



UNIVERSITATEA BABEȘ-BOLYAI
Facultatea de Matematică și Informatică



PROIECT - INGINERIA SISTEMELOR SOFTWARE

AN UNIVERSITAR 2022-2023

SEMESTRUL 2

Aplicație Monitoring System

Panduru Mihai

Grupa 225 / 1

Informatică română

Coordonator:

Prof. Sima Ioan



Cuprins

Prezentarea cerinței	4
Modelul funcțional	5
Diagrama cazurilor de utilizare	5
Descrierea cazurilor de utilizare	5
Modelul conceptual	12
Diagrama de clase a modelului	12
Diagrama de clase finisate	13
Diagrama bazei de date	13
Modelul dinamic	14
Diagramele de secvență	14
Diagramele de comunicare	16
Funcționalități	17
Angajat	17
Șef	17
Etapele proiectării	18
Analiza cerințelor	18
Cerințe funcționale	18
Cerințe nefuncționale	18
Proiectarea sistemului	19
Object design	19
Implementarea	19
Testarea	19
Tehnologii folosite	20
Bază de date	20
Mediu de proiectare	20
Limbaj de programare	20



ORM	21
Proiectarea diagramelor	21
GUI	21
Comunicare client-server	21
Testarea codului	22
Scenariu de utilizare	22
Angajat	22
Șef	23
Tutorial	24
Server	24
Deschidere server	24
Conectare client	24
Acțiune luată pentru un proces	24
Deconectare client	25
Client	25
Fereastră de logare	25
Fereastră principală	26
Fereastră principală cu rezultatele scanării	27
Fereastră secundară cu detaliile unui proces	28
Bibliografie	29



Prezentarea cerinței

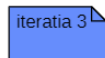
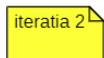
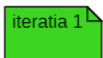
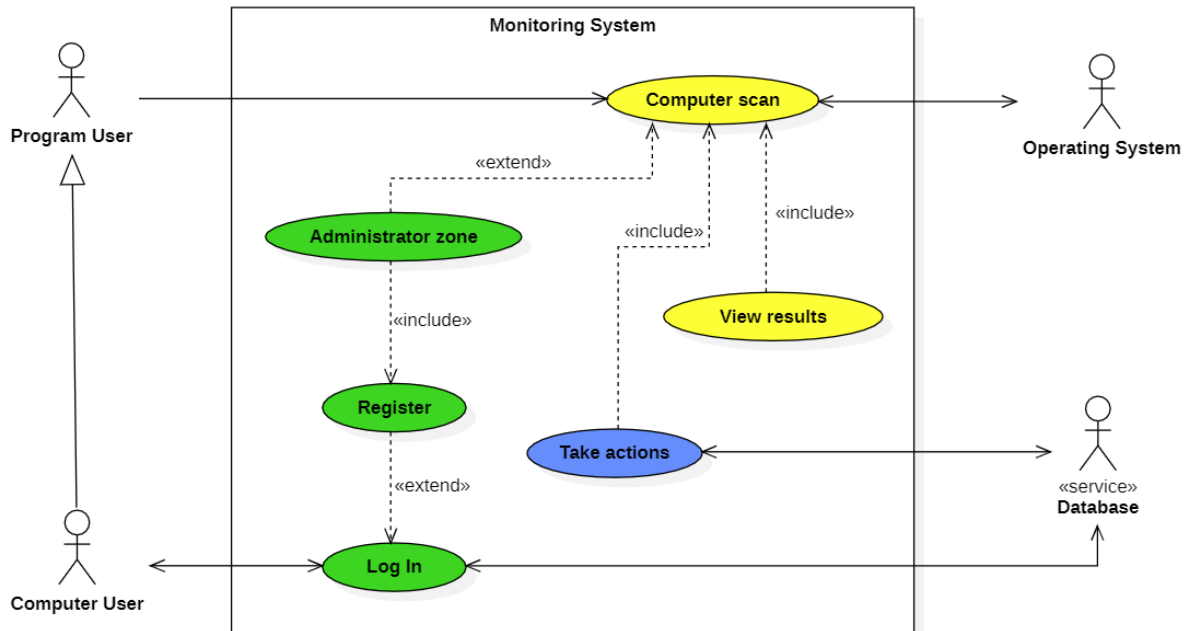
În peisajul digital actual, malware-ul reprezintă o amenințare deosebit de serioasă. În România, suntem confrunțați cu o creștere alarmantă a numărului de infecții cu malware. Aceste programe malițioase pot avea consecințe devastatoare, precum furtul de informații sensibile, blocarea sistemelor sau preluarea controlului acestora. Pentru a contracara această amenințare, este vital să luăm măsuri preventive și să implementăm soluții avansate de securitate. Totodată, educarea în domeniul securității cibernetice joacă un rol esențial în protejarea împotriva malware-ului și în asigurarea siguranței noastre în mediul online.

O firmă și-a creat o infrastructură prin care monitorizează aplicațiile folosite de angajații săi. Firma are o aplicație care oferă:

- *o fereastră pentru șef*, fereastră de control, cu ajutorul căreia șeful vede lista programelor în execuție în timp real pentru fiecare angajat al său care este în timpul lucrului. De asemenea, șeful poate lua o acțiune pentru orice astfel de proces în caz că i se pare că ceva nu este în regulă astfel: selectează angajatul, procesul și acțiunea și o declanșează apăsând un buton “Take action”. Imediat după apăsarea butonului, toți angajații sunt notificați și se actualizează procesele pentru fiecare dintre ei.
- *câte o fereastră pentru fiecare angajat*: atunci când angajatul vine la serviciu, el deschide aplicația cât să poată fi supravegheat și să își desfășoare munca în siguranță. În orice moment, acesta poate lansa o scanare de procese și poate vedea detalii pentru fiecare proces în parte. De asemenea, poate lua diferite acțiuni pentru procese pe care le consideră suspecte.

Modelul funcțional

Diagrama cazurilor de utilizare



Descrierea cazurilor de utilizare

ID and name	UC-1: Computer scan		
Primary actor	Program User	Secondary actors	Operating System Monitoring System
Description	The program user should be able to initiate a computer scan at any time. It can be stopped by the user or by the administrator at any time. If the administrator is the one initiating the scan, the user can not stop it. Once the scan is completed, the results are shown including the safe processes and the less safe/unsafe ones. The user can perform some actions for the latter processes.		



Trigger	The user requests a computer scan by clicking the 'Scan computer' button.
Preconditions	Pre-1: The user is logged into the program. Pre-2: The user has a valid license for the program.
Postconditions	Post-1: The user can see the scan results. Post-2: The user is able to perform actions on less safe/unsafe processes.
Normal flow	1.0 Computer scan <ol style="list-style-type: none"> 1. The user logs in to view the Computer scan section. 2. The user starts the scan by pressing the 'Scan computer' button and waits for the results. 3. The results are shown in the results section and the user can take actions on some processes he considers.
Alternative flows	1.1 Multiple computer scans <ol style="list-style-type: none"> 1. The user wants another scan after a scan has already been completed. 2. Return to step 2 of normal flow.
Exceptions	None

ID and name	UC-2: Administrative users		
Primary actor	Program Administrator	Secondary actors	Monitoring System Program User
Description	The administrator should be able to view all the users computers' status and to initiate a computer scan for any of the users at any time. It can not be stopped by the user but by the administrator at any time.		
Trigger	The administrator enters this zone via log in.		
Preconditions	Pre-1: The user is logged into the program and has administrator privileges.		
Postconditions	Post-1: The administrator can see all the user's status, licenses and actions performed. Post-2: The user is able to start/stop a computer scan for any of the users. Post-3: The taken actions for users are recorded in the database.		



Normal flow	2.0 User's Computer scan <ol style="list-style-type: none"> 1. The administrator selects a user and starts the scan for him. 2. The administrator is the only one who can stop the scan. 3. The administrator waits the scan to complete. 4. The results are shown in the results section and the administrator can take actions on some user's processes he considers. 5. The program perform the action specified by the administrator. (see 2.0.E1) 6. The program sends the action taken to the server. 7. The program updates the results after the action is performed.
Alternative flows	2.1 Multiple computer scans <ol style="list-style-type: none"> 1. The administrator wants another scan after he had taken some actions for processes of the previous scan. 2. Return to step 1 of normal flow.
Exceptions	2.0.E1 Action denied <ol style="list-style-type: none"> 1. The program informs the administrator that the selected action can not be performed on the considered process due to process is vital to the system. 2. The administrator has to reconsider the action he wants to perform.

ID and name	UC-3: Log In		
Primary actor	Computer User	Secondary actors	Monitoring System
Description	The computer user must authenticate in order to use the program. He needs a valid username and password. A valid license is not necessarily needed.		
Trigger	The computer user starts the program.		
Preconditions	Pre-1: The computer user is not currently logged in.		
Postconditions	Post-1: The computer user is now a program user. Post-2: The program user can initiate a scan.		



Normal flow	3.0 Log In <ol style="list-style-type: none"> 1. The computer user starts the program. 2. The user specifies its username and password and clicks the log in button. (see 3.0.E1) 3. The computer user is logged in and can use the program.
Alternative flows	None
Exceptions	3.0.E1 User is already logged in <ol style="list-style-type: none"> 1. The program informs the computer user that there is an active instance where it is logged in. 2. The user has to use the active session.

ID and name	UC-4: Register		
Primary actor	Program Administrator	Secondary actors	Monitoring System Program User
Description	The administrator should be able to register an account for a normal user. He can not modify or delete details of the account after the creation has been done.		
Trigger	The administrator clicks the 'Create account' button found in the Administrator's panel.		
Preconditions	Pre-1: The user is logged into the program and has administrator privileges. Pre-2: The administrator enters valid credentials for the new account.		
Postconditions	Post-1: The account is created and can be used to log in. Post-2: The administrator sees the new user in the users list.		



Normal flow	4.0 Register account <ol style="list-style-type: none"> 1. The computer user starts the program. 2. The user specifies its username and password and clicks the log in button. (see 4.0.E1) 3. The program user is logged in having the administrator privileges and can use the program. 4. The administrator clicks the 'Create account' button and enters valid credentials for the new account. (see 4.0.E2) 5. The account is created and can be used to log in by a user.
Alternative flows	None
Exceptions	4.0.E1 User is already logged in <ol style="list-style-type: none"> 1. The program informs the computer user that there is an active instance where it is logged in. 2. The user has to use the active session. 4.0.E2 Credentials are not valid or already taken <ol style="list-style-type: none"> 1. The program informs the computer user that the entered credentials are not valid or already in use. 2. The user has to enter another set of credentials.

ID and name	UC-5: View results		
Primary actor	Program User	Secondary actors	Operating System Monitoring System
Description	The program user should be able to see details for every process on the system. The running processes are shown in the Results list after a scan has been performed. The details the user sees are meant to help with threat intelligence.		
Trigger	The user selects a process by clicking it in the list and then the 'Detail view' button.		



Preconditions	<p>Pre-1: The user is logged into the program.</p> <p>Pre-2: The user has a valid license for the program.</p> <p>Pre-3: A scan has already been performed.</p>
Postconditions	<p>Post-1: The user can see the scan results.</p> <p>Post-2: The user is able to perform actions on less safe/unsafe processes.</p> <p>Post-3: The taken actions are recorded in the database.</p>
Normal flow	<p>5.0 View results</p> <ol style="list-style-type: none"> 1. The user logs in to view the Computer scan section. 2. The user has already performed a scan and sees the results. 3. The user selects a process to view the details for and the 'Detail view' button gets enabled. 4. The user clicks the 'Detail view' button and a separate window with details for the process selected is shown.
Alternative flows	<p>5.1 Detail view for multiple processes</p> <ol style="list-style-type: none"> 1. The user wants to see details for another process. 2. Return to step 3 of normal flow.
Exceptions	None

ID and name	UC-6: Take actions		
Primary actor	Program User	Secondary actors	<p>Operating System</p> <p>Monitoring System</p>
Description	The program user should be able to take several actions to processes he considers. Some actions might be denied due to low privileges or protected processes that are vital to the system.		
Trigger	The user selects a process by clicking it in the list and then selects the action from the actions group. After both are selected, the user clicks the 'Take action' button.		



Preconditions	<p>Pre-1: The user is logged into the program.</p> <p>Pre-2: The user has a valid license for the program.</p> <p>Pre-3: A scan has already been performed.</p>
Postconditions	<p>Post-1: The user can see either the selected action taken for the considered process or a message denying the action.</p> <p>Post-2: All the logged users see the updated list with processes after the action has been performed.</p> <p>Post-3: The taken action is recorded in the database.</p>
Normal flow	<p>6.0 Take action on process</p> <ol style="list-style-type: none">1. The user logs in to view the Computer scan section.2. The user has already performed a scan and sees the results.3. The user selects a process and an action to be performed on that process.4. The user clicks the 'Take action' button and waits either for the action to be performed or an denying informative message. (see 6.0.E1)
Alternative flows	<p>6.1 Multiple actions to be taken</p> <ol style="list-style-type: none">1. The user wants to perform an action multiple times for the same process or for other processes.2. Return to step 3 of normal flow.
Exceptions	<p>6.0.E1 Action denied</p> <ol style="list-style-type: none">1. The program informs the program user that the selected action can not be performed on the considered process due to several reasons: process is vital to the system or higher privileges are required.2. The user has to reconsider the action he wants to perform.

Modelul conceptual

Diagrama de clase a modelului

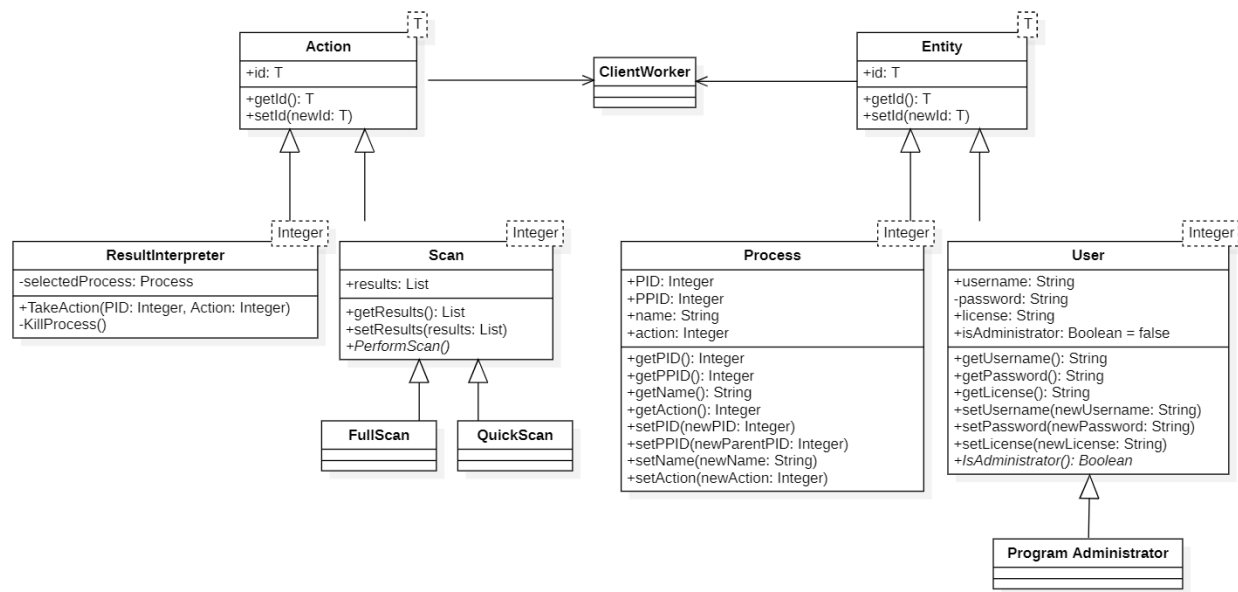


Diagrama de clase finisate

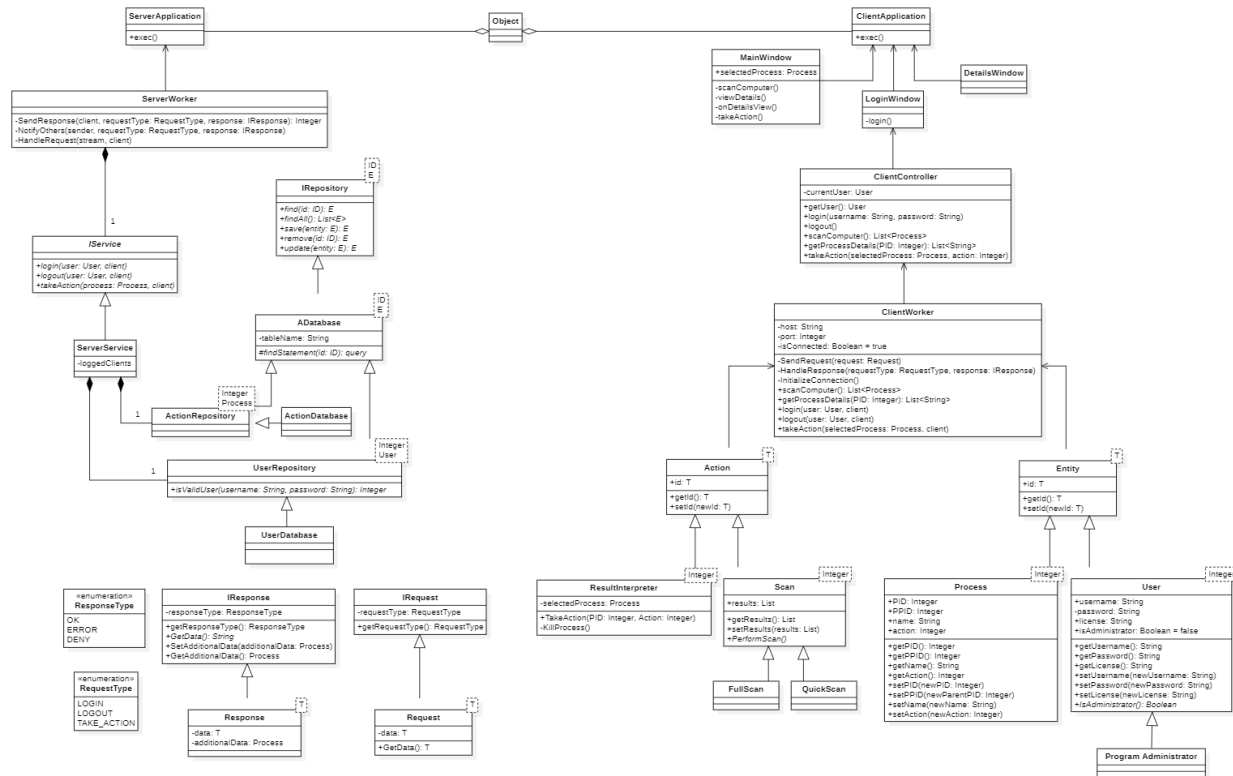
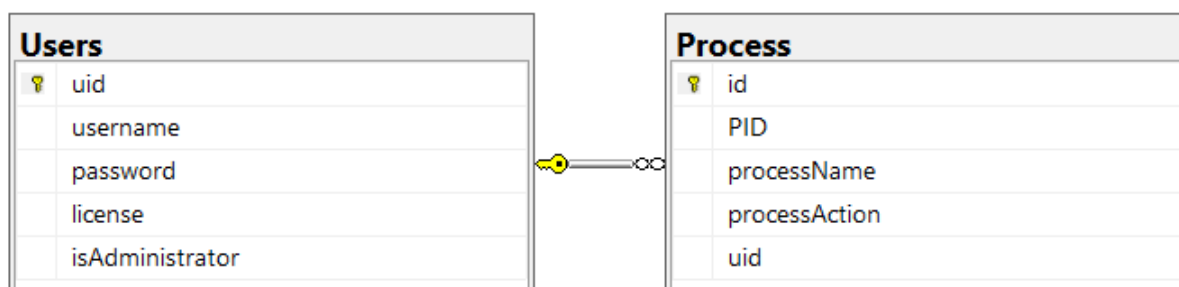
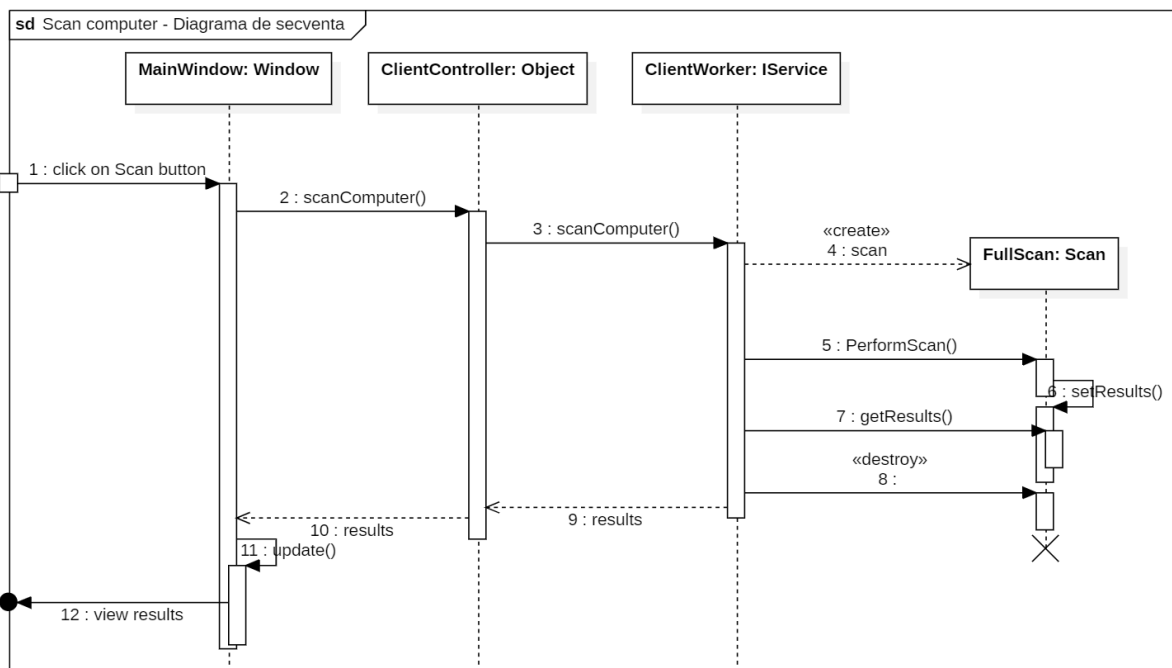
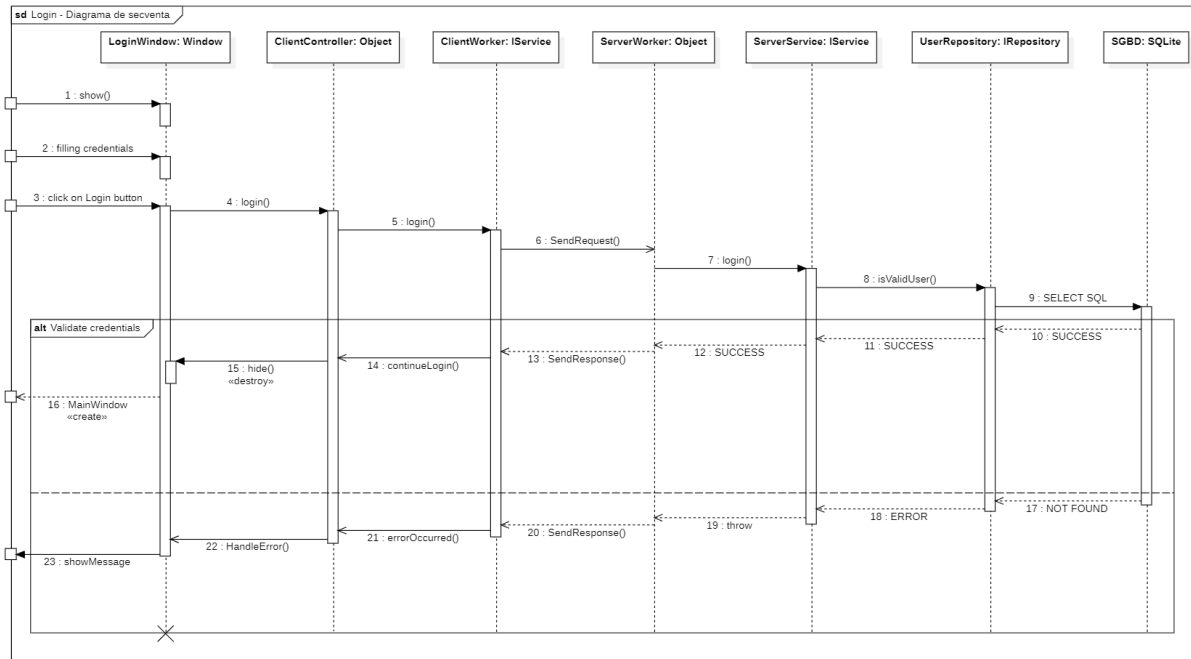


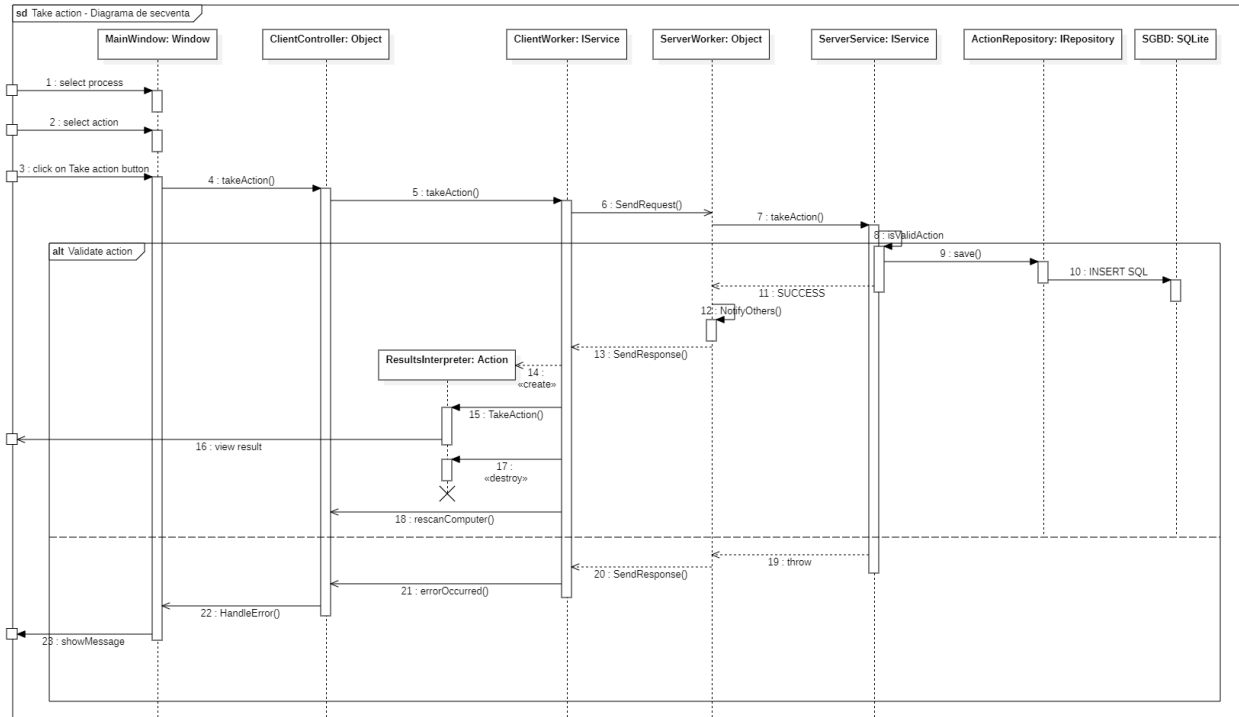
Diagrama bazei de date



Modelul dinamic

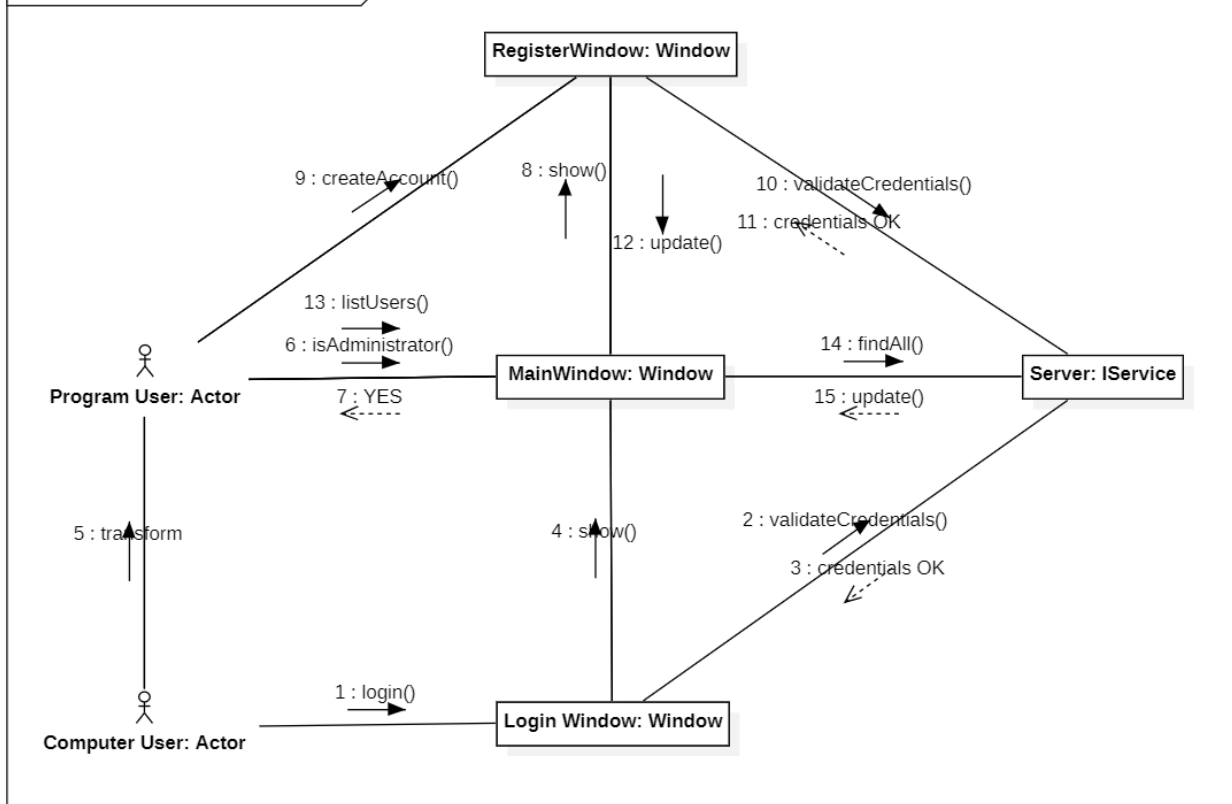
Diagramele de secvență



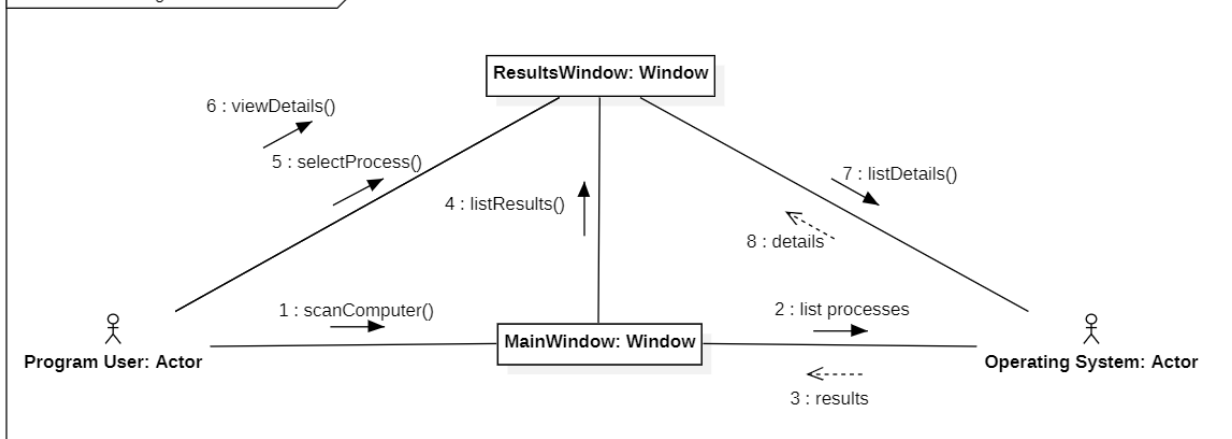


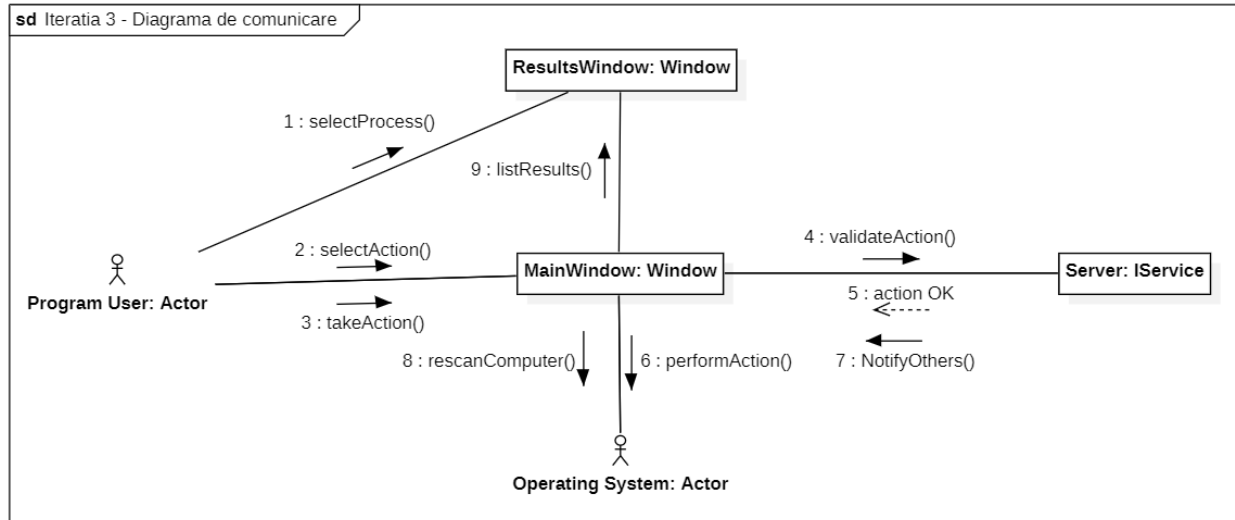
Diagramele de comunicare

sd Iteratia 1 - Diagrama de comunicare



sd Iteratia 2 - Diagrama de comunicare





Funcționalități

Angajat

- poate să devină un utilizator al aplicației prin logare
- ca utilizator, poate iniția o scanare de sistem și vizualiza lista tuturor proceselor. De asemenea poate vedea detalii despre fiecare proces în parte. Mai mult decât atât, poate lua diferite măsuri pentru procesele de pe sistemul personal.

Șef

- poate să devină un utilizator cu drepturi administrative al aplicației prin logare
- în urma logării, poate vedea toți ceilalți angajați logați în aplicație care sunt la muncă



- ca utilizator cu drepturi administrative, poate iniția o scanare de sistem și vizualiza lista tuturor proceselor pentru orice angajat logat în aplicație. De asemenea poate vedea detalii despre fiecare proces în parte. Mai mult decât atât, poate lua diferite măsuri pentru procesele de pe sistemul oricărui angajat.
- șeful poate crea un cont nou pentru un nou angajat

Etapele proiectării

Analiza cerințelor

Pe lângă cerințele impuse și funcționalități, am identificat și constrângeri: un angajat are un singur cont creat la angajare, doar șeful poate vedea procesele tuturor angajaților, o acțiune pentru un proces nu se poate realiza în cazul în care procesul este unul vital pentru funcționarea sistemului sau în cazul în care privilegiile sunt prea scăzute.

Cerințe funcționale

Aplicația trebuie să permită angajatului să vadă lista proceselor active pe sistemul personal în timpul muncii astfel încât acesta să știe dacă este în siguranță. Pentru fiecare astfel de proces, aplicația trebuie să permită vizualizarea în detaliu a informațiilor prin care se poate diferenția un program normal de un program malițios.

Cerințe nefuncționale

Aplicația trebuie să fie scrisă în C++ și să fie ușor de utilizat. Aplicația trebuie să folosească Qt framework împreună cu QxOrm pentru a



beneficia de o bază stabilă de dezvoltare ulterioară dată de interacțiunea complementară, de actualitate, actualizată și susținută de o comunitate mare a celor două componente.

Proiectarea sistemului

Am folosit StarUML pentru a dezvolta diferite tipuri de diagrame menite să ne ușureze ulterior munca. Am dezvoltat diagrama cazurilor de utilizare, diagramele de clase, diagrama bazei de date, diagramele de comunicare și de secvență.

Object design

Clasele din domeniul problemei au fost modelate astfel încât să permită ne-excluderea reciprocă a mai multor tipuri de constrângeri. Astfel, au fost utilizate clase generice pentru o entitate și o acțiune ca mai departe să poată fi adăugate altele noi cu ușurință.

Implementarea

Am început prin a implementa baza de date. După aceea am continuat cu clasele din domeniul problemei și apoi am continuat cu fereastra de logare încât să pot testa codul scris până în acel moment. După ce sistemul de logare a fost funcțional, am continuat cu următoarea fereastră și anume fereastra principală care conține scanarea de procese, rezultatele în urma scanării, vizualizarea detaliilor pentru un anumit proces și acțiunile care se pot lua asupra unui proces. În cele din urmă, am creat serverul care să valideze toate acțiunile necesare prin conexiunea acestuia la baza de date.

Testarea

Codul a fost testat progresiv astfel rezolvând toate problemele identificate la fiecare parte de implementare. Fiecare funcție a aplicației



a fost testată atât individual cât și în diferite combinații cu altele pentru a asigura funcționarea corectă și completă a aplicației.

Tehnologii folosite

Bază de date

Am folosit SQLite ca și bază de date deoarece este foarte simplu de configurat și ușor de folosit pentru o aplicație până în nivel mediu. De asemenea, este foarte portabilă și compactă ușurând ulterior orice modificare se dorește.

Mediu de dezvoltare

Am ales să folosesc Microsoft Visual Studio Community 2022 deoarece sunt mai familiarizat cu acesta, este ușor de folosit și îmi place personal mai mult.

Limbaaj de programare

Limbaajul de programare ales este C++ deoarece este foarte permisiv și tind să înțeleg mai bine ceea ce se întâmplă în spate lovindu-mă mult mai des de probleme. Prin rezolvarea problemelor, prin gestiunea corectă a memoriei, prin cunoașterea în detaliu a sistemului de operare și a limbajului se poate obține un program cu capacități extraordinare. Un alt motiv a fost acela al performanței, care este necesară pentru acest tip de aplicație; este foarte rapid, performant fiind un limbaj low-level de nivel înalt.



ORM

Object Relational Mapping (ORM) este o tehnica de programare care facilitează interacțiunea între bazele de date relaționale și obiectele dintr-un limbaj de programare orientat obiect. Scopul principal al unui ORM este de a realiza o mapare automată între structurile de date și tabelele din baza de date, astfel încât programatorul să poată manipula datele utilizând obiecte și metode specifice limbajului de programare, în loc de a scrie interogări SQL direct.

Am ales să folosesc QxOrm deoarece conlucrează foarte bine cu Qt framework și pentru faptul că este de actualitate, actualizat. Mai mult decât atât, este ușor de folosit și are foarte multe funcționalități.

Proiectarea diagramelor

Am ales să folosesc StarUML la proiectarea diagramelor pentru multitudinea de funcționalități ale programului și pentru ușurința de folosire a acestuia.

GUI

Am folosit o extensie la IDE numită Qt VS Tools care activează Qt framework direct în IDE și astfel am putut crea interfețele în Qt direct în mediul de dezvoltare fără a fi nevoie de alte programe.

Comunicare client-server

Structura proiectului este organizată pe mai multe niveluri:

- Domeniu: conține declarații de entități folosite în aplicație.
- Persistență: aplicația comunică cu baza de date printr-o clasă abstractă ce implementează toate operațiile CRUD dintr-o



interfață. Orice clasă ce servește drept repository moștenește din această clasă abstractă.

- Servicii: conțin și controlează interfețele aplicației, fac legătura dintre utilizator și aplicația propriu-zisă.
- Server: validează operațiile primite de la un client, comunică cu baza de date, implementează peristența.
- Client: conține toate interfețele vizibile utilizatorului și controlează cererile către server.

Testarea codului

Codul a fost testat minuțios, manual, prin inspecția datelor la fiecare pas și prin afișarea multor mesaje de tip checkpoint atât în consola serverului cât și utilizatorului atunci când este cazul.

Scenariu de utilizare

Angajat

La deschiderea aplicației este întâmpinat de fereastra de logare unde trebuie să își introducă propriile credențiale obținute de la șef la angajare. După ce acestea au fost validate de server, o altă fereastră se deschide ce conține butonul de scanare a sistemului, o listă cu rezultatele scanării, butonul de vizualizare în detaliu a unui proces și lista cu acțiuni ce pot fi aplicate unui proces. Utilizatorul aplicației inițiază o scanare de sistem și vede lista proceselor active pe calculatorul personal, apoi selectează un proces din această listă și vizualizează în detaliu informațiile pentru acesta, iar în cele din urmă, dacă consideră



că procesul este suspect, alege una dintre acțiuni (omoară procesul, suspendă procesul, reia procesul) și o aplică procesului. Acțiunea este trimisă la server pentru validare și înregistrare și dacă este una permisă atunci este aplicată procesului. Toți ceilalți utilizatori sunt notificați și văd lista actualizată a proceselor imediat.

Şef

La deschiderea aplicației este întâmpinat de fereastra de logare unde trebuie să își introducă propriile credențiale primite de la dezvoltatorul aplicației. După ce acestea au fost validate de server, o altă fereastră se deschide ce conține zona de administrare a angajaților, butonul de scanare a sistemului, o listă cu rezultatele scanării, butonul de vizualizare în detaliu a unui proces și lista cu acțiuni ce pot fi aplicate unui proces. Şeful selectează un angajat din listă și inițiază o scanare de sistem pentru acesta urmând să vadă lista proceselor active pe calculatorul angajatului, apoi selectează un proces din această listă și vizualizează în detaliu informațiile pentru acesta, iar în cele din urmă, dacă consideră că procesul este suspect, alege una dintre acțiuni (omoară procesul, suspendă procesul, reia procesul) și o aplică procesului. Acțiunea este trimisă la server pentru validare și înregistrare și dacă este una permisă atunci este aplicată procesului. Toți ceilalți utilizatori sunt notificați și văd lista actualizată a proceselor imediat.

De asemenea, zona administrativă îi permite şefului să creeze un cont nou pentru un nou angajat introducând credențialele acestuia.



Tutorial

Server

Deschidere server

```
Server started.  
Waiting for clients...  
_
```

Conectare client

```
Server started.  
Waiting for clients...  
A client has connected. 880  
[QxOrm] qx::QxSqlDatabase : create new database connection in thread '0x296c' with key '{42cf6959-99d9-4d84-ac26-0e05af842620}'  
[QxOrm] sql query (total: 12.7 ms, db_exec: 0.261 ms, db_prepare: 0.438 ms, db_next(2): 0.0275 ms, db_open: 11.5 ms, db_transaction: 0 ms, build_relations: 0 ms, build_sql: 0 ms, build_cpp: 0 ms, read_cpp: 0 ms) :  
SELECT *  
  FROM Users  
 WHERE username = :username  
    AND password = :password  
_
```

Acțiune luată pentru un proces

```
Server started.  
Waiting for clients...  
A client has connected. 880  
[QxOrm] qx::QxSqlDatabase : create new database connection in thread '0x296c' with key '{42cf6959-99d9-4d84-ac26-0e05af842620}'  
[QxOrm] sql query (total: 12.7 ms, db_exec: 0.261 ms, db_prepare: 0.438 ms, db_next(2): 0.0275 ms, db_open: 11.5 ms, db_transaction: 0 ms, build_relations: 0 ms, build_sql: 0 ms, build_cpp: 0 ms, read_cpp: 0 ms) :  
SELECT *  
  FROM Users  
 WHERE username = :username  
    AND password = :password  
  
[QxOrm] sql query (total: 4.49 ms, db_exec: 1.78 ms, db_prepare: 1.32 ms, db_next(0): 0 ms, db_open: 0.376 ms, db_transaction: 0 ms, build_relations: 0 ms, build_sql: 0.712 ms, build_cpp: 0 ms, read_cpp: 0.0558 ms) :  
INSERT INTO Process (PID, processName, processAction)  
  VALUES (:PID, :processName, :processAction)  
_
```




Deconectare client

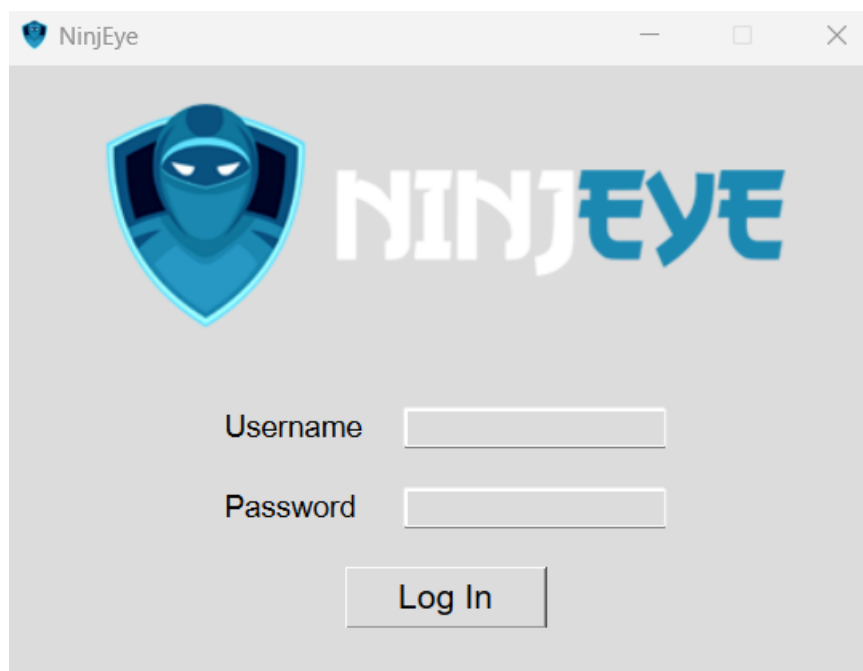
```
Server started.
Waiting for clients...
A client has connected. 880
[QxOrm] qx::QxSqlDatabase : create new database connection in thread '0x296c' with key '{42cf6959-99d9-4d84-ac26-0e05af842620}'
[QxOrm] sql query (total: 12.7 ms, db_exec: 0.261 ms, db_prepare: 0.438 ms, db_next(2): 0.0275 ms, db_open: 11.5 ms, db_transaction: 0 ms, build_relations: 0 ms, build_sql: 0 ms, build_cpp: 0 ms, read_cpp: 0 ms) :
SELECT *
  FROM Users
 WHERE username = :username
    AND password = :password

[QxOrm] sql query (total: 4.49 ms, db_exec: 1.78 ms, db_prepare: 1.32 ms, db_next(0): 0 ms, db_open: 0.376 ms, db_transaction: 0 ms, build_relations: 0 ms, build_sql: 0.712 ms, build_cpp: 0 ms, read_cpp: 0.0558 ms) :
INSERT INTO Process (PID, processName, processAction)
  VALUES (:PID, :processName, :processAction)

A client has disconnected. QTcpSocket(0x25229f7e6b0)
```

Client

Fereastră de logare



The image shows a login window titled "NinjEye". It features a blue shield icon with a white ninja mask on the left and the "NINJEYE" logo in large, stylized blue letters on the right. Below the logo, there are two input fields: "Username" and "Password". At the bottom center, there is a "Log In" button.



Câmpurile din dreapta textelor “Username” și “Password” sunt menite a fi completate de utilizatorul aplicației cu propriile credențiale de acces. Prin apăsarea butonului “Log In” fereastra de logare se va închide și fereastra principală se va deschide.

Fereastră principală

NinjEye for mihai

Scan computer

Results

Process Name	Process ID
--------------	------------

Detail view

Action

☒ Suspend process
☐ Resume process
☐ Kill process

Take action



Prin apăsarea butonului “Scan computer” se va iniția o scanare de sistem, iar rezultatele vor apărea dedesubt în secțiunea marcată corespunzător “Results”.

Fereastră principală cu rezultatele scanării

The screenshot shows a window titled "NinjEye for mihei". Inside, there is a large button labeled "Scan computer". Below it, the "Results" section contains a table with the following data:

	Process Name	Process ID
1	[System Process]	0
2	System	4
3	Secure System	172
4	Registry	220
5	smss.exe	1016
6	csrss.exe	1328
7	wininit.exe	1468

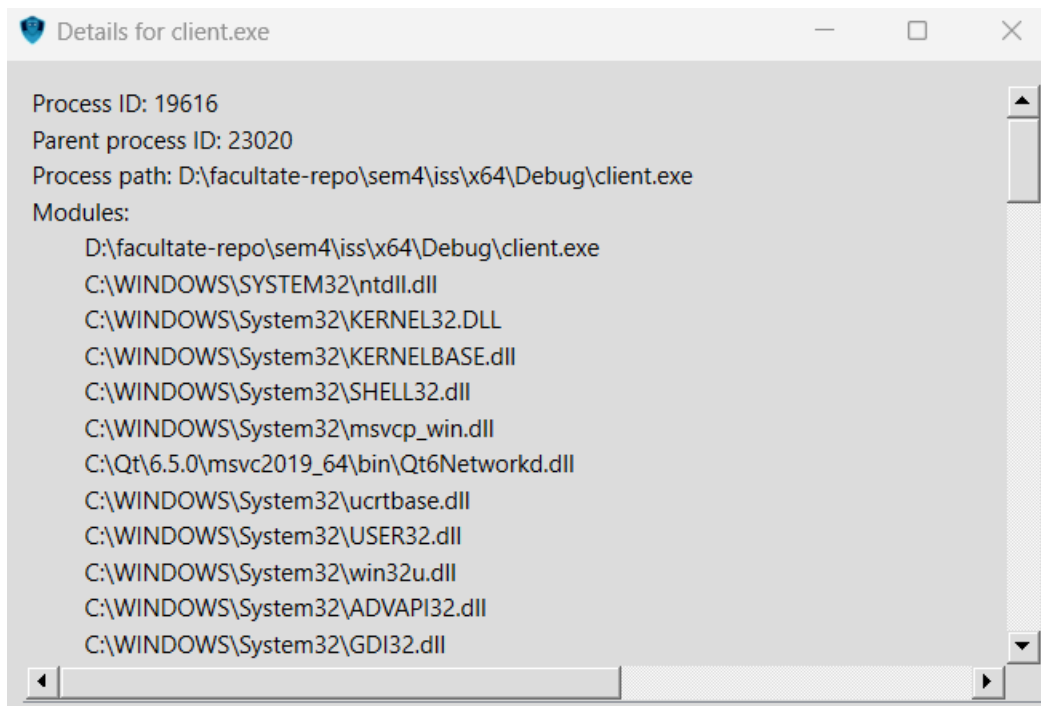
Below the table is a button labeled "Detail view". At the bottom, the "Action" section has three radio buttons: "Suspend process", "Resume process", and "Kill process" (which is selected). To the right of these buttons is a button labeled "Take action".



Prin selectarea unui proces din lista de rezultate, butonul “Detail view” se activează. Prin apăsarea acestuia, o nouă fereastră se deschide ce conține detalii pentru procesul selectat.

De asemenea, având un proces selectat, se poate selecta una dintre acțiunile disponibile, acțiune care va fi aplicată procesului prin acționarea butonului “Take action”.

Fereastră secundară cu detaliile unui proces





Bibliografie

1. <https://doc.qt.io/qt.html>
2. https://www.qxorm.com/qxorm_en/manual.html
3. <https://github.com/QxOrm/QxOrm>
4. <https://docs.staruml.io>
5. <https://github.com/vreauladudu/ubb/tree/main/sem4/mpp>
6. <https://blog.visual-paradigm.com/what-are-the-six-types-of-relationships-in-uml-class-diagrams>
7. <https://simaioan.wordpress.com>
8. Notițele de curs, seminar și laborator de la Ingineria sistemelor software, Medii de proiectare și programare, Proiectare orientată obiect, Sisteme de operare, Baze de date și Rețele de programare.