Backend dokumentáció
(Cinema jegyfoglaló)

Készítette:
Németh Krisztián (JB9MRJ)
Mihalik Máté (QHXKZE)

Használt technológiák

A projekt első lépéseként egy git repository-t hoztunk létre a github oldalon. A git egy verziókezelő rendszer, amely megkönnyíti a csoportmunkát és a fejlesztést. Nyomon követhetőek ill. visszavonhatóak az elvégzett változtatások, mindenki ugyanazon a kódbázison dolgozhat egyszerre, komolyabb adminisztratív teher nélkül. A kód külön létezik a fejlesztők helyi gépén, illetve egy masternek nevezett központi tárhelyen. A git repository beállítását még a félév elején az első beadandó keretében elvégeztük, ezért ezt nem részletezzük tovább.
A backend java nyelven íródik, melynek egyik alapvető technológiájaként a Spring-et használjuk. A Spring nagyon sok mindent magában foglal, számunkra azonban az IOC (Inversion Of Control) aspektusa a leginkább fontos elem. Ennek lényege, hogy a programunk egymással interfészen keresztül összekapcsolt komponensekből áll (függőségek). Ha egy komponensnek egy függőségre van szüksége, akkor azt nem saját maga hozza létre, hanem valamilyen módon megkapja azt futásidőben akkor, amikor neki kell. Ezt a munkát elvégzi nekünk a Spring (@Autowired). Ennek számos előnye van a könnyen lecserélhető implementációktól kezdve a hibakeresés egyszerűsödéséig (a hiba helye komponensekre izolálható).
Egy kicsit más vizekre evezünk a Lombok használatával, mellyel az entitásoknál gyakran leírt, favágó kódot spórolhatjuk meg különféle annotációk használatával. Automatikusan létrehozza nekünk az adattagokhoz tartozó getter/setter függvényeket, paraméter nélküli konstruktort, összes paramétert tartalmazó konstruktort.
Szükségünk van még egy adatbázisra is, erre a H2 Database Engine nyújt megoldást. Egyik, főleg a kezdeti tesztelés szempontjából előnyös tulajdonsága, hogy lehet vele memóriában tárolt adatbázist is létrehozni, ami így csak a program futásának erejéig őrzi meg az adatokat. Előfordul, hogy kezdetekben az adatbázis sémáján többször is csiszolni kell, ezért hasznos ha az elején ezt még rugalmasan tudjuk kezelni.

Fejlesztői környezet

A repository eléréséhez szükség van egy git kliensre. Erre többféle megoldás létezik (konzolos és GUI-s is), pl. SourceTree, TortoiseGit...stb. Mi a Git-SCM (git-scm.com) konzolos változatát használjuk. Telepítés után a git shell-t elindítva létre kell hoznunk egy mappát a kódbázisunk számára. Ez lesz a saját, lokális repository-nk amin dolgozhatunk. A *git clone <repo cím>* parancs kiadásával a távoli gépen elérhető kódbázist lehúzzuk a saját gépünkre. A *git status*-sal ellenőrizhető a lokális repo állapota (változások a masterhez képest a legutóbbi információk alapján). A *git fetch* szerzi meg az előző parancs számára szükséges infókat a masterről, a változásokat pedig a *git pull* segítségével tudjuk letölteni a gépünkre. Ha mi végzünk változtatásokat, akkor ezeket ún. commit-okba szervezve küldjük fel a masterre. Először a *git add .* meghívásával az összes változtatást hozzáadjuk a commithoz, majd a *git commit -m "commit message"* véglegesíti a változtatásokat. Ekkor még csak a helyi gépen létezik a változtatás, ahhoz hogy ez a masteren is reflektálódjon, a *git push* parancsot kell kiadni.
A kezdeti projektünket a start.spring.io oldalon hozzuk létre, Maven projektként. A Maven egy projektkezelő, melynek a konfigurációs fájlja a pom.xml. Ide kerülnek be az oldalon beállított dolgok, például hogy milyen függőségekre lesz szüksége a programunknak, mi a projekt neve, milyen verziójú java-t használunk...stb. A generált projektet elmentjük a gépre, majd egy tetszőleges IDE-vel megnyitjuk (pl. IntelliJ, NetBeans, STS...stb). Mi az STS-t használjuk. Szerencsére az STS csak minimális kezdeti konfigurációt igényel. A lombok használatához szükség van egy plugin-ra ami a lombok hivatalos oldaláról letölthető, majd a fejlesztői

környezethez hozzáadható. Miután mindezekkel megvagyunk, jobb gombbal a projekt nevére kattintva kiválasztjuk az Update Maven Project lehetőséget. Az IDE felkutatja a pom.xml-ben megadott dependency-ket, és rövid idő múltán a projekt mellett már fogjuk látni a [spring] feliratot. Ekkor már futtatni is tudjuk, a fejlesztői konzolban megjelenik a spring elindulásával együtt járó debug üzenetek sokasága, és elindul a tomcat webszerver is.

Alkalmazott könyvtárstruktúra

A projekt alapvetően egy Maven project, mely konvencionális könyvtárstruktúrát követ. Ennek utána lehet olvasni az interneten, ebben a dokumentációban csak a projekt specifikus struktúra lesz kifejtve.
A forrásfájlok a cinema.cinema csomagban találhatóak. Ezen belül 3 további csomag van:
-entity: az adatbázis entitásokat, illetve a kapcsolódó enumokat tartalmazza
-repository: az adatbázis entitások CRUD műveleteinek elvégzéséhez szükséges interfészek
-service: az üzleti logika által megkövetelt funkcionalitások ide kerülnek

Adatbázis terv

[todo]