# Workflow GIT

## Programe necesare:

- Scene builder 8 ( ! IMPORTANT ) - http://gluonhq.com/products/scene-builder/thanks/?dl=/download/scene-builder-windows-x64/ - nu merge cu java 9, deoarece proiectul e facut in java 8
- Java JDK 1.8.0_162 - http://www.oracle.com/technetwork/java/javase/downloads/java-archive-javase8-2177648.html?printOnly=1

## Fonturi necesare:

(Pana reusesc sa aflu cum se pun embeded in aplicatie fara sa trebuiasca descarcate)

- https://www.dropbox.com/s/fs3b1vlmmu402r1/Fonts.rar?dl=0 - sunt mai multe fonturi, nu se vor folosi toate, doar ca nu stiu inca pe care le vom folosi si ca sa nu va pun sa le descarcati de mai multe ori. Pentru instalare, se dezarhiveaza, se selecteaza toate -> click dreapta -> install

### Daca nu aveti descarcat proiectul atunci rulati comanda

```
git clone https://github.com/nesarares/ISS-Donare-Sange.git
```

### Configurare GIT:

```
git config --global user.name "numele"
git config --global user.email emailul
```

## Workflow - pasii ce trebuie respectati cand se lucreaza la applicatie:

1. **Se aduc din repository toate modificarile (potentiale)** - In functie de branchiul pe care lucrati (sau unde ati lucrat ultima oara si vreti sa continuati). Se ruleaza comenzile urmatoare:

```
git checkout master
git fetch
git pull
```

Daca vreti sa faceti pull pe un anumit branch care stiti ca s-a modificat puteti sa faceti asa:

```
git pull origin nume-branch
```

2. **Se creeaza un branch nou pe care vreti sa lucrati** - Ca si conventie numele branchurilor va fi de forma: modul-nume-branch (ex: *client-ui-login*, *server-login*, *services-interfaces*, etc.)

```
git checkout -b nume-branch
```

In caz ca uitati sa faceti branch inainte sa modificati orice fisier se poate face si inainte sa faceti commit, dar **e important sa faceti commit doar pe branch separat** pentru a evita problemele de versionare.

3. **Se lucreaza la functionalitati in mod normal**
4. **Se adauga toate fisierele din proiect in "working directory" - adica modul git-ului de a retine fisierele ce vor fi adaugate in commit**

```
git add .
```

5. **Pentru a publica modificarile se face un commit cu un mesaj cu modificarile efectuate**

```
git commit -m "Mesaj despre modificari"
```

Exemplu de mesaj de commit: "Added GUI for login component and registration form", "Fixed bug relating two same users logging in at the same time".

6. **IMPORTANT - Treceti pe branchul parinte (de obicei master) pentru a face *merge***

```
git checkout master
```

7. **Se repeta pasul 1 pentru a aduce eventualele modificari**

```
git fetch
git pull origin master
```

8. **Se face merge cu branchul pe care ati lucrat**

```
git merge nume-branch
```

9. **Se rezolva conflictele daca exista intre diferitele versiuni de cod (in caz ca s-a lucrat pe aceleasi fisiere si git-ul nu a reusit sa faca merge singur)** - Pentru a rezolva conflictele se foloseste IDE-ul - IntelliJ (daca exista - se va vedea in git bash la numele branchului ceva de genul "master|MERGING"). Puteti cauta pe google "Resolving conflicts in intellij idea" - https://www.jetbrains.com/help/idea/resolving-conflicts.html (exemplu de tutorial)
10. **IMPORTANT - verificati codul sa functioneze cum trebuie dupa ce se rezolva conflictele si inainte de a face push**

11. **Daca totul functioneaza ok se pun in repository modificarile** - Daca faceti push la alt branch se inlocuieste "master" cu numele branchului respectiv

```
git push origin master
```

> Daca vreti puteti sa cautati si alte workflow-uri pe net, doar ca mie asa mi s-a parut cel mai usor. Toate chestiile astea se pot face in mod normal si din editor, doar ca eu nu stiu cum, dar daca vi se pare greu puteti sa cautati pe net. Site util de informare