Proiect de Programare / C++

-= Rețea socializare jobs =-

Studenți: Câdu Daria Maria Arinton Raluca Ioana

I. Enunț

Studentul 1 se ocupă de anunțurile de joburi:

- adăugare, ștergere și modificare anunțuri de job
- vizualizare candidaților care au aplicat

Studentul 2 se ocupă de interacționarea cu anunțurile:

- poate vizualiza lista cu joburi disponibile
- poate filtra joburile în funcție de un skill necesar
- poate aplica la un job

II. Structura datelor folosite de echipă

Vom folosi următoarele clase pentru modelarea datelor:

- **Job**: conține `string titlu`, `string companie`, `vector<string> skilluri_necesare` și `int numar_aplicanti`
- **Candidat**: conține `string nume`, `vector<string> skilluri` și `string email`
- **Aplicare**: conține `string nume_aplicant`, `string email_aplicant`, `vector<string> skilluri` și jobul la care aplică

Structura fișierelor utilizate de aplicație

- `joburi.txt` – conține lista de joburi disponibile:
  - Prima linie: numărul total de joburi.
  - Pentru fiecare job:

    ```
    <titlu_job> <nume_companie> <nr_skilluri>
    <skill_1> <skill_2> ... <skill_n>
    ```

- `aplicari.txt` – conține lista aplicațiilor înregistrate:
  - Prima linie: numărul total de aplicări.
  - Pentru fiecare aplicare:

    ```
    <nume_aplicant> <email_aplicant> <index_job> <nr_skilluri>
    <skill_1> <skill_2> ... <skill_n>
    ```

- `Job.h / Job.cpp` – definește clasa `Job`, care conține:
  - Atribute: `titlu`, `companie`, `vector<string> skilluri`

  - Metode: constructori, `necesitaSkill()`, `operator«`
- `Aplicant.h` – definește structura `Aplicant` și `Aplicare`:
  - `Aplicant`: `nume`, `email`, `vector<string> skilluri`
  - `Aplicare`: `Aplicant aplicant`, `int jobIndex`
- `utils.h / utils.cpp` – funcții pentru salvare și încărcare:
  - `salveazaJoburi()`, `incarcaJoburi()`
  - `salveazaAplicari()`, `incarcaAplicari()`
  - `validIndex()` – verifică dacă un index e valid
- `template_utils.h` – definește o funcție template:
  - `afiseazaVector()` – afișează conținutul oricărui vector de obiecte cu operator « suprascris
- `main.cpp` – punctul de intrare în aplicație:
  - Interpretează comenzile din linia de comandă.
  - Apelează funcții specifice pentru fiecare acțiune: adăugare job, aplicare, filtrare, afișare etc.
  - La final, salvează toate modificările în fișierele `joburi.txt` și `aplicari.txt`.

III. Interacțiunea cu executabilele

Aplicația 1 - Angajator

- Adaugă un job nou cu titlul, compania și lista de skilluri necesare:
  `./proiect_jobs --adauga-job "TitluJob" "Companie" "Skill1" "Skill2" ...`

- Șterge jobul cu titlul specificat
  `./proiect_jobs --stergere-joburi "TitluJob"`

- Afișează toate joburile disponibile:
  `./proiect_jobs --afiseaza-joburi`

- Afișează lista candidaților care au aplicat la jobul specificat:
  `./proiect_jobs --aplicanti-job "IndexJob"`

Aplicația 2 - Utilizator

- Afișează toate joburile disponibile:
  `./proiect_jobs --afiseaza-joburi`

- Afișează joburile care cer skill-ul specificat:

```
./proiect_jobs --filtreaza "Skill"
```

- Aplică la jobul selectat cu datele candidatului și lista de skilluri:
  `./proiect_jobs --aplica "Nume" "Email" "IndexJob" "Skill1" "Skill2" ...`

IV. Salvarea datelor

Datele despre joburi și aplicări sunt salvate local în fișierele:

- `joburi.txt` – conține toate joburile
- `aplicari.txt` – conține toate aplicările
- `Comenzi.txt` – conține toate comenzile/interactiunea cu executabilele

Datele sunt încărcate la pornirea programului și salvate după fiecare modificare efectuată.