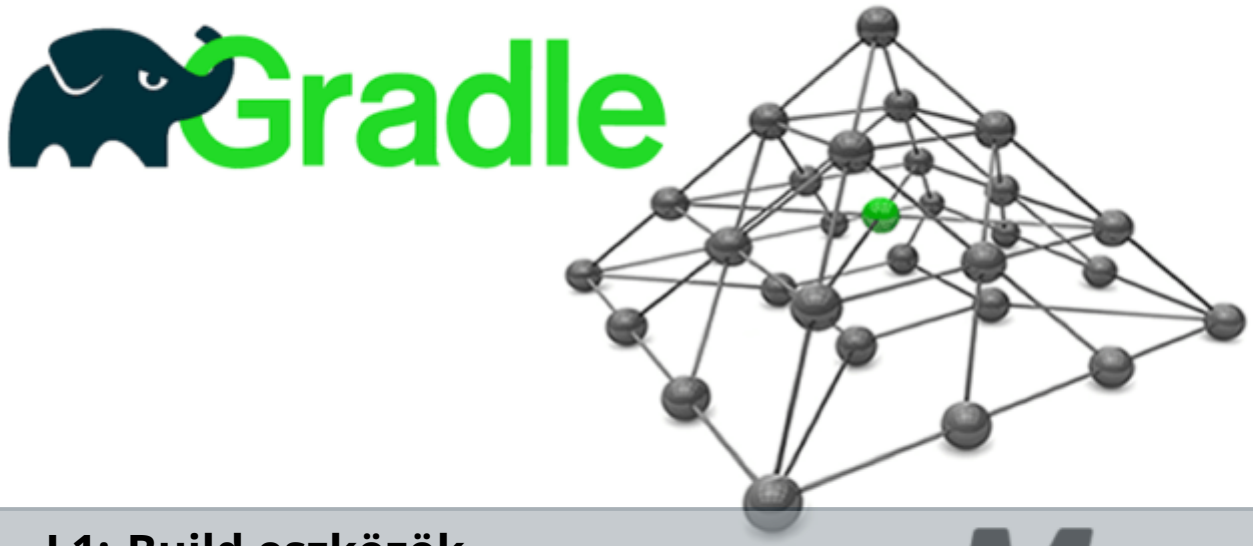

L1: Build eszközök

Maven™

- Az első és mádosik részben leírtak mindenkire vonatkoznak. A harmadik részben a használt adattípusok teszik egyedivé a feladatokat. A harmadik részből minden diák a sorszámának (**Törzskönyv szám utolsó számjegye + 1**) megfelelő specifikációnak kell eleget tegyen.
- A laborfeladat elfogadásának feltétele, hogy a `check` taszk **ne jelezzen hibákat** (ld. második rész).

Első rész: Multi-modul Gradle alkalmazás

Készítsünk egy **multimodul Gradle alkalmazást** a következő specifikáció alapján:

- Az alkalmazás 2 modulból áll, amely a git tárolónkon 2 almappában él. Hozzuk létre ezekhez a megfelelő `build.gradle` és `settings.gradle` deszkriptor-állományokat.
- Az első modul egy szerveralkalmazást jelképez, amely *adatelérési* műveleteket hivatott elvégezni. Az adatokat jelen laborfeladat esetén tároljuk *memóriában*. A laborfeladat harmadik részében találhatóak a személyre szabott adattípusok. **Készítsünk *interfészeket* a CRUD műveletek absztraktizálásához.**
- A második modul jelképez egy kliensalkalmazást, amely *megjeleníti* a szervertől letöltött adatokat (függőség szükséges). Hozzunk létre (tetszőleges technológiával) egy grafikus felületet, amely táblázat formájában megjeleníti az adatokat. A felület tartalmazzon egy gombot, melynek eseménykezelője inicializáljon egy „kapcsolatot” a szerveroldalhoz, és töltse be az adatokat a felületen található táblázatba. A backend és frontend **nem kell külön folyamatban fusson**, s **nem szükséges hálózaton keresztül kommunikáljanak**.
- A projektek Maven-stílusú 3-részes ID-ja legyen a következő:
  group: `edu.bbte.idde.<diákid>`, *name*: `<diákid>-backend`/`<diákid>-desktop`, *version*: `1.0-SNAPSHOT`
- Mindkét modullal tudjuk elvégezni konzolból egy-egy Gradle parancs segítségével a **kompilálást** és a **feltöltést** a lokális Maven tárolóba. Emellett a kliens modul esetén tudjuk **futtatni** is az alkalmazást, ugyancsak egy paranccsal, Gradle segítségével (ld. `maven` és `application` pluginok).
- Mindkét modul alkalmazzon megfelelő szintű **naplózás**t. Ehhez használjunk egy `slf4j` API-val kompatibilis implementációt. Konfiguráljuk a naplózást úgy, hogy különböző *szintet* és *formátumot* alkalmazva naplózzunk *konzolra* és *állományba* egyaránt.

Második rész: Statikus kódelemzés

- Tanulmányozzuk a lehetséges statikus kódelemző eszközöket a `01-build/gradle-staticcodeanalysis` példaprojektben, majd másoljuk projektünk gyökerébe a következő állományokat:
  - `lint.gradle`
  - `checkstyle.xml`
  - `pmd-ruleset.xml`
  - `spotbugs-filter.xml`
- Aktiváljuk a `lint.gradle`-t minden Java forráskódot tartalmazó projektünkben: `apply from: 'lint.gradle'` (relatív elérésért `apply from: "$rootDir/lint.gradle"`).
- Próbáljuk ki a statikus kódellenőrző eszközök futtatását a `gradle check` taszkkal. Javítsuk a potenciálisan fellépő hibákat.
- Opcionálisan telepíthetünk IntelliJ pluginokat az ellenőrző eszközöknek:
  - `Settings` → `Plugins` → `Browse repositories`
  - Keresendő ingyenes plugin-nevek: *„Checkstyle-IDEA”*, *„FindBugs-IDEA”*, *„PMDPlugin”*
  - Minden telepített pluginnak adjuk meg a megfelelő átmásolt konfigurációs állományt (`Settings` → `Other Settings`), hogy egyezzenek a beállítások a konzolfuttatással.

Harmadik rész: Adatmodell

Használjuk az alábbi adattípust az előző részekben leírt feladat megoldásához. Az adattípust tartalmazzon legalább 5 különböző releváns adattagot. Minden diák csak a sorszámának megfelelő adattípust kell implementálja.

1. Használt autókereskedés platformhoz tartozó **használt autó hirdetés**.
2. Online számítógépes hardverüzlethez tartozó **alkatrész**.
3. Ingatlanokat forgalmazó platformhoz tartozó **ingatlan hirdetés**.
4. Catering céghez tartozó **menü**.
5. Túrázóknak szánt platformhoz tartozó **túra**.
6. Használt autókereskedés platformhoz tartozó **használt autó hirdetés**.
7. Online számítógépes hardverüzlethez tartozó **alkatrész**.
8. Ingatlanokat forgalmazó platformhoz tartozó **ingatlan hirdetés**.
9. Catering céghez tartozó **menü**.
10. Túrázóknak szánt platformhoz tartozó **túra**.

Feltöltés

- A feladatot töltsük fel a saját git tárolónkra, egy **dedikált ágra**. Az ág neve tartalmazza a laborfeladat sorszámát. További információk a `ubb-idde-lab0-setup.pdf` állományban.
- Hozzunk létre egy **merge request**et, amely tartalmazza a laborfeladat számát, állítsuk a merge requestet a jelen feladathoz tartozó **csoportos milestone-ra**, majd linkjét **adjuk le Canvasen**. Végső leadási időpontnak tekintjük az utolsó commit, az utolsó push, és a Canvasre való linkfeltöltés közül a **legkésőbbi** mozzanatot.