



Saptamana 1

Partea 2

Programare Front-End

4. Work coordination across members/teams - Git

Cum lucram impreuna la acelasi proiect ?

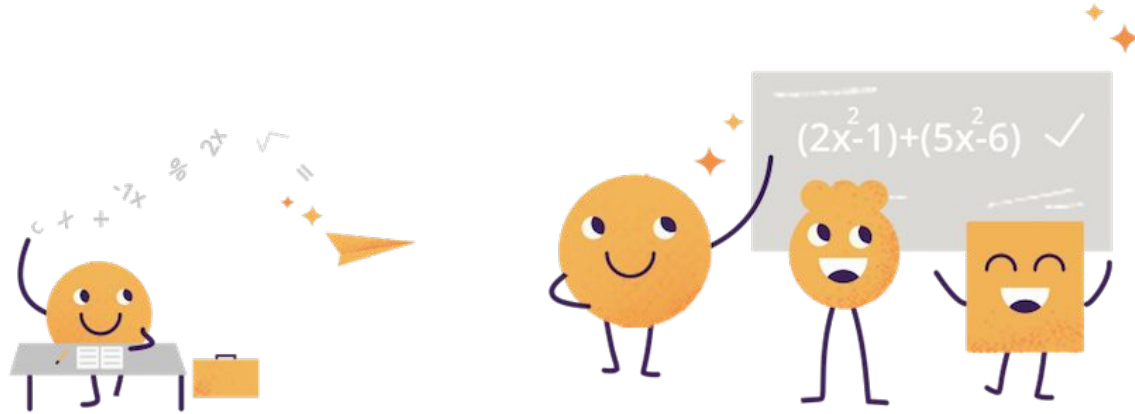
Cum ne sincronizam codul ?

Cum evitam “conflictele” ?

Git

- cel mai popular **version control system**
- dezvoltat de catre Linus Torvalds, celebrul creator al *Linux kernel*
- este un sistem de versionare distribuit, ceea ce inseamna ca fiecare membru al unui *repository* care tine intreg codul corespunzator unei aplicatii, poate avea o copie locala a acestuia, complet functionala; acest lucru permite lucrul la distanta sau offline: schimbarile asupra codului se fac local pentru ca ulterior sa fie sincronizate cu cele din repository-ul principal, de pe server
- un alt tip de sisteme de versionare sunt cele centralizate, care presupun ca mai intai clientii sa se sincronizeze cu server-ul inainte sa faca schimbari

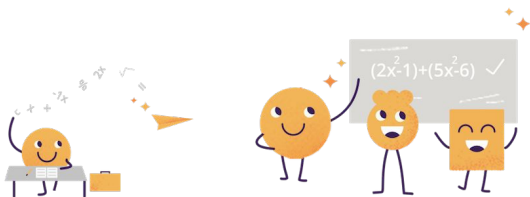
PRACTICE: Git Basics



PRACTICE: Git Basics – Part 1

Cerinte: (veti lucra in grupe de cate doi)

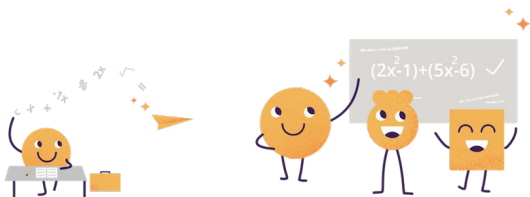
1. Accesati <https://github.com/> si creati-va un cont personal
2. Descarcati GIT de la adresa <https://git-scm.com/> si instalati utilitarul
3. Descarcati GitExtensions de la adresa <https://github.com/gitextensions/gitextensions> si instalati utilitarul
4. Logati-va pe github.com folosind contul anterior creat si creati un nou **repository (proiect)** cu denumirea **"Practica Front-End Wantsome"**
5. Clonati acest proiect folosind utilitarul GitExtensions



PRACTICE: Git Basics – Part 2

Cerinte: (veti lucra in grupe de cate doi)

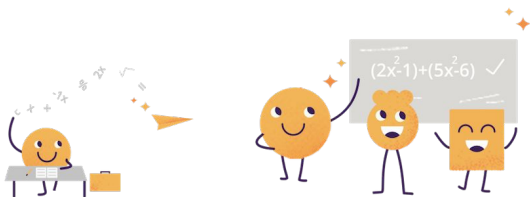
1. Creati un nou folder cu denumirea **"Saptamana 1 - Practica Git Basics"** in cadrul proiectului clonat
2. Creati un nou fisier in cadrul directorului anterior creat ce denumirea **"NumePrenume.txt"**
3. Editati fisierul anterior creat si adaugati in el textul **"Practica pe bune la Wantsome. Azi invatam Git!"**
4. Folositi GitExtensions pentru a crea un **commit local** folosind fisierul anterior modificat - mesajul commit-ului trebuie sa descrie in ansamblu modificarile si scopul acestuia. **Actualizati** branch-ul **remote** cu modificarile produse local.



PRACTICE: Git Basics – Part 3

Cerinte: (veti lucra in grupe de cate doi)

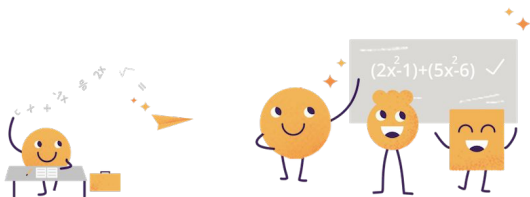
1. Creati un nou branch denumit **"1234_my-first-feature-branch"** pornind din branch-ul master
2. Creati un nou fisier denumit **"new-feature-file-1.txt"** si adaugati ca si continut definitia Git luata de pe Google
3. Creati un nou folder denumit **"public"** si adaugati un nou fisier in acesta denumit **"new-feature-file-2.txt"**. Adaugati ca si continut definitia unui **branch** si a unui **commit** luate de pe Google
4. Creati un nou **commit local** pe noul branch creat care sa contina fisierele anterior create. **Actualizati** branch-ul **remote** cu modificarile produse local.
5. Faceti **merge** noului branch **cu** branch-ul principal **master**. Mai intai local, apoi actualizati remote.



PRACTICE: EXTRA

Cerinte: (veti lucra in grupe de cate doi)

1. Creati **doua branch-uri** separate in cadrul carora sa modificati acelasi fisier si dati **commit pe fiecare branch**.
2. Dati **checkout pe unul** din cele doua branch-uri si faceti o noua modificare a fisierului anterior implicat.
3. Realizati ca voiati sa faceti modificarile pe celalalt branch. Ca sa nu pierdeti modificarile, folositi **Stash** pentru a le "duce" pe celalalt branch. Dati commit local si apoi actualizati branch-ul remote.



HOMEWORK: **Play with Git**

In case of fire



1. `git commit`



2. `git push`



3. leave building