2 Git (continuare) și Documentul de specificare a cerințelor

I. Git

1. Recapitulăm comenzile de git învățate în laboratorul precedent

- Setări de configurare inițială pentru comanda git folosită în CLI

`git config --global user.name "numele de utilizator de git"`	Configurare adresă de email
`git config --global user.email "adresa email cont git"`	Configurare nume de utilizator
`git config --global core.editor "nume_editor"`	Configurare editor de text pentru editare - `notepad` în Windows `sau` - `vim` în sisteme bazate pe Linux - `code` pentru VSCode: `"code –wait"`
`git config --global init.defaultBranch main`	Configurare git astfel încât la crearea unui nou repo branch-ul creat să fie numit main
`git config --global core.autocrlf true`	Configurare git astfel încât caracterul de sfârșit de linie (EOL – End of line) să fie „CRLF"
`git config --global -e`	Deschidere fișier de configurări pentru editare
`git config --global --list`	Listare configurări salvate

- Setări de gestionare repo

`git init`	Creare repo local. În directorul curent (care devine pentru git *working directory*) se creează directorul .git, care va gestiona tot ce ține de git

`git add`	Adăugare conținut din *working directory* în *staging area*
	Se poate folosit cu un fișier, mai multe fișiere sau cu directoare, mai multe directoare dau tot conținutul *working directory*
	Ex: `git add fișier1`
	sau
	`git add fișier2 fișier3`
	sau
	`git add .`

`git status`	Verificare ce este modificat în working directory și ce se află în staging area Are și forma scurtă: `git status -s`
`git commit`	Sincronizare conținut din staging area (sau direct din working directory) în snapshot Se folosește opțiunea `-m` pentru a specifica un mesaj informativ cu privire la conținutul modifcărilor. Dacă opțiunea lipsește, se deschide editorul de text pentru a se introduce textul explicativ. Se pot introduce două mesaje: un

	mesaj scurt de max. 50 de caractere, urmat pe alt rând de un mesaj mai lung dacă se consideră util. `git commit -a` se folosește pentru a face commit pentru toate modificările (din fișierele tracked – cel puțin o dată adăugate cu comanda `git add nume_fișier` sau `git add .`) indiferent dacă modificările sunt în *staging area* sau nu.
`git remote add origin`	Stabilire repo remote care va fi folosit cu numele de origin `git remote add origin "URL_repo_remote"`
`git push`	Ex: `git push origin main` Copiere conținut din main (branch-ul curent) în origin
`git pull`	Ex: `git pull origin main` Copiere conținut din origin în main

- Comenzi de ajutor:

`git help`	Listare comenzi de bază
`git help -a`	Listarea tuturor comenzilor (se navighează cu săgețile pentru a derula în sus/ jos câte o comandă sau cu tasta *space* pentru a derula câte o pagină)
`git help -g`	Afișează o listă a ghidurilor disponibile
`git help tutorial`	Deschide ghidul numit *tutorial* ***Notă: o idee bună de parcurs, împreună cu tutorial-2, workflows și alte pagini de manual***
`git help nume_comandă`	Informații despre utilizare *nume_comandă*

2. Exerciții pentru familiarizarea cu conceptul de branch-uri

branch (engl.) = ramură (ro.)

main (engl.) = principal (ro.)

a. Comenzi de bază

`git branch`	Listare nume branch-uri existente. Branch-ul curent este marcat cu *
`git branch -a`	Listare branch-uri remote
`git branch nume_branch`	Creare branch *nume_branch*
`git branch -m nume_nou`	Redenumire branch
`git switch nume_branch`	Mutare pe branch-ul *nume_branch* Există și opțiunea --create care creează și apoi se mută pe branch-ul precizat: `git switch --create nume_branch`
`git merge nume_branch`	Unire conținut branch main/ branch curent cu branch *nume_branch*
`git branch -d nume_branch`	Ștergere branch *nume_branch* Se șterge *nume_branch* doar dacă modificările din el sunt deja în main/ branc-ul curent
`git branch -D nume_branch`	Ștergere branch *nume_branch* Se șterge *nume_branch* indiferent dacă modificările se regăsesc în branch-ul curent
`git log --graph`	Arată graficul commit-urilor și merge-urilor efectuate Opțiunea `--abbrev-commit` scurtează numărul asignat fiecărui commit

b. Exerciții

1. Creați un folder numit Lab2_Asipsi.
2. Creați un nou folder creat retetaLimonada2 în care recreați fișierele ingrediente.txt și instructiuni.txt.
3. Creați un repo în folder-ul retetaLimonada2 cu branch-ul principal cu denumirea main (`git init -b main`)
4. Listați branch-urile existente:
   `git branch`.
   Observație: branch-ul se creează abia după primul commit.
5. Faceți un commit pentru cele două fișiere (`git add *` și `git commit -m „primul commit”`) și listați din nou branch-urile existente.
6. Creați un nou branch cu denumirea citronada (`git branch citronada`).
7. Listați branch-urile existente în acest repo și verificați care este branch-ul curent – este marcat cu caracterul `‚*’ : git branch`
8. Schimbați branch-ul curent în branch-ul citronada: `git switch citronada`
9. Modificați fișierul ingrediente.txt adăugând „o portocală” și fișierul instructiuni.txt adăugând textul „se stoarce portocala”.
10. Realizați un commit cu modificările realizate în branch-ul curent – citronada: `git add *` și `git commit -m „am adăugat citronada”`
11. Uniți acest branch cu branch-ul main:
    ```
    git switch main
    git merge citronada
    ```
12. Verificați din nou branch-urile existente și branch-ul curent.
13. Ștergeți branch-ul citronadă.
14. Vizualizați graficul branch-urilor.
15. Creați un nou branch numit `exotic`. Faceți acest branch curent
    ```
    git branch exotic
    git switch exotic
    ```
    sau
    `git switch –create exotic`
16. Modificați fișierul inigrediente adăugând `„suc ananas”` și adăugați un fișier numit `„despre.txt”`. Faceți `commit` pentru modificările acestea (branch-ul curent este `exotic`).
17. Vizualizați conținutul fișierelor nou create.
18. Schimbați branch-ul în `main`. Listați conținutul directorului `retetaLimonada2` și vizualizați conținutul fișierului `ingrediente.txt`.
19. Deschideți o fereastră de GUI pentru a vizualiza conținutul directorului `retetaLimonada2`.
20. Uniți branch-ul `main` cu branch-ul `exotic`. Listați conținutul directorului `retetaLimonada2` și vizualizați conținutul fișierului `ingrediente.txt`. Deschideți o fereastră de GUI pentru a vizualiza conținutul directorului `retetaLimonada2`.

II. Documentul de specificare a cerințelor

Creați pe GitHub un nou repo cu numelele ASIPSI-nume proiect licenta.

Folosind documentul template de specificare a cerințelor, creați un document word pe care sa il completati pe baza temei de licenta și urcati-l pe GitHub. Documentul template este disponibil pe Teams.