PROIECT - INGINERIA SISTEMELOR SOFTWARE

AN UNIVERSITAR 2022-2023

SEMESTRUL 2

APLICATIE
SISTEM DE PONTAJ & MANANGEMENT DE TASK-URI

Nume si Prenume Student

LEONTE CEZAR-COSMIN

GRP. 831P

Matematica-Informatica

Coordonator:

Prof. Sima Ioan

PREZENTAREA CERINTEI

O firma si-a creat o infrastructura prin care seful monitorizeaza angajatii prezenti la lucru si le traseaza sarcini individuale. Firma are o aplicatie care ofera:

• o fereastra pentru sef, cu ajutorul careia seful vede lista angajatilor prezenti în firma, un element din lista precizând numele angajatului si ora la care s-a logat în sistem. De asemenea, seful poate transmite o sarcina unui angajat prezent astfel: selecteaza angajatul din lista, introduce o descriere a sarcinii si declanseaza un buton "transmite sarcina". Imediat dupa transmiterea unei sarcini, aceasta poate fi consultata de catre angajatul respectiv.

• câte o fereastra pentru fiecare angajat: Atunci când angajatul vine la serviciu, introduce ora sosirii si declanseaza un buton "prezent". Imediat dupa declansarea butonului, seful vede în lista lui ca angajatul este prezent. În continuare, cât timp angajatul sta la serviciu, el primeste si, în consecinta, vede în fereastra lui, sarcinile transmise de sef. La plecare, angajatul închide fereastra, moment în care seful este notificat de delogarea acestuia din sistem.

FUNCTIONALITATILE

Angajat:

- Poate sa se logheze in aplicatie pe baza datelor de logare
- Poate sa vizualizeze toate task-urile active asignate acestuia
- Poate sa muta un task intre urmatoarele statusuri: READY TO TAKE <-> IN PROGRESS <-> IN REVIEW -> DONE
- Poate sa se ponteze IN / OUT in system

Sef:

- Poate sa se logheze in aplicatie pe baza datelor de logare
- Poate sa vizualizeze toti angajatii activi la momentul actual
- Poate crea un task si sa-l asigneze la un angajat active / inactive

Etapele proiectarii

Analiza cerintelor:

- Am indentificat cerintele impuse, functionalitatiile (un user se logheaza in aplicatie: daca este angajat normal poate sa vizualizeze si sa actualizeze task-uri asignate de sef; daca este sef poate sa vada cine a venit la munca si sa asigneze task-uri)

Analiza:

- *Cerinte functionale:* Aplicatia trebuie sa ii permita utilizatorului sa interactioneze cu ferestre dintr-un executabil, in asa fel incat sa permita toate actiunile necesare: creare task, vizualizare task, vizualizare angajati activi, pontare in/out
- *Cerinte nefunctionale:* Aplicatia trebuie sa fie scrisa in Python. Aplicatia trebuie sa fie usor de utilizat iar codul usor de scalat.

Proiectarea sistemului:

- Am folosit StarUML pentru a dezvolta diagrame pentru a ne usura munca. Am dezvoltat diagrama deutilizare, diagrama de stare, diagrama de activitate, diagrama de comunicare/colaborare, diagrama de arhitectura, diagrama bazei de date, diagrama de clase.

Object design: # Implementare Curenta

- Clasele de Employee & Boss NU vor avea acces la baza de date, ele vor folosii un Controller
- Clasa Controller va putea doar sa PRIMEASCA date din baza de date
- Clasa TaskGenerator (TaskEngine) va putea doar sa INSEREZE si sa ACTUALIZEZE date in baza de date

Object design: # Implementare CORECTA (lucru realizat prea tarziu asa ca il fac teoretic doar)

- Clasele de Employee & Boss NU vor avea acces la baza de date, ele vor folosii un Controller

- Clasa Controller va putea VALIDEZE datele primite de la Engine-uri si sa transmita mesaje catre UI
- Clasa EntityEngine este o clasa abstracta care permite operatii CRUD pentru toate entitatile in discutie: Employee, Boss, Task, Clocker
- Subclasele copii din EntityEngine (EmployeeEngine, TaskEngine etc.) o sa contina doar operatiile specifice claselor respective, care nu se pot generalizare:
    - EmployeeEngine: generare user activi/inactivi
    - TaskEngine: generare task-uri asignate per user specific
    - ClockEngine: clock IN/OUT din system

Implementarea:

- Crearea modelelor de UI
- Crearea bazei de date
- Crearea modelelor conceptuale SQLAlchemy dupa baza de date
- Implementarea claselor cu acces la baza de date
- Implementarea evenimentelor din UI
- Validarea datelor la nivel de UI # Pentru urmatoarea versiune: muta la nivel de Controller
- Legarea componentelor intre ele

Testarea:

- *Testare manuala:* testarea manuala a tuturor functionalitatilor
- *Testare unitara:* crearea testelor unitare si rularea lor # Not Done but mentioned
- *Testare automata:* deschidere ENDPOINTS catre servicii de extragere, transformare si incarcare de date (ETL) & testarea cu date in masa # Not Done but mentioned

Technologii Folosite

FrontEnd (GUI):

- PyQT
- QT-5
- QT-Designer

Am folosit aceste technologii deoarece ofera o metoda simpla de a crea un UI si a face legatura cu back-end. Technologii ideale pentru aplicatii cu scop local, care se pot folosii fara acces la internet, doar cu o conexiune LAN.

BackEnd:

- Python
- SQLAlchemy
- PyQT5

Am folosit aceste technologii deoarece:

  - Python: cod usor de scris, usor de inteles, usor de scalat
  - SQLAlchemy: framework care permite cu usurinta conectivitatea la o baza de date si poate sa fie scalat pentru date imense
  - PyQT5: pentru a face legatura cu front-end

Baza de date:

- PostgreSQL

O baza de date stabila, care ofera viteza decenta si scalabilitate pentru scopul aplicatiei.

Mediu de dezvoltare:

- VS Code

Lightweight, simple, and efficient.

Proiectarea diagramelor:

- StarUML

Am folosit StarUML la proiectarea diagramelor pentru usurinta si multitudinea de avantaje ale programului.

Source Control:

- **GitHUB**

Am folosit github pentru technologie de SC. Aceasta aplicatie nu necesita gestionari speciale asa ca am mers cu varianta cea mai simpla.

Comunicare Client-Server:

Structura proiectului este organizata pe niveluri: FrontEnd (GUI) - Client, BackEnd, Controller, Engines, Server

FrontEnd-Client: contine formularele de acces pentru users precum si datele de la Server

BackEnd: contine validarea de date din formularele de FE

Controller: gestioneaza datele primite dintre BE si Engines

Engines: creaza query-uri pe care le trimite cu SQLAlchemy la server

Server: contine DB + executarea query-urilor

Testarea codului:

Pentru testare s-a folosit PyTest. # Not implemented but mentioned

Scenariu de utilizare

Employee:

- Deschide aplicatia: insereaza datele de logare, apasa pe butonul de logare. Daca datele sunt goale sau eronate, o fereastra de eroare este aparuta. Se apasa pe butonul “OK !” pentru a inchide fereastra. Dupa inserarea datelor corecte si a apasarii de logare, fereastra de logare se inchide si se deschide UI-ul de Employee
- Datele angajatului sunt afisate in partea de sus a ferestrei
- Task-urile asignate current (numele lor) angajatului sunt afisate in partea stanga a ferestrei
- Pe partea de mijloc, angajatul poate sa scrie numele unui task si sa apese pe butonul de “Get Task Details” pentru a i se afisa detaliile legate de acel task. Daca se scrie un task gresit la nume, o sa se afiseze o fereastra de eroare unde utilizatorul trebuie sa apese pe butonul “OK !” pentru a o inchide.
- In partea de jos sunt 2 butoane pentru a gestiona statusul unui task. Se scrie numele task-ului ca si la pasul precedent si se folosesc butoanele pentru a muta task-ul la un status anterior sau precedent.
- Daca un task ajunge la statusul “DONE”, acesta nu mai poate sa fie intors, fiind marcat ca si finalizat

Boss:

- Deschide aplicatia: insereaza datele de logare, apasa pe butonul de logare. Daca datele sunt goale sau eronate, o fereastra de eroare este aparuta. Se apasa pe butonul “OK !” pentru a inchide fereastra. Dupa inserarea datelor corecte si a apasarii de logare, fereastra de logare se inchide si se deschide UI-ul de Boss
- Datele angajatului sunt afisate in partea de sus a ferestrei
- Angajatii current activi sunt afisati in partea stanga a ferestrei
- Pe partea de mijloc, seful poate sa insereze date pentru un task nou si sa il asigneze la un angajat dupa nume. Daca vreun camp este lasat gol sau

angajatul nu exista, o fereastra de eroare o sa apara cu mesajele aferente. Seful trebuie sa apese pe butonul “OK !” pentru a inchide fereastra

Diagrama cazurilor de utilizare:

Diagrama cazurilor de utilizare Employee:

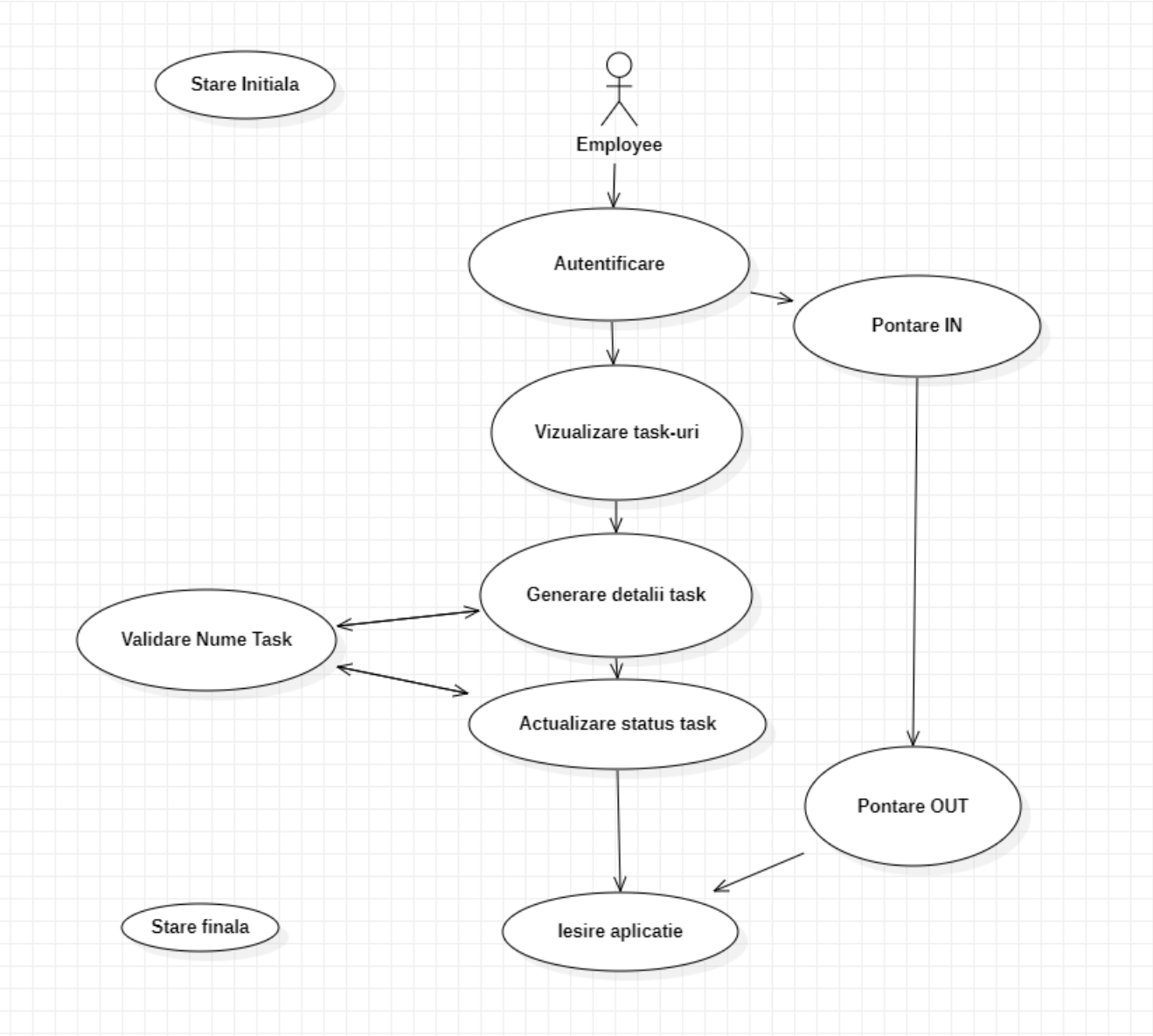


Diagrama cazurilor de utilizare Boss:

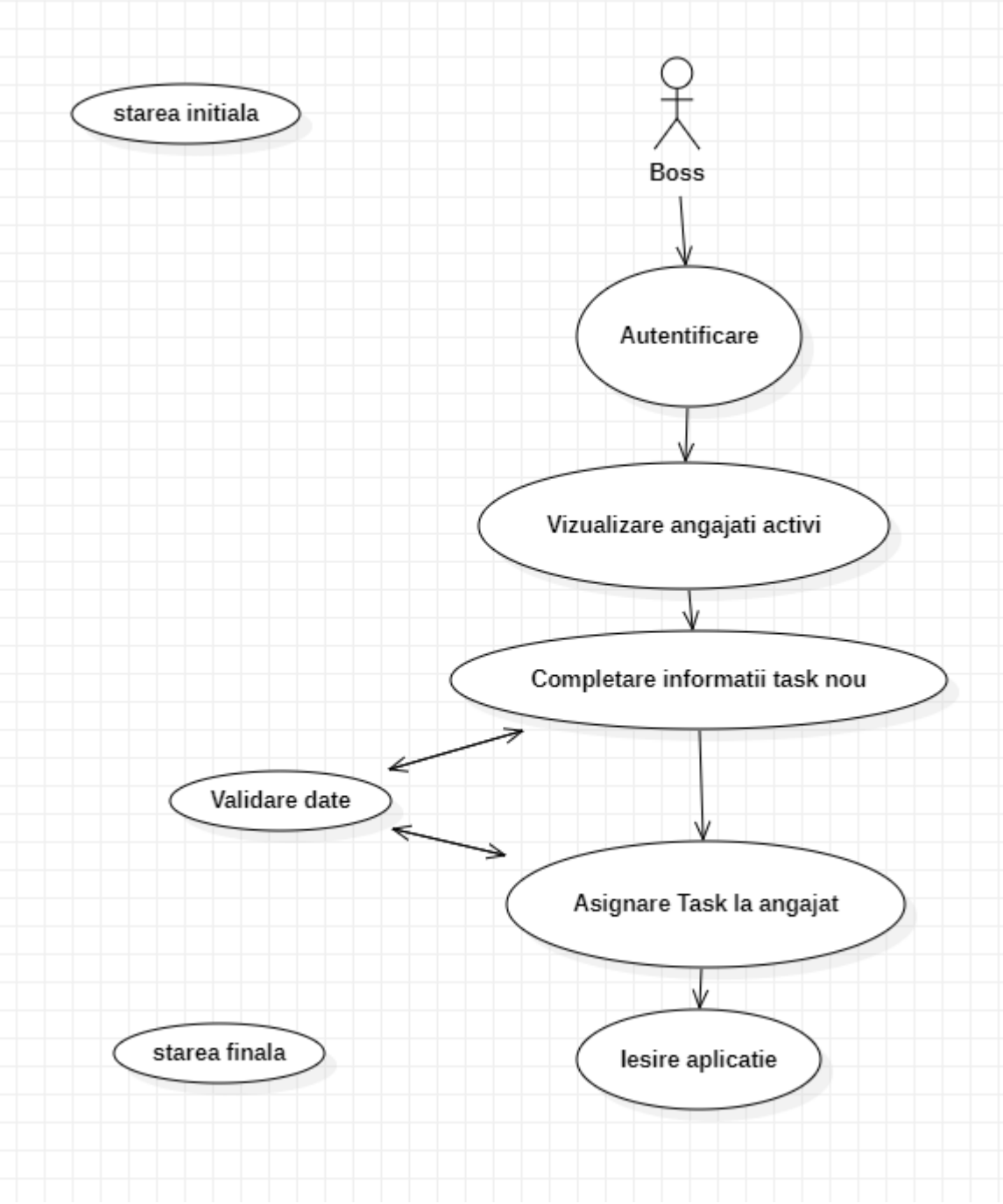


Diagrama workflow:

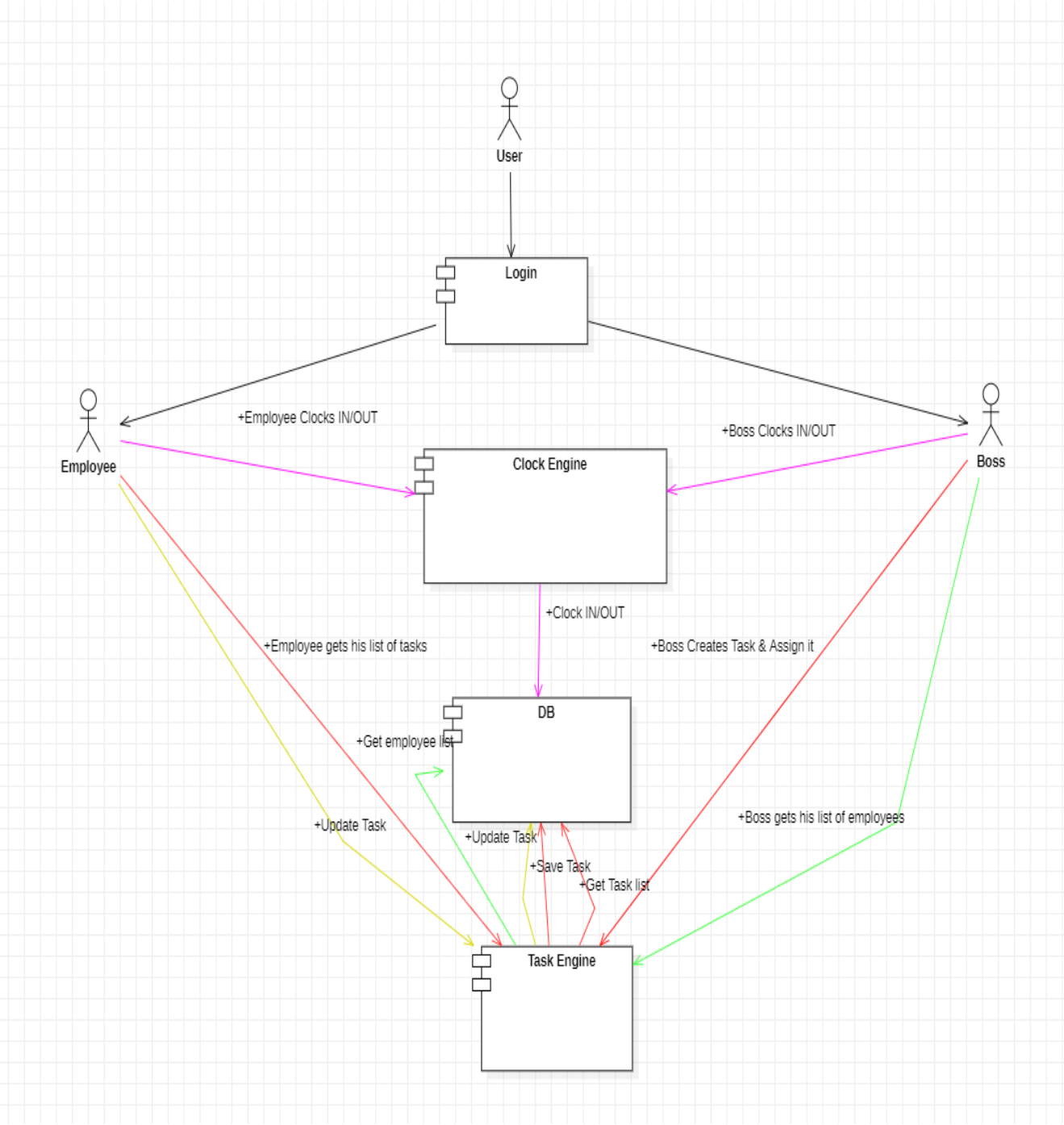


Diagrama de clase:

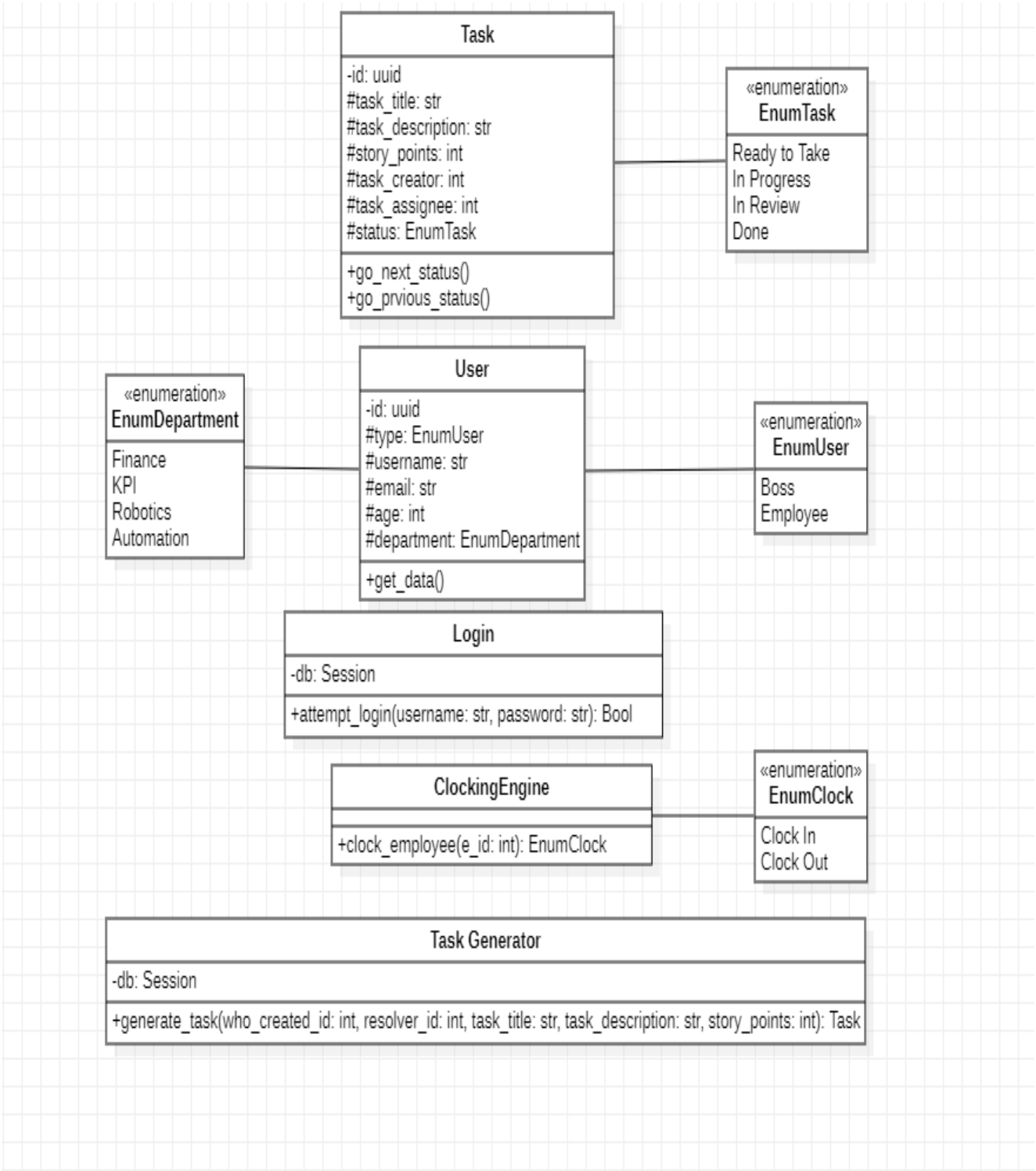
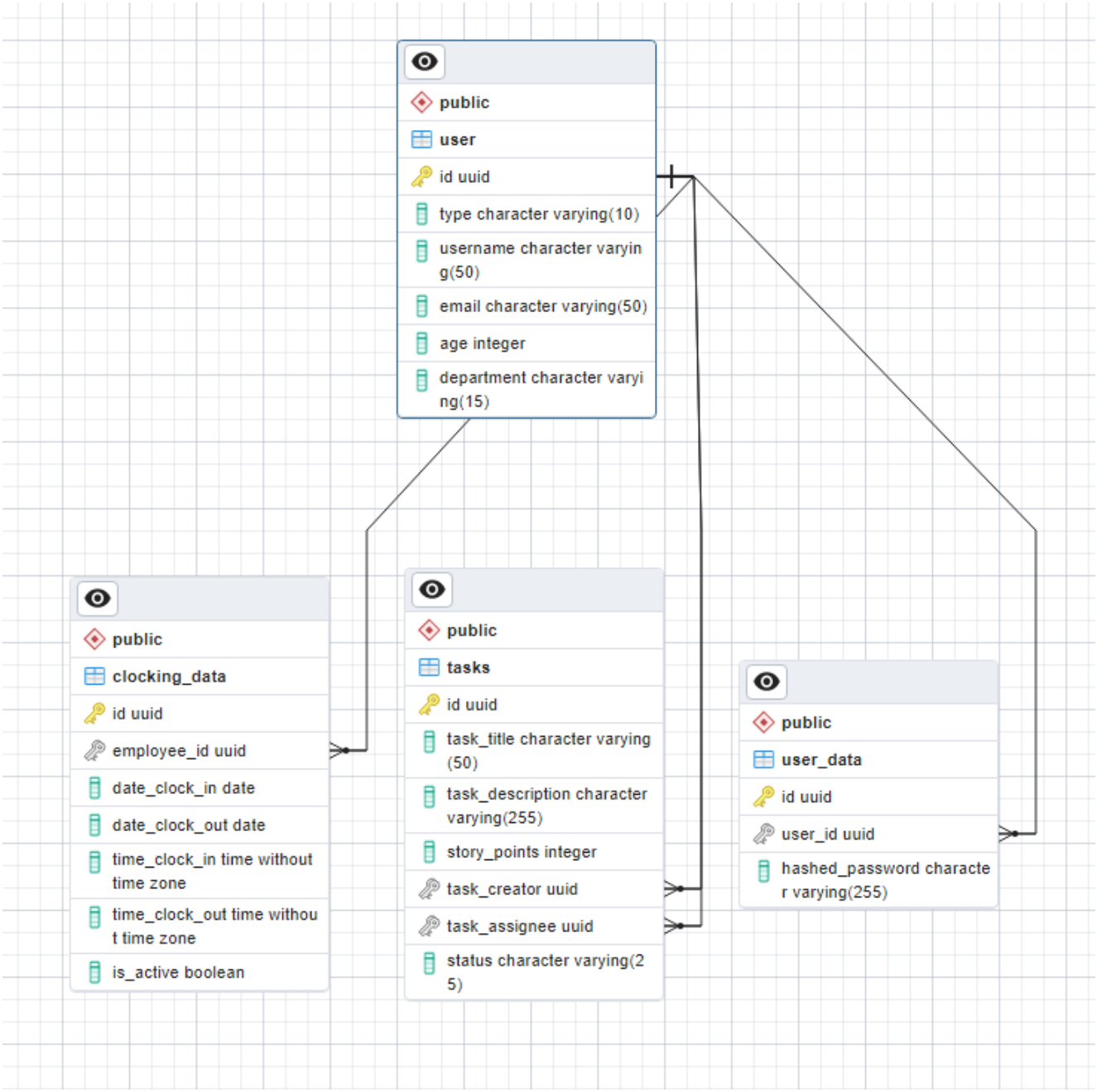


Diagrama bazei de date:



Documentatia tehnica

1. Clonarea proiectului de pe github:
   `git clone https://github.com/LCCosmin/iss_lab.git`
2. Dechiderea unui terminal (Git Bash pentru Windows, orice terminal pentru Linux/MacOS)
3. Asigurare ca aveti python 3.10+ instalat
   `python3 -V`

```
Cosmin@DESKTOP-Q6CDSQN MINGW64 ~
$ python3 -V
Python 3.10.11
```

4. Mergeti la locatia repo-ului creat in terminal si asigurati-va ca aveti fisierul `requirements.txt`
   `ls -la`

```
Cosmin@DESKTOP-Q6CDSQN MINGW64 ~/Desktop/Lab 3 ISS
$ ls -la
total 498
drwxr-xr-x 1 Cosmin 197121      0 May 21 22:39 ./
drwxr-xr-x 1 Cosmin 197121      0 May 22 11:14 ../
-rw-r--r-- 1 Cosmin 197121 337179 Apr  2 18:37 Lab_ISS.mdj
drwxr-xr-x 1 Cosmin 197121      0 May 21 23:16 UI_QT/
drwxr-xr-x 1 Cosmin 197121      0 May 22 00:26 __pycache__/
-rw-r--r-- 1 Cosmin 197121  12095 May 21 23:13 boss.py
-rw-r--r-- 1 Cosmin 197121   4643 May 22 00:22 controller.py
-rw-r--r-- 1 Cosmin 197121  31065 Apr  2 18:58 db_uml_diagram.pgerd
-rw-r--r-- 1 Cosmin 197121  18237 May 22 00:25 employee.py
drwxr-xr-x 1 Cosmin 197121      0 May 21 20:34 enums/
-rw-r--r-- 1 Cosmin 197121   2030 May 21 20:49 error.py
drwxr-xr-x 1 Cosmin 197121      0 Apr  2 19:16 iss_env/
-rw-r--r-- 1 Cosmin 197121  17920 Apr  2 19:10 iss_lab
-rw-r--r-- 1 Cosmin 197121   5101 May 21 23:07 login.py
-rw-r--r-- 1 Cosmin 197121    328 May 21 21:43 main.py
drwxr-xr-x 1 Cosmin 197121      0 May 21 20:34 models/
-rw-r--r-- 1 Cosmin 197121    292 May 22 00:27 requirements.txt
-rw-r--r-- 1 Cosmin 197121   1204 May 22 00:12 task_generator.py
```

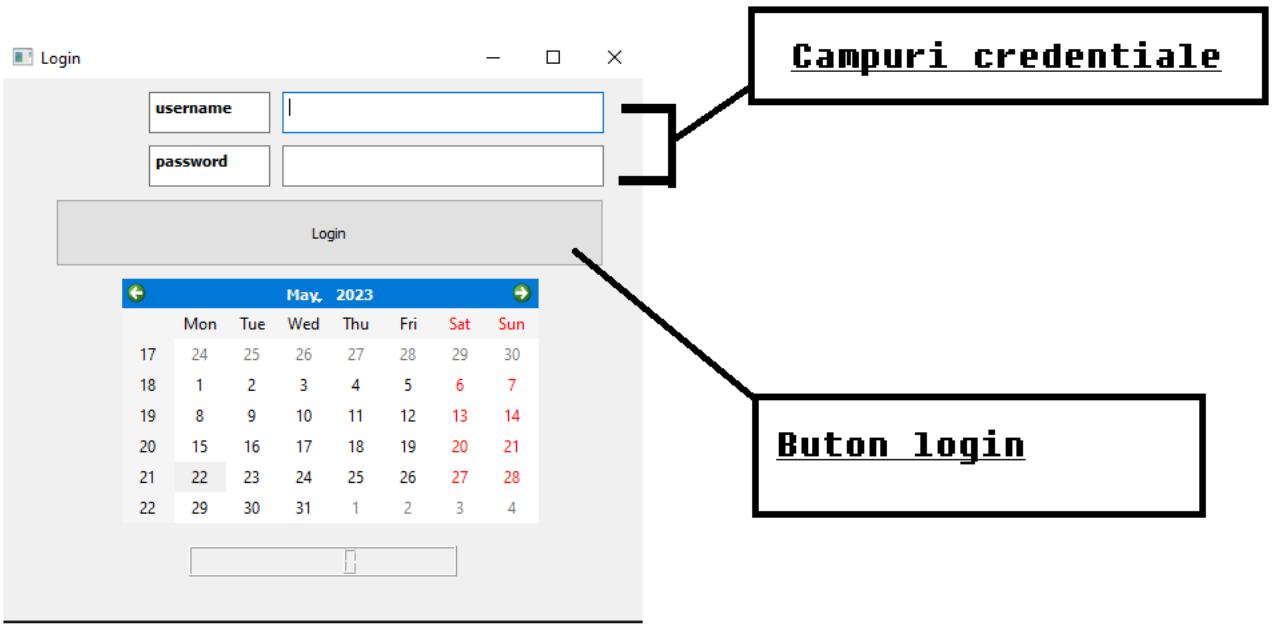
5. Creati un virtual environment nou
   `python3 -m venv /path/to/new/virtual/environment`
6. Activati noul virtual environment
   `source /path/to/new/virtual/environment/bin/activate` - Linux / MacOS
   `source /path/to/new/virtual/environment/Scripts/activate` - Git Bash Windows

7. Asigurati-va ca aveti `pip` instalat
   `pip` in terminal

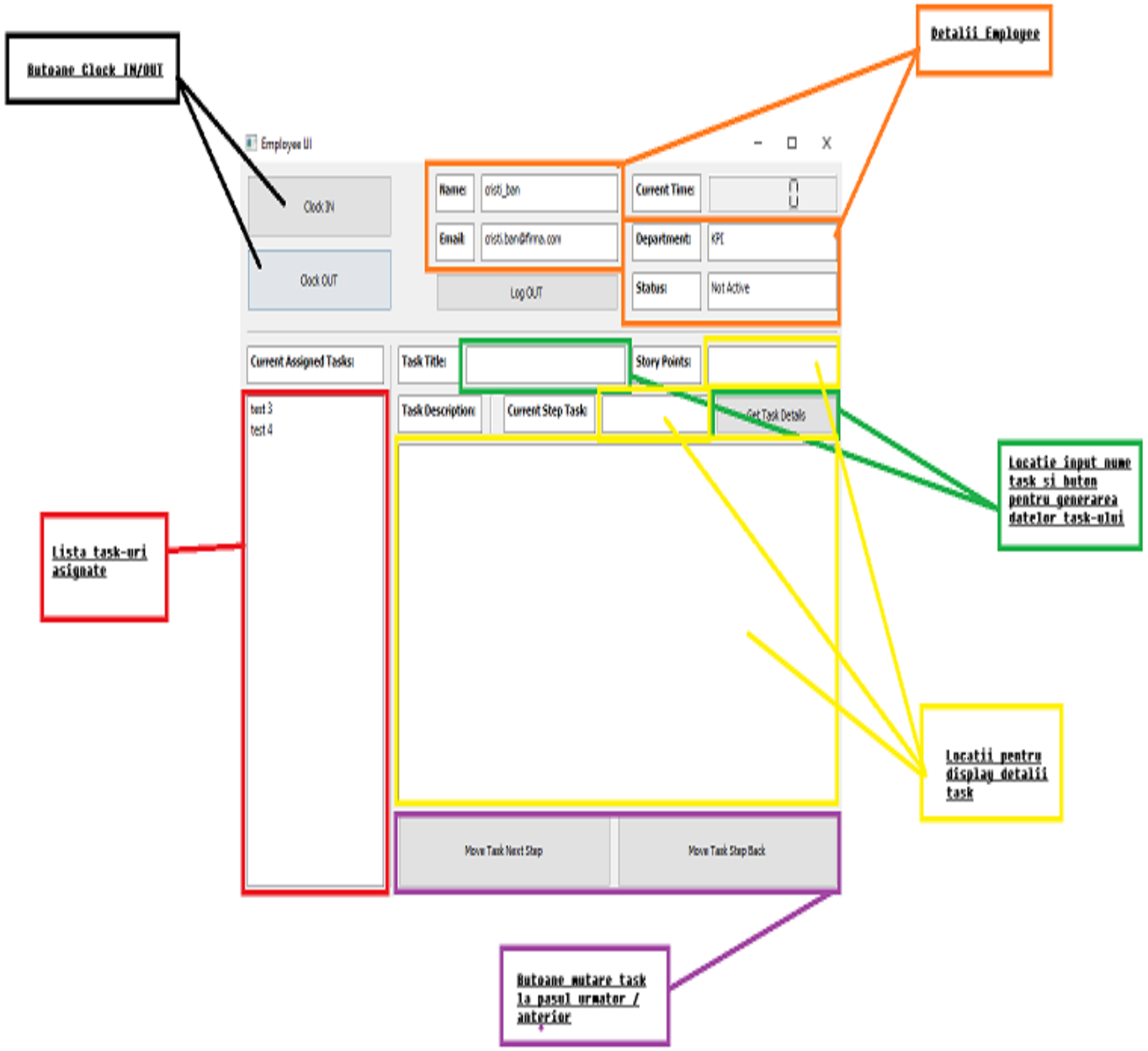
8. Instalati dependentele la proiect
   `pip install -r requirements.txt`
9. Lansati proiectul:
   `python3 main.py`

Ajutor utilizator

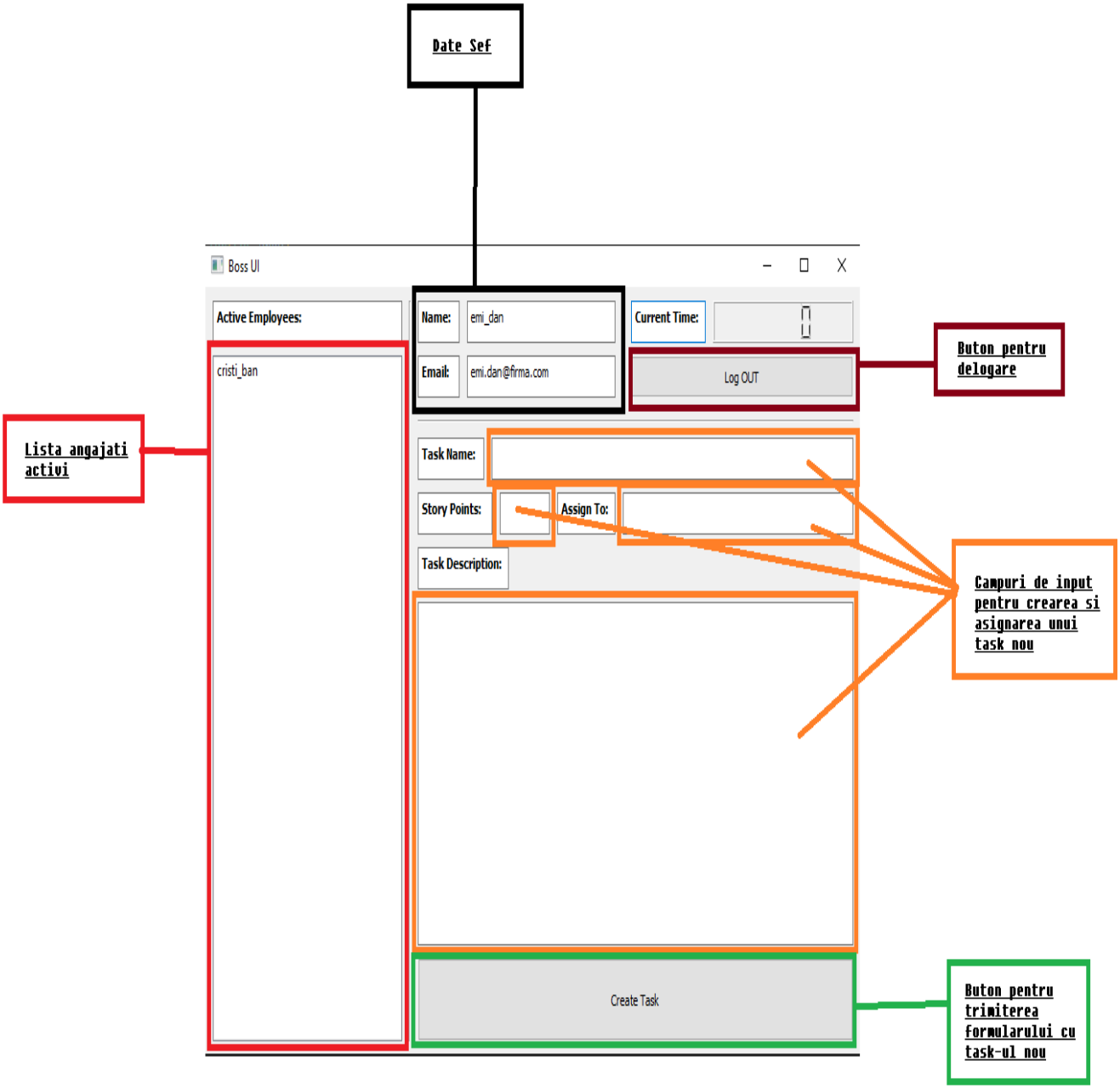
Login:



Employee UI:


Butoane Clock IN/OUT
Detalii Employee
Employee UI
Clock IN
Clock OUT
Name:
oristi_ban
Email:
oristi.ban@firma.com
Log OUT
Current Time:
Department:
KPI
Status:
Not Active
Current Assigned Tasks:
Task Title:
Story Points:
Task Description:
Current Step Task:
Get Task Details
test 3
test 4
Move Task Next Step
Move Task Step Back
Lista task-uri asignate
Locatie input nume task si buton pentru generarea datelor task-ului
Locatii pentru display detalii task
Butoane mutare task la pasul urmator / anterior

Boss UI:


Date Sef
Boss UI
Active Employees:
cristi_ban
Name:
emi_dan
Email:
emi.dan@firma.com
Current Time:
Log OUT
Task Name:
Story Points:
Assign To:
Task Description:
Create Task
Lista angajati activi
Buton pentru delogare
Campuri de input pentru crearea si asignarea unui task nou
Buton pentru trimiterea formularului cu task-ul nou

Bibliografie

1. Tutoriale PyQT:
    a. Management of windows: https://www.youtube.com/watch?v=OmZ-WUDr4JQ&ab_channel=Codemy.com
    b. MenuBars: https://www.youtube.com/watch?v=HYV81L7qd6M&ab_channel=TechWithTim
    c. How to use QT Designer: https://www.youtube.com/watch?v=FVpho_UiDAY&ab_channel=TechWithTim
2. Documentatie SQLAlchemy: https://docs.sqlalchemy.org/en/20/
3. Documentatie PyQT: https://doc.qt.io/qtforpython-6/
4. GRASP Design Patterns Principles: https://www.youtube.com/watch?v=fGNF6wuD-fg&t=767s&ab_channel=ArjanCodes
5. Curs OOP: https://www.udemy.com/course/java-the-complete-java-developer-course/learn/lecture/34997018?start=0#overview