Követelményspecifikáció

Fájlalapú dokumentumnyilvántartó rendszer készítése desktopra
Szoftverarchitektúrák tárgy házi feladat

Feladatkiírás

Az elkészítendő szoftver egy asztali alkalmazás, amely képes dokumentumok (cikkek, könyvek, folyóiratok, weboldalak) rendezett, rendszerezett tárolására.

A dokumentumok megjelenítése az operációs rendszer és egyéb, külső beszállítók által készített programok feladata, nem képezi jelen projekt témáját. Az elkészítendő szoftver célja a kereshetőség biztosítása sok dokumentum esetén. Bővebb leírás a feladatleírás részben olvasható.

A fejlesztői csapat

A csapat tagjai:

csapattag neve	Neptun-kód	E-mail cím

A csapatban dedikált szerepek kiosztását a csapat kis mérete miatt nem tartottuk fontosnak.

Részletes feladatleírás

A projekt során célunk egy olyan alkalmazás készítése, amely képes egy felhasználó által használt dokumentumok (fájlok, weblapok, stb.) egységes környezetben történő kezelésére, hozzájuk tartozó metaadatok tárolására és ezek alapján történő keresésre, szűrésre.

A nyilvántartott dokumentumok életciklusa a következő:

- dokumentum beindexelése (kézi hozzáadás alapján vagy megadott mappák figyelése alapján automatikusan egy dokumentum létrehozásakor)
- dokumentum kezelése, amely magában foglalja:
    - dokumentumról tárolt metaadatok megtekintése, módosítása
    - dokumentum másolása, áthelyezése (helyben tárolt fájlok esetén)
    - dokumentum törlése (helyben tárolt fájlok esetén)
    - dokumentum megnyitásának kezdeményezése (az operációs rendszerben hozzárendelt programmal)
- dokumentumok keresése, szűrése, amely magában foglalja:
    - dokumentumok keresése metaadataik alapján (szabadszavas keresés az összes metaadat alapján)
    - dokumentumok listájának megjelenítése címkéik alapján
        - egy megadott címkével ellátott dokumentumok listázása

        - megadott címkék mindegyikével ellátott dokumentumok listázása („és kapcsolat”)
        - megadott címkék valamelyikével ellátott dokumentumok listázása („vagy kapcsolat”)
        - bizonyos metaadat alapján megadott tartományba eső dokumentumok listázása (pl. a 2006–2008 közti megjelenési év adattal rendelkező dokumentumok megjelenítése)
    - dokumentumok tárolt metaadatainak exportálása BibTeX formátumban
- dokumentum eltávolítása a program adatbázisából

Technikai paraméterek

A definiált alkalmazást Java platformra készítjük el annak érdekében, hogy több operációs rendszeren (Windows, Mac OS) is lehessen használni. Az alkalmazás az adatait egy SQLite (vagy más hasonló könnyűsúlyú) adatbázisban tárolja annak érdekében, hogy ne legyen szükség komoly infrastruktúra üzemeltetésére. A program további, magától nem értetődő követelményeket nem fog támasztani a futtató számítógéptől.

Szótár

Dokumentum: olyan adat (fájl vagy webcím), amely által reprezentált entitást a szoftverünk segítségével kezelni, nyilvántartani szeretnénk. Jelenleg dokumentumnak tekintjük az alábbi típusú fájlokat és webcímeket:

- Word fájl (.doc vagy .docx kiterjesztéssel)
- PDF-fájl (.pdf kiterjesztéssel)
- tetszőleges weboldal (teljes URL-jével megadva)
- Evernote jegyzet (azonosítójával megadva)

Dokumentumok kezelése: dokumentumok metaadatainak tárolása a program adatbázisában, melyek alapján szűrések, keresések végezhetők ezen, illetve a dokumentumok a programból megnyithatók, áthelyezhetők, törölhetők.

Dokumentum beindexelése: dokumentum felvétele a program adatbázisába, a dokumentum kezelése innentől kezdődhet.

Dokumentum típusa: a dokumentum jellege, amely meghatározza a hozzá tárolandó metaadatok körét (pl. konferenciacikk típusú dokumentum esetén tárolandó metaadat a konferencia neve, míg egy folyóiratcikk típusú dokumentum esetén nincs alapból „konferencia neve” metaadat – kézzel hozzáadható).

Dokumentumról tárolt metaadatok: a dokumentumhoz fűzött adatok, melyeket a program adatbázisában tárol. Elemei lehetnek: dokumentum szerzője, dokumentum címe, dokumentum DOI-ja, dokumentumhoz fűzött megjegyzése, dokumentum eredeti helye, megjelenés éve, konferencia neve vagy folyóirat címe, dokumentumcímkék. A tárolt metaadatok köre függ a dokumentum típusától.

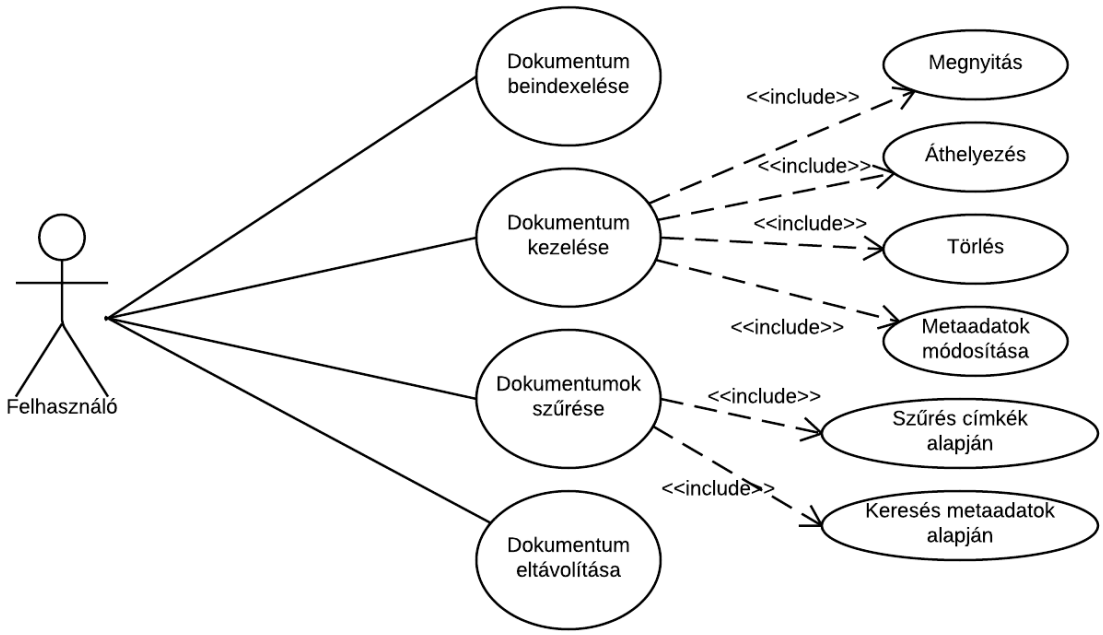
Dokumentumcímke: egy szöveg (pl. „munka”, „egyetem”), amely egy vagy több dokumentumhoz lehet rendelve. A dokumentumok ennek alapján csoportosíthatók. Egy dokumentumhoz több címke is tartozhat.

DOI: Digital Object Identifier, digitális objektumazonosító, a http://www.doi.org/ oldalon található szabványoknak és ajánlásoknak megfelelően.

Indexelt (figyelt) mappa: olyan helyi könyvtár a számítógépen, amelyben az újonnan létrehozott dokumentumokat automatikusan beindexeli a program futása közben.

Essential use-case-ek

Use-case diagram



DocuMaison

Rendszerterv

DocuMaison

Fájlalapú dokumentumnyilvántartó rendszer készítése desktopra

Szoftverarchitektúrák tárgy házi feladat

Készítették:

Tartalom

A rendszer célja, funkciói és környezete

Feladatkiírás

Az elkészítendő szoftver egy asztali alkalmazás, amely képes dokumentumok (cikkek, könyvek, folyóiratok, weboldalak) rendezett, rendszerezett tárolására.

A dokumentumok megjelenítése az operációs rendszer és egyéb, külső beszállítók által készített programok feladata, nem képezi jelen projekt témáját. Az elkészítendő szoftver célja a kereshetőség biztosítása sok dokumentum esetén.

A feladat részletes specifikációja a követelményspecifikáció dokumentumban olvasható.

A rendszer által biztosítandó tipikus funkciók

Vázlatosan az alábbi funkciók biztosítását várjuk el a rendszertől. (A funkciók részletes definíciója szintén a követelményspecifikáció dokumentumban olvasható.)

- dokumentum beindexelése (kézi hozzáadás alapján vagy megadott mappák figyelése alapján automatikusan egy dokumentum létrehozásakor)
- dokumentum kezelése, amely magában foglalja:
    - dokumentumról tárolt metaadatok megtekintése, módosítása
    - dokumentum másolása, áthelyezése (helyben tárolt fájlok esetén)
    - dokumentum törlése (helyben tárolt fájlok esetén)
    - dokumentum megnyitásának kezdeményezése (az operációs rendszerben hozzárendelt programmal)
- dokumentumok keresése, szűrése, amely magában foglalja:
    - dokumentumok keresése metaadataik alapján (szabadszavas keresés az összes metaadat alapján)
    - dokumentumok listájának megjelenítése címkéik alapján
        - egy megadott címkével ellátott dokumentumok listázása
        - megadott címkék mindegyikével ellátott dokumentumok listázása („és kapcsolat”)
        - megadott címkék valamelyikével ellátott dokumentumok listázása („vagy kapcsolat”)

        - bizonyos metaadat alapján megadott tartományba eső dokumentumok listázása (pl. a 2006–2008 közti megjelenési év adattal rendelkező dokumentumok megjelenítése)
    - dokumentumok tárolt metaadatainak exportálása BibTeX formátumban
- dokumentum eltávolítása a program adatbázisából

A program környezete

A szoftvert vastagkliens alkalmazásként készítettük el. Annak érdekében, hogy több operációs rendszeren (Windows, Mac OS stb.) is futtatható legyen, platformfüggetlen megoldásokat választottunk. A programot Java nyelven fejlesztettük, a grafikus felülethez az SWT-t használtuk, az adatbázis rétegben pedig az SQLite adatbázismotort használtuk fel. Ezek mind platformfüggetlen komponensek.

A DocuMaison program képes a kezelt dokumentumok megnyitására is, azaz laza kapcsolatban áll a használt dokumentumok megjelenítőprogramjaival is. Mivel a felhasználón múlik, hogy milyen dokumentumokat tárol a rendszerben, ezért csak annyi követelményt támasztunk, hogy a dokumentumokhoz tartozó fájlokhoz az operációs rendszer rendeljen hozzá megfelelő megjelenítőprogramokat. A DocuMaison program mindössze a fájlok megnyitását kezdeményezi az operációs rendszernél, ennél szorosabb kapcsolatban nem áll a megjelenítőprogramokkal.

A program más, a manapság használatos számítógépek által nem teljesített követelményt nem támaszt.

Megvalósítás

Az alkalmazást a feladatkiírásnak megfelelően egy többrétegű alkalmazásként készítettük el. Bár az egyes rétegek jól definiáltan különválnak egymással, mégsem valósítanak meg pl. kliens-szerver felépítést, erre egy vastagkliens alkalmazásnál nincs is szükség.

Az általunk elkészített programot DocuMaison (kiejtése: [dokumɛzɔ̃]) névre kereszteltük, utalva arra, hogy a fontos dokumentumainknak szeretnénk egy „házat” építeni, ahol boldogan élhetnek.

A fejezetben áttekintést adunk a program architektúrájáról, bemutatjuk az egyes komponensek feladatait és felelősségeit, továbbá részletesen ismertetjük a használt adatmodellt és a grafikus felhasználói felület felépítését.

Architektúra

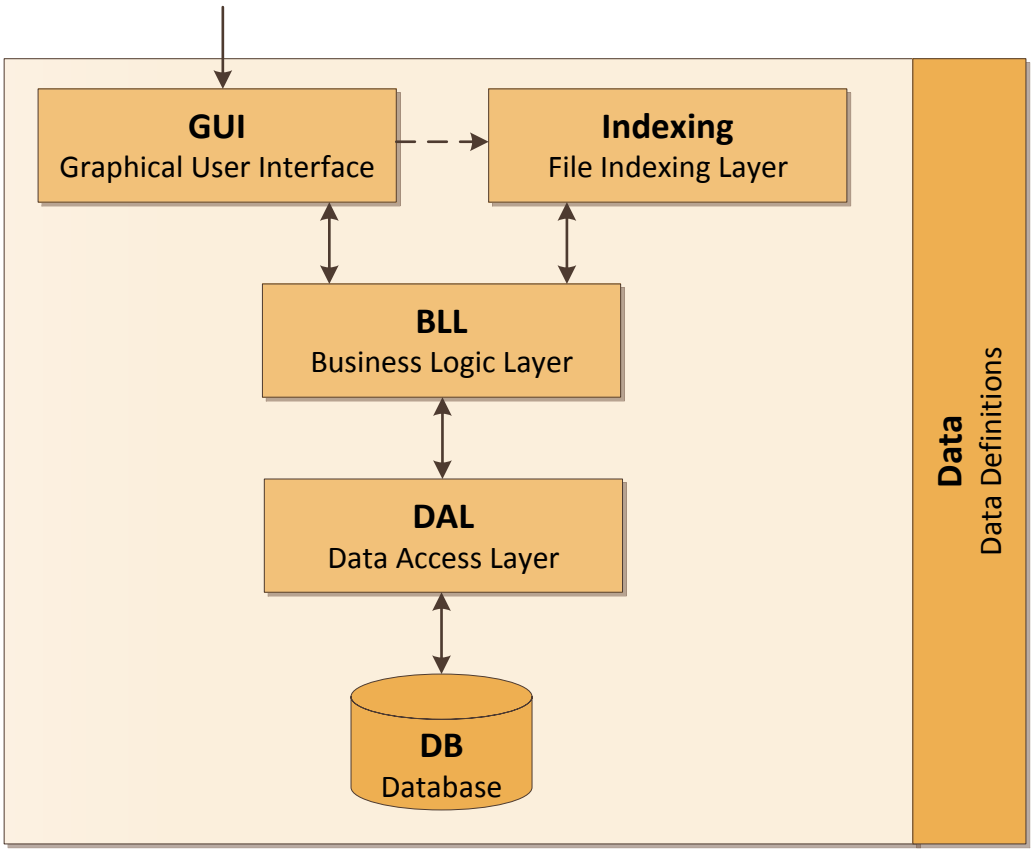
A DocuMaison architektúrája 6 különálló modulra bontható. Ebből négy a klasszikus N-rétegű alkalmazás architektúrának felel meg:

- adatbázisréteg (Database Layer, DB)
- adatelérési réteg (Data Access Layer, DAL)
- üzleti logika rétege (Business Logic Layer, BLL)
- felhasználói felület (Graphical User Interface, GUI)

Ezeket kiegészítettük még két különálló komponenssel:

- indexelést biztosító réteg (File Indexing Layer),
- adatdefiníciók (Data Definitions).

Az egyes komponensek kapcsolatait mutatja be az 1. ábra. A fejezet hátralévő részében az egyes komponensek feladatait és felelősségeit ismertetjük.



1. ábra A szoftver architektúrája

Adatbázis réteg

Célja: Az adatbázis réteg felel az adatok perzisztálásáért.

Erre mi az SQLite [1] adatbáziskezelő rendszert választottuk, a következők miatt:

- Nem szükséges hozzá kiszolgálóalkalmazás telepítése (pl. Microsoft SQL Server), ami egy felhasználói programnál felesleges terhet jelentene a felhasználók számára, nem beszélve a túlzott erőforrásigényről.
- A kiszolgáló nélküli adatbáziskezelők közt az SQLite viszonylag elterjedt, számos területen használják.
- Nyílt forráskódú, ingyenesen elérhető rendszer.

Számos SQLite driver készült Java platformra. Ezek közül – a használt ORM-rendszer ajánlásai alapján – a Xerial által készített változatot használtuk, amelyet a [2] címről lehet beszerezni.

Megjelenés a kódban: tekintve, hogy ez nem egy általunk fejlesztett komponens, így a forráskódját nem használtuk fel és nem is mellékeltük. A használt binárisok a `hu.documaison.dal\*.jar` fájlok.

Az alkalmazás által használt konkrét adatbázis felépítése az Adatbázisterv fejezetben olvasható (18. oldal).

Adatdefiníciók

Célja: Annak érdekében, hogy a program minden komponensében elérhetők legyenek az egyes adatdefiníciók, ezt egy külön komponensként készítettük el. Három részre tagolható a komponens:

- adatentitások (üzleti logikához kötődő fogalmak, mint pl. dokumentum, címke stb.),
- kereséshez kötődő osztályok (pl. keresőkifejezés),
- programspecifikus kivételek.

Az egyes adatentitásokat 1–1 Java osztályként valósítottuk meg. Annak érdekében, hogy az adathozzáférési réteg megfelelően tudja ezeket perzisztálni, alkalmaztuk az ORMLite objektumrelációs leképezést biztosító keretrendszer annotációit az entitásokat leíró osztályokban, így például itt vannak megadva az egyes mezők és osztályok táblákra és rekordokra történő leképezései is.

Ügyeltünk a tervezés során arra is, hogy az itt definiált entitások amellett, hogy közvetlenül perzisztálhatók legyenek az adatbázisba, használhatók legyenek felsőbb szinteken, egészen a grafikus felhasználói felületig. Ennek megfelelően választottuk meg az egyes mezők és metódusok láthatóságát (így például az adatbázisban az entitás azonosítására használt azonosító később már nem változtatható meg).

Mivel az itt definiált entitások és az adatbázis felépítése egymásnak jól megfeleltethető, ezért ezeket együttesen, az Adatbázisterv fejezetben mutatjuk be (18. oldal).

Ebben a komponensben kerültek definiálásra a **keresőkifejezések** is. Egy keresőkifejezés (SearchExpression) n darab atomi kifejezésből (Expression) áll. Megadható, hogy a keresőkifejezés az atomi kifejezések ÉS vagy VAGY kapcsolatából álljon elő. Egy atomi keresőkifejezés egy metaadatnévből (pl. „szerző”), egy értékből (pl. „George Orwell”) és egy operátorból (pl. „=”) áll. A támogatott operátorok halmaza a következő: egyenlő, nem egyenlő, kisebb, nagyobb, kisebb vagy egyenlő, nagyobb vagy egyenlő, tartalmazza, hasonló (like).

Ez alapján például ki lehet listázni az összes Donald Knuth összes 1970 előtti könyvét: (szerző=Donald Knuth) ∧ (kiadási év<1970), ahol a zárójelben szereplő kifejezések 1-1 atomi kifejezésnek felelnek meg.

Megjelenés a kódban: a közös adatelemek definíciója a `hu.documaison.data` projektben történt meg.

Adathozzáférési réteg (Data Access Layer)

Célja: a külső, objektumrelációs leképezést (ORM) biztosító eszközzel együttműködve adathozzáférés biztosítása a felsőbb rétegek számára.

Ennek megfelelően a réteg feladatai:

- **üres adatbázis** létrehozása (az adatdefiníciós komponensben definiáltak alapján, az ORM eszköz felhasználásával),
- **új entitások** létrehozása az adatbázisban (az adatdefiníciós komponensben definiáltak alapján, az ORM eszköz felhasználásával),
- igény szerinti **adathozzáférés biztosítása** felsőbb rétegnek az adatbázishoz (az adatdefiníciós komponensben definiáltak alapján, az ORM eszköz felhasználásával).

Megjelenés a kódban: az adathozzáférési réteg megvalósítása a `hu.documaison.dal` projektben történt meg.

A réteg által nyújtott szolgáltatások a felsőbb rétegek számára (kódban a `DalInterface` interfészben került definiálásra):

- Új entitás példányosítása és perzisztálása.
    - Az alábbi entitások példányosítására van lehetőség:
        - dokumentum (`Document`)
        - dokumentumsablon (`DocumentType`)
        - metaadat (`Metadata`)
        - alapértelmezett metaadat (`DefaultMetadata`)
        - megjegyzés (`Comment`)
        - címke (`Tag`)
    - Ekkor általános esetben mindössze az entitás egyedi azonosítója (id) lesz kitöltve. Kivétel ez alól a dokumentum, ahol a dokumentumsablon mező

is ki lesz töltve, illetve a címke, ahol a címke nevét tartalmazó mező ki lesz töltve.

- Entitás törlése (azonosítója alapján).
  - Az alábbi entitások törlésére van lehetőség:
    - dokumentum (`Document`)
    - dokumentumsablon (`DocumentType`)
    - metaadat (`Metadata`)
    - alapértelmezett metaadat (`DefaultMetadata`)
    - megjegyzés (`Comment`)
    - címke (`Tag`)
- Entitás tulajdonságainak módosítása.
  - Az alábbi entitások módosítására van lehetőség:
    - dokumentum (`Document`)
    - dokumentumsablon (`DocumentType`)
    - metaadat (`Metadata`)
    - alapértelmezett metaadat (`DefaultMetadata`)
    - megjegyzés (`Comment`)
    - címke (`Tag`)
- Dokumentumok lekérése.
  - Az alábbiak szerint biztosít az adatelérési réteg lehetőséget dokumentumok lekérdezésére:
    - egy dokumentum lekérése az azonosítója alapján,
    - egy adott címkével rendelkező dokumentumok lekérése,
    - egy adott címkehalmaz bármelyikével rendelkező dokumentumok lekérése,
    - az összes létező dokumentum lekérése,
    - egy adott keresőkifejezés által meghatározott dokumentumok lekérése (a 8. oldalon ismertetett módon).
  - Bár az adatelérési réteg viszonylag alacsonyszintű és kevés szolgáltatást nyújt, a lekérdezések gazdag támogatására szükség van, hiszen így elkerülhető, hogy túlzottan sok adatot kelljen az adatbázisból felolvasni. Az-

zal, hogy csak az aktuálisan szükséges adatokat kérjük le az adatbázisból, erőforrást takarítunk meg, ami nagyszámú dokumentumnál jelentős lehet.

- Címkék lekérése.
    - Lehetőségek:
        - összes címke lekérése,
        - egy adott nevű vagy adott azonosítójú címke lekérése.
- Egy dokumentumhoz címke hozzáadása vagy elvétele.

A réteg fő feladata a fenti funkciók megvalósítása az ORMLite által nyújtott lehetőségek felhasználásával. Az ORMLite viszonylag gazdag lehetőségeket biztosít az adatbázisműveletek végrehajtására, például lehetőség van a lekérdezéseket is Java nyelven felépíteni. Így az adathozzáférési rétegben nem található nehezen karbantartható és biztonságilag aggályos SQL-lekérdezések.

Ez a réteg valósít meg néhány segédfunkciót is, mint az üres adatbázis létrehozása. Robosztussági okokból minden alkalommal, amikor nem létező adatbázisból próbál az alkalmazás adatot elérni, létrehozunk egy új, üres adatbázist.

Üzleti logikai réteg (Business Logic Layer)

Célja: az adatelérési rétegtől kapott adatok alapján kiszolgálni a grafikus felhasználói felületet. Lényegében egy *wrapper facade*-ként viselkedik.

Esetünkben ez a réteg viszonylag kevés feladatot lát el, hiszen az üzleti logika nagyon egyszerű a DocuMaison programban. Azonban ennek ellenére külön definiáltuk, hiszen a program továbbfejlesztése során később még hasznos lehet.

Jelenleg a felsőbb rétegek felé nyújtott interfésze (amely a `BllInterface` interfészben került definiálásra) ugyanazokat a műveleteket teszi lehetővé, mint az előző fejezetben ismertetett DAL interfész (`DalInterface`), csak esetenként kényelmesebben elérhető módon (pl. több paraméterrel).

Egy jelentős különbség van a DAL és a BLL által nyújtott szolgáltatásokban, ez pedig az új dokumentum létrehozása. A DAL rétegben ez csak a dokumentum azonosítóját és a dokumentumsablont állítja be. A BLL esetén emellett az új dokumentum megkapja a dokumentumsablonból következő alapértékeit is (alapértelmezett metaadatok, alapértelmezett bélyegkép). Emellett BLL szintjén végezzük el például a dokumentumok mozgatásának műveletét is (összefogja a tényleges fájlműveletet és az adatbázisműveletet egy műveletté).

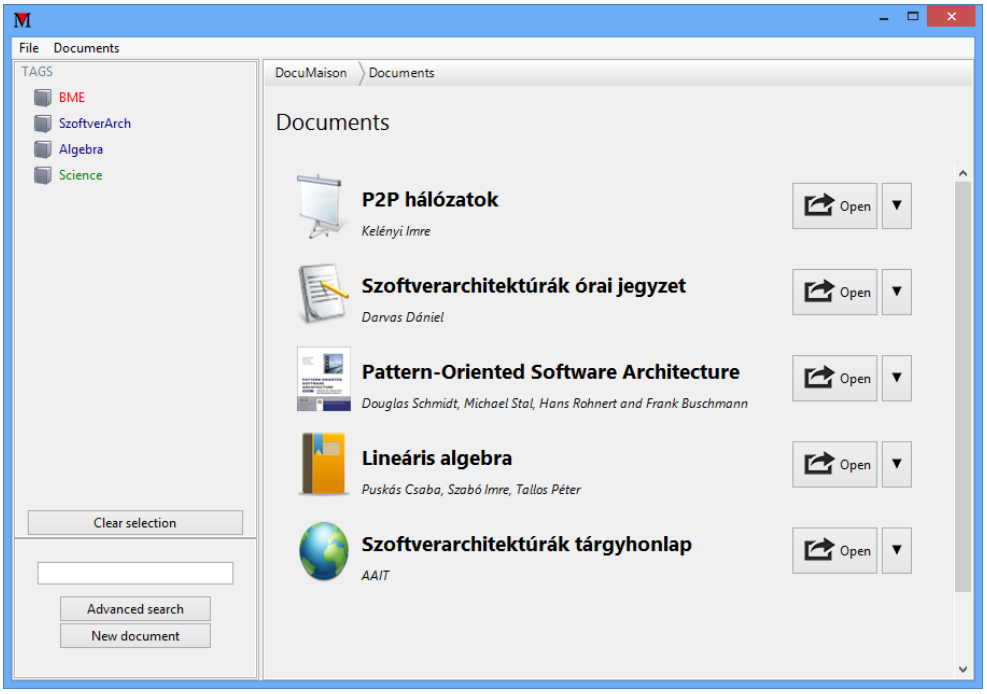
Megjelenés a kódban: az üzleti logikai réteg megvalósítása a `hu.documaison.bll` projektben történt meg.

Grafikus felhasználói felület

Célja: A felhasználók számára egyszerű, könnyen átlátható felületet nyújtani, az összes funkciót elérhetővé tenni.

A DocuMaison grafikus felületét a változatos platformokon elérhető SWT, azaz *Standard Widget Toolkit* könyvtárral készítettük el. A grafikus felület egy *egyablakos* megoldás, azaz falugró ablakok helyett, minden a főablakban jelenik meg (leszámítva néhány esetet, ahol a felhasználói interakció úgy a legkönnyebben megvalósítható)

A program indulásakor a felhasználót az összes dokumentum kilistázására alkalmas képernyővel fogadja az alkalmazás (lásd 2. ábra)

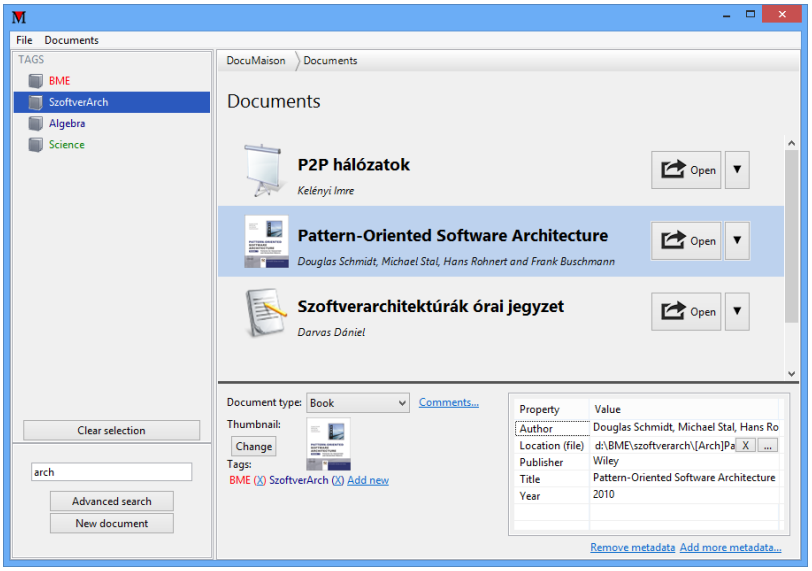


2. ábra: Kezdőképernyő az összes felvett dokumentummal

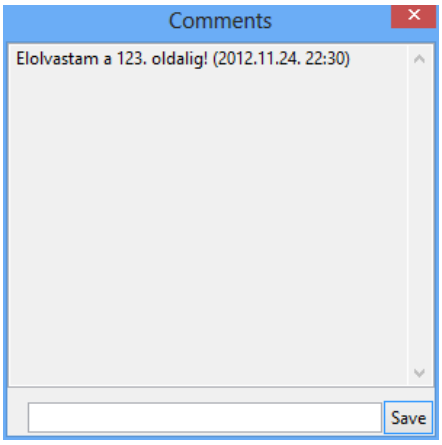
A képernyő bal oldalán az adatbázisban található címkék vannak felsorolva. Ezekre kattintva kilistázhatóak azon dokumentumok, amelyekhez a kiválasztott címke van rendelve. Lehetőség van több címke kiválasztására is (a Shift gomb lenyomásával).

Egy dokumentum részletesebb adatai a dokumentumra kattintva érhetjük el (lásd 3. ábra). A részletes adatok közt megtalálható az összes hozzárendelt metaadat és annak értéke, a címkéi, a bélyegképe valamint a típusa. Bármelyik adata módosítható, valamint

további metaadatok rendelhetőek hozzá. A *Comments* hivatkozással lehetőség van megjegyzések hozzáfűzésére is (lásd 4. ábra)

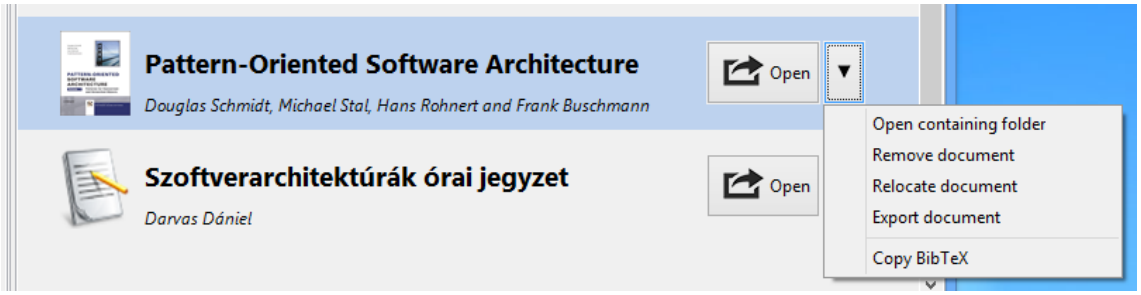


3. ábra: Dokumentum részletes adatainak megtekintése



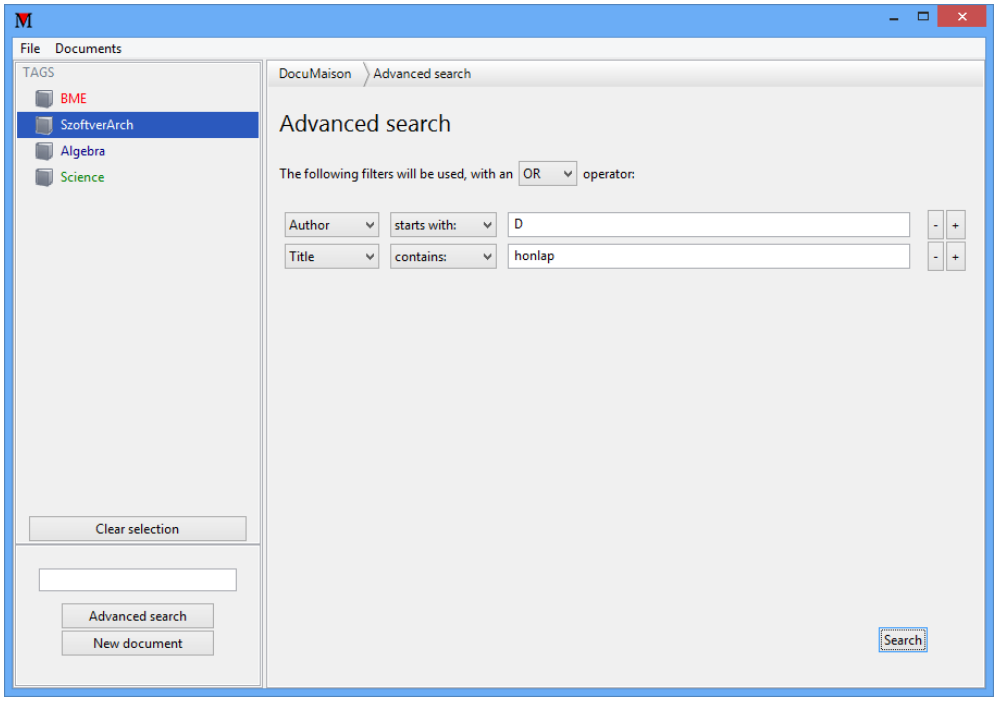
4. ábra: Dokumentumhoz fűzött megjegyzések

A dokumentumot az *Open* gomb megnyomásával lehet megnyitni (ilyenkor az operációs rendszerben hozzárendelt alkalmazás nyitja meg). Ezen felül az egyes dokumentumokkal néhány további műveletet is el lehet végezni: például a dokumentumok exportálása, áthelyezése, törlése vagy a *BibTex* hivatkozás másolása (lásd 5. ábra)



5. ábra: Dokumentumokkal végezhető műveletek

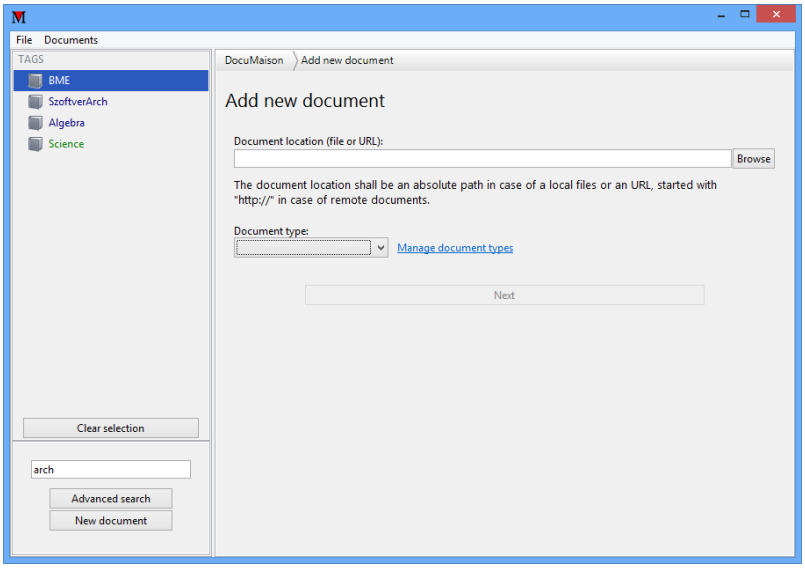
A címkék alatt egy általános kereső mező található meg, amely szabadszavas keresőként működik. A keresőszót beírva indíthatunk keresést. Az alkalmazás a beírt keresőszót az összes metaadatban keresi, az eredményeket a dokumentumlistázóban (a képernyő jobb oldalán) jeleníti meg. Lehetőség van részletesebb keresést is végezni, ahogy a 6. ábrán látható.



6. ábra: Részletes keresés

A részletes keresésben tetszőleges metaadatra lehet megkötéseket tenni: például pontos egyezést, meg nem egyezést vagy akár intervallumilleszkedést is kiköthetünk. A különböző feltételek ÉS vagy VAGY kapcsolatban lehetnek egymással.

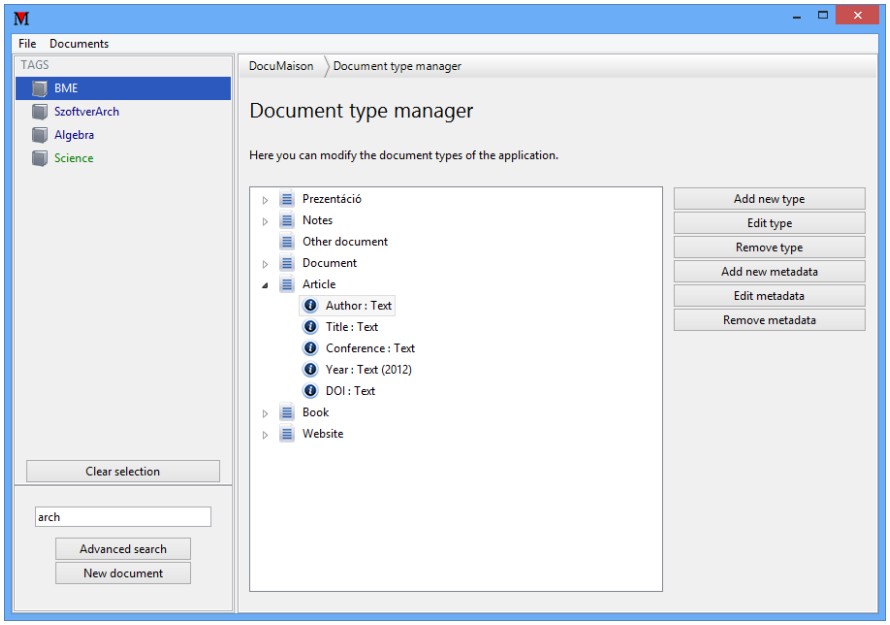
A képernyő bal alsó sarkában levő gombra kattintva új dokumentumot vehetünk fel az alkalmazásba. Az ehhez tartozó „varázsló” két ablakból áll. Az első a dokumentum helyét és típusát kéri be, ahogy a ábrán látható.



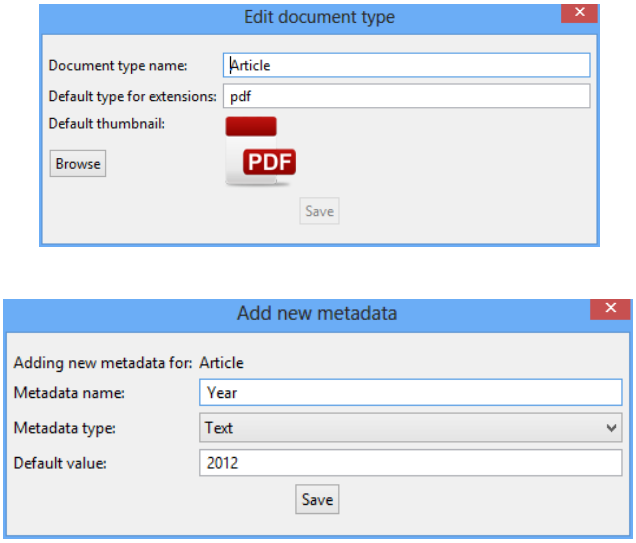
7. ábra: Új dokumentum hozzáadása ablak

A második ablakon a különböző metaadatok értéke adható meg, illetve a dokumentumhoz rendelt metaadatok halmaza tetszőleges bővíthető.

A dokumentumok típussal rendelkeznek. A típusok menedzselését egy külön ablakon lehet elvégezni (lásd ábra). Ez elérhető az új dokumentum létrehozása ablakból, vagy a *Documents* menüpontból.

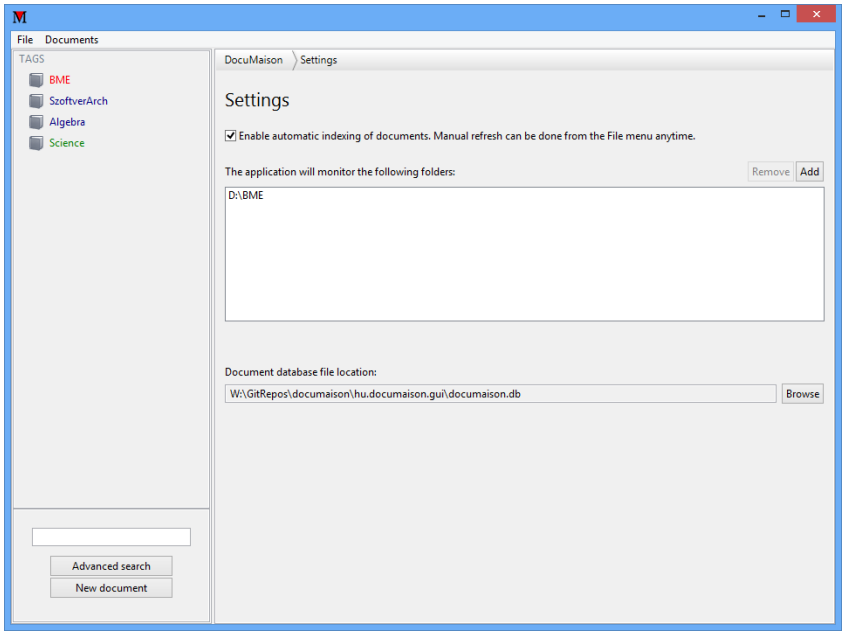


8. ábra: Dokumentum típusok menedzselése



9. ábra: Dokumentum típus módosítása, új metaadat hozzáadása

Az alkalmazás beállításait a *File* menü *Preferences* menüpontjából érhetjük el (lásd 10. ábra). Ebben az ablakban beállíthatjuk, hogy az alkalmazás hol tárolja az adatbázisát, illetve, hogy milyen könyvtárakat indexeljen. Lehetőség van az indexálás letiltására is – ilyenkor manuálisa a *File* menüből tudunk frissíteni.



10. ábra: DocuMaison beállításai

Megjelenés a kódban: A grafikus felület a `hu.documaison.gui` projektben van implementálva. Ez a projekt indul el és hivatkozik a többire.

Indexelő réteg

Célja: Megadott könyvtárak tartalmának összevetése az adatbázisban szereplő dokumentumokkal és az esetleges változások követése.

A jelenlegi verzióban egy egyszerű indexelő modult készítettünk el. Ez az alábbi főbb funkciókat látja el:

- A beállításokból beolvassa az indexelendő (figyelendő) könyvtárak listáját. (A könyvtárak változásait on-demand figyeljük, az operációs rendszer által biztosított esetleges változásértesítést nem használjuk ki többek közt a Java 6 platform limitációi miatt.)
- Amennyiben egy, az adatbázisban szereplő dokumentum nem található meg a lemezen, akkor az törlésre kerül az adatbázisból.
    - A dokumentum csak azon a számítógépen törlődhet az adatbázisból, ahol az bekerült. Így elejét vehetjük annak, hogy ha más gépen megnyitjuk az adatbázist és ott nem állnak rendelkezésre a dokumentumok a lemezen, akkor kitörlődjenek az adataink.
- Ha a figyelt mappákban olyan fájlt találunk, ami az adatbázisban eddig még nem szerepelt, megkíséreljük felvenni. Ha van olyan dokumentumtípus, amihez tartozó alapértelmezett kiterjesztések közt szerepel az újonnan talált fájl kiterjesztése, akkor ezt felhasználva felvesszük az adatbázisba, különben nem.

Természetesen számos továbbfejlesztési ötletünk van ehhez a részhez, azonban ez túlmutat a tárgy keretein.

Annak érdekében, hogy a dokumentumok hozzáadására és adatbázisból eltávolítására a program többi része megfelelően reagálhasson, szükséges bizonyos kapcsolatok bevezetése modulok közt. Túlzottan szoros függést azonban nem szerettünk volna bevezetni (például az indexelő rétegnek ne kelljen például a grafikus interfészről specifikus ismeretekkel rendelkeznie), így itt a megvalósítás során *Interceptor* mintát használtunk.

Megjelenés a kódban: az indexelő réteg megvalósítása a `hu.documaison.indexing` projektben történt meg.

Adat- és adatbázisterv

Ebben a fejezetben ismertetjük az egyes üzleti entitásokat és ezek adatbázisra történő leképzését.

Az ORMLite által támogatott kapcsolatmodell

Az adatdefiníciók előtt célszerű megismerkedni azzal, hogyan támogatja az általunk használt ORM keretrendszer, az ORMLite az entitások közti kapcsolatokat.

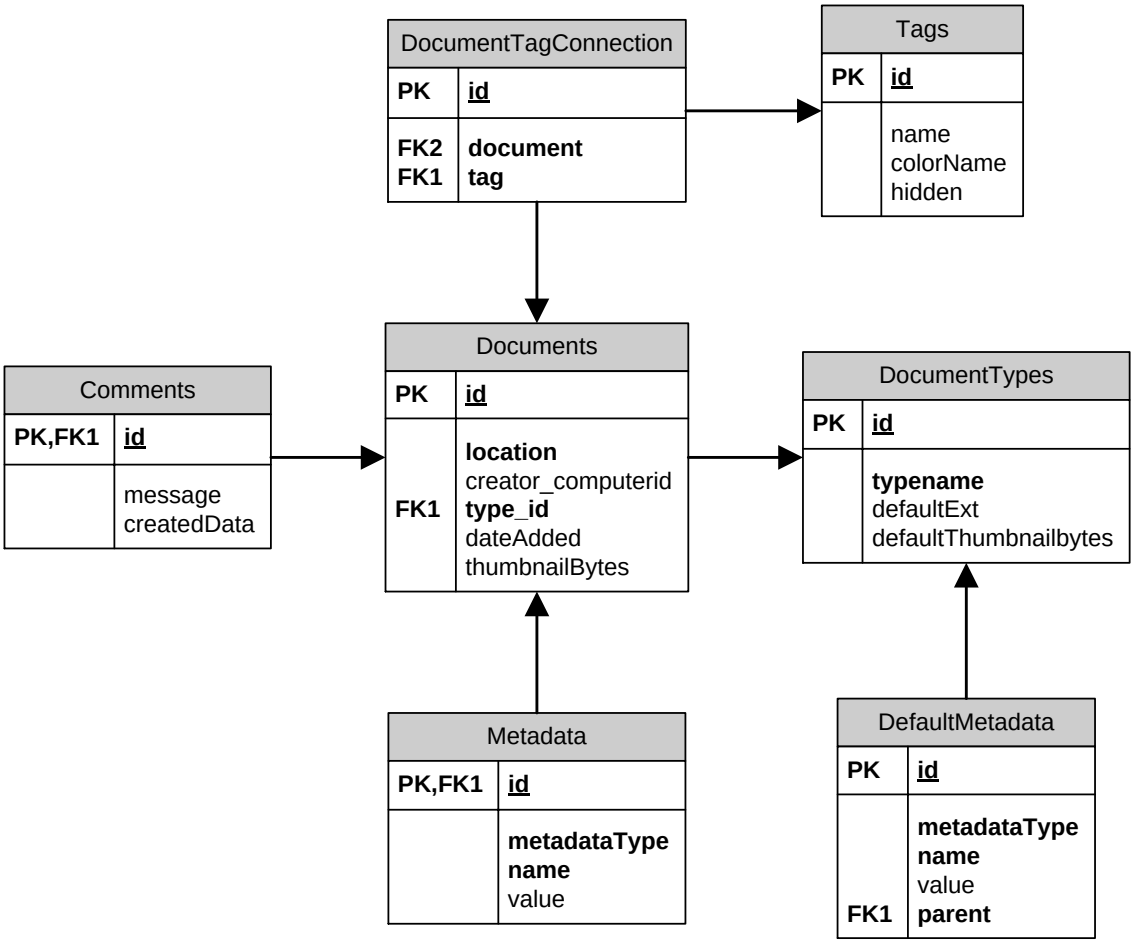
Az ORMLite rendszerben 1:1 és 1:N kapcsolatok definiálhatók, N:N kapcsolatok definíciójára nincs lehetőség.

Az 1:1 kapcsolatok esetén mindkét entitás tárol referenciát a kapcsolat másik oldalán található entitásra. Az adatbázisban ilyenkor mindkét entitás tárolni fogja a párjának kulcsát idegen kulcsként.

Amennyiben 1:N kapcsolatot alkalmazunk, úgy az N végen lévő entitások tartalmazni fognak 1-1 referenciát a kapcsolat túloldalán lévő entitásra, illetve az adatbázisban is rögzítésre kerülnek a megfelelő idegen kulcsok. Az N oldalon viszont az adatbázisban nem fogunk kulcsokat tárolni. Ennek ellenére az ORMLite lehetővé teszi, hogy Javaban mint kollekció láthassunk a kapcsolat N oldalán résztvevő entitásokat. Ehhez egy speciális, `ForeignCollection` osztályt kell használnunk, ennek a példányosítását az ORMLite végzi el. Ahol ilyen `ForeignCollection<X>` adattípust használtunk, azt a későbbiekben szereplő táblázatban „X gyűjtemény”-ként tüntetjük fel.

Az adatbázis entitás-relációs diagramja

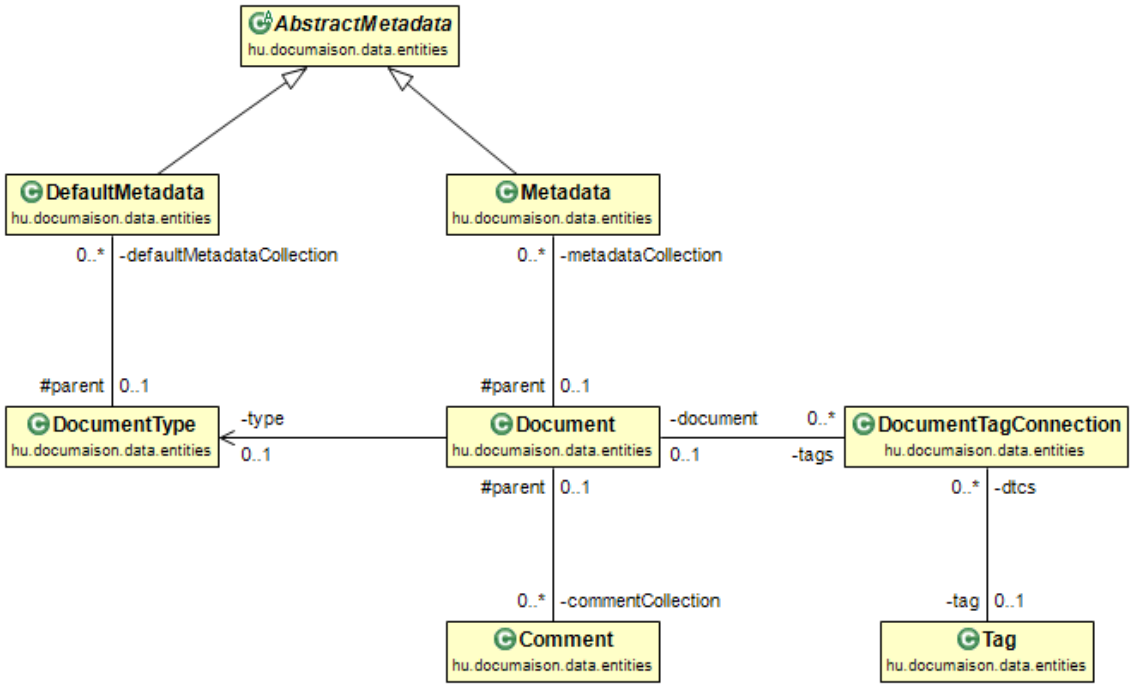
A következő alfejezetekben részletezésre kerülő adatbázis entitás-relációs diagramját mutatja az alábbi ábra (1. ábra).



1. ábra Az adatbázis felépítésének áttekintőábrája

A program objektummodellje

A következő alfejezetekben részletezésre kerülő adatmodell osztálydiagramját mutatja a 2. ábra. Megjegyzendő, hogy az ábrán szereplő összes osztály a saját `DatabaseObject` osztályból származik, azonban az áttekinthetőség érdekében ezt nem jelenítettük meg az ábrán.

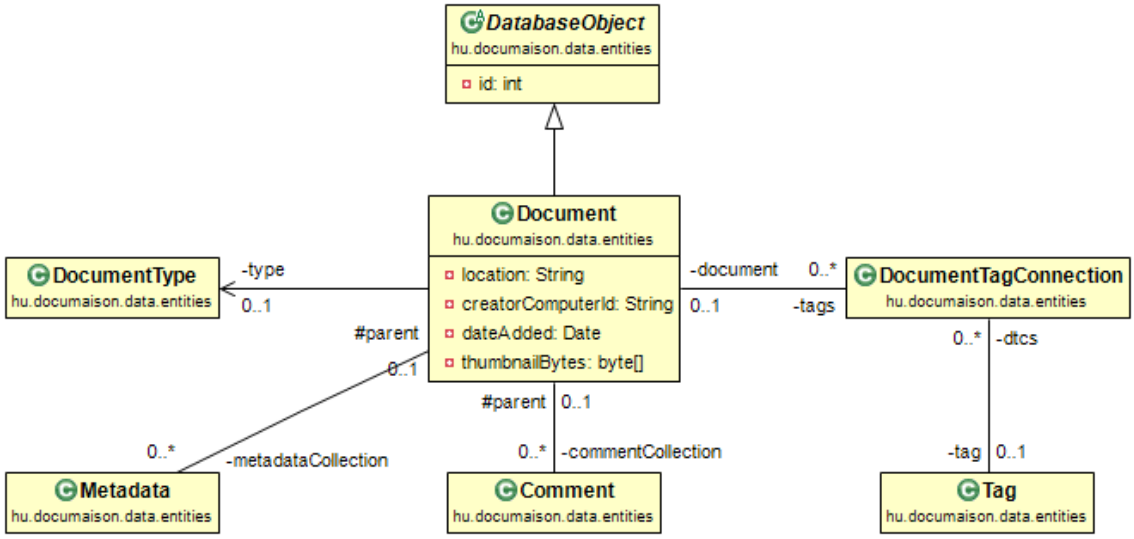


2. ábra Az adatmodell áttekintő osztálydiagramja

Dokumentum entitás

Célja: egy dokumentum (például könyv, konferenciacikk, lementett weboldal) adatainak reprezentálása.

Leképzése a kódban: a `Document` osztály. (Környezetét mutatja a 3. ábra.)



3. ábra A Document osztály környezete

Leképzése az adatbázisban: a `Documents` tábla 1-1 sorára.

Tulajdonságai

Java mezőnév	Java adattípus	SQLite adattípus
id	int	INTEGER PRIMARY KEY
location	String	TEXT
creatorComputerId	String	TEXT
type	DocumentType	INTEGER (idegen kulcs)
dateAdded	Date	TIMESTAMP
thumbnailBytes	byte[]	BLOB
tags	DocumentTagConnection gyűjtemény	–
metadataCollection	Metadata gyűjtemény	–
commentCollection	Comment gyűjtemény	–

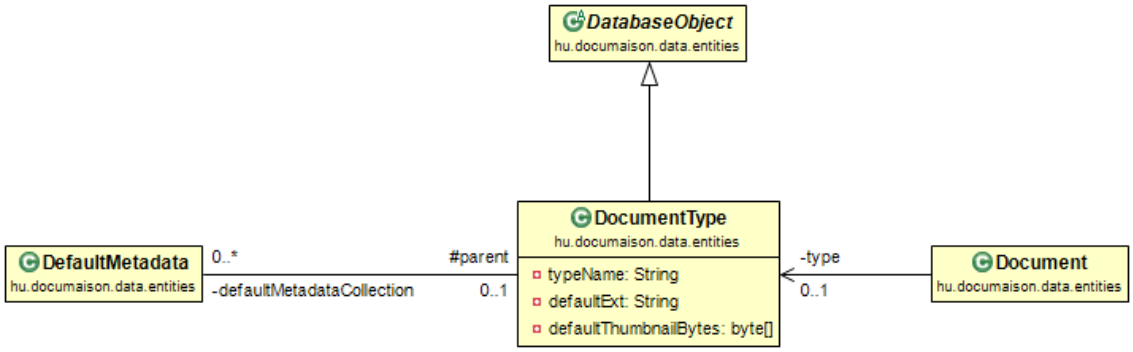
- Az `id` mező tárolja az entitás egyedi azonosítóját. Ez alapján lehet az adatbázisban az entitást azonosítani.
- A `location` mező tárolja a dokumentum elérési útját.
- A creatorComputerId mező tárolja annak a számítógépnek az azonosítóját, ami az adott dokumentum entitást létrehozta.
- A `type` mező tárolja a dokumentumhoz tartozó dokumentumsablont.
- A `dateAdded` mező tárolja a dokumentum létrehozásának idejét.
- A `thumbnailBytes` mező tárolja a dokumentum bélyegképeként megjelenítendő képet, png formátumban.

- A `tags` mező megadja a dokumentumhoz rendelt címkék gyűjteményét.
- A `metadataCollection` mező megadja a dokumentumhoz rendelt metaadatok gyűjteményét.
- A `commentCollection` mező megadja a dokumentumhoz rendelt megjegyzések gyűjteményét.

Dokumentumtípus entitás

Célja: egy dokumentum típusának (pl. konferenciacikk, weboldal stb.) reprezentálása.

Leképzése a kódban: a `DocumentType` osztály. (Környezetét mutatja a 4. ábra.)



4. ábra A DocumentType osztály környezete

Leképzése az adatbázisban: a `DocumentTypes` tábla 1-1 sorára.

Tulajdonságai

Java mezőnév	Java adattípus	SQLite adattípus
id	int	INTEGER PRIMARY KEY
typeName	String	TEXT
defaultExt	String	TEXT
defaultThumbnailBytes	byte[]	BLOB
defaultMetadataCollection	DefaultMetadata gyűjtemény	–

- Az `id` mező tárolja az entitás egyedi azonosítóját. Ez alapján lehet az adatbázisban az entitást azonosítani.
- A `type` mező tárolja a dokumentumtípus nevét.
- A `defaultExt` mező tárolja a dokumentumtípushoz tartozó fájlok kiterjesztéseit.

- A `defaultThumbnailBytes` mező tárolja a dokumentum bélyegképeként alapértelmezetten megjelenítendő képet, png formátumban.
- A `defaultMetadataCollection` mező megadja a dokumentumhoz alapértelmezetten rendelt metaadatok gyűjteményét.

Címke entitás

Célja: egy dokumentum egy címkéjének (tagének) reprezentálása (pl. „munka”).

Leképzése a kódban: a `Tag` osztály.

Leképzése az adatbázisban: a `Tags` tábla 1-1 sorára.

Tulajdonságai

Java mezőnév	Java adattípus	SQLite adattípus
id	int	INTEGER PRIMARY KEY
name	String	TEXT
colorName	String	TEXT
hidden	boolean	INTEGER

- Az `id` mező tárolja az entitás egyedi azonosítóját. Ez alapján lehet az adatbázisban az entitást azonosítani.
- A `name` mező tárolja a címke nevét (ez kerül megjelenítésre).
- A `colorName` mező tárolja a címkéhez tartozó színt (szövegesen, html színkódként).
- A `hidden` mező tárolja a címke rejtettségét, azaz hogy meg kell-e jeleníteni a címkét.

Metaadat entitás

Célja: egy dokumentum egy metadadat-példányának (lényegében egy dokumentumhoz rendelt kulcs-érték párnak) reprezentálása (pl. „szerző = George Orwell”).

Leképzése a kódban: a `Metadata` osztály.

Leképzése az adatbázisban: a `Metadata` tábla 1-1 sorára.

Tulajdonságai

Java mezőnév	Java adattípus	SQLite adattípus
id	int	INTEGER PRIMARY KEY
metadataType	MetadataType	INTEGER
name	String	TEXT
value	String	TEXT
parent	Document	INTEGER (idegen kulcs)

- Az `id` mező tárolja az entitás egyedi azonosítóját. Ez alapján lehet az adatbázisban az entitást azonosítani.
- A `name` mező tárolja a metaadat kulcsát (nevét, pl. „szerző”).
- A `value` mező tárolja a metaadat kulcsához tartozó értéket (pl. „George Orwell).
- A `metadataType` mező tárolja a metaadat értékének típusát (lehetőségek: Text, Integer, Date).
- A `parent` mező tárolja, hogy melyik dokumentumhoz tartozik a metaadat.

Alapértelmezett metaadat entitás

Célja: egy dokumentumtípus egy alapértelmezett metadadat-példányának (lényegében egy dokumentumhoz rendelt kulcs-érték párnak) reprezentálása (pl. „év = null”). Egy újonnan létrehozott dokumentum alapértelmezetten tartalmazni fogja a dokumentumtípusában definiált metaadatokat.

Leképzése a kódban: a `DefaultMetadata` osztály.

Leképzése az adatbázisban: a `DefaultMetadata` tábla 1-1 sorára.

Tulajdonságai

Java mezőnév	Java adattípus	SQLite adattípus
id	int	INTEGER PRIMARY KEY
metadataType	MetadataType	INTEGER
name	String	TEXT
value	String	TEXT
parent	DocumentType	INTEGER (idegen kulcs)

- Az `id` mező tárolja az entitás egyedi azonosítóját. Ez alapján lehet az adatbázisban az entitást azonosítani.
- A `name` mező tárolja az alapértelmezett metaadat kulcsát (nevét, pl. „év”).

- A `value` mező tárolja az alapértelmezett metaadat kulcsához tartozó értéket. Nem kötelezően kitöltendő.
- A `metadataType` mező tárolja a metaadat értékének típusát (lehetőségek: Text, Integer, Date).
- A `parent` mező tárolja, hogy melyik dokumentumtípushoz tartozik a metaadat.

Megjegyzés entitás

Célja: egy dokumentumhoz rendelt megjegyzés reprezentálása.

Leképzése a kódban: a `Comment` osztály.

Leképzése az adatbázisban: a `Comments` tábla 1-1 sorára.

Tulajdonságai

Java mezőnév	Java adattípus	SQLite adattípus
id	int	INTEGER PRIMARY KEY
message	Date	TEXT
createdDate	String	TIMESTAMP
parent	Document	INTEGER (idegen kulcs)

- Az `id` mező tárolja az entitás egyedi azonosítóját. Ez alapján lehet az adatbázisban az entitást azonosítani.
- A `message` mező tárolja a megjegyzés szövegét.
- A `createdDate` mező tárolja a megjegyzés létrehozásának időpontját.
- A `parent` mező tárolja, hogy mely dokumentumhoz tartozik a megjegyzés.

Dokumentum–címke összerendelés technikai entitás

Célja: egy dokumentumok és címkék összerendelésének tárolása. (Az ORMLite nem támogatja a több-több kapcsolatot, így azt fel kellett bontanunk két egy-több kapcsolatra és erre a kapcsolótáblára.)

Leképzése a kódban: a `DocumentTagConnection` osztály.

Leképzése az adatbázisban: a `DocumentTagConnection` tábla 1-1 sorára.

Tulajdonságai

Java mezőnév	Java adattípus	SQLite adattípus
id	int	INTEGER PRIMARY KEY
document	Document	INTEGER (idegen kulcs)
tag	Tag	INTEGER (idegen kulcs)

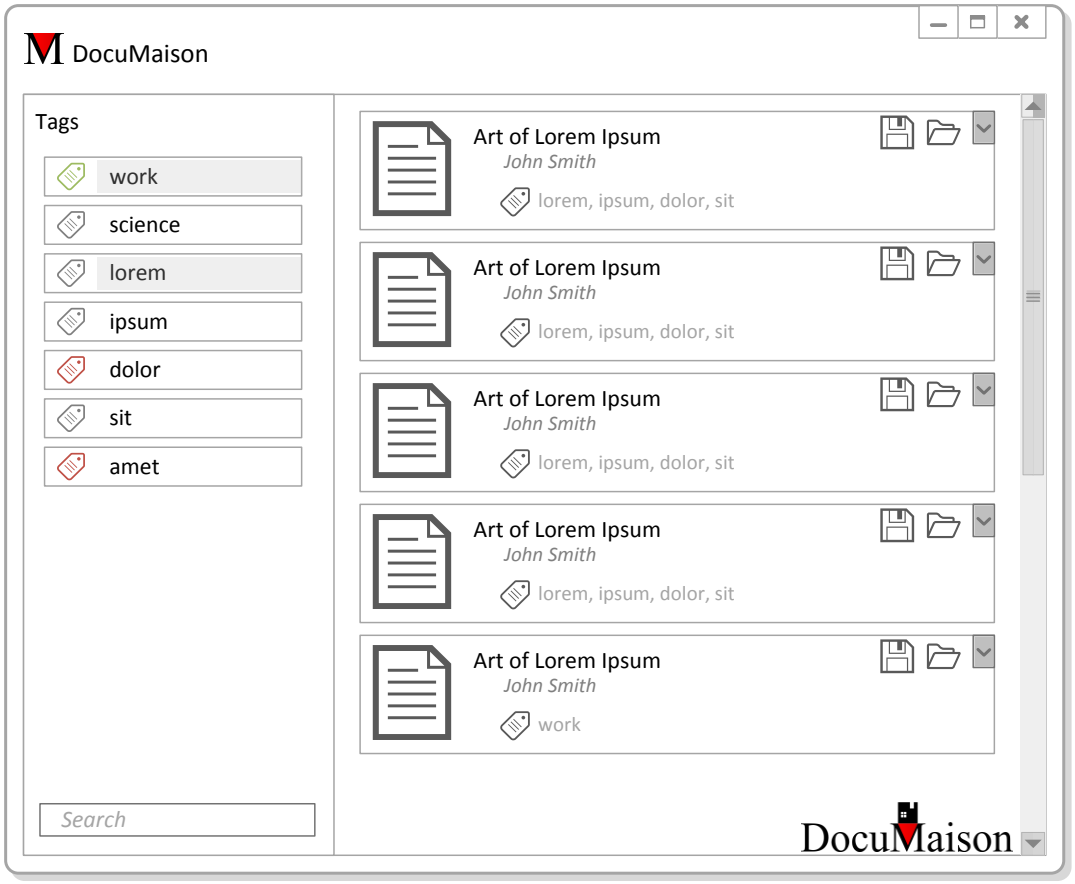
- Az `id` mező tárolja az entitás egyedi azonosítóját. Ez alapján lehet az adatbázisban az entitást azonosítani.
- A `document` mező tárolja az összerendelésben szereplő dokumentumot.
- A `tag` mező tárolja az összerendelésben szereplő címkét.

GUI-terv

A program tervezésekor az egyszerű felépítést tartottuk szem előtt. Ennek megfelelően a program érdemi működését egyetlen ablak látja el. Ennek felépítését mutatja az 5. ábra.

A képernyő bal oldalán a dokumentumok címkéi vannak felsorolva, illetve egy keresőmező található. A képernyő jobb oldalán a kiválasztott címkéknek vagy a keresőkifejezésnek megfelelő dokumentumok fontosabb adatai láthatók.

Itt csak az alapkoncepcióra igyekeztünk példát mutatni. A dokumentumok részletes adatainak felvételére szolgáló ablak, a beállítások ablaka és egyéb, specifikusabb felhasználói felületek leírása a „Grafikus felhasználói felület” című fejezetben olvasható (12. oldal).



5. ábra A grafikus felhasználói felület felépítése

Telepítési leírás

A DocuMaison program készítésekor hangsúlyt fektettünk az egyszerű kezelhetőségre és üzembehelyezhetőségre. A szoftvert úgy készítettük el, hogy az első indításnál mind az üres adatbázist, mind egy alapértelmezett tartalmú konfigurációs fájlt a megfelelő helyeken létrehozzon. Így a program telepítése mindössze annyiból áll, hogy a közzétett fájlokat egy könyvtárba kicsomagoljuk, majd elindítjuk a `DocuMaison_win_32.jar` fájlt (64 bites Windows esetén a `DocuMaison_win_64.jar` indítható). Az átadott bináris állomány mellett egy példa adatbázist is mellékeltünk – a tesztelést, kipróbálást támogatva –, amennyiben ez a bináris mellett van, az első induláskor ezt tölti be az alkalmazás.

A program **rendszerkövetelményei**:

- Java futtatókörnyezet (legalább 1.6 verziójú)
    - A program tetszőleges operációs rendszeren futtatható, amelyre a Java futtatókörnyezet és az SWT ablakozó rendszer elérhető, azonban csak Windows és Mac OS operációs rendszereken teszteltük. Eltérő platformokhoz a platformhoz szükséges SWT könyvtárral újra kell fordítani az alkalmazást.
- 1 GHz vagy gyorsabb processzor
- legalább 128 MB szabad memória
- legalább 100 MB szabad lemezterület

A program készítése során felhasznált eszközök

- Eclipse Juno
  - Felhasználás: fejlesztőkörnyezet.
- SQLite [1]
  - Felhasználás: adatbázis-kezelő rendszer.
- OrmLite [3]
  - Felhasználás: az adatok objektum-relációs leképezésére.
- Nebula Project [4]
  - Felhasználás: grafikus felhasználói felület elemei.
- Opal [6]
  - Felhasználás: grafikus felhasználói felület elemei.
- Microsoft Visio
  - Felhasználás: a dokumentum ábráinak előállítására.
- Microsoft Word
  - Felhasználás: a dokumentáció elkészítésére.
- ObjectAid UML Explorer [5]
  - Felhasználás: UML osztálydiagramok előállítása a Java forráskód alapján.
- Java Architecture for XML Binding (JAXB) [7]
  - Felhasználás: a konfigurációs fájl mentésére és betöltésére.
- GitHub
  - Felhasználás: verziókezelésre, csapatmunka támogatására.

Összefoglalás

Munkánk során megterveztük, implementáltuk illetve dokumentáltuk a DocuMaison nevű dokumentumnyilvántartó rendszert. Az elkészített alkalmazás segítségével dokumentumokat tárolhatunk és rendszerezhetünk.

A megvalósított alkalmazás 3 rétegű architektúrát használ: adatbázis réteg, üzleti logikai réteg és felhasználói felület. Az alkalmazás az adatokat egy helyi fájlban az SQLite adatbázis kezelő-rendszer tárolja. A grafikus megjelenítést az SWT platformfüggetlen megvalósítással oldottuk meg. Az alkalmazásunk képes dokumentumok automatikus indexelésére is, azaz új dokumentumok – igény esetén – automatikusan bekerülnek az adatbázisba.

Munkánk során részletes terveket készítettünk – részük jelen dokumentum tartalmát képezik – és jelentős mennyiségű implementációs munkát is végeztünk. Ennek eredményeképpen egy jól működő és megbízható alkalmazást készítettünk el, amely az elvárt alapvető igényeknek megfelel, feladatát képes ellátni.

Továbbfejlesztési lehetőségek

A DocuMaison alkalmazással kapcsolatban számos továbbfejlesztési lehetőséget látunk. Ezek közül számosat tervezünk is megvalósítani a tárgy keretein túlmenően is.

Néhány továbbfejlesztési lehetőség:

- Metaadatok automatikus beolvasása a beindexelt fájlokból. (Ez kihívásnak tűnik, különösen a többplatformúság miatt.)
- Bélyegképek automatikus készítése a beindexelt fájlok alapján.
- Együttműködés ebookolvasókkal: exportálás segítése, támogatás a szinkronizálásra stb.
- Testreszabhatóság növelése: lehetőség létrehozása pluginekkel történő bővítésre.

A fejlesztés folyamán is odafigyeltünk ezekre a továbbfejlesztési irányokra és igyekeztünk olyan tervezői döntéseket hozni, amelyek segítik a program fejlesztésének folytatását.

Hivatkozások

[1] SQLite honlapja
http://www.sqlite.org/

[2] Xerial SQLite driver
http://www.xerial.org/maven/repository/artifact/org/xerial/sqlite-jdbc/

[3] OrmLite – Lightweight Object Relational Mapping Java Package honlapja
http://ormlite.com/

[4] Nebula Project honlapja
http://www.eclipse.org/nebula/

[5] ObjectAid UML Explorer honlapja
http://www.objectaid.com/home

[6] Opal honlapja
http://code.google.com/a/eclipselabs.org/p/opal/

[7] Java Architecture for XML Binding (JAXB) honlapja
http://www.oracle.com/technetwork/articles/javase/index-140168.html

DocuMaison

DocuMaison Telepítési leírás

A DocuMaison program készítésekor hangsúlyt fektettünk az egyszerű kezelhetőségre és üzembehelyezhetőségre. A szoftvert úgy készítettük el, hogy az első indításnál mind az üres adatbázist, mind egy alapértelmezett tartalmú konfigurációs fájlt a megfelelő helyeken létrehozzon. Így a program telepítése mindössze annyiból áll, hogy a közzétett fájlokat egy könyvtárba kicsomagoljuk, majd elindítjuk a `DocuMaison_win_32.jar` fájlt (64 bites Windows esetén a `DocuMaison_win_64.jar` indítható). Az átadott bináris állomány mellett egy példa adatbázist is mellékeltünk – a tesztelést, kipróbálást támogatva –, amennyiben ez a bináris mellett van, az első induláskor ezt tölti be az alkalmazás.

A program **rendszerkövetelményei**:

- Java futtatókörnyezet (legalább 1.6 verziójú)
    - A program tetszőleges operációs rendszeren futtatható, amelyre a Java futtatókörnyezet és az SWT ablakozó rendszer elérhető, azonban csak Windows és Mac OS operációs rendszereken teszteltük. Eltérő platformokhoz a platformhoz szükséges SWT könyvtárral újra kell fordítani az alkalmazást.
- 1 GHz vagy gyorsabb processzor
- legalább 128 MB szabad memória
- legalább 100 MB szabad lemezterület