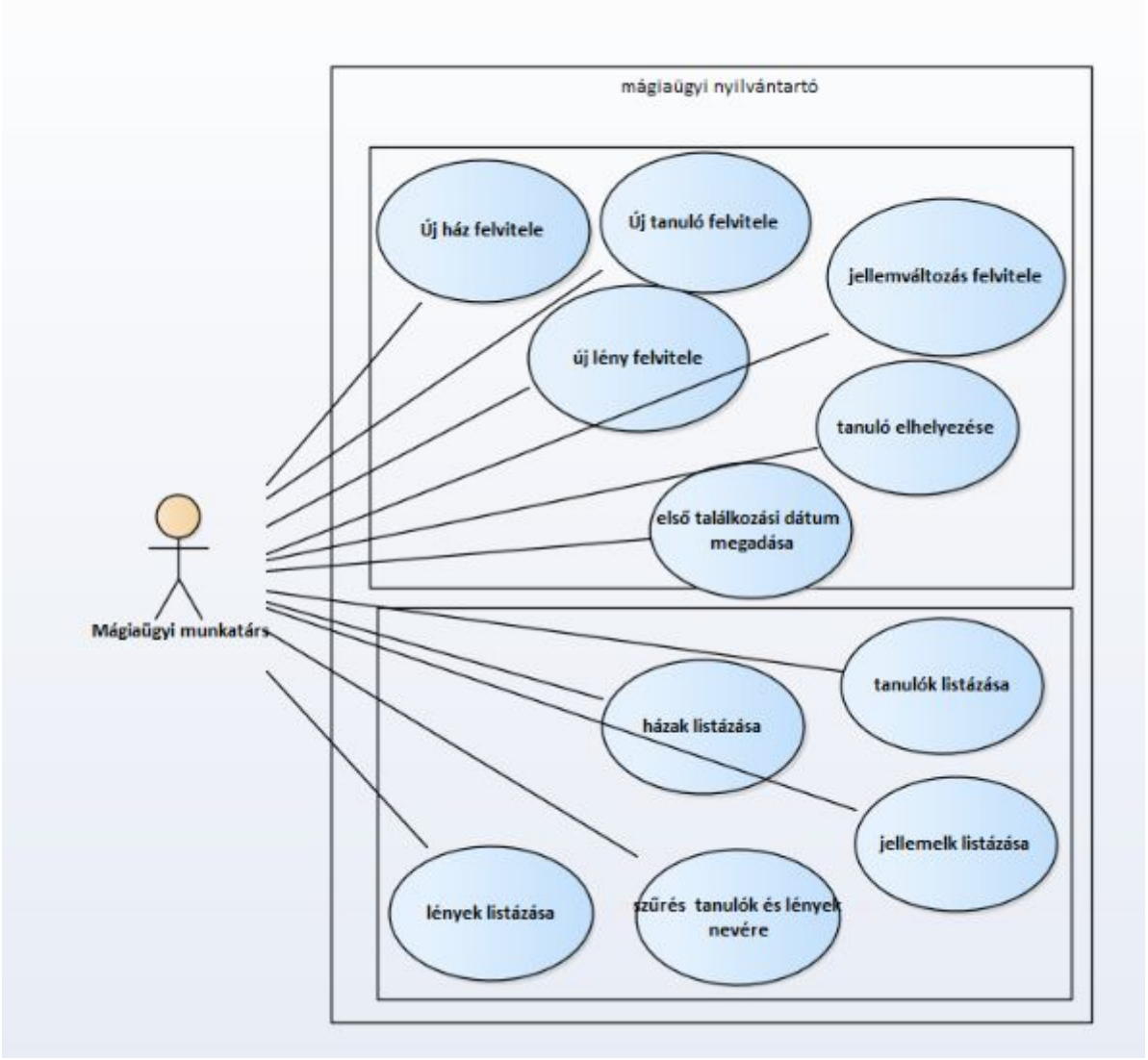
• ház (név, címer, tanulók száma)
• jellem (megnevezés)
• tanuló (név, életkor, jellem)
• lény (név, első találkozás dátuma, jelleme)

3.) **Követelmény elemzése**

A nyilvántartó rendszer fő funkciói adatbevitel, módosítás és szűrés.
Aktorok: a mágiaügyi munkatárs.

1. Funkcionális követelmények

    1. Használati eset (Use case) diagram



2. részletes funkciók:

A.) új adatok felvitele:

-új ház felvitele
-új tanuló felvitele
-új lény felvitele
-első találkozási dátum felvitele (lények esetén)

B.) adatok szűrése:

-házak/tanulók/jellemek/lények listázása
-szűrés a tanulók és lények nevére

C.) adatok módosítása:

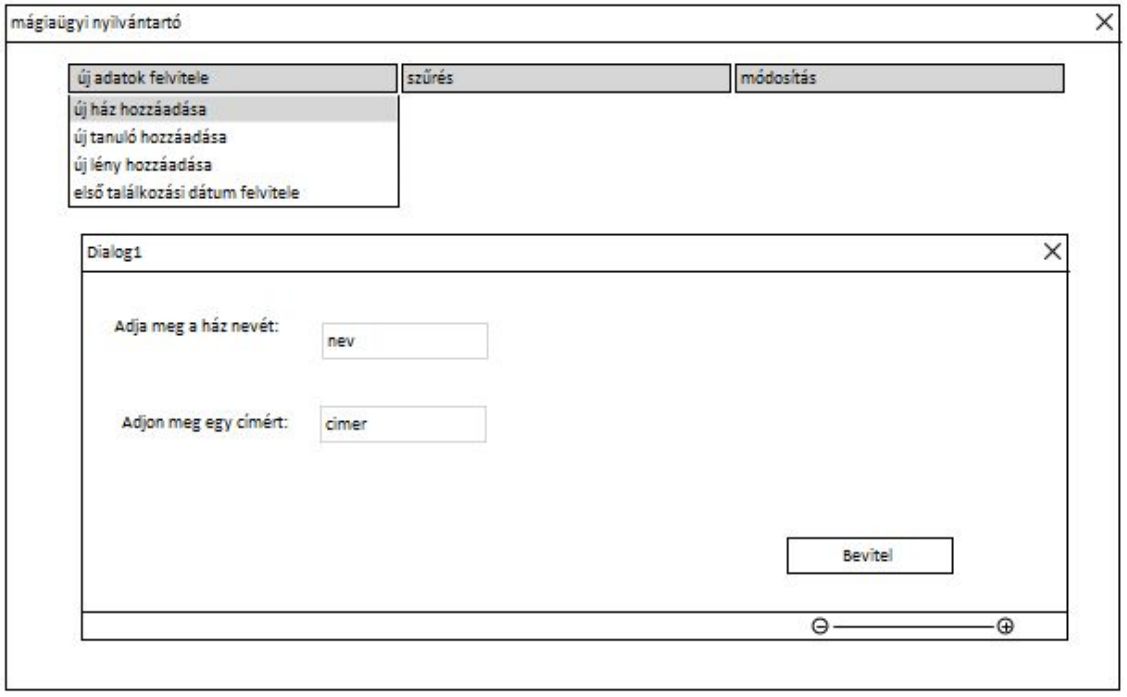
-tanulók/lények jellemének módosítása
-tanuló elhelyezése

3. Felhasználói történetek (User story)

A.) 1. új adatok felvitele/új ház hozzáadása:

Elsődleges Actor	mágiaügyi munkatárs
Előfeltétel	felhasználó elindította az alkalmazást
Sikertelen eredmény	az új ház adatai nem kerülnek rögzítésre. Hibaüzenet jelenik meg a probléma leírásával.
Sikeres eredmény	az új ház adatai rögzítésre kerülnek.
Kiváltja	Ismét a főoldal jelenik meg.
Események	1.)az alkalmazás főoldalán a felső menüsorból kiválasztotta az *új adatok felvitele menü* almenüjét: *új ház hozzáadása*-t. Ez betölt egy formot az alkalmazás alsó részén, a felhasználó kitölti (név és címér megadásával) és a *Bevitel* gombra kattint. 2.) A rendszer ellenőrzi, hogy van e ilyen nevű ház, ha nincs, sikeres az új ház felvitele.
kiegészítés	ha a név foglalt hibát dob.

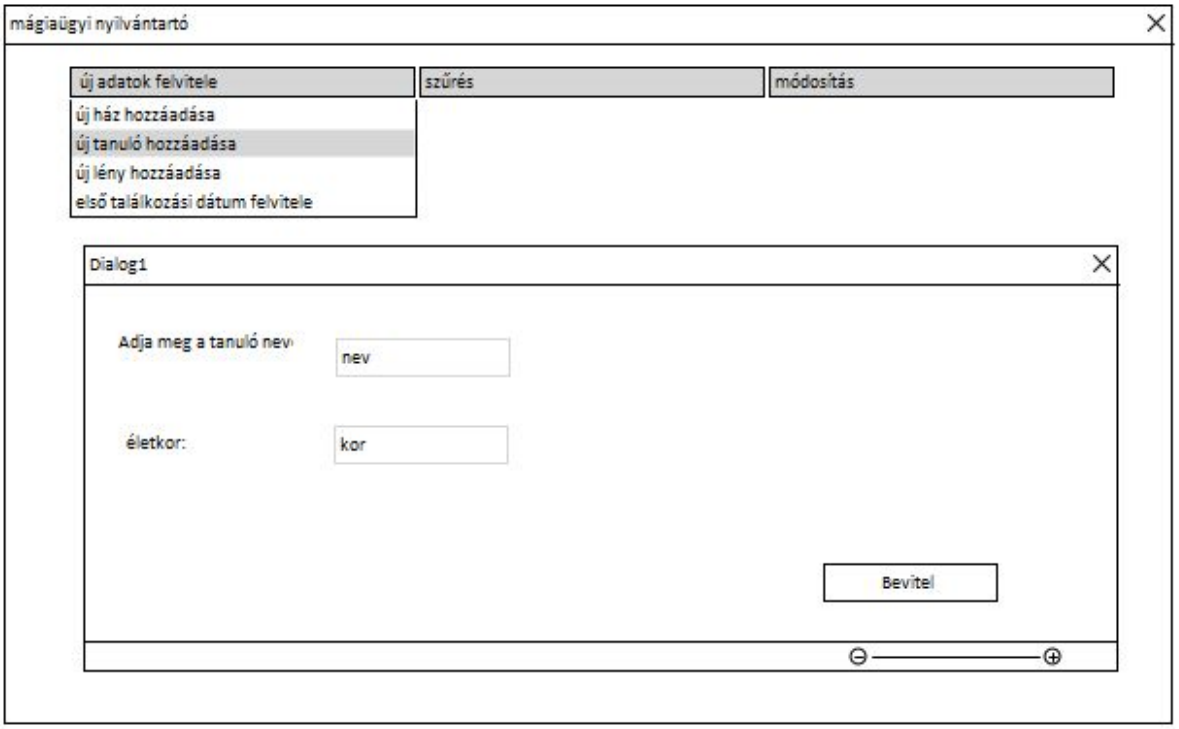
Wireframe:A-1:



2. új adatok felvitele/új tanuló hozzáadása:

Elsődleges Actor	mágiaügyi munkatárs
Előfeltétel	felhasználó elindította az alkalmazást
Sikertelen eredmény	az új tanuló adatai nem kerülnek rögzítésre. Hibaüzenet jelenik meg a probléma leírásával.
Sikeres eredmény	az új ház adatai rögzítésre kerülnek.
Kiváltja	Ismét a főoldal jelenik meg.
Események	1.)az alkalmazás főoldalán a felső menüsorból kiválasztotta az *új adatok felvitele menü* almenüjét: *új tanuló hozzáadása*-t. Ez betölt egy formot az alkalmazásalsó részén, a felhasználó kitölti (név és életkor megadásával) és a *Bevitel* gombra kattint. 2.) A rendszer ellenőrzi, hogy van e ilyen nevű tanuló, és ellenőrzi hogy az életkor helyességét. ha nincs ilyen nevű tanuló még, akkor sikeres az új tanuló felvitele.
kiegészítés	ha a név foglalt hibát dob.

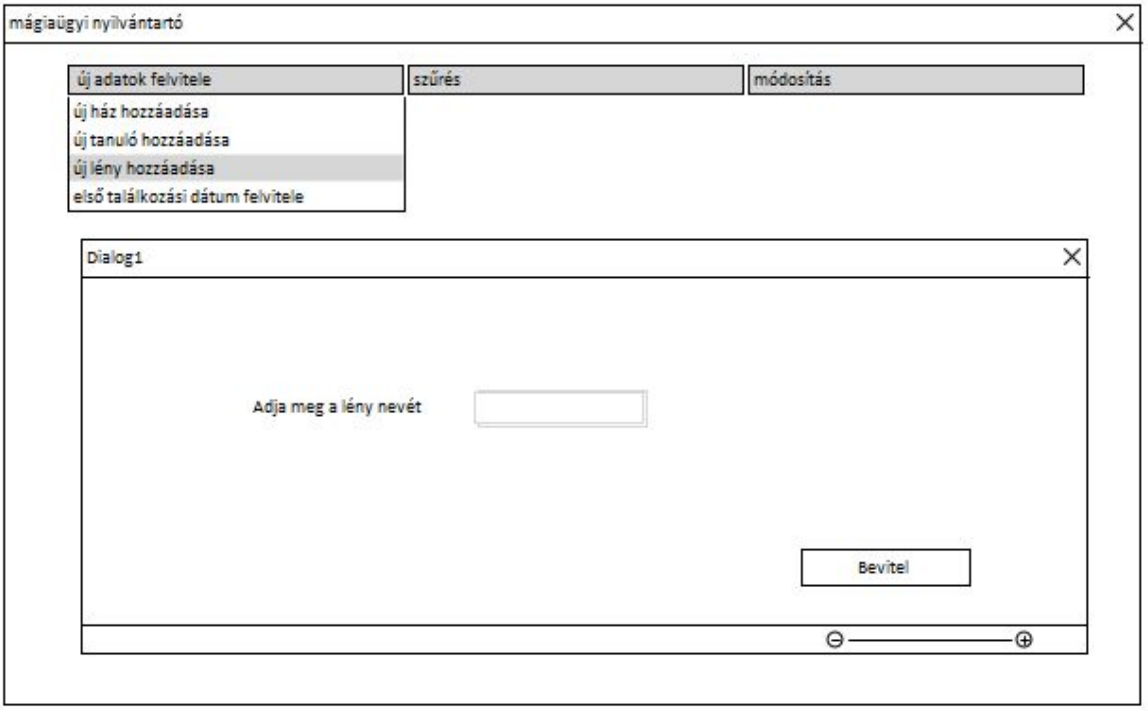
Wireframe A-2:



3. új adatok felvitele/új lény hozzáadása:

Elsődleges Actor	mágiaügyi munkatárs
Előfeltétel	felhasználó elindította az alkalmazást
Sikertelen eredmény	az új lény adatai nem kerülnek rögzítésre. Hibaüzenet jelenik meg a probléma leírásával.
Sikeres eredmény	az új lény adatai rögzítésre kerülnek.
Kiváltja	Ismét a főoldal jelenik meg.
Események	1.)az alkalmazás főoldalán a felső menüsorból kiválasztotta az *új adatok felvitele menü* almenüjét: *új lény hozzáadása*-t. Ez betölt egy formot az alkalmazás alsó részén, a felhasználó kitölti (név megadásával) és a *Bevitel* gombra kattint. 2.) A rendszer ellenőrzi, hogy van e ilyen nevű lény, ha nincs, akkor sikeres az új lény felvitele.
kiegészítés	ha a név foglalt hibát dob.

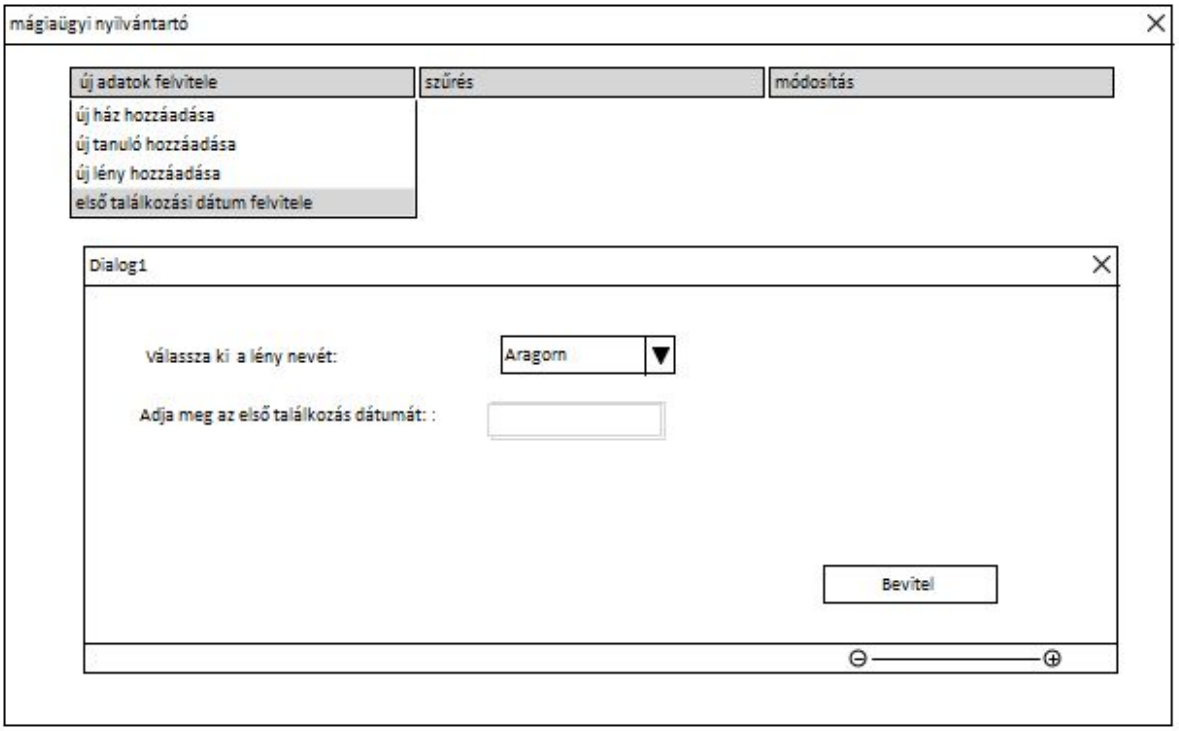
Wireframe A-3:



4. új adatok felvitele/első találkozási dátum felvitele:

Elsődleges Actor	mágiaügyi munkatárs
Előfeltétel	felhasználó elindította az alkalmazást
Sikertelen eredmény	az első találkozási dátum nem kerül rögzítésre. Hibaüzenet jelenik meg a probléma leírásával.
Sikeres eredmény	az első találkozási dátum rögzítésre kerül.
Kiváltja	Ismét a főoldal jelenik meg.
Események	1.)az alkalmazás főoldalán a felső menüsorból kiválasztotta az *új adatok felvitele menü* almenüjét: *első találkozási dátum rögzítése*-t. Ez betölt egy formot az alkalmazás alsó részén, a felhasználó kitölti (név kiválasztásával és dátum megadásával) és a *Bevitel* gombra kattint. 2.) A rendszer ellenőrzi, hogy van e már dátum megadva, illetve a dátum formátum helyességét.
kiegészítés	ha a dátum helytelen vagy már meg volt adva, hibát dob.

Wireframe A-4:

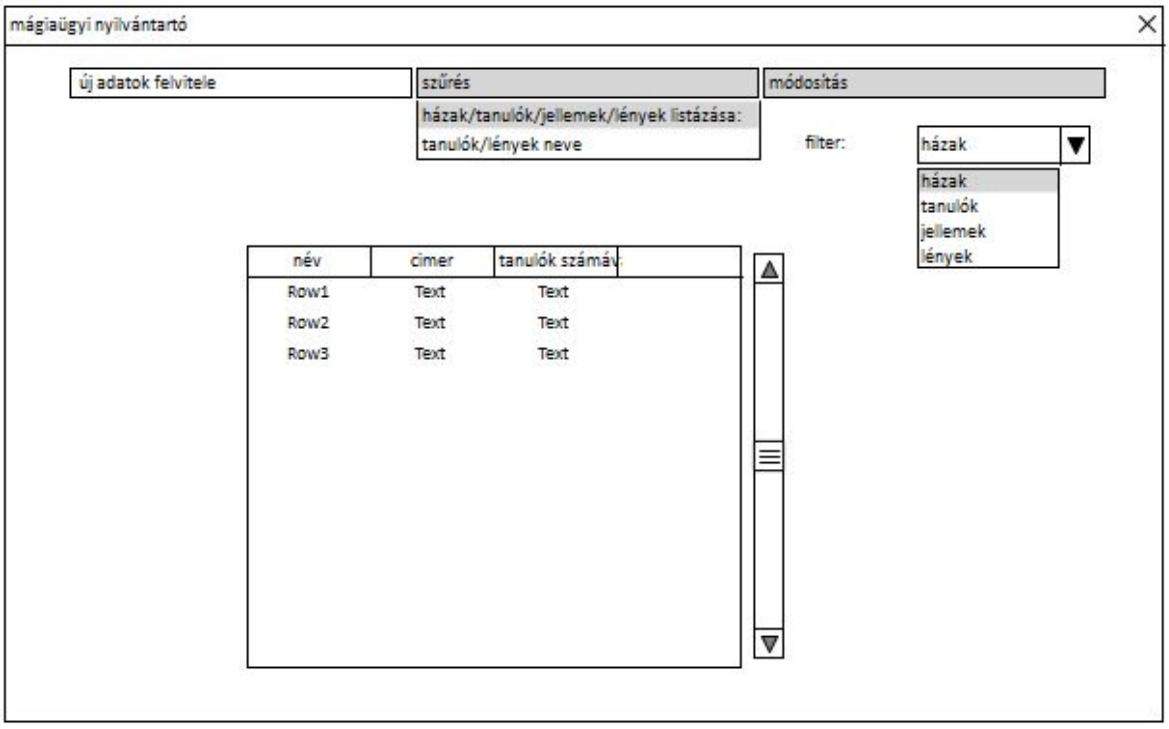


B.) 1. szűrés/ házak/tanulók/jellemek/lények listázása:

Elsődleges Actor	mágiaügyi munkatárs
Előfeltétel	felhasználó elindította az alkalmazást
Sikertelen eredmény	a lekérdezés eredménye táblázatos formában nem jelenik meg a menü alsó részén. Hibaüzenet jelenik meg a probléma leírásával.
Sikeres eredmény	a lekérdezés eredménye táblázatos formában jelenik meg a menü alsó részén. -Ház esetén: név, cimer, tanulók számával. -Tanuló esetén: név, életkor, jellemével. -Lények esetén: név, első találkozás dátuma, jellemével. -Jellemek esetén: megnevezéssel.
Kiváltja	-
Események	1.)az alkalmazás főoldalán a felső menüsorból kiválasztotta az *szűrés menü* almenüjét: *házak/tanulók/jellemek/lények listázása*-t. Ez betölt egy oldalt a jobb oldali téren felül egy dropdown listával, a felhasználó kiválasztja

	házak/tanulók/jellemek/ vagy lények -et szeretne szűrni. 2.) A rendszer ellenőrzi, hogy van e ilyen típusú adat és a szűrés gomb alatti mezőben megjeleníti az eredményt.
kiegészítés	ha nincs ilyen típusú adat akkor hibát dob.

Wireframe B-1:

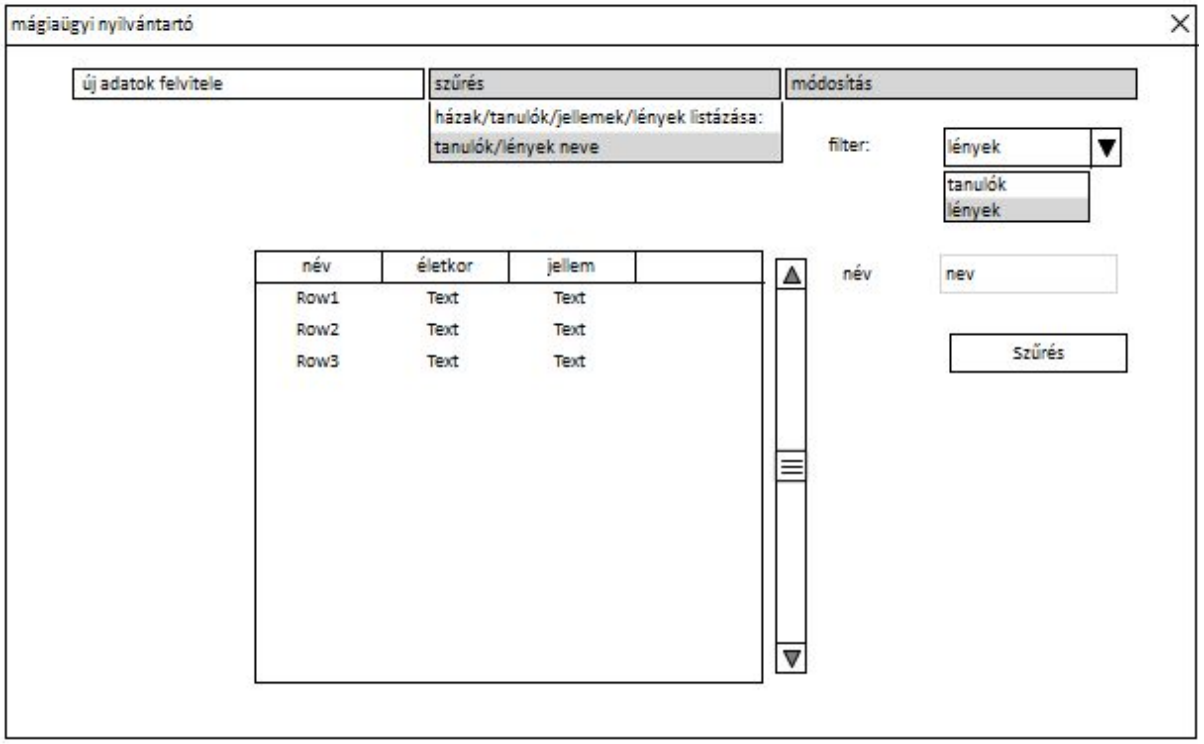


2. szűrés/ *tanulók/lények neve*:

Elsődleges Actor	mágiaügyi munkatárs
Előfeltétel	felhasználó elindította az alkalmazást
Sikertelen eredmény	a lekérdezés eredménye táblázatos formában nem jelenik meg a menü alsó részén. Hibaüzenet jelenik meg a probléma leírásával.
Sikeres eredmény	a lekérdezés eredménye táblázatos formában jelenik meg a menü alsó részén. -Tanuló esetén: név, életkor, jellemével. -Lények esetén: név, első találkozás dátuma, jellemével.

Kiváltja	-
Események	1.)az alkalmazás főoldalán a felső menüsorból kiválasztotta az *szűrés menü* almenüjét: *tanulók/lények neve*-t. Ez betölt egy oldalt a jobb oldali téren felül egy dropdown listával, a felhasználó kiválasztja *tanulók vagy lények -et* szeretne szűrni és a *mezőbe beírja a nevet*, és a *Szűrés* gombra kattint. 2.) A rendszer ellenőrzi, hogy van e ilyen nevű tanuló vagy lény és a szűrés gomb alatti mezőben megjeleníti az eredményt.
kiegészítés	ha nincs ilyen nevű lény vagy tanuló, akkor hibát dob.

Wireframe B-2:

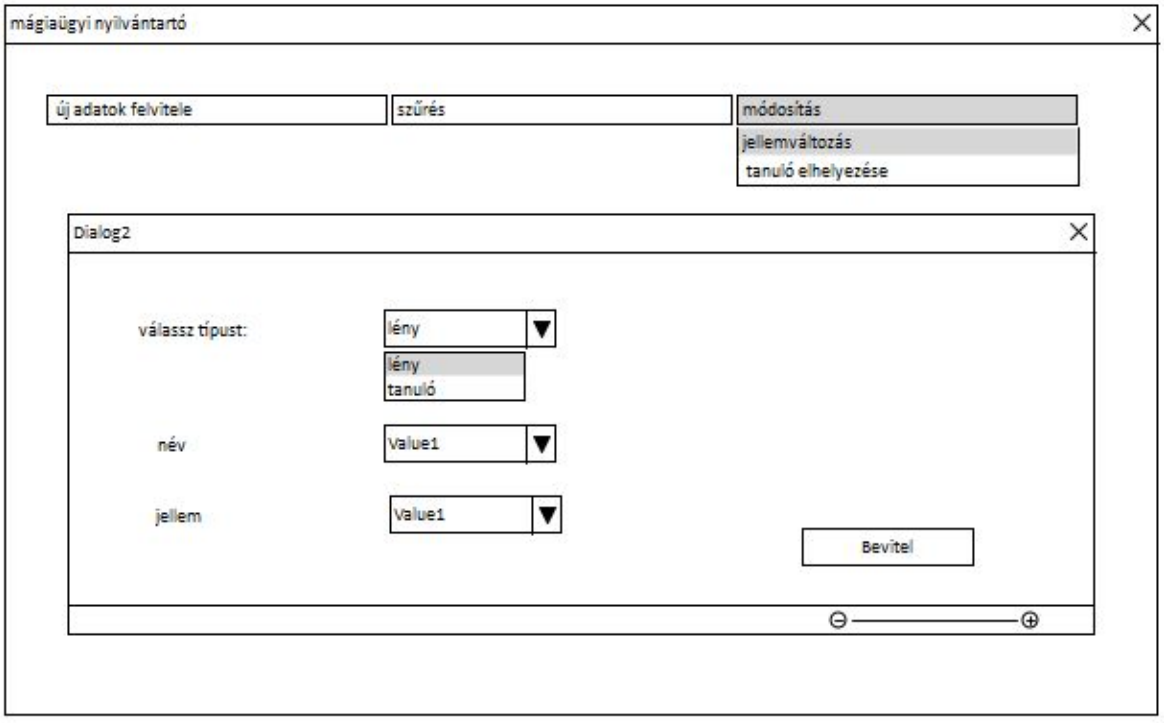


C.) 1. módosítás/ *jellemváltozás*:

Elsődleges Actor	mágiaügyi munkatárs
Előfeltétel	felhasználó elindította az alkalmazást

Sikertelen eredmény	a jellemváltozás nem kerül rögzítésre. Hibaüzenet jelenik meg a probléma leírásával.
Sikeres eredmény	a jellemváltozás rögzítésre kerül.
Kiváltja	Ismét a főoldal jelenik meg.
Események	1.)az alkalmazás főoldalán a felső menüsorból kiválasztotta az *módosítás menü* almenüjét: *jellemváltozás*-t. Ez betölt egy formot az alkalmazás alsó részén, a felhasználó kiválasztja a dropdown listából hogy tanuló vagy lény jellemét szeretné megváltoztatni és kiválasztja a lény/tanuló nevét majd kiválasztja a jellemet és a *Bevitel* gombra kattint. 2.) A rendszer ellenőrzi, hogy van e ilyen nevű lény illetve tanuló, ha van, sikeres az új jellem megadása.
kiegészítés	ha nincs, hibát dob.

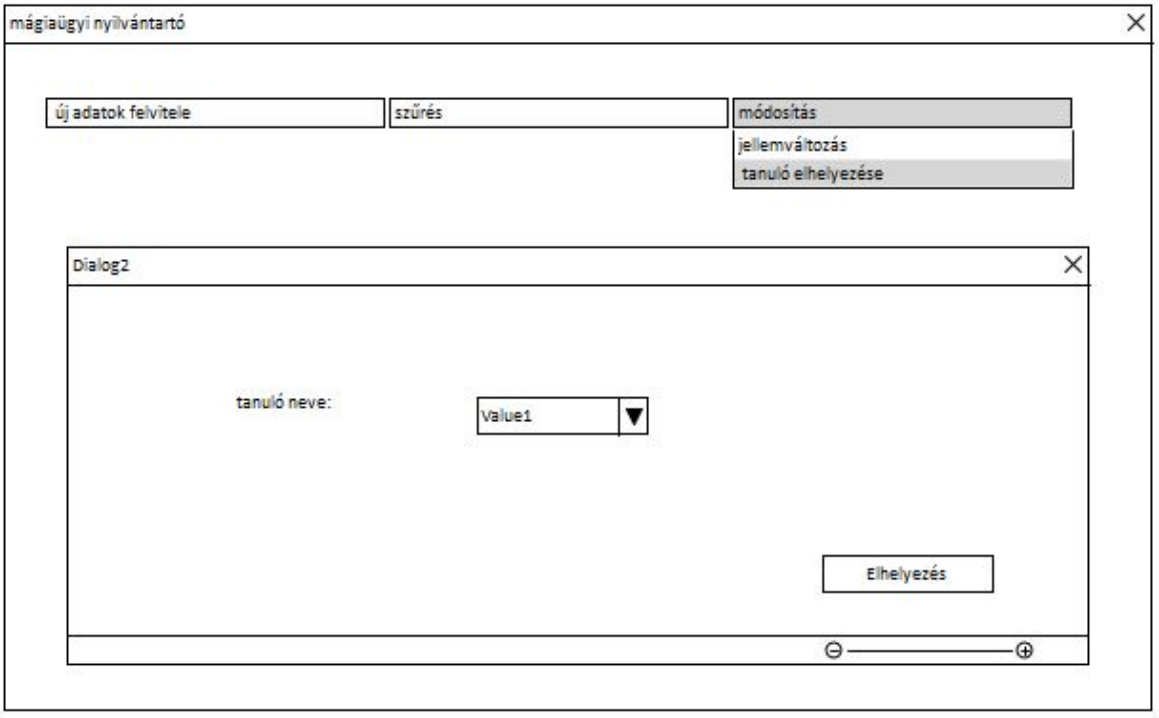
Wireframe C-1:



2. módosítás/ tanuló elhelyezése:

Elsődleges Actor	mágiaügyi munkatárs
Előfeltétel	felhasználó elindította az alkalmazást
Sikertelen eredmény	Hibaüzenet jelenik meg a probléma leírásával.
Sikeres eredmény	a változás rögzítésre kerül.
Kiváltja	Ismét a főoldal jelenik meg.
Események	1.)az alkalmazás főoldalán a felső menüsorból kiválasztotta az *módosítás menü* almenüjét: *tanuló elhelyezése*-t. Ez betölt egy formot az alkalmazás alsó részén, a felhasználó kiválasztja a dropdown listából hogy melyik tanuló *Elhelyezés* gombra kattint. 2.) A rendszer ellenőrzi hogy nincs e már házhoz rendelve a hallgató. Ha nincs: A rendszer visszaad egy ház nevet, az adott házhoz rendeli a hallgatót, megnöveli a házhoz tartozó tanulók számát.
kiegészítés	Ha a hallgató már hozzá van rendelve egy házhoz, akkor hibát dob.

Wireframe C-2:



2. Nem funkcionális követelmények

   Az alkalmazás hardverigénye : Intel Core i5 és Windows10 operációs rendszeren fut. A fejlesztés során: NetBeans fejlesztő környezetben készül az alkalmazás Java nyelven a Clean Code alkalmazásával.

3. Felhasználói-felület modell

   A képernyőtervek készítése során Enterprise Architect-et használtam. User-story-nként látható egy-egy wireframe.

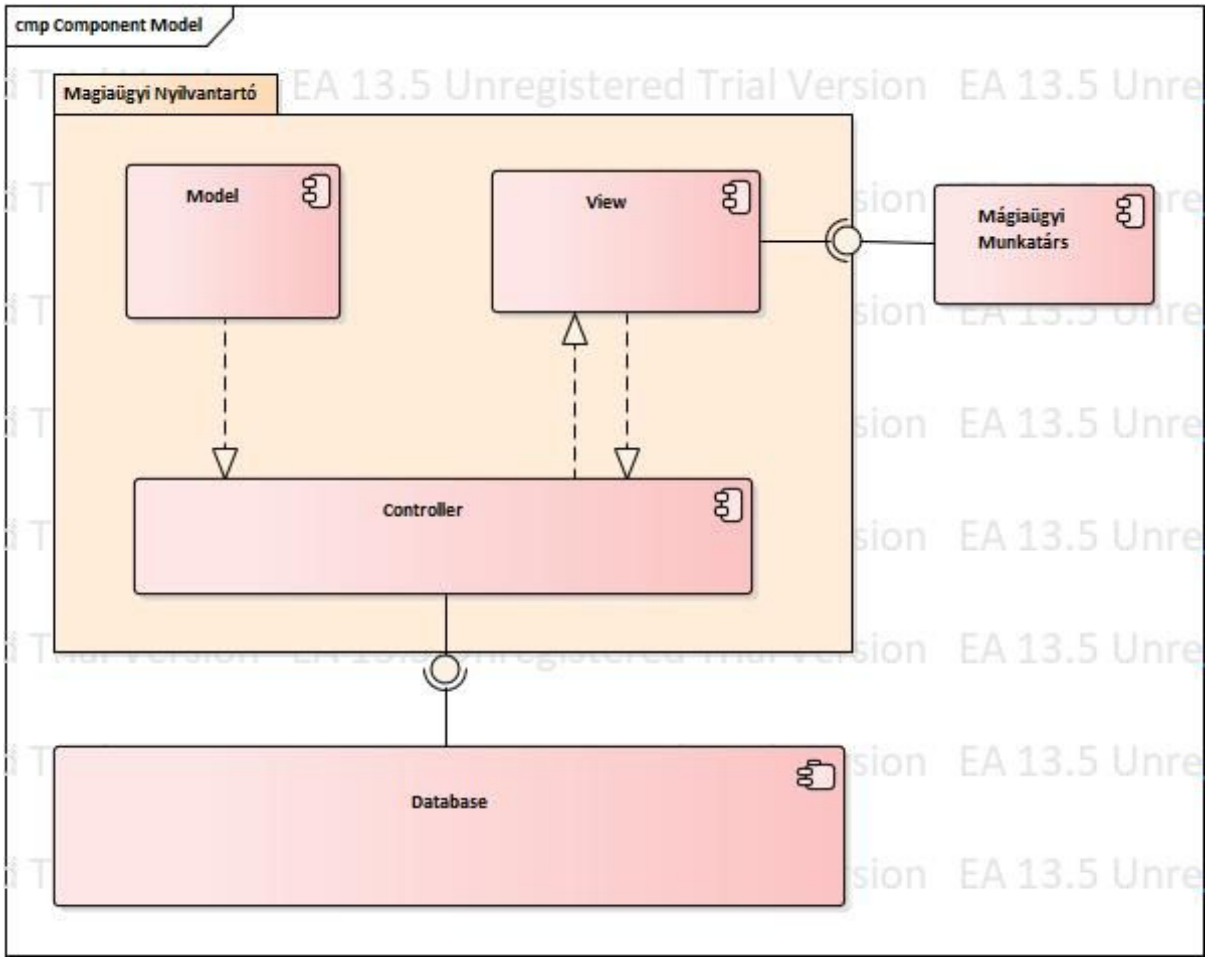
Fejlesztői dokumentáció (2. rész)

4.) **Tervezés**

A rendszer architectúrája

Első sorban nagy rendszerek esetén érdemes dokumentálni, hogy milyen alrendszerekből épül fel a szoftverarchitektúránk.
Segítségünkre lehetnek az átlátható tervezéshez az implementációs szempont szerinti UML diagramok:

- **Komponensdiagram :**



A Mágiaügyi Nyilvántartó több komponensből áll. MVC modellt követem a fejlesztés során, így az egyes osztályok is Model, View és Controller alegységekbe csoportosul. A Controller tart kapcsolatot az adatbázissal, az itt megvalósuló kommunikáció TCP Protokollal történik.

Egy másik ám igen fontos kapcsolat a View és User közötti kapcsolat.

A View és Controller kommunikációja többnyire valamely események által vezérelt. Míg a Controller használja a Modellben szereplő adatosztályokat a View komponenssel történő kommunikációban.

Adatbázis terv

- **Adatbázis sémák leírása**

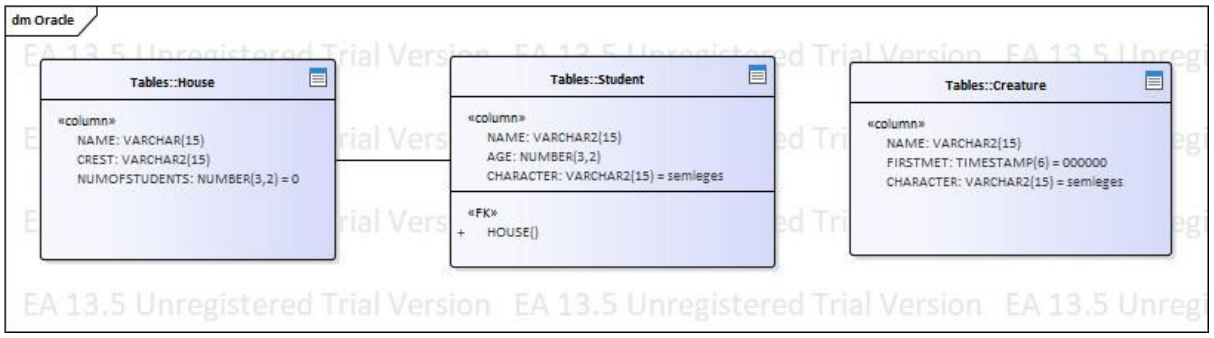
Tábla neve		
attributumok	**típusok**	**leírás**

HOUSE_TABLE		
NAME	**VARCHAR2(15 CHAR)**	**A ház azonosítója/neve , elsődleges kulcs.**
CREST	**VARCHAR2(15 CHAR)**	**A ház címere**
NUMOFSTUDENTS	**NUMBER(3)**	**A házhoz tartozó tanulók száma**

CREATURE_TABLE		
NAME	**VARCHAR2(15 CHAR)**	**A lény azonosítója/neve , elsődleges kulcs.**
FIRSTMET	**TIMESTAMP(6)**	**A lénnyel az első találkozás dátuma**
CHARACTER	**VARCHAR2(15 CHAR)**	**A lény jelleme**

STUDENT_TABLE		
NAME	**VARCHAR2(15 CHAR)**	**A tanuló azonosítója/neve , elsődleges kulcs.**
AGE	**NUMBER(3)**	**A tanuló életkora**
CHARACTER	**VARCHAR2(15 CHAR)**	**A tanuló jelleme**
HOUSE_NAME	**VARCHAR2(15 CHAR)**	**Idegenkulcs (FK) a tanulóra**

- **Egyed-kapcsolat diagram / Kapcsolati tábla**:
- Entity-relationship model:



Statikus (Szerkezeti, Struktúrális) szempont szerinti nézetrendszer alapján

Megmondja, hogy a rendszer milyen egységekből épül fel, mi ezeknek az egységeknek a feladata, milyen kapcsolatban vannak egymással a megoldás elérésének az érdekében. Jellemzően kétféle diagramot szoktak használni a rendszer ezen szerkezetének leírásához.

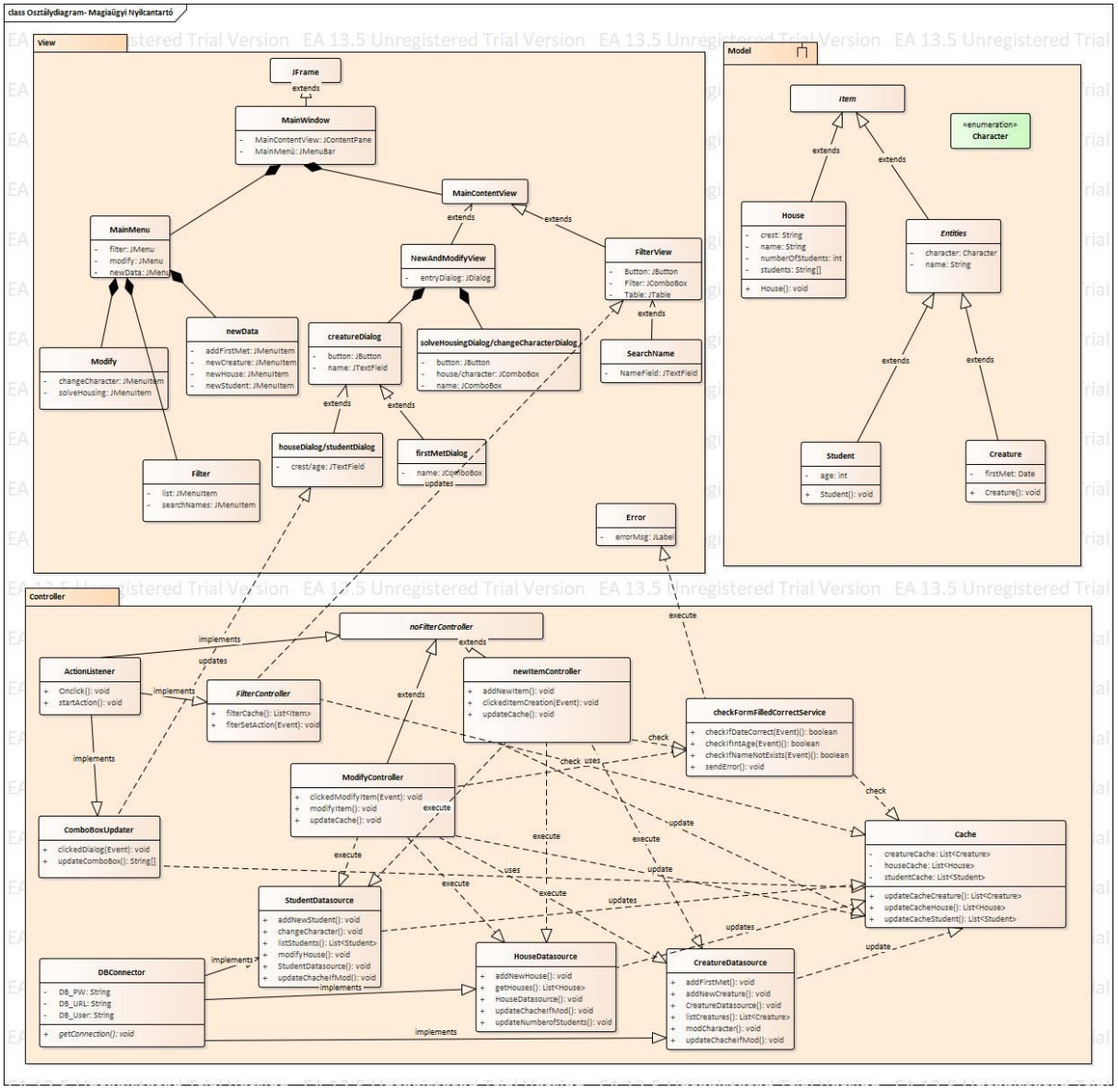
Az MVC modellt alkalmazva:

A View package tipikus Swing elemeket tartalmaz, fő célja a grafikus felület létrejötte, amelyben a Menü tagoltságának köszönhetően adat változtatás és / új adat bevitel esetén az egyes Dialogusok kapnak szerepet, amelyeket a NewAndModifyView jelenít meg. A Dialógusok adatbevitel/módosítás céljaiknak megfelelően többfélék is lehetnek, így a class absztrakciós szintjén is eltérést mutatnak, öröklődési hierarchiájuk is látható az alábbi Diagrammban.
A szintén MainContentViewból származó FilterView a különböző filteropciókat jeleníti meg ennek egy speciális változata SearchName. A FilterView-ek különböző Táblákat jelenítenek meg a szűrési feltételeknek megfelelően.

A View package Swing elemeinek különböző eseményeit figyeli az ActionListenert implementáló noFilterController, FilterController, ComboBoxUpdater, amelyek már a Controller packageben szerepelnek. A szűrőfeltételek beállításával aktiválódik a filterSetAction, mely meghívja saját filterCache függvényét, hogy lekérje az adott Item egy listáját majd betöltse ezt a FilterView JTable-ebe.
A newItemController az új Itemek hozzáadását végzi azaz a Form kitöltését követően ellenőrzi a checkFormFilledCorrectService meghívásával az adatok helyességét, majd az adott Item típusának megfelelő Datasource osztályt meghívva az csatlakozik az implementált DBConnector segítségével az Adatbázishoz és a

megfelelő metódusának hívásával az új Item hozzáadásának megfelelő SQL parancs meghívódik.
Ezután a NewItemController frissíti a cachet is ugyanazon Item Datasource updateCacheIfMod() metódusának meghívásával.
A modifyController osztály hasonlóan csak adatváltoztatásokra, így más Datasource metódusok hívását végzi.
Már csak a ComboBoxUpdater maradt ki az ActionListenert implementálók sorából. Amint az ügyfél egy formot meghívó menüre kattint ami Comboboxot valósit meg, a ComboBoxUpdater erre reagálva szűri meg a cachet és updateli a ComboBox listáját (updateComboBox metódus).

A Model az adatstrucktúrát modellezi , az adatok osztályainak viszonyát. Ezeket az összetett adattípusokat használja a Controller, és a View, amikor adatokat manipulál illetve szűr. Mindezt az **Osztálydiagram** szemlélteti aláb:



Fejlesztői dokumentáció (3. rész)

Implementáció

1. **Technológiák és API-k**

   MySQL adatbázishoz Connector/J (mysql.jdbc 5.1.5.jar)
   Teszteléshez:
   a. junit-4.12.jar
   b. mockito-all-1.9.5.jar

2. **Algoritmusok**

   A beadandó feladata nm követelt meg egy két if elágazás ill for cikluson kívül más nevezetesebb algoritmust.

3. **Csomagszerkezet és metódusok**

   lásd javadoc-kal generálva.

Fejlesztői dokumentáció (4. rész)

Tesztelés

1. **Manuális tesztek**

új lény felvitele	**lilla**	**workbrenchben ellenőrizve, majd lények listazasanal is megjelent**
új karakter felvitele	**rosszmaju**	**workbrenchben ellenőrizve,majd karakterek listazasanal is megjelent**
új tanuló felvitele	**Juci**	**workbrenchben**

		ellenőrizve, hallgatók szűrésénél is megjelent.
névre szűrés ellenőrzése mindkét esetben, létező és nem létező névre.	**nem létező névre hibát dob JDialogban: lény esetén Juci**	**létező névre megjeleníti táblázban. tanulo esetén Juci**
karakterváltoztatás	**Juci semleges karakterrel lett felvéve és char 1 re sikeresen változott.**	**nem létező karakter nem választható JComboBoxból.**
hallgató elhelyezése	**Jucihoz nem rendel létrehozáskor houseId-t elhelyezéskor illabereket , azaz az 1-es Id-t rendelte, random**	**workbrenchben ellenőrizve**

2. **Egység tesztek**

```
package harry.potter.controller.service;

import harry.potter.controller.datasource.HouseDatasource;
import harry.potter.controller.datasource.StudentDatasource;
import harry.potter.model.House;
import harry.potter.model.Student;
import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.mockito.Mock;
import org.mockito.Mockito;
import org.mockito.runners.MockitoJUnitRunner;

import java.sql.Struct;
import java.util.LinkedList;
import java.util.List;

import static org.junit.Assert.assertEquals;
import static org.mockito.Matchers.anyInt;
import static org.mockito.Matchers.anyString;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.spy;
import static org.mockito.Mockito.when;
```

```
public class StudentServiceTest {

    private StudentService testObject = null;
    private StudentDatasource sd = null;
    private HouseDatasource hd = null;


    @Before
    public void init() {
        sd = mock(StudentDatasource.class);
        hd = mock(HouseDatasource.class);
        testObject = new StudentService(hd, sd);
    }

    @Test
    public void addStudentToHouse() {

        // WHEN
        Student student = new Student(1, "Bela");

        House house = new House(2, "Griffendel", "Lion");
        Student excepted = new Student(20, "CHARACTER", "Harry
Potter", house.getId());


when(hd.getHouseIdByName(anyString())).thenReturn(house.getId(
));
        when(sd.addStudentToHouse(anyString(),
anyInt())).thenReturn(excepted);

        // THEN
        Student actual =
testObject.setStudentHouse(excepted.getName(),
house.getName());

        // ASSERT
        assertEquals(excepted, actual);
    }

}
```