2.dia

A program írásához NetBeans-t és SceneBuilder programokat használtunk.

A NetBeans-ben a New projekt > Java FX, Java FXML Applicationt választva hoztuk létre a fájlt. Ez létrehoz 3 fájlt amik FXML a dizájnért, egy osztály az irányításért és a maint az egész elindításért felel.

3. dia

A program az MVC modelre épül… de mi az az MVC? van 3 rétege a model ami a DB osztályunk az adatbázisba adatokat tesz be vesz ki töröl és módosít. A View ez a kinézete a programnak ez lesz az FXML fájl és akkor a Kontroller réteg a nevében is benne van hogy ez irányít fogadja az adatokat és a felhasználói utasításokat kívűlről feldolgozza őket majd visszaküldi a választ a képernyőre és kommunikál a model réteggel is közben. A három fájl az nem a modell három régtege csak a fxml fájl ami a kinézet és document kontroller ami a controller innen még hiányzik a model réteg ami később lett hozzá írva.

4. dia

A használt API-ok (éjpíáj) Application Programing Interface a JDBC (dzséjdíbíszí) ez nem látszik a jdk-ban van, könyvesek az adatbázis konnektorai az itext nevű a pdf-be mentésért a többi pedig az exelbe mentésért felelős Egy előre java nyelven megírt fájlok amit csak beimportálunk és szabadon használhatunk.  
ezekről csak ennyit a bendegúz programozó kolega ugyis erről tart előadást

POJO

getter setter ebből következik hogy osztály változókat és konstruktorokat. Így végülis egy adatbázis rekordot fejez ki így ebben a formában Ennek segítségével az adatbázisból kinyert rekordokat objektummá tudjuk alakítani ha minden oszlophoz tartozik egy osztályváltozó és a továbbiakban használni vagy épp ezen keresztül tudjuk átadni a model rétegnek az objektumot ami majd mindent a megfelelő oszlopba pakol az adatbázisunkban

A benne lévő osztályváltozók egy speciális típusú stringek amiket az a lista tud fogadni amelyek felkerülnek a táblázatra és így van megoldva hogy bármeik oszlop szerint lehessen rendezni az adatait

6. dia

Ez felel a kinézetért ez a fájl egy külső SceneBuilder programmal van szerkesztve amiben csak be kell huzkodni az elemeket és itt tudunk adni id-t nekik beállítani a hozzájuk tartozó css-t és akár az eseménykezelőket is hozzá tudjuk adni.

A main osztály nagy részét a NetBeans maga generálja le. Amit mi írtunk hozzá az a két CSS fájl hozzá adása a fejlécben megjelenő név és ikon hozzá adása és még annyi hogy be lett állítva hogy ne lehessen átméretezni az ablakot.

A többi magától generálódik de pár szóban azért hogy mi micsoda… a TanuloiNyilvantarto osztály kiterjeszti az Application osztályt. Ennek van egy start nevű absztrakt metódusa amit felül kell írnunk minden képpen (@Override)

JDBC

Ezt az API-t megtaláljuk a jdk-ban de még szügségünk lesz hozzá egy konnektorra (driver-re

A híd a kód és az adatbázis között

Képes kommunikálni az összes sql adatbázissal

amit ennél a programnál használtunk egy belső beépített adatbázis a derby amit azért választottunk mert ahol a java fent van ott meg lesz a derby is

Connection

fontos létrehozni a kapcsolatot (Connection-t a jdbc-vel) ez egy osztály amiből objektum lesz aminek utána tudjuk használni már a metódusait osztályváltozóit.

Ez egyébként mindig ugyan így fog kinézni

Connection létrehozásához szükség lesz az elérési címre az adatbázis nevére egy felhasználó névre és egy jelszóra és a megfelelő konnektorra

URL: JDBC figyelj egy derby lesz mondom a nevét és a create legyen true vagyis ha még nincs akkor ő készít egyet magának

Ezeket osztály változokként hozzuk létre

létrehozzuk a connection aminek null értéket adnuk

Statementek

3 fajta a createstatement a preparedstatement és a callablestatement ő meg tudja hívni az adatbázisba mentet adatbázis fügvényeket a többiről később

utasításokat tudunk neki ádani amiket végrehajtanak

ha a connection a híd akkor a statementek rajta a teherautók amik föl alá mennek a híd 2 vége között

metaadatok ezek információk az adatbázisunkról

konstruktor

Az összes adatbázis műveletet le kell hiba kezelni ez egy try catch-el lett megoldva

itt jön létre a kapcsolat amit a drivermanager.getconnection-el hozunk létre neki adjuk át az url-t, ha szükséges akkor a felszhasználó nevet és a jelszót… de itt most nem bonyolítottuk túl ezért ilyeneket nem kér most így csak egy url-t adtunk át a getconnection-nek

a createstatementet a connectionon keresztül tudjuk létrehozni vagyis ha a conn az nem null vagyis van kapcsolat akkor létrehozza a createstatementet

tábla létrehozása

a meta adatok lekérésével majd a resultset-el meggyőződik a program hogy van e már létrehozva benne tábla

ha az rs.next-re már az egyből nem jön vissza eredmény akkor még nincsenek táblák

ekkor a createstatement execute metódusának át adjuk stringként a táblázat létrehozó sql utasítást

getallStudent

egy függvény amely egy arraylistet ad vissza tele a tanulok adataival

van benne egy String ami az sql utasítást tartalmaz egy a createstatementnek átadjuk közben létrejön egy arraylist ami a megadott objektumot fogja tárolni ez jelenleg üres egy ciklus segítségével végig megyünk az adatbázison a ciklus egész addig fut amig az rs.next fügvény talál adatot ha nem talál false értéket ad vissza és kilép a ciklusból

a ciklusban létrejön egy objektum a konstruktorában az rs.getint és rs.getstring parancsokkal kivesszük az oszlopokból az adatokat és átadjuk nekik az oszlopok neveit ahonnan szeretnék hogy kivegye az adatot létre jön az objektumunk és ezt hozzá adjuk az arraylistunk-hoz és így töltődik fel ugye mint minden adatbázis műveletet ezt is hibakezeléssel el kell látni és ha bármi baj történne akkor a catch ágon jelezni fogja program és végül a függvényünk vissza returnöli a megtöltött arraylistet

newStudent

itt is van egy string amiben szintén egy sql utasítás van ezt a preparedStatementnek adjuk át majd setString parancsal megadjuk az oszlop számát ahova a pojo getterén keresztül beletöltjük az objektumot egy adatbázis rekodba

azért kell preparedStatementnek használnunk mert így meghatározhatjuk hogy milyen adattípust is akarunk küldeni plusz createstatementel veszélyes lehet mert ilyen módon bármit küldhet a user akár egy sql utasítást is ami töröl dolgokat az adatbázisunkból így nem az sql utasíton kereszül hanem azon kívűl érkezik az adat és valamilyen szintű védelmet nyújt az ilyen rosszalkodások ellen is a másik előnye még annyi hogy gyorsabban is dolgozik

a végén az execute-al végre hajtatjuk vele a mentést

majd végül megint egy kis hiba kezelés hátha gikszer csúszik a műveletbe

törlés

megint csak egy sql utasítást tartalmazó string amit átadunk a createstatement nem

majd egy másik create statementel pedig hasonló módon mint az elején létrehozza a tanulo táblánkat kicsit levan egyszerűsítve az adatok törlése de ennyi fért a budget-be