MEGOLDÁS

1. USE CASE DIAGRAM

Első lépésként keressük meg, milyen szereplők lesznek a rendszerben. Lesz egy látogató, illetve, mivel van bejelentkezési lehetőség is, így biztosan lesz egy bejelentkezett felhasználó is. Ezen kívül vegyük fel az adatbázist, mint szereplőt.

Ezután gyűjtsük ki a funkciókat a rendszer leírásából. Biztosan kell, mivel szerepel a szövegben:

- bejelentkezés
- „Emlékezz rám” funkció
- regisztráció
- mappa listázás
- címkére szűkítés
- jegyzet listázás, módosítás és törlés
- új jegyzet, mappa és címke felvétele
- kijelentkezés
- részletes keresés
- egyszerű keresés

A fenti funkciókon kívül még a szövegből kikövetkeztethető, hogy lehetőség van a rendszer funkcióinak elolvasására (mivel a főoldalon bejelentkezés előtt ez látszik). Valamint, mivel adatbázis is szerepel a diagramon, így egy használati eset az adatot kezel lesz.

Miután felvettük minden fenti funkciónak egy-egy használati esetet, már csak a kapcsolatokat kell megállapítani közöttük. A látogató kapcsolatban áll a funkciók olvasása, bejelentkezés, regisztráció használati esetekkel.

A bejelentkezett felhasználó kapcsolatban áll (használja) a mappák listázása és kijelentkezés használati esetekkel. A három új elem létrehozását egyszerűbb kezelni, ha egy használati eset speciális eseteként tekintünk rájuk. Így hozzunk létre egy új elemet létrehoz használati esetet, és kapcsoljuk specializáló kapcsolattal hozzá az új jegyzet, új mappa és új címke eseteket.

A keresésnél két megközelítést is használhatunk. Egyrészt, kezelhetjük itt is speciális esetként, ekkor fel kell venni egy keresés használati esetet, amelynek a leszármazottjai lesznek az egyszerű és a részletes keresés. De felfoghatjuk úgy is az oldalt, hogy alapértelmezetten egyszerű keresést használjuk, és onnan lehetőségünk van további mezőket megadni a részletes kereséshez. Ekkor a részletes keresés az egyszerű kereséssel <<extend>> kapcsolatban áll. Az emlékezz rám funkció a bejelentkezést terjeszti ki, tehát vele áll <<extend>> kapcsolatban.

A mappában lévő jegyzeteket csak úgy tudjuk megnézni, ha előtte kilistázzuk az elérhető mappákat, így itt érdemes tartalmazás kapcsolatot (<<include>>) felvenni a két használati eset közé. A „címkékre szűrés” a mappa listázás kiterjesztett funkcionalitása lesz (<<extend>>). A jegyzet módosítás és törlés funkciók kiterjesztik a jegyzet listázást. De úgy

is lehet értelmezni, hogy a módosítás és törlés funkciók csak a listázást követően érhetőek el, így a listázás magában foglalja ezt a kettőt. Ezesetben a módosítás/törlés is <<include>> kapcsolatban áll az adatbázissal.

Végül minden olyan használati esetet, amelyik elment valamit az adatbázisba vagy kiolvas valamit az adatbázisból, kössünk össze az adatot kezel használati eset <<include>> kapcsolattal.

2. CLASS DIAGRAM

A három egyed osztály (Note, User, Notebook) felvétele magától értetődik. Mivel ezek az osztályok alkalmazás független adatot tárolnak, amelyek hosszú életűek, ezért a sztereotípus <<Entity>> lesz.

A NoteDAO insertNote függvénye a címet és a szöveget várja paraméterben, ezért két Stringet kap. Mivel ez csak egy adatbázis beszúrást végez, ezért nem tér vissza semmilyen értékkel, illetve void lesz a visszatérési érték. A getAllNote függvény – ahogy a szövegben is szerepel – egy felhasználót vár paraméterül, és egy Note típusú listát ad vissza. A fillNoteBook egy Notebook objektumot kap paraméterben, aminek feltölti a listáját. Mivel „helyben” végzi a műveletet, ezért itt is void a visszatérési érték. Ez az osztály a rendszer környezete és belseje közötti kommunikációt valósítja meg, ezért <<Boundary>> a sztereotípusa.

A NoteControl osztály egy User és egy Note adattagot tartalmaz. A createNewNote függvény nem vár paramétert, mivel üres jegyzetet ad vissza. Visszatérési értéke az új jegyzet, tehát egy Note típusú objektum. A deleteNote az adattagban tárolt jegyzetet törli, tehát nem vár paramétert és void a visszatérési értéke. Az addTagToNote egy szöveg típusú paramétert vár, ez lesz a jegyzet címkéje. Visszatérési érték itt is void. Ez az osztály használati esetek szekvenciális viselkedését valósítja meg (CRUD-szerű műveletek), ezért <<Control>> sztereotípus lesz.

Végül már csak a kapcsolatok maradtak. Adattagként a Note objektum szerepel a NoteControl és a Notebook osztálynál, a User pedig szintén a NoteControl osztálynál, így ott aggregáció típusú kapcsolatot érdemes jelölni (nem erős tartalmazás). Megfontolandó az erős tartalmazás kapcsolat a Notebook és a Note osztály között, ha olvasatunkban egy adott Notebook törlésekor a benne található jegyzetek is törlésre kerülnek.

A NoteDAO mindhárom egyed osztállyal asszociációs kapcsolatban áll. A NoteDAO jelen állapotában nincs kapcsolatban a NoteControl osztállyal, mert a jegyzeten végrehajtott műveletekkel kapcsolatban nem volt szó arról, hogy a módosítások adatbázisban is tárolódnak. De ez is egy elfogadható megoldás (indoklással), ekkor a NoteDAO is kapcsolatban áll a NoteControl osztállyal.