**Dokumentáció**

1. **Felhasználói dokumentáció**

Az alkalmazás, egy egyszerű teendő lista (todo) app. Használata egyszerű, mégis sokat segíthet a napi feladatok észben tartásánál. A felső szövegmezőben egy leírást lehet készíteni a teendőről, majd a Hozzáad gomb megnyomásával bekerül a listába.

Amennyiben valamelyiket teljesítettük, úgy törölni lehet a szöveg melletti Törlés gomb megnyomásával, így átláthatóbb lesz a lista.

Az alkalmazás csak portré módban, azaz álló helyzetben használható.

1. **Fejlesztői dokumentáció**

Négy osztály, és két felületet leíró xml található a forráskódban.

* 1. **MainActivity.java**
     1. onCreate()

Az onCreate() metódusban példányosítom az adatbázist kezelő osztályt, illetve a ListView-t, melyről a későbbiekben bővebben is írok. Ezek után meghívásra kerül az updateUI() metódus.

* + 1. updateUI()

A list privát változóba lekérem az adatbázisban lévő összes feladatot, majd ezt a listát átadom a TodoAdapter osztálynak, amely feldolgozza, és végül megjeleníti.

* + 1. addTaskNow()

Ez a függvény a Hozzáad gomb megnyomására megkeresi a beviteli szövegmezőt, és kinyeri belőle a beírt szöveget. Amennyiben ez a szöveg üres, akkor egy Toast üzenetet küld, hogy írja be a felhasználó a teendő leírását. Ha beírta, és megnyomta a gombot, akkor példányosít egy Task osztályt, és átadja neki a beírt szöveget, majd elmenti az adatbázisba. Végül egy Log bejegyzést készít, kiüríti a beviteli mezőt és frissíti a felületet.

* + 1. deleteTask()

Bármely törlés gomb megnyomására megkeresi az adott gomhoz tartozó task\_id-t mely egy egyedi kulcs az adatbázisban. Ezt az értéket átadja az adatbázist kezelő osztály deleteTask metódusának, mely törli a táblából. Végül frissíti a felületet, így látható, hogy kikerült a táblából is az elem.

* + 1. getDate()

Ebben a metódusban a felületre kihelyezett dátum választó gomb megnyomásakor megjelenő felugró ablak létrehozását és megjelenítését kezelem. Gyakorlatilag annyi történik, hogy az adott napi (mai) dátumot jelenítem meg.

* 1. **Task.java**

Ez az osztály reprezentálja a Task (feladat) objektumot. Két privát adattagja van, ezek a taskName (feladat neve/leírása) és az id (egyedi azonosító).

Néhány egyszerű metódust találunk benne, melyekkel kinyerhetők az adattagok értékei, illetve beállíthatók azok (get, set).

* 1. **TodoAdapter.java**

Ez az osztály azért került létrehozásra, mert a felületre valahogy ki kellett helyezni a feladat leírását, és egy láthatatlan mezőbe a feladat egyedi azonosítóját. Erre azért volt szükség, mert ha két teljesen azonos leírás kerül a listába, és valamelyiket törölni szeretnénk, akkor azt a legkönnyebben úgy tudjuk megtenni, hogy az egyedi azonosító alapján távolítjuk el a táblából. Ezt a getView() metódusban készítettem el, ahol egyesével kerülnek kiírásra a listában szereplő elemek.

* 1. **DbHelper.java**

Ebben az osztályban valósítottam meg az SQLite adatbázishoz való kapcsolódást, illetve a CRUD (Create, Read, Update, Delete) műveleteket.

Az osztály elején létrehoztam néhány statikus változót, melyek az adatbázis főbb információit tartalmazzák. Erre azért volt szükség, hogy ne kelljen minden esetben leírni a tábla és mezők nevét, így minimalizálva a félre gépelésből adódó hibákat. Ezek a mezők az alábbiak:

* DATABASE\_VERSION melynek értéke 1, ezt a példányosításkor kell használni
* DATABASE\_NAME, ami az adatbázis nevét tartalmazza
* TABLE\_TASK, ami a tábla nevét tartalmazza
* KEY\_ID, az egyedi kulcs oszlopának nevét tartalmazza
* KEY\_TASKNAME, a feladat nevét tartalmazó oszlop neve
  + 1. onCreate()

Ez a függvény minden példányosításkor lefut, és amennyiben nem létezik még a tábla, akkor létrehozza azt, a fenti két mezővel. Fontos kiemelni, hogy az ID mező értéke autoincrement tulajdonsággal bír, így nem kell azzal törődni, hogy milyen érték kerül be, mert az adatbázis motor ezt automatikus elvégzi.

* + 1. onUpgrade()

Ez is minden esetben lefut, és amennyiben verzióváltás történik, akkor törli a régi táblát, és meghívja az onCreate() függvényt, mely létrehozza az újat.

* + 1. addTask()

Lekérdezi az írható táblákat, majd egy ContentValues típusú változóba beírja, hogy melyik mezőbe szeretnénk eltárolni a megadott feladat leírást, ezt is eltárolja a változóban, végül az insert metódus meghívásával beírja a táblába.

* + 1. getAlltasks()

Egy feladatokat tartalmazó listával tér vissza. Gyakorlatilag lekérdezi az összes, tasks táblában lévő sort, majd ezeket egyesével 1-1 Task osztályba rendezi, és ezeket az osztályokat adja át a visszatérési listának.

* + 1. updateTask()

Ezt a függvényt csak azért hoztam létre, hogy ha később tovább fejlesztem az alkalmazást egy olyan funkcióval, melyben módosíthatom az adott feladat tulajdonságait, akkor már csak meghívni kelljen ezt a metódust.

Itt annyi történik, hogy megkeresi ID alapján, hogy melyik Taskról van szó, és módosítja a megfelelő mezőket.

* + 1. deleteTask()

ID alapján megkeresi a megfelelő sort a táblában, és törli azt.

* 1. **activity\_main.xml**
  2. **todo\_list.xml**
  3. **Egyéb módosítások**