

## Heroku használata SZTE SED Rendszerfejlesztés-I kurzus keretein belül a webalkalmazás kitelepítéséhez

### Spring Boot alkalmazás

A Spring Boot egy nyílt forráskódú Java alkalmazás keretrendszer, amellyel web alkalmazásokat is készíthetünk. Ebben a példában egy kutya nyilvántartó rendszert mutatunk be, amely képes az adatbázisba adatot menteni, kiolvasni, módosítani és törölni.

A kiinduláshoz az alkalmazást itt találhatjuk:

<http://gitlab-okt.sed.hu/markusa/rf1-pelda>

A fejlesztéshez Debian alapú OS-t ajánlunk, de természetesen más OS is használható.

Ezen kívül szükséges még:

Apache Maven 3.3+ (*sudo apt install maven*)

Snap (*sudo apt install snapd*)

Heroku (*sudo snap install --classic heroku*)

JDK 1.8+

Eclipse Spring Tool 4-el:

<http://marketplace.eclipse.org/content/spring-tools-4-spring-boot-aka-spring-tool-suite-4/>

A kutya nyilvántartó alkalmazás forrás fájl- és mappa struktúrája:

```
|— pom.xml
|— doc
|   |— heroku.pdf
|— src
|   |— main
|   |   |— java
|   |   |   |— application
|   |   |   |   |— Application.java
|   |   |   |   |— controller
|   |   |   |   |   |— DogController.java
|   |   |   |   |— model
|   |   |   |   |   |— Dog.java
|   |   |   |   |— service
|   |   |   |   |   |— DogService.java
|   |— resources
|   |— application.properties
|   |— data.sql
|   |— schema.sql
|   |— templ
|       |— index.html
|       |— update-dog.html
```

Az Apache Maven egy projektmenedzsment eszköz, amellyel projektek build-elést tudjuk automatizálni. Egy nagyobb szoftver projekt témérdek programcsomagot tartalmazhat, melyeket összefoglaló néven függőségeknek (dependency) szoktunk nevezni. A maven tehát képes a külső könyvtárakat (pl. SQL, Bootstrap, stb.) kezelni, hogy a projektben használhassuk.

Az alapja a POM.xml, amelyben ezek a függőségek definiálva vannak.

A Spring Boot alkalmazás belépési pontja az *Application.java* osztályban van. Hierarchikusan ennek kell a legmagasabb szinten lenni. A *DogController.java* fájl tartalmazza a REST végpontokat, azaz azokat az útvonalakat ahol a GET, POST, stb. műveletek meg vannak valósítva. A *Dog.java* lényegében az adatbázistábláinkról készített model, mindig egy ilyen modelbe konvertáljuk a lekérdezések eredményét. A *DogService.java* tartalmazza a tényleges SQL utasításokat, amelyeket a Controller osztály metódusai hívják meg.

A lényeges része még a Spring Boot alkalmazásnak a megjelenítés, adatbázis táblák és az adatok létrehozása is. A *resource* mappában találhatóak az ezért felelős fájlok. Az *application.properties* tartalmazza az adatbázis azonosítását (elérhetőség, felhasználónév, jelszó).

A *data.sql* illetve *schema.sql* fájlokat a Spring Boot alkalmazás automatikusan érzékeli és végrehajtja, ha sikeres az adatbázishoz való kapcsolódás.

Végezetül a *templ* mappa tartalmazza a HTML fájlokat az adatok megjelenítéséhez.

Ahhoz, hogy fejleszteni tudjunk, a legegyszerűbb, ha megnyitjuk Eclipse a programot a következő módon, majd futtatjuk az *Application.java* fájlt:

File -> Import -> Maven -> Existing Maven Projects , majd adjuk meg a "Browse" gombra kattintva a pom.xml elérési útvonalát és nyomjuk meg a "Finish" gombot.

Amennyiben minden lépésünk sikeres, úgy a <http://localhost:8080/> -on fog futni a programunk.

### Mi is az a Heroku?

Lényegében egy platform szolgáltatás (PaaS) webalkalmazások hosztolásához, ahol virtuális gépek bérletére van lehetőségünk, amelyeken előre van telepítve a web alkalmazás futtatásához szükséges erőforrások (pl. webszerver). Ezenkívül biztosít számos ingyenes szolgáltatást is, többek között PostgreSQL relációs adatbázis szerveret, illetve egy "hobby" virtuális gépet, amely mellé saját aldomain jár, pl. valami.herokuapp.com.

### Mi is ez a dokumentum és miért van erre szükség?

A Heroku több programozási nyelvet támogat, többek között Node, Java, PHP. Mivel a kurzus során UML megértése és diagramok készítése a tárgy teljesítésének feltétele, így csak objektumorientált programozási nyelv választható a kötelező program elkészítéséhez (különben nem lenne szükségünk pl. Osztály diagramra).

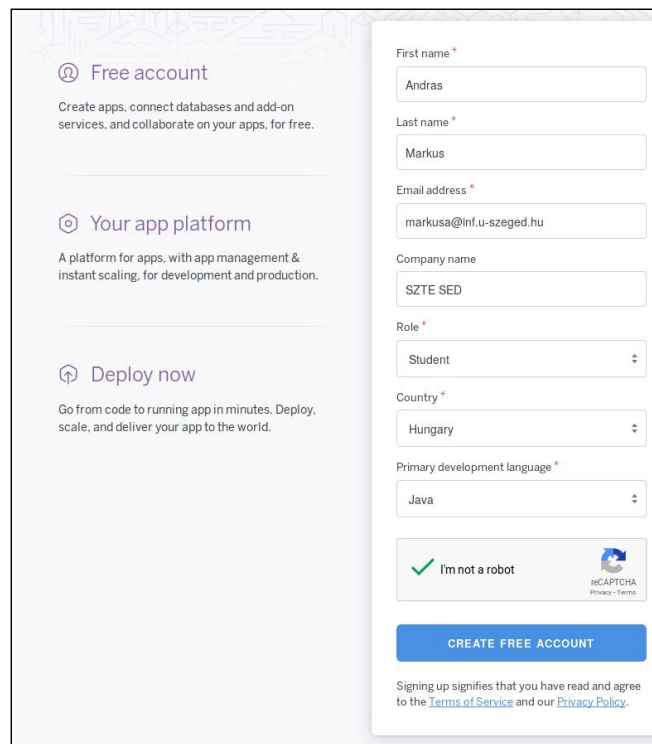
Ebben a dokumentumban tehát 2 minta alkalmazást hozunk létre Heroku telepítéssel és GitLab használattal. Ezek használata nem kötelező a félév során, ha pl. van saját szervered

és domain-ed, azonban a 4. mérföldkő során már élesben kell bemutatni a program aktuális állapotát.

## Első lépések

Mivel főlegesen minden csapattagnak Heroku-s hozzáféréssel rendelkeznie, így elegendő, ha a csapat 1 tagja (pl. a projektmenedzser) regisztrál csak.

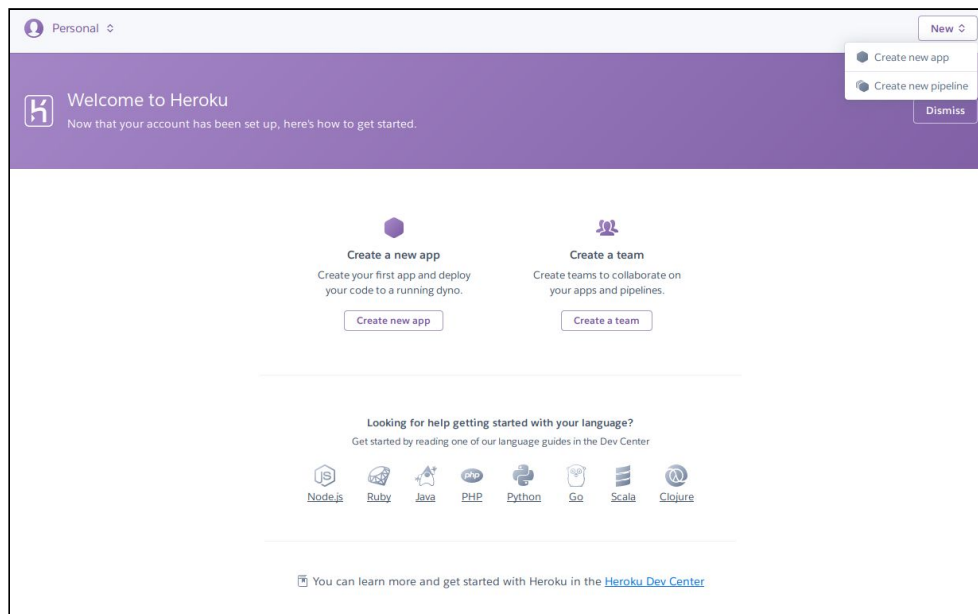
Nyissuk meg a következő oldalt: <https://www.heroku.com/>, és regisztráljunk a jobb felső “Sign Up” gombra kattintva. Töltsük ki az űrlapot (lásd 1. ábra), majd a megadott címre érkező e-mail segítségével állítsuk be jelszavunkat és jelentkezünk is be.



The image shows the Heroku sign-up page. On the left, there are three sections: 'Free account' (Create apps, connect databases and add-on services, and collaborate on your apps, for free.), 'Your app platform' (A platform for apps, with app management & instant scaling, for development and production.), and 'Deploy now' (Go from code to running app in minutes. Deploy, scale, and deliver your app to the world.). On the right, there is a registration form with the following fields: 'First name' (Andras), 'Last name' (Markus), 'Email address' (markusa@inf.u-szeged.hu), 'Company name' (SZTE SED), 'Role' (Student), 'Country' (Hungary), and 'Primary development language' (Java). Below these fields is a reCAPTCHA checkbox labeled 'I'm not a robot' and a 'CREATE FREE ACCOUNT' button. At the bottom, there is a note: 'Signing up signifies that you have read and agree to the [Terms of Service](#) and our [Privacy Policy](#).'

1. ábra: Heroku login

Ha minden jól ment a 2. ábrán látható felülethez jutunk, kattintsunk a “Create a new app” gombra.



2. ábra: Új alkalmazás létrehozása

Adjunk meg egy egyedi azonosítót, válasszuk ki Európát és kattintsunk a “Create app” gombra (lásd 3. ábra).

3. ábra: Alkalmazás létrehozása

Ahhoz, hogy a saját web alkalmazásunkat ki tudjuk deploy-olni Heroku-ba, telepíteni kell a Heroku CLI-t, amely segítségével terminálból tudjuk menedzselni a telepítést.

A Heroku CLI telepítést itt találod: <https://devcenter.heroku.com/articles/heroku-cli>

A telepítés befejeztével ne felejtjük el ellenőrizni annak sikerességét, illetve bejelentkezni (azonosítani a regisztrált felhasználót):

`heroku --version`

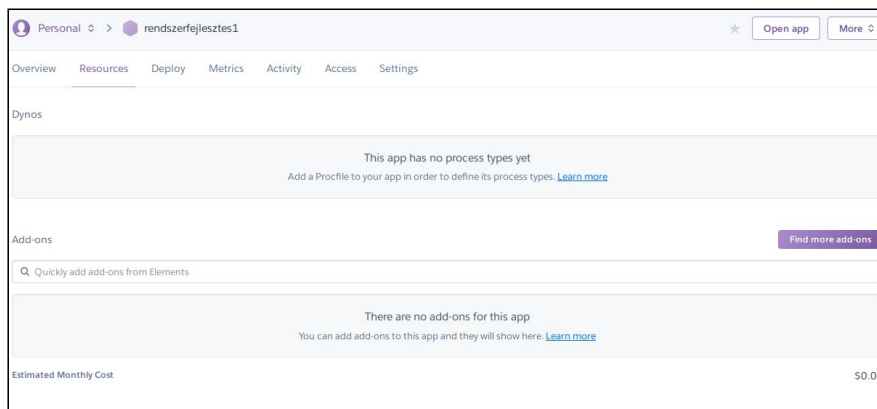
`heroku login -i`

## PostgreSQL

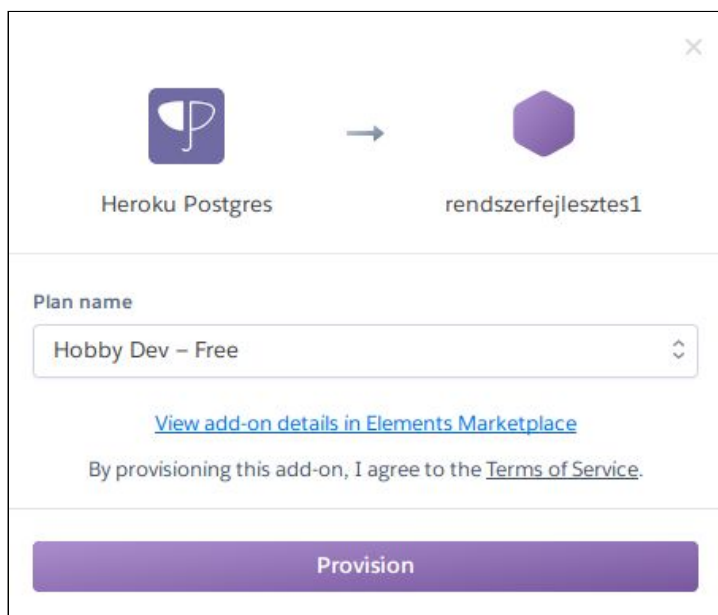
Lépünk fel a következő oldalra:

<https://dashboard.heroku.com/apps>

Majd válasszuk ki a létrehozott projektünket, és navigáljunk át a “Resources” menüre (lásd 4. ábra). Az “Add-ons” kereső mezőjébe kezdjük el begépelni, hogy “Heroku Postgres” és kattintsunk is rá, ekkor az 5. ábrán látható felülethez jutunk. kattintsunk a “Provision” gombra.



4. ábra: Heroku Add-ons felülete



5. ábra: Postgres hozzákapcsolása a projekthez

Ha minden jól ment, az “Add-ons”-oknál meg is jelent az adatbázisunk, kattintsunk is rá, navigáljunk át az újonnan megnyíló oldalon a “Settings” fülre és kattintsunk a “View Credentials” gombra, ekkor hozzáférünk az adatbázis kapcsolódáshoz szükséges információkra.

## GitLab

Menjünk fel a következő oldalra:

<http://gitlab-okt.sed.hu/markusa/rf1-pelda/>

Klónozzuk le a minta Spring Boot alkalmazást és az azt tartalmazó git Repository-ban adjuk ki a következő utasítást (értelemszerűen a saját projekt nevével helyettesítve):

```
heroku git:remote -a rendszerfejlesztes1
```

Ekkor a GitLab mellett a Heroku-n lévő távoli tárolót is hozzáadtuk projektünkhöz.

Mivel a feladat, hogy a csapat által használt GitLab repository-ba dolgozzatok, így a jelenlegi remote repository-t lecseréljük a csapat által használt tárolóra:

```
git remote set-url origin URL
```

Ha helyesen dolgoztunk, akkor a `git remote -v` utasítás hatására valami hasonlót kell látnunk:

```
heroku https://git.heroku.com/rendszerfejlesztes1.git (fetch)
```

```
heroku https://git.heroku.com/rendszerfejlesztes1.git (push)
```

```
origin http://gitlab-okt.sed.hu/markusa/rf1-pelda.git (fetch)
```

```
origin http://gitlab-okt.sed.hu/markusa/rf1-pelda.git (push)
```

A deploy-hoz a következő utasításra van szükségünk:

```
git push heroku
```

Ha hiba nélkül fordult a programunk, a következő üzenetet kapjuk (ellenkező esetben magát a hibaüzenetet):

```
remote:      [INFO] -----
remote:      [INFO] BUILD SUCCESS
remote:      [INFO] -----
remote:      [INFO] Total time: 18.602 s
remote:      [INFO] Finished at: 2019-09-04T13:04:51+00:00
remote:      [INFO] Final Memory: 42M/330M
remote:      [INFO] -----
remote: -----> Discovering process types
remote:      Procfile declares types      -> (none)
remote:      Default types for buildpack -> web
remote:
remote: -----> Compressing...
remote:      Done: 81.5M
remote: -----> Launching...
remote:      Released v8
remote:      https://rendszerfejlesztes1.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/rendszerfejlesztes1.git
```

\* *[new branch]*      *master -> master*

Azaz a BUILD sikeres, a honlapunk elérhető a következő URL-en:

<https://rendszerfejlesztes1.herokuapp.com/>

Pár hasznos tanács:

- Heroku nyilvántartja a deploy kísérleteket, ezáltal ha egy korábbi állapotot szeretnénk visszaállítani aktuálisnak, akkor a következő utasítást adjuk ki:  
*heroku rollback vX*  
ahol X a fentebb látható release száma
- Abban az esetben a módosítottunk valamit a programon, akkor ne felejtsük el, hogy a lokális repository-hoz 2 másik remote is tartozik, tehát a GitLab-ra *git push origin*, míg a Heroku-ra a *git push heroku* utasításokkal tudjuk a változásokat érvényesíteni (a *git add* illetve a *git commit -m “..”* parancsok után)
- A Heroku az ingyenes szolgáltatását leállítja inaktivitás esetén. Előfordulhat, hogy akár 30-40 mp is szükséges ahhoz, hogy az alkalmazást újraindítsa, a lassú kapcsolódás nem (feltétlen) hibás működés.
- Mivel egyszerre 2 távoli Repository-t is menedzselünk (*origin* és *heroku*) ezért előfordulhat, hogy más-más verziónál járnak. Ekkor érdemes egy új *branch*-et létrehozni a Heroku *master branch*-ének kezelésére.

Utoljára módosítva: 2019.09.09.

Gyakorlatvezetők