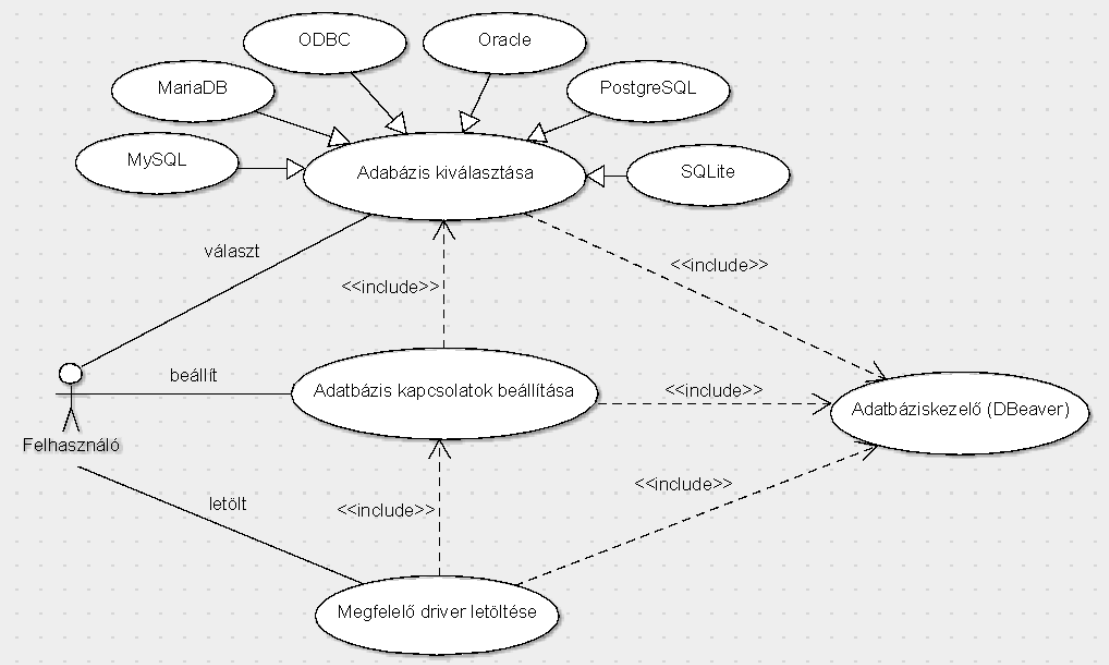
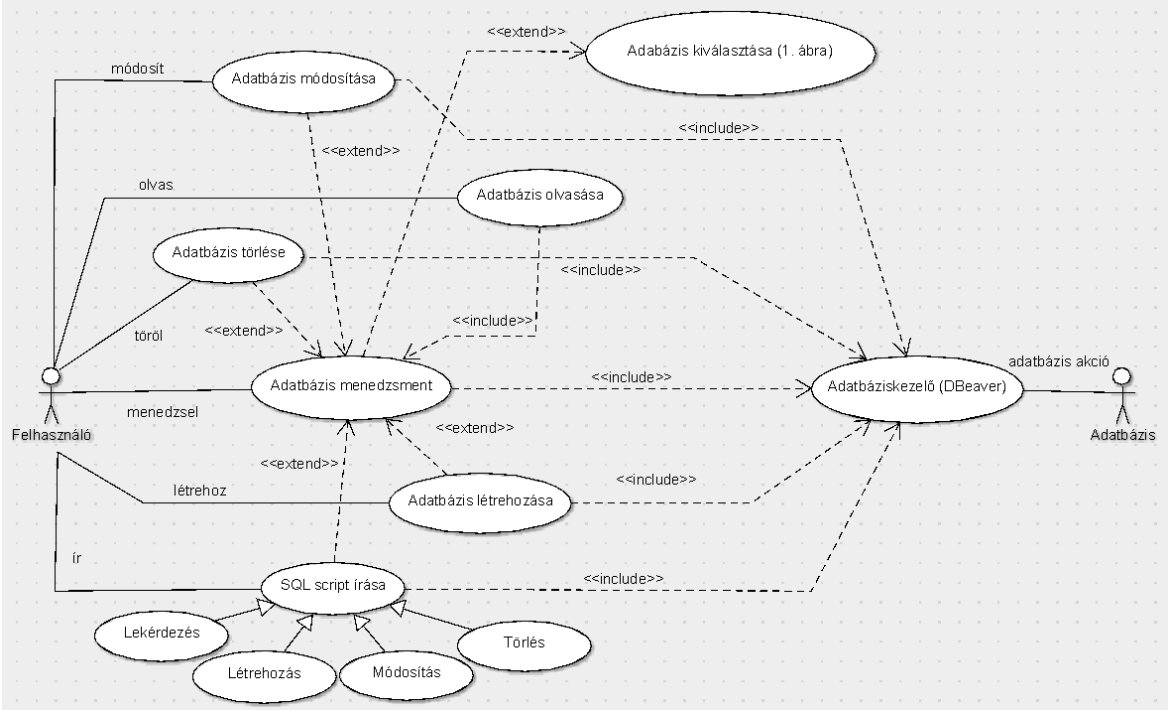
A választott projekt bemutatása

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

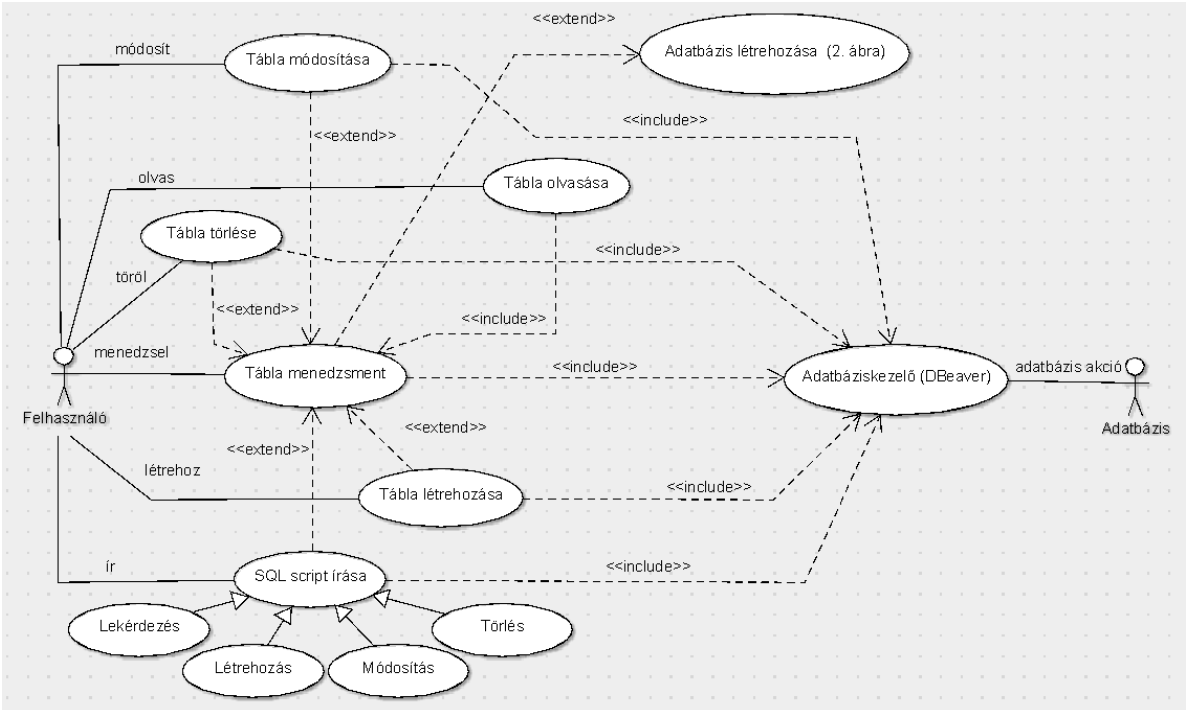


2. ábra: Adatbázis menedzsment
    - Az adatbázis kiválasztását követően menedzselni lehet azt
        - A meglévő adatbázisok megtekinthetőek (olvashatóak)
        - Új adatbázist lehet létrehozni
        - Meglévő adatbázist lehet módosítani
        - Meglévő adatbázist lehet törölni
        - Az olvasás (lekérdezés), létrehozás, módosítás, törlés műveletek SQL szkriptel is megvalósíthatóak
    - Az adatokat a DBeaver tárolja a választott adatbázisba



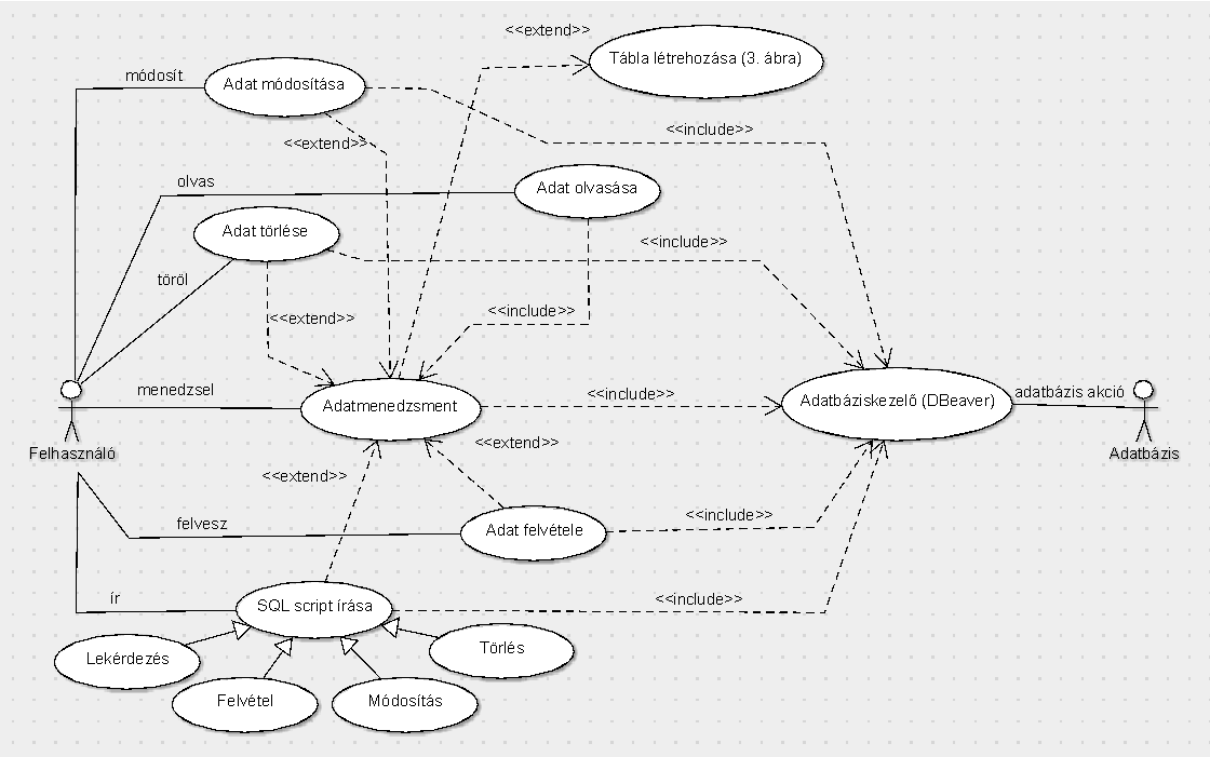
3. ábra: Tábla menedzsment

- 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 szkriptel is megvalósíthatóak
- Az adatokat a DBeaver tárolja a választott adatbázisba



4. ábra: Adatmenedzsment

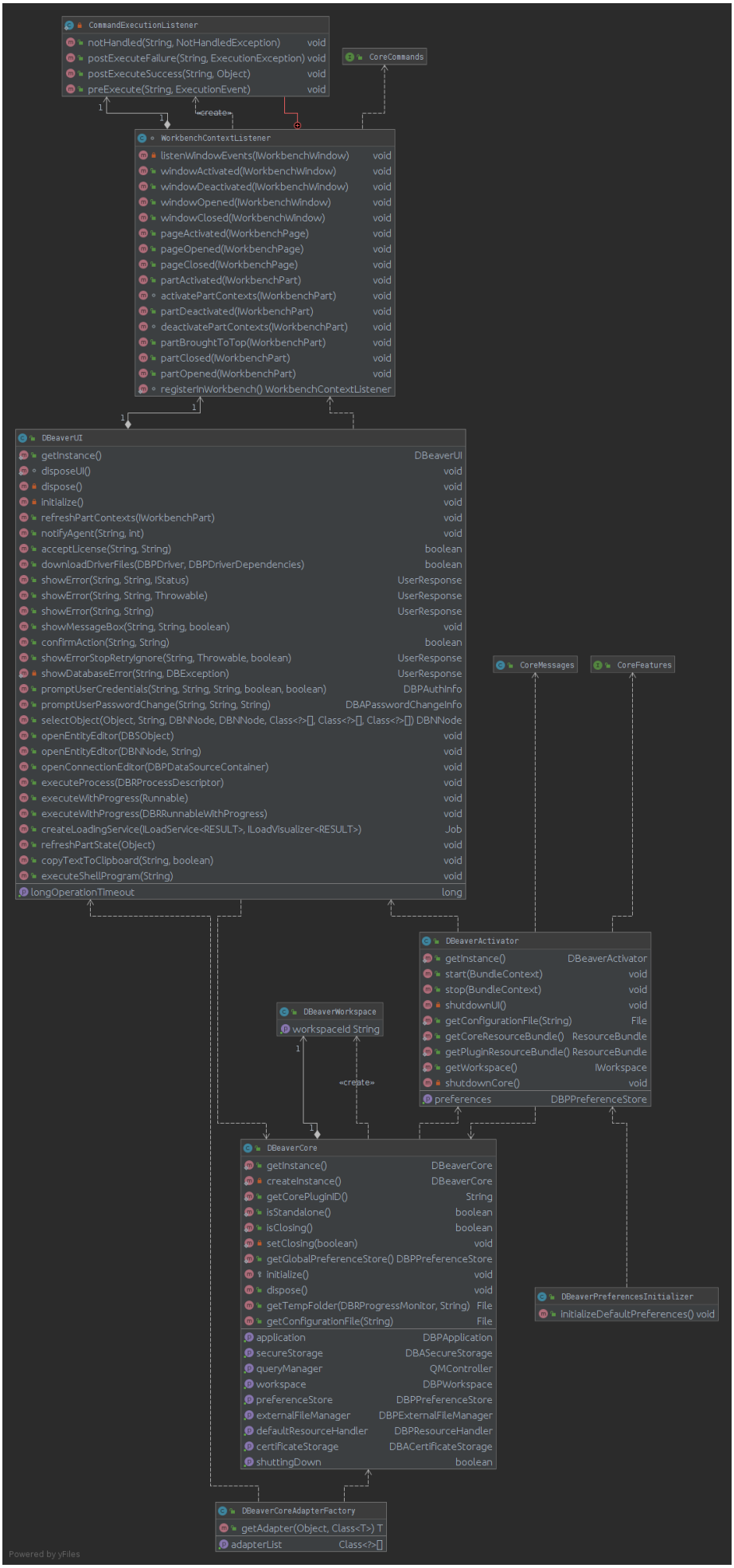
- Az tábla létrehozását követően menedzselni lehet annak sorait/rekordjait (adatait)
  - A meglévő rekordot megtekinthetőek (olvashatóak)
  - Új rekordot lehet létrehozni
  - Meglévő rekordot lehet módosítani
  - Meglévő rekordot lehet törölni
  - Az olvasás (lekérdezés), létrehozás, módosítás, törlés műveletek SQL szkriptel 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.


CommandExecutionListener
notHandled(String, NotHandledException) void
postExecuteFailure(String, ExecutionException) void
postExecuteSuccess(String, Object) void
preExecute(String, ExecutionEvent) void
CoreCommands
«create»
WorkbenchContextListener
listenWindowEvents(IWorkbenchWindow) void
windowActivated(IWorkbenchWindow) void
windowDeactivated(IWorkbenchWindow) void
windowOpened(IWorkbenchWindow) void
windowClosed(IWorkbenchWindow) void
pageActivated(IWorkbenchPage) void
pageOpened(IWorkbenchPage) void
pageClosed(IWorkbenchPage) void
partActivated(IWorkbenchPart) void
activatePartContexts(IWorkbenchPart) void
partDeactivated(IWorkbenchPart) void
deactivatePartContexts(IWorkbenchPart) void
partBroughtToTop(IWorkbenchPart) void
partClosed(IWorkbenchPart) void
partOpened(IWorkbenchPart) void
registerInWorkbench() WorkbenchContextListener
DBeaverUI
getInstance() DBeaverUI
disposeUI() void
dispose() void
initialize() void
refreshPartContexts(IWorkbenchPart) void
notifyAgent(String, int) void
acceptLicense(String, String) boolean
downloadDriverFiles(DBPDriver, DBPDriverDependencies) boolean
showError(String, String, IStatus) UserResponse
showError(String, String, Throwable) UserResponse
showError(String, String) UserResponse
showMessageBox(String, String, boolean) void
confirmAction(String, String) boolean
showErrorStopRetryIgnore(String, Throwable, boolean) UserResponse
showDatabaseError(String, DBException) UserResponse
promptUserCredentials(String, String, String, boolean, boolean) DBPAuthInfo
promptUserPasswordChange(String, String, String) DBAPasswordChangeInfo
selectObject(Object, String, DBNNode, DBNNode, Class<?>[], Class<?>[], Class<?>[]) DBNNode
openEntityEditor(DBSObject) void
openEntityEditor(DBNNode, String) void
openConnectionEditor(DBPDataSourceContainer) void
executeProcess(DBRProcessDescriptor) void
executeWithProgress(Runnable) void
executeWithProgress(DBRRunnableWithProgress) void
createLoadingService(ILoadService<RESULT>, ILoadVisualizer<RESULT>) Job
refreshPartState(Object) void
copyTextToClipboard(String, boolean) void
executeShellProgram(String) void
longOperationTimeout long
CoreMessages
CoreFeatures
DBeaverActivator
getInstance() DBeaverActivator
start(BundleContext) void
stop(BundleContext) void
shutdownUI() void
getConfigurationFile(String) File
getCoreResourceBundle() ResourceBundle
getPluginResourceBundle() ResourceBundle
getWorkspace() IWorkspace
shutdownCore() void
preferences DBPPreferenceStore
DBeaverWorkspace
workspaceId String
«create»
DBeaverCore
getInstance() DBeaverCore
createInstance() DBeaverCore
getCorePluginID() String
isStandalone() boolean
isClosing() boolean
setClosing(boolean) void
getGlobalPreferenceStore() DBPPreferenceStore
initialize() void
dispose() void
getTempFolder(DBRProgressMonitor, String) File
getConfigurationFile(String) File
application DBPApplication
secureStorage DBASecureStorage
queryManager QMController
workspace DBPWorkspace
preferenceStore DBPPreferenceStore
externalFileManager DBPExternalFileManager
defaultResourceHandler DBPResourceHandler
certificateStorage DBACertificateStorage
shuttingDown boolean
DBeaverPreferencesInitializer
initializeDefaultPreferences() void
DBeaverCoreAdapterFactory
getAdapter(Object, Class<T>) T
adapterList Class<?>[]
Powered by yFiles

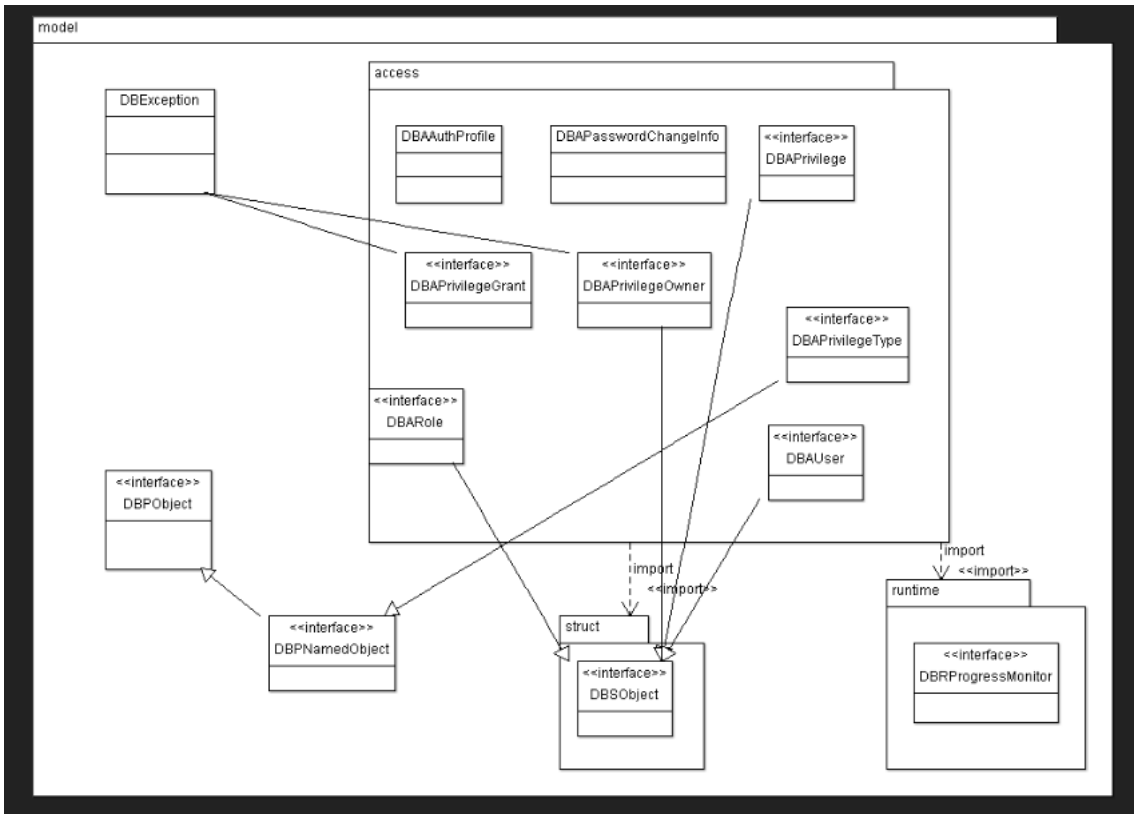
Package diagram - Halász Gábor

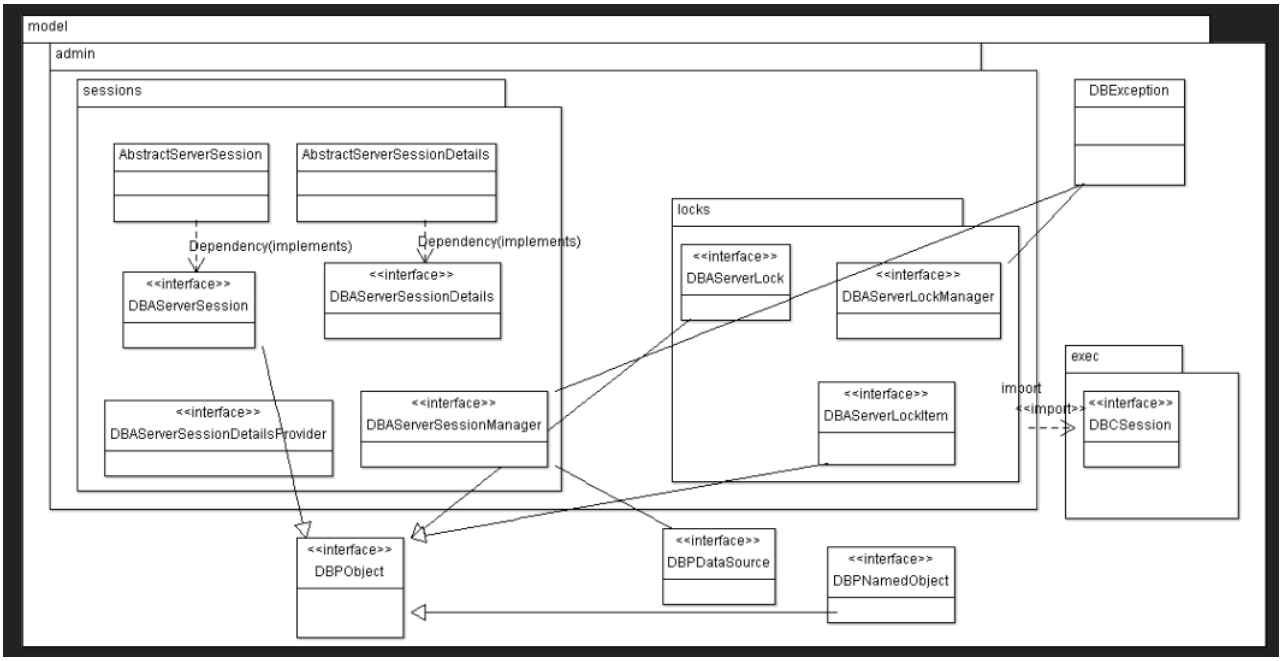
A projekt 7 fő csomagból áll:

Forráskód szerkezete (fő package-ek):

- docs – Többnyire elavult dokumentációt tartalmaz.
- features – Funkciók (feature) leírása. Nem tartalmaz forráskódot. A termékben található plugin-ok és függőségek strukturálására.
- bundles – Nagyon alap plugin-okat tartalmaz.
- plugins – Fő forráskód itt helyezkedik 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)





Access:

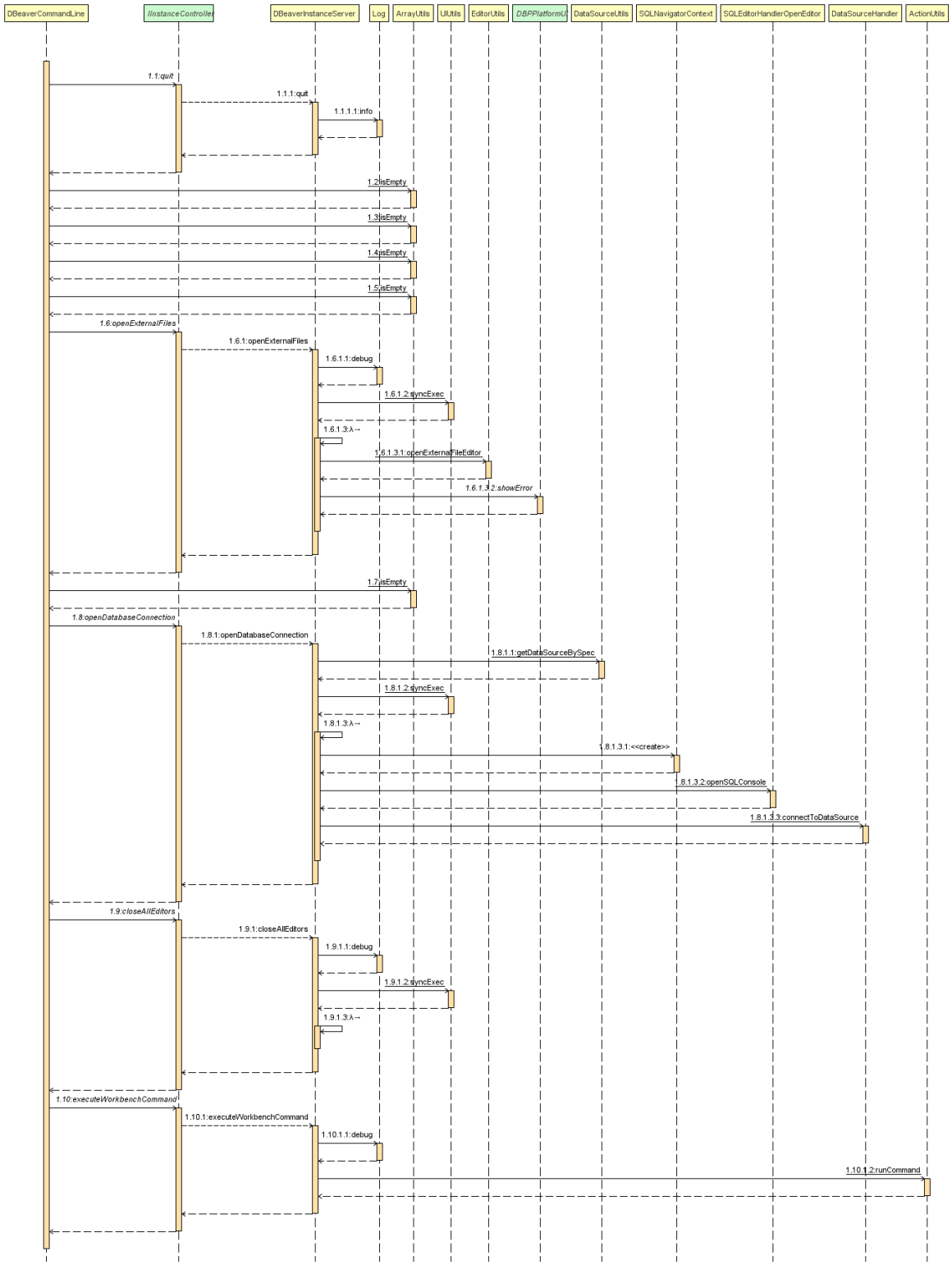
- DBAuthProfil – Profil authentikáció
- DBAPasswordChangeInfo -Jelszó megváltoztatási információk (régi, új jelszó)
- DBAPrivilege – Adatbázis előnyök (SELECT, CREATE, DROP, CONNECT etc.)
- DBAPrivilegeGrant
- DBAPrivilegeOwner – Előny tulajdonos
- DBAPrivilegeType – Típus
- DBARole – Szerep
- DBAUser – Felhasználó

Admin:

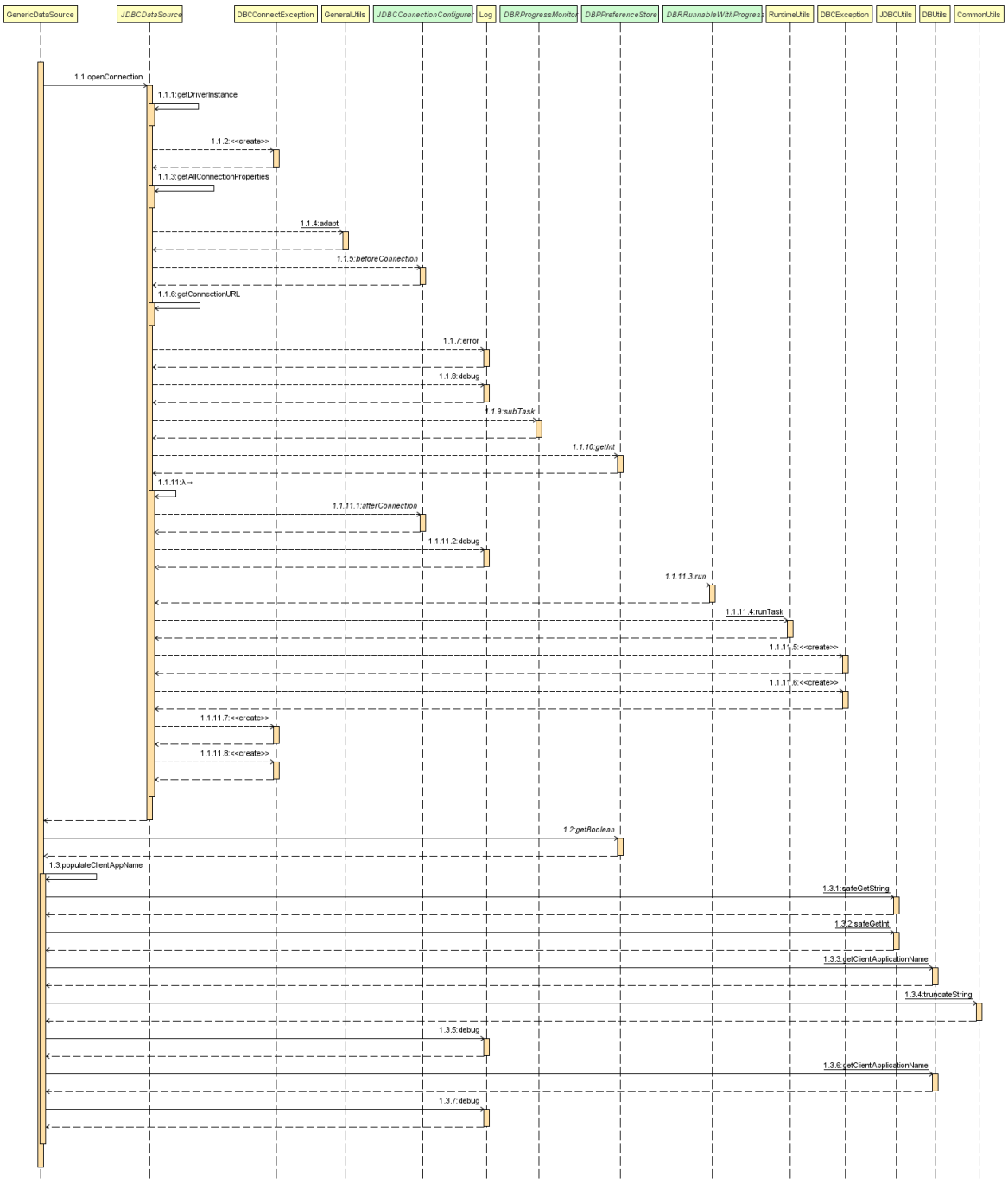
- Session:
  - AbstractServerSession – Absztrakt implementációja a szerver sessionnek.
  - AbstractServerSessionDetails – A szervert további részletekkel látja el
  - DBAServerSession – Szerver session
  - DBAServerSessionDetails - A szervert további részletekkel látja el
  - DBAServerSessionDetailsProvider – Session manager
  - DBAServerSessionManager - Session manager
- Locks:
  - DBAServerLock – Server lock interfész
  - DBAServerLockItem – Server lock részlet elemek
  - DBAServerLockManager – Server lock manager

Sequence diagram - Kuba Bence

Az executeCommandLineCommands metódus diagramja megmutatja hogy milyen úton hajtódik végre egy általunk a konzolban kiadott parancs. Lényegében az instance controller megkapja a parancsot az tovább jutatja az adott instance szerverére és ott végrehajtódik a parancs. Ezt követően logol.



Az openConnection metódus diagramján jól létszanak egy kapcsolat megnyitásához szükséges lépések. Elöször kell egy adatforrás majd egy driver és csak ezekután éri el a tényleges adatbázist.



Activity diagram - Horváth Marcell

A DBeaver megnyitása után kiválaszthatjuk a megfelelő adatbázist, amivel dolgozni szeretnénk (Oracle, MySQL, Mariadb, ODBC). Létrehozást követően új kapcsolatokat lehet menedzselni. Meglévő adat törlése, módosítása, lekérdezése, vagy újat is létrehozhatunk, amit SQL szkript írásával is megtehetünk. Ezek után a DBeaver letárolja a megadott adatokat (a kiválasztott adatbázisba), amivel később dolgozni szeretnénk