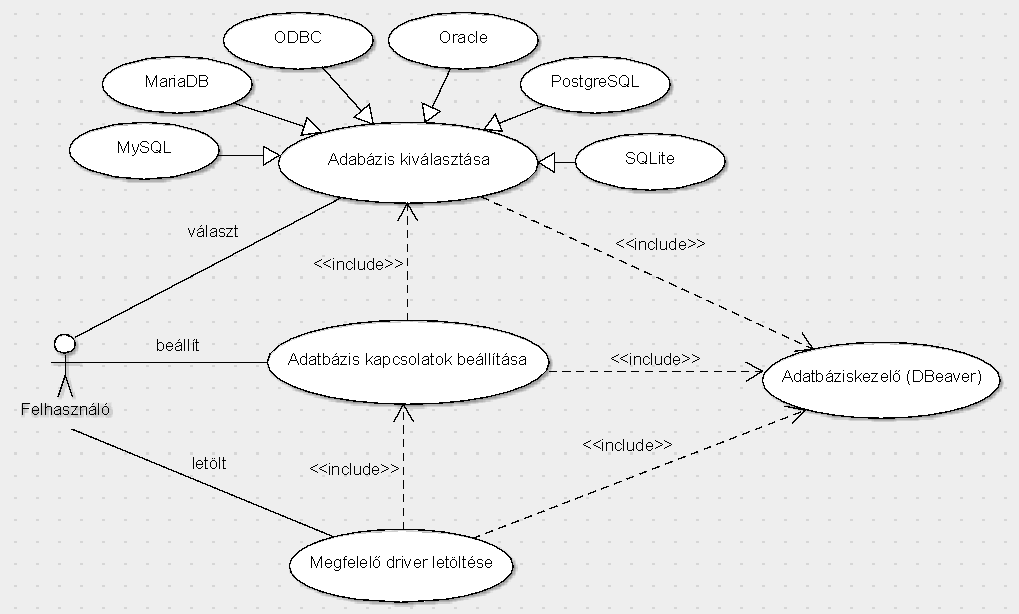
Projekthez tartozó diagramok bemutatása

Use-case diagram: - Dobó Gergely:

1. ábra: Adatbázis kiválasztása

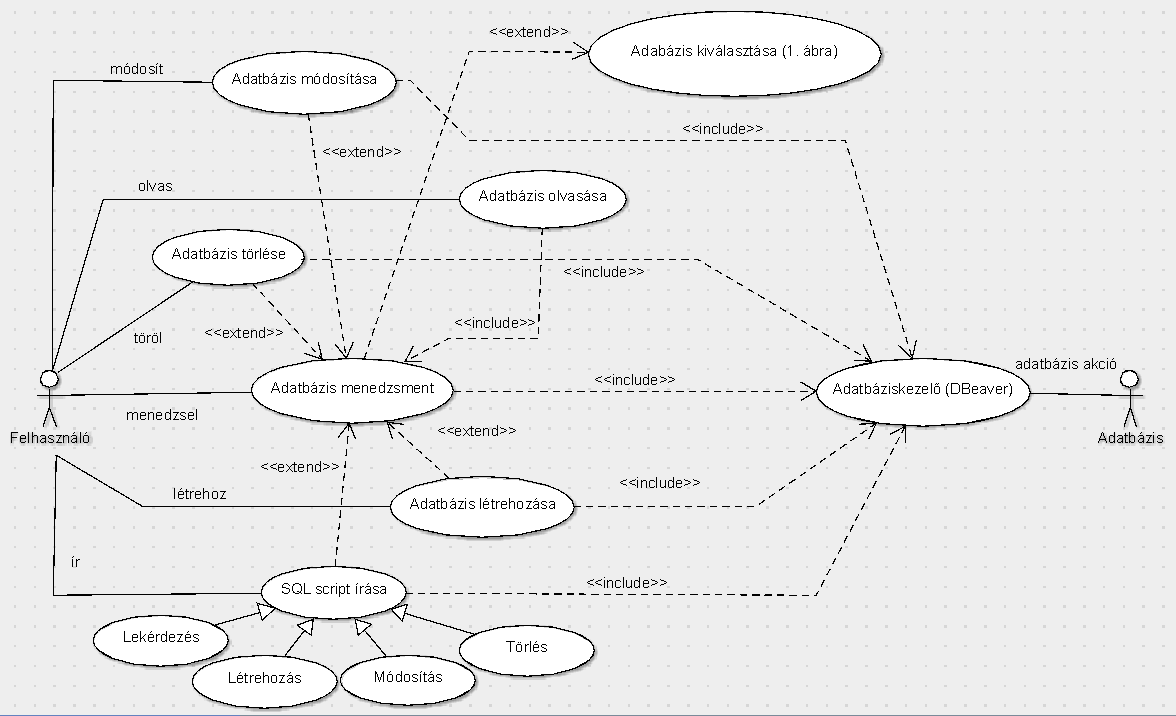
- A DBeaver elindítása után számos adatbázisból lehet választani

- A választást követően az adatbázis kapcsolatokat be kell állítani

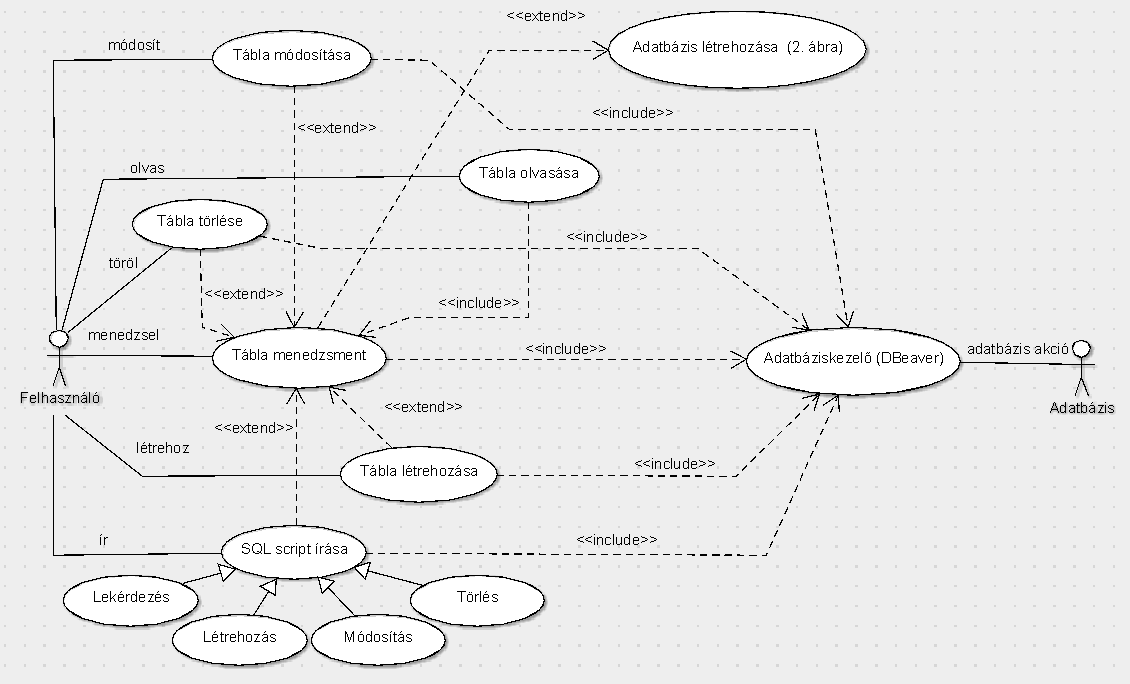
 - A kapcsolatok beállítása után le kell tölteni a megfelelő adatbázis driver-t

2. ábra: Adatbázis menedzsment

* Az adatbázis kiválasztását követően menedzselni lehet azt
  + A meglévő adatbázisok megtekinthetőek (olvashatóak)
  + Új adatbázist lehet létrehozni
  + Meglévő adatbázist lehet módosítani
  + Meglévő adatbázist lehet törölni
  + Az olvasás (lekérdezés), létrehozás, módosítás, törlés műveletek SQL script-el is megvalósíthatóak
* Az adatokat a DBeaver tárolja a választott adatbázisba

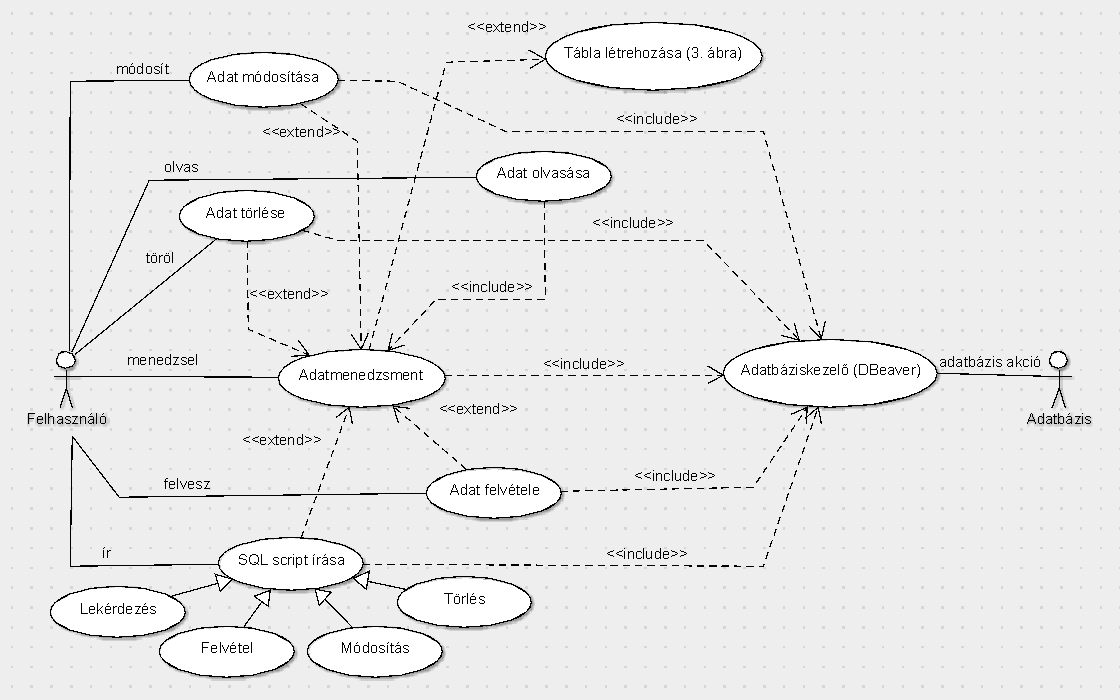


3. ábra: Tábla menedzsment

* Az adatbázis létrehozását követően menedzselni lehet annak tábláit
  + A meglévő táblák megtekinthetőek (olvashatóak)
  + Új táblát lehet létrehozni
  + Meglévő táblát lehet módosítani
  + Meglévő táblát lehet törölni
  + Az olvasás (lekérdezés), létrehozás, módosítás, törlés műveletek SQL script-el is megvalósíthatóak
* Az adatokat a DBeaver tárolja a választott adatbázisba

4. ábra: Adatmenedzsment

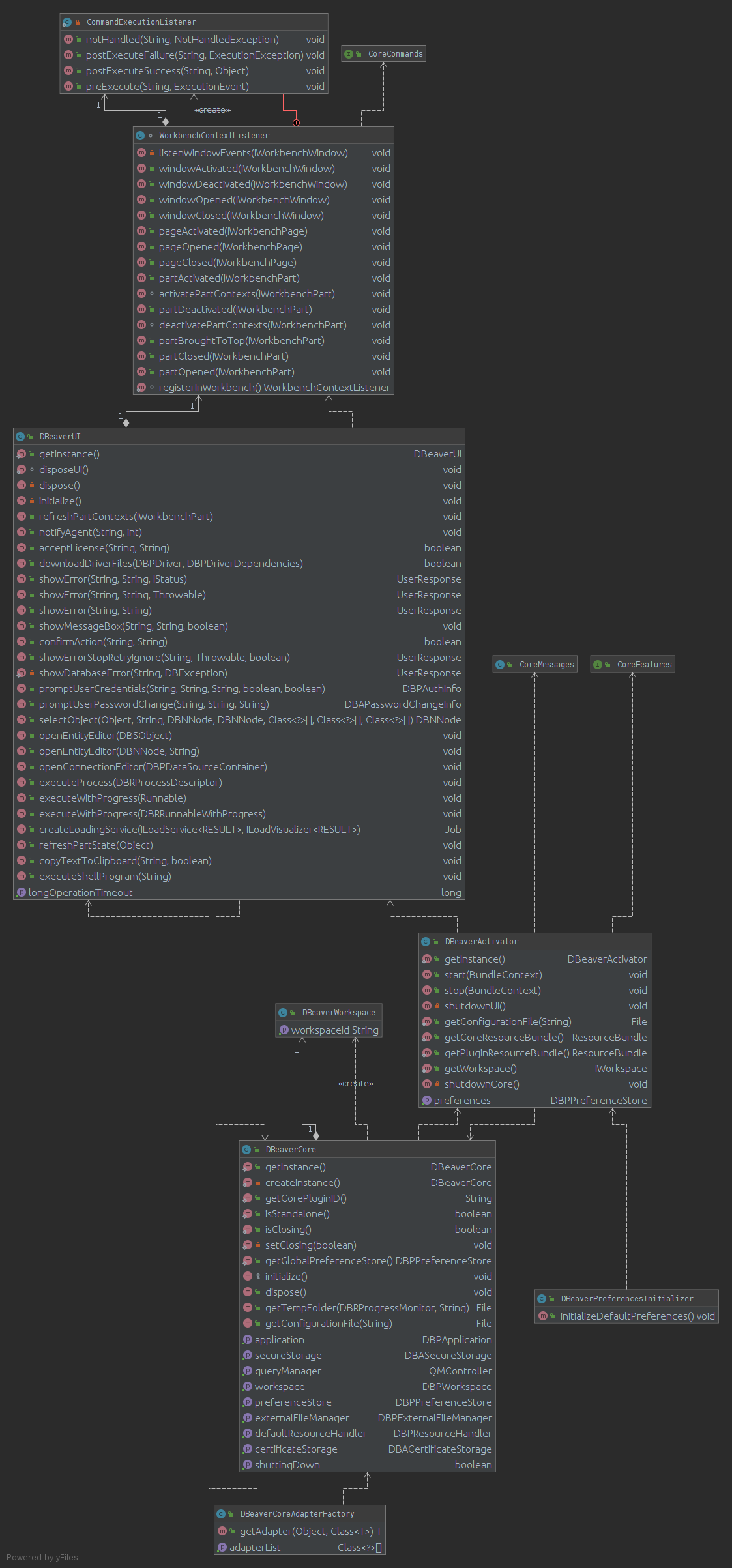
* Az tábla létrehozását követően menedzselni lehet annak sorait/rekordjait (adatait)
  + A meglévő rekordot megtekinthetőek (olvashatóak)
  + Új rekordot lehet létrehozni
  + Meglévő rekordot lehet módosítani
  + Meglévő rekordot lehet törölni
  + Az olvasás (lekérdezés), létrehozás, módosítás, törlés műveletek SQL script-el is megvalósíthatóak
* Az adatokat a DBeaver tárolja a választott adatbázisba



Class diagram: - Szabó Tamás

Mivel maga az alkalmazás eléggé összetett (elég sok pluginból,packageből és ezáltal még több java fájlból áll), ezért úgy döntöttem, hogy csak egy package-nek csinálom meg az osztály diagramját, még pedig a core-nak, mivel az alkalmazás wikipédiája alapján az tűnt az egyik fő modulnak. A diagramot az Intellij IDE segítségével generáltam, megjelenítve az osztályok közötti kapcsolatokat, illetve az osztályok metódusait és a propertyket (property = adattag, ha van gettere). Az összes adattag megjelenítése nem lett volna jó ötlet, mivel a CoreMessages osztály elég sok adattagja van, és ez így elrontotta volna a diagram összképét. A konstruktorokat szintén nem ábrázoltam a diagramon, mivel többnyire egy külön metódus a felelős az objektum inicializálásáért. Az elkövetkezőkben megpróbálom leírni a osztályok főbb feladatait:

* CoreCommands - a főbb module parancsokat tartalmazza
* CoreFeatures - a főbb featureket tartalmazza
* CoreMessages - az üzeneteket tartalmazza
* DBeaverActivator - ez class irányítja a plugin életciklusát, többek között betölti a CoreFeatures-t és a CoreMessages-t. Emellett itt találhatóak meg a az UI-t és Core-t leállító metódusok is.
* DBeaverPreferencesInitializer - DBeaver beállításainak inicializálása
* DBeaverCore - Ahogy a class neve is jelzi, ez a "magja" az alkalmazásnak, objektum létrehozása a singleton tervezési mintának megfelelő (getInstance(),createInstance(),initialize()), így mindig csak egy ilyen objektum létezik. A megvalósítás threadsafe(synchronized). Emellett tartalmazza az objektum megfelelő bezárásáról gondoskodó metódust, illetve különböző propertykhez tartozó gettereket is.
* DBeaverCoreAdapterFactory - A kódból (és névből) ítélve ez az adapter tervezési mintát valósítaná meg, de komment szerint még nincs teljesen kész. A class felhasználja mind a DBeaverCore és DBeaverUI class-t.
* DBeaverWorkspace - Wrapper osztály az Eclipse workspace számára
* DBeaverUI - Ez a package fő UI classja. A DBeaverCore-hoz hasonlóan ez is megvalósítja a Singleton tervezési mintát. Ez az osztály felel (a nevének megfelelően) a felhasználói felületen történő dolgokért (üzenetek,errorok megjelenítése, megerősítések stb).
* WorkbenchContextListener - több fajta Listener-t is implementál, így különböző aktivációkért és deaktivációkért, illetve a navigator és sql szerkesztő környezetének aktiválásáért felel. Az osztálynak van egy belső osztálya is, a CommandExecutionListener, melyből csak a postExecuteSuccess metódus van ténylegesen implementálva, mely a kód alapján a megadott azonosítójú feature (DBFeature) használatát/regisztrációját teszi lehetővé.
* Ezenkívül a package tartalmaz még egy CoreResources.properties fájlt is, ami a lokalizációért felel.

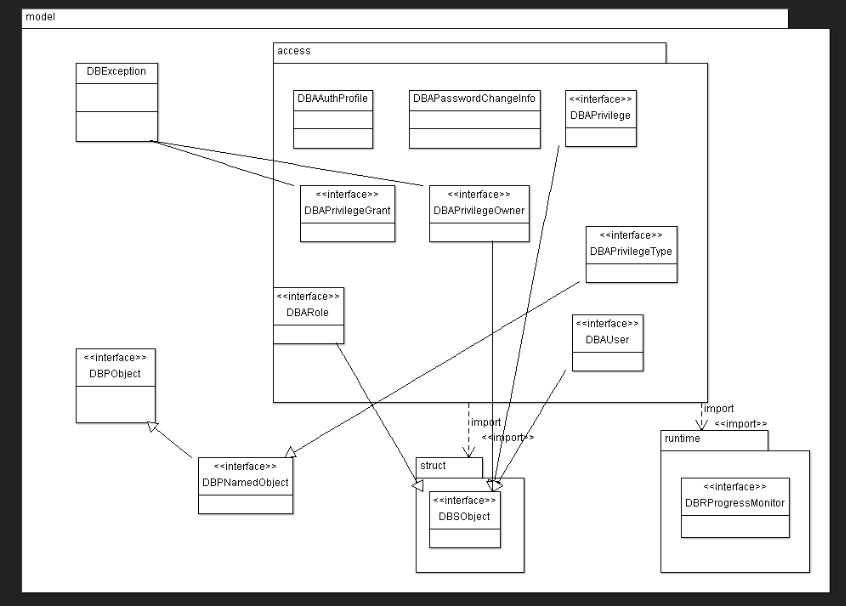


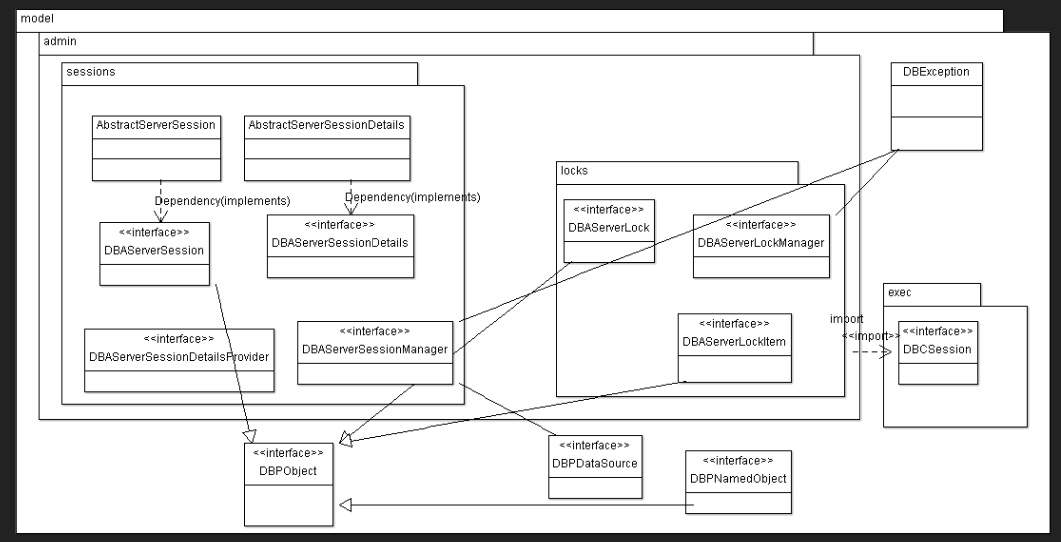
Package diagram: - Halász Gábor

A projekt 7 fő csomagból áll:

Forráskód szerkezete(fő packagek):

* docs – Többnyire elavult dokumentációt tartalmaz.
* features – Funkciók(feature) leírása. Nem tartalmaz forráskódot. A termékben található pluginok és függőségek strukturálására.
* bundles – Nagyon alap pluginokat tartalmaz.
* plugins – Fő forráskód itt helyzekedik el:
  + org.jkiss.dbeaver.model – Model API és base osztályok. Nem tartalmaz felhasználói felület(UI) függőségeket csak tiszta adat model.
  + org.jkiss.dbeaver.core – Fő DBeaver modul. A legtöbb alapvető felhasználó felületi osztályt(UI) ez a package tartalmazza.
  + org.jkiss.dbeaver.core.application – Relatíve kicsi modul, mely konfigurálja az alappevtő DBeaver applikációt.
  + org.jkiss.dbeaver.core.eclipse - Fő Eclipse plugin. Néhány extrea menüt/nézetet ad a standard Eclipse IDE-hez.
  + org.jkiss.dbeaver.ext.\* - DBeaver kiterjesztések.
* product – Végleges termék(önálló és Eclipse plugin) konfiguráció.
* test – Teszteléshez kapcsolódó unit teszteket tartalmazza.

Az átláthatóság érdekében e plugin.org.jkiss.dbeaver.model csomagon belül 2 csomagot fejtettünk ki, ugyanis közel száz csomagot tartalmaz maga a projekt. Ezen belül az access(különböző autentikáció, 1.ábra) és az admin(admin jogok, 2.ábra)



Sequence diagram: - Kuba Bence