

MINISTERUL EDUCAȚIEI ȘI CERCETĂRII
UNIVERSITATEA DE STAT „BOGDAN PETRICEICU HASDEU” DIN CAHUL
FACULTATEA DE ECONOMIE, INGINERIE ȘI ȘTIINȚE APLICATE
CATEDRA DE INGINERIE ȘI ȘTIINȚE APLICATE
SPECIALITATEA INFORMATICĂ

Raport

La practica de specialitate: tehnologică

Student: Sîrbu David

Grupa: IT-2201

Locul practicii: Halley Soft S.R.L.

Coordonatorul practicii de la unitatea: Moraru Nicoleta, Administrator

Coordonatorul practicii de la USC: Vișcu Irina, Asistent Universitar

Cahul, 2025

Cuprins

Introducere.....	3
Prezentarea companiei Halley KM.....	5
Descrierea aplicației „Înregistrarea clientului cu probleme tehnice”.....	8
3.1. Obiectivele aplicației	9
3.2. Funcționalitățile principale	9
4.1. Limbajul de programare Python	10
4.2. Baza de date SQL – Structură și modelare	11
4.3. Biblioteci și framework-uri utilizate.....	12
Arhitectura aplicației	14
5.1. Structura generală a aplicației	14
5.2. Fluxul de date și interacțiunea utilizatorului.....	15
5.3. Modul de gestionare a cererilor de reparație.....	16
Implementarea funcționalităților.....	17
6.1. Modulul de înregistrare a clienților	17
6.2. Modulul de diagnosticare a problemelor tehnice	18
Testare și optimizare	19
7.1. Scenarii de testare	20
7.2. Probleme întâmpinate și soluții aplicate.....	21
7.3. Optimizarea performanței bazei de date și codului	22
Concluzii și direcții de dezvoltare viitoare	23
8.1. Evaluarea generală a proiectului	23
8.2. Posibile îmbunătățiri și extinderea funcționalităților	24
Caiet de sarcini Halley Soft	25

Introducere

1.1. Scopul raportului de practică

Acest raport de practică documentează procesul de dezvoltare a unei aplicații software destinate gestionării înregistrărilor clienților care solicită reparații pentru imprimante și calculatoare în cadrul companiei **Halley KM**. Aplicația este concepută pentru a optimiza și digitaliza procesul de preluare a cererilor de service tehnic, oferind o evidență clară a solicitărilor, statusului reparațiilor și istoricului intervențiilor.

Scopul principal al acestui raport este de a detalia etapele de proiectare, dezvoltare și implementare a soluției software, precum și impactul pe care aceasta îl are asupra activității companiei. Totodată, raportul evidențiază beneficiile aduse de digitalizare în domeniul service-ului tehnic și analizează îmbunătățirile aduse procesului de gestionare a reparațiilor.

Pe parcursul raportului, vor fi prezentate tehnologiile utilizate, arhitectura sistemului, modul de organizare a bazei de date și implementarea funcționalităților cheie ale aplicației. De asemenea, se va analiza eficiența sistemului, problemele întâmpinate în timpul dezvoltării și posibilele îmbunătățiri viitoare.

Într-o lume în care eficiența și rapiditatea în gestionarea problemelor tehnice sunt esențiale, utilizarea unei soluții informatice specializate devine o necesitate. Prin acest raport, se urmărește evidențierea modului în care un software personalizat poate îmbunătăți procesele operaționale dintr-o companie de service IT, reducând erorile umane și oferind o mai bună organizare a informațiilor.

1.2. Importanța digitalizării proceselor de service tehnic

În prezent, digitalizarea este un element esențial în dezvoltarea oricărei industrii, iar domeniul service-ului tehnic nu face excepție. Gestionarea manuală a solicitărilor de reparații, utilizarea fișelor de hârtie și a înregistrărilor disparate pot duce la pierderi de timp, erori și lipsa unei evidențe clare asupra activităților desfășurate. Prin urmare, implementarea unei soluții informatice dedicate aduce multiple avantaje atât pentru companie, cât și pentru clienți.

Eficiența operațională și reducerea timpului de răspuns

Unul dintre cele mai importante beneficii ale digitalizării în domeniul service-ului tehnic este creșterea eficienței operaționale. O aplicație software permite gestionarea automată a cererilor de

service, reducând astfel timpul necesar pentru înregistrarea, procesarea și alocarea resurselor pentru fiecare reparație. Astfel, tehnicienii pot avea acces rapid la informațiile necesare, fără a fi nevoiți să caute în registre fizice sau să verifice manual fiecare solicitare.

În plus, utilizarea unei baze de date centralizate asigură o organizare eficientă a informațiilor, permițând acces instantaneu la istoricul reparațiilor pentru fiecare client. Acest lucru facilitează diagnosticarea problemelor recurente și permite oferirea unor soluții rapide și eficiente.

Reducerea erorilor umane și creșterea preciziei

Erorile umane sunt frecvente în procesele de gestionare manuală a solicitărilor. Informațiile incomplete, pierderea fișelor de service sau confuziile legate de statusul unei reparații pot duce la întârzieri și nemulțumiri din partea clienților. Prin intermediul unei aplicații informatice, fiecare solicitare este înregistrată într-un mod structurat, cu date precise despre problemă, client și intervențiile realizate.

Un alt avantaj major este posibilitatea de a seta notificări automate pentru a reaminti tehnicienilor și clienților despre statusul reparațiilor. Astfel, se elimină riscul ca o solicitare să fie uitată sau întârziată din cauza unei omisiuni.

Accesibilitate și monitorizare în timp real

O aplicație software permite accesul rapid la date de oriunde, permițând managerilor și tehnicienilor să monitorizeze activitatea în timp real. Aceasta este o funcționalitate esențială mai ales în cazul companiilor cu mai multe puncte de service sau cu echipe mobile de intervenție. Prin digitalizare, se poate urmări exact cine a preluat o solicitare, cât timp a durat reparația și care este statusul actual al fiecărei cereri.

Monitorizarea în timp real permite o mai bună coordonare a resurselor și o alocare eficientă a tehnicienilor în funcție de volumul de muncă. De asemenea, clienții pot fi informați rapid cu privire la progresul reparației, ceea ce crește satisfacția acestora.

Automatizarea procesului de generare a rapoartelor

Un alt avantaj semnificativ al digitalizării este posibilitatea de a genera automat rapoarte detaliate despre activitatea companiei. Aceste rapoarte pot include informații despre numărul de solicitări procesate, tipurile de probleme întâlnite, durata medie a unei reparații și gradul de satisfacție al clienților.

Aceste date sunt esențiale pentru îmbunătățirea continuă a serviciilor oferite. Pe baza rapoartelor, managerii pot identifica problemele recurente, pot optimiza procesul de reparație și pot lua decizii strategice pentru creșterea eficienței și rentabilității companiei.

Impactul asupra experienței clienților

Digitalizarea procesului de service tehnic nu aduce beneficii doar companiei, ci și clienților. Prin utilizarea unei aplicații informatice, aceștia pot beneficia de:

- **Timp de așteptare redus** – cererile sunt procesate mai rapid, iar reparațiile sunt efectuate într-un timp mai scurt.
- **Transparență** – clienții pot primi actualizări automate despre statusul reparației, eliminând astfel incertitudinea și necesitatea apelurilor telefonice frecvente.
- **Accesibilitate** – posibilitatea de a trimite cereri online sau de a verifica statusul reparației direct dintr-o platformă dedicată.

Prezentarea companiei Halley KM

2.1. Domeniul de activitate

Compania **Halley KM** activează în domeniul **service-ului IT și al reparațiilor tehnice**, oferind soluții profesionale pentru diagnosticarea, întreținerea și repararea echipamentelor IT, precum **calculatoare, laptopuri, imprimante și alte dispozitive electronice**. Cu o experiență semnificativă în industrie, Halley KM și-a consolidat poziția ca un furnizor de servicii fiabil, adaptându-se constant la noile tehnologii și cerințe ale pieței.

Activitatea companiei se axează pe **identificarea și remedierea problemelor hardware și software**, atât pentru **persoane fizice**, cât și pentru **firmе și instituții** care necesită întreținerea periodică a echipamentelor informatice. Într-o eră digitalizată, unde funcționalitatea echipamentelor IT este esențială pentru buna desfășurare a activităților, compania Halley KM oferă suport rapid și eficient pentru a minimiza timpul de nefuncționare al dispozitivelor clienților săi.



Fig.1.1

2.2. Structura și organizarea companiei

Halley KM este organizată într-un mod eficient pentru a asigura o gestionare optimă a cererilor de service și a intervențiilor tehnice. Structura companiei include mai multe departamente și funcții esențiale, fiecare având un rol bine definit în desfășurarea activității:

- **Departamentul de Recepție și Asistență Clienți**
 - Preia solicitările de service de la clienți și oferă informații despre costuri și termene de reparație.
 - Oferă suport tehnic de bază și îndrumă clienții spre soluții rapide dacă problema poate fi remediată fără intervenție fizică.
- **Departamentul Tehnic (Service IT)**
 - Responsabil cu diagnosticarea și repararea echipamentelor IT.
 - Include tehnicieni specializați pe diferite categorii de reparații: hardware, software, imprimante, rețelistică.
 - Efectuează testări înainte și după reparație pentru a asigura funcționarea corectă a echipamentului.
- **Departamentul de Management și Administrare**
 - Coordonează activitatea companiei și alocă resursele necesare pentru desfășurarea operațiunilor.
 - Gestionează achizițiile de piese și componente pentru reparații.
 - Monitorizează performanța echipei și implementează strategii de îmbunătățire a serviciilor.
- **Departamentul de Facturare și Logistică**
 - Se ocupă cu emiterea facturilor și gestionarea plăților pentru serviciile oferite.

- Organizează livrările și returnarea echipamentelor reparate către clienți.

Compania este organizată astfel încât să ofere **un flux de lucru eficient**, reducând timpul de așteptare al clienților și optimizând intervențiile tehnice. De asemenea, utilizarea unui **sistem digital de gestionare a cererilor de reparație** permite monitorizarea în timp real a statusului fiecărei solicitări.

2.3. Serviciile oferite în domeniul reparațiilor IT

Halley KM oferă **o gamă variată de servicii IT**, menite să acopere atât problemele hardware, cât și pe cele software, pentru **calculatoare, laptopuri și imprimante**.

Reparații hardware pentru PC și laptopuri

- **Diagnosticare hardware** – identificarea problemelor legate de componentele interne ale calculatorului sau laptopului.
- **Înlocuire și reparație componente** – reparația sau înlocuirea **plăcilor de bază, procesoarelor, memoriei RAM, surselor de alimentare, hard disk-urilor și SSD-urilor**.
- **Curățare și întreținere** – eliminarea prafului și a depunerilor de pe componente pentru prevenirea supraîncălzirii și îmbunătățirea performanței.
- **Reparații ecrane laptop** – înlocuirea display-urilor sparte sau defecte.

Reparații software pentru calculatoare și laptopuri

- **Instalare și configurare sisteme de operare** – Windows, Linux, macOS.
- **Eliminare viruși și malware** – scanare și curățare completă pentru protejarea datelor.
- **Recuperare date** – salvarea fișierelor pierdute de pe hard disk-uri și alte dispozitive de stocare.
- **Optimizare performanță** – creșterea vitezei de funcționare a dispozitivului prin ajustarea setărilor și curățarea fișierelor inutile.

Service imprimante și echipamente de birou

- **Diagnosticare și depanare imprimante** – identificarea și remedierea problemelor legate de alimentarea cu hârtie, erorile de imprimare sau problemele de conectivitate.
- **Reparații și întreținere imprimante laser și inkjet** – înlocuirea cartușelor, reumplerea tonerului și curățarea componentelor interne.

- **Configurare rețea pentru imprimante** – instalare și setare pentru conectarea imprimantei la rețea, astfel încât să poată fi utilizată de mai multe dispozitive.

Suport IT pentru companii și instituții

- **Mentenanță periodică pentru echipamente IT** – verificarea și întreținerea calculatoarelor și imprimantelor pentru a preveni defecțiunile.
- **Configurare și administrare rețele** – instalarea și gestionarea conexiunilor de rețea pentru birouri și spații comerciale.
- **Soluții de securitate IT** – implementarea programelor antivirus, firewall-urilor și măsurilor de protecție a datelor.

Avantajele serviciilor oferite de Halley KM

Timp de răspuns rapid – diagnosticare și reparație într-un timp cât mai scurt.

Tehnicienii calificați – specialiști cu experiență în reparațiile IT.

Piese de schimb originale – utilizarea componentelor de calitate pentru o durabilitate sporită.

Garanție pentru serviciile efectuate – reparațiile sunt acoperite de o garanție, oferind siguranță clienților.

Soluții personalizate – adaptarea serviciilor în funcție de nevoile fiecărui client.

Prin aceste servicii, compania Halley KM își propune să ofere **soluții eficiente, fiabile și accesibile** pentru întreținerea și repararea echipamentelor IT, contribuind la optimizarea activităților clienților săi.

Descrierea aplicației „Înregistrarea clientului cu probleme tehnice”

În cadrul activităților desfășurate de compania Halley KM, gestionarea eficientă a solicitărilor de service este esențială pentru asigurarea unui flux de lucru optim și pentru creșterea satisfacției clienților. Aplicația „Înregistrarea clientului cu probleme tehnice” a fost dezvoltată pentru a facilita procesul de preluare, organizare și monitorizare a cererilor de reparații pentru imprimante și calculatoare. Aceasta oferă o soluție digitalizată care elimină necesitatea utilizării unor metode tradiționale precum înregistrarea manuală a solicitărilor pe hârtie, reducând astfel erorile și timpul de procesare.

3.1. Obiectivele aplicației

Scopul principal al aplicației este de a automatiza și optimiza procesul de înregistrare și gestionare a solicitărilor de reparații primite de compania Halley KM. Aceasta își propune să ofere un sistem centralizat în care toate informațiile legate de clienți și echipamentele lor defecte să fie stocate și accesibile în timp real pentru personalul tehnic și administrativ.

Obiectivele specifice ale aplicației includ:

1. **Îmbunătățirea gestionării cererilor de reparații**
 - Crearea unei baze de date structurate pentru toate solicitările primite.
 - Organizarea eficientă a cererilor în funcție de prioritate, tipul problemei și statusul reparației.
2. **Reducerea timpului de răspuns și creșterea productivității echipei**
 - Oferirea unei platforme intuitive care să permită tehnicienilor acces rapid la detaliile fiecărui caz.
 - Automatizarea procesului de alocare a cazurilor către specialiștii disponibili.
3. **Creșterea transparenței și îmbunătățirea comunicării**
 - Posibilitatea de a vizualiza în timp real progresul fiecărei solicitări.
 - Notificarea automată a clienților privind statusul reparației.
4. **Stocarea și analizarea datelor pentru îmbunătățirea serviciilor**
 - Generarea de rapoarte detaliate asupra tipurilor de probleme întâlnite și a timpului de soluționare.
 - Identificarea celor mai frecvente defecțiuni pentru optimizarea stocului de piese și echipamente.

3.2. Funcționalitățile principale

Aplicația include mai multe funcționalități esențiale, menite să sprijine atât personalul companiei, cât și clienții, oferind un sistem bine organizat pentru gestionarea solicitărilor de service.

1. **Înregistrarea solicitărilor de reparație**
 - Clienții pot introduce detalii despre echipamentul defect și natura problemei.
 - Sistemul permite atașarea de imagini sau documente relevante.
2. **Generarea automată a unui număr de înregistrare pentru fiecare solicitare**
 - Fiecare cerere primește un ID unic pentru urmărire și referință ulterioară.

3. Categorișirea și prioritizarea cererilor

- Solicitățile sunt grupate în funcție de tipul dispozitivului (PC, laptop, imprimantă) și nivelul de urgență.

4. Actualizarea statusului reparației

- Aplicația permite modificarea statusului solicitării (În așteptare, În proces, Finalizat, Anulat).
- Notificările sunt trimise automat către client în momentul în care se înregistrează o schimbare a statusului.

5. Gestionarea bazei de date a clienților și echipamentelor

- Păstrarea unui istoric detaliat al tuturor reparațiilor efectuate pentru fiecare client.
- Posibilitatea de a accesa informații despre problemele recurente ale unui anumit echipament.

6. Facturare și generare rapoarte

- Sistemul permite generarea automată a facturilor pentru serviciile prestate.
- Crearea de rapoarte statistice pentru analiza eficienței echipei și identificarea tendințelor privind problemele tehnice. Tehnologii utilizate

În dezvoltarea aplicației „Înregistrarea clientului cu probleme tehnice”, au fost utilizate mai multe tehnologii moderne, fiecare contribuind la eficiența și performanța aplicației. În continuare, vom analiza limbajul de programare Python, baza de date SQL, în special SQLite, precum și bibliotecile și framework-urile utilizate pentru a crea o interfață prietenoasă și ușor de utilizat.

4.1. Limbajul de programare Python

Python este limbajul de programare ales pentru dezvoltarea aplicației datorită simplității sale, versatilității și sprijinului comunității. De-a lungul decadelor, Python s-a impus ca unul dintre cele mai populare și utilizate limbaje pentru dezvoltarea rapidă a aplicațiilor software, având o sintaxă clară și ușor de învățat.

1. Ușurința în dezvoltare

Python se remarcă prin sintaxa sa simplă și intuitivă, care permite dezvoltarea rapidă a aplicațiilor. Aceasta înseamnă că dezvoltatorii pot scrie cod eficient, fără a fi necesar să se concentreze excesiv pe detalii complexe. De asemenea, Python permite utilizarea unui număr mare de biblioteci externe, care pot extinde funcționalitățile aplicațiilor.

2. **Flexibilitate și scalabilitate**

Python este un limbaj care poate fi folosit atât pentru aplicații mici și prototipuri, cât și pentru sisteme complexe. Acesta permite integrarea ușoară cu diferite baze de date, sisteme externe și tehnologii, lucru esențial pentru aplicațiile care necesită scalabilitate și funcționalități avansate.

3. **Biblioteci pentru dezvoltare rapidă**

Python dispune de o gamă largă de biblioteci și framework-uri care facilitează dezvoltarea aplicațiilor. În cazul aplicației noastre, Python a fost utilizat pentru a gestiona logica aplicației și interacțiunile cu baza de date, iar bibliotecile externe au fost folosite pentru a crea interfața grafică a utilizatorului (GUI) și pentru a manipula datele.

4. **Suport pentru baze de date SQL**

Python oferă o integrare excelentă cu bazele de date relaționale, în special cu SQL, prin biblioteci precum `sqlite3`. Acest lucru este esențial pentru aplicația „Înregistrarea clientului cu probleme tehnice”, unde stocarea și gestionarea datelor clienților și a solicitărilor de reparație sunt critice.

4.2. Baza de date SQL – Structură și modelare

În aplicația „Înregistrarea clientului cu probleme tehnice”, baza de date SQL joacă un rol crucial în stocarea și gestionarea datelor. Pentru această aplicație, a fost aleasă utilizarea SQLite, un sistem de gestionare a bazelor de date relaționale (RDBMS) ușor de implementat și gestionat, care se integrează perfect cu Python.

1. **SQLite**

SQLite este un RDBMS compact, care nu necesită un server separat pentru a funcționa, ceea ce îl face ideal pentru aplicații mici și mijlocii. Acesta este inclus ca bibliotecă în Python, facilitând accesul rapid la baze de date. În plus, SQLite este o alegere bună pentru aplicațiile care nu au nevoie de complexitatea unei baze de date mari sau distribuite.

2. **Structura bazei de date**

Baza de date este organizată într-o manieră logică pentru a facilita stocarea și gestionarea informațiilor despre clienți și solicitările de reparație. Aceasta include mai multe tabele, fiecare având un rol specific în procesul de înregistrare și procesare a cererilor de reparație. Exemple de tabele includ:

- **Tabela „Clienți”:** Stocază informațiile esențiale ale clientului, cum ar fi numele, adresa, numărul de telefon și email.

- **Tabela „Solicitari”**: Conține informațiile despre fiecare solicitare de reparație, inclusiv ID-ul clientului, tipul echipamentului, descrierea problemei și data solicitată pentru reparație.
- **Tabela „Reparatii”**: Detaliază statusul fiecărei reparații, numele tehnicianului responsabil, piesele folosite și data finalizării.

3. Modelarea datelor

Procesul de modelare a bazei de date a fost realizat pentru a asigura relaționarea corectă a datelor între tabele și pentru a permite interogări eficiente. Astfel, am utilizat relații de tipul „1 la N” între tabele, de exemplu, un client poate avea mai multe solicitări de reparație, dar fiecare solicitare este asociată unui singur client. Structura bazei de date a fost gândită astfel încât să ofere performanță ridicată și să permită extinderea ușoară a aplicației pe măsură ce numărul de clienți și solicitări crește.

4. Interogări SQL

Aplicația utilizează diverse interogări SQL pentru a adăuga, actualiza și extrage date din bază, incluzând operațiuni de inserare a noilor solicitări, actualizarea statusului acestora și crearea de rapoarte detaliate asupra progresului reparațiilor. Exemple de interogări includ:

- **Inserare solicitări noi**: `INSERT INTO Solicitari (id_client, tip_echipament, descriere_problema) VALUES (?, ?, ?);`
- **Actualizare status reparație**: `UPDATE Reparatii SET status = ? WHERE id_solicitare = ?;`
- **Interogare status reparație**: `SELECT * FROM Reparatii WHERE status = 'Finalizat';`

4.3. Biblioteci și framework-uri utilizate

Pentru a crea o aplicație ușor de utilizat și intuitivă, am folosit mai multe biblioteci și framework-uri Python, fiecare având un rol specific în construirea interfeței grafice și a funcționalităților aplicației.

1. sqlite3

Biblioteca `sqlite3` este utilizată pentru a integra aplicația Python cu baza de date SQLite. Aceasta oferă o modalitate ușoară de a interacționa cu bazele de date SQL, permițând crearea tabelelor, executarea interogărilor și manipularea rezultatelor. `sqlite3` este esențială pentru gestionarea datelor utilizatorilor și a cererilor de reparație în aplicația noastră.

2. Tkinter

Tkinter este biblioteca de bază a Python pentru construirea interfeței grafice a utilizatorului (GUI). Aceasta permite crearea de feronerie vizuală, precum feronerie de introducere a

datelor, butoane, etichete, câmpuri de text și alte elemente grafice. Tkinter este ușor de utilizat și nu necesită biblioteci externe suplimentare pentru a construi o aplicație simplă și eficientă. În cadrul aplicației, Tkinter este folosit pentru a crea formulare pentru înregistrarea solicitărilor și a vizualiza rapoartele generate.

3. **ttkbootstrap**

ttkbootstrap este un framework care extinde funcționalitățile Tkinter, oferind un set mai avansat de componente vizuale și un aspect modern pentru aplicațiile GUI. Utilizând ttkbootstrap, aplicația a dobândit un aspect mai estetic și un design coerent, cu teme predefinite care îmbunătățesc experiența utilizatorului. Acesta a fost utilizat pentru a crea interfețe de utilizator mai rafinate, cu butoane, feronerie și meniuri personalizabile.

4. **Alte biblioteci și instrumente auxiliare**

Pe lângă bibliotecile principale, au fost utilizate și alte instrumente Python pentru optimizarea performanței și a procesului de dezvoltare. De exemplu, pentru gestionarea erorilor și a excepțiilor, au fost implementate tehnici standard Python, iar pentru gestionarea fișierelor externe, s-au folosit funcționalități built-in de citire și scriere a fișierelor.

```
import sqlite3
import ttkbootstrap as tb
from ttkbootstrap.constants import *
from tkinter import messagebox
from tkinter import ttk
```

Fig.2.1

Aceste tehnologii și instrumente, combinate într-o soluție integrată, oferă aplicației „Înregistrarea clientului cu probleme tehnice” o funcționalitate robustă și un design plăcut, având un impact semnificativ asupra eficienței operaționale ale companiei Halley KM.

Arhitectura aplicației

Arhitectura aplicației „Înregistrarea clientului cu probleme tehnice” este concepută pentru a răspunde nevoilor specifice ale companiei Halley KM, oferind o soluție eficientă pentru gestionarea cererilor de reparație și a datelor clienților. În această secțiune, vom descrie structura generală a aplicației, fluxul de date și interacțiunea utilizatorului, precum și modul în care sunt gestionate cererile de reparație.

5.1. Structura generală a aplicației

Aplicația este organizată într-o arhitectură modulară, care facilitează întreținerea, extinderea și gestionarea eficientă a codului. Structura generală include mai multe componente esențiale, care sunt conectate între ele pentru a asigura o funcționare fluidă:

1. Interfața Grafică a Utilizatorului (GUI)

Aplicația utilizează Tkinter pentru a crea o interfață vizuală prietenoasă, iar framework-ul ttkbootstrap este folosit pentru a îmbunătăți aspectul grafic și pentru a oferi un design mai modern. Interfața este alcătuită din mai multe feronerie (formulare), fiecare având un rol specific, precum:

- Formulare pentru introducerea informațiilor clienților (nume, contact, echipament etc.)
- Meniuri și butoane pentru navigarea prin aplicație
- Rapoarte vizuale care permit tehnicienilor și altor utilizatori să vizualizeze solicitările și să actualizeze statusul acestora.

2. Logica Aplicației

Partea logică a aplicației este implementată în Python și gestionează procesele de manipulare a datelor. Aceasta include validarea informațiilor introduse, interacțiunea cu baza de date SQL pentru stocarea și extragerea datelor, precum și actualizarea statusului cererilor de reparație.

Logica aplicației controlează interacțiunea utilizatorilor cu formularele și actualizează informațiile în baza de date pe măsură ce utilizatorii introduc sau modifică datele.

3. Baza de date SQLite

Aplicația utilizează o bază de date SQLite pentru a stoca informațiile despre clienți, solicitările de reparație și statusul acestora. Baza de date este accesată și manipulată de aplicație prin intermediul bibliotecii sqlite3 din Python. Fiecare solicitare de reparație este

asociată unui client, iar datele sunt structurate în tabele interconectate pentru a asigura integritatea și performanța operațiunilor de interogare.

4. Interacțiunea între componente

- **Utilizatorul** interacționează cu aplicația prin intermediul GUI-ului, introducând date despre clienți și cereri de reparație.
- **Logica aplicației** validează datele și le trimite către baza de date, unde sunt stocate și gestionate.
- **Baza de date SQLite** primește și păstrează informațiile, asigurându-se că sunt accesibile rapid pentru interogări ulterioare.

5.2. Fluxul de date și interacțiunea utilizatorului

Fluxul de date în aplicație urmează un proces clar definit, care permite interacțiunea eficientă între utilizatori și sistemul de gestionare al cererilor de reparație. Procesul include mai multe etape, fiecare având un rol esențial:

1. Înregistrarea unui client

- Utilizatorul accesează interfața grafică a aplicației și completează formularul pentru înregistrarea unui client nou. Acesta va furniza informații precum numele, adresa, numărul de telefon și detalii despre echipamentul care necesită reparație.
- Aplicația validează datele introduse pentru a se asigura că sunt complete și corecte. În caz contrar, va afișa un mesaj de eroare pentru a ghida utilizatorul să corecteze informațiile.
- După validare, aplicația trimite datele către baza de date SQLite pentru a înregistra clientul într-o tabelă dedicată.

2. Înregistrarea cererii de reparație

- După înregistrarea unui client, utilizatorul poate adăuga o cerere de reparație. În formularul aferent cererii, utilizatorul va specifica tipul echipamentului, descrierea problemei și data dorită pentru reparație.
- Aplicația validează informațiile introduse și le trimite în baza de date, asociindu-le cu ID-ul clientului corespunzător.
- În această etapă, sunt generate automat notificări (în cazul în care aplicația este configurată pentru a le trimite) care informează tehnicianul sau personalul relevant despre solicitarea de reparație.

3. Actualizarea și gestionarea cererilor de reparație

- Tehnicienii sau utilizatorii autorizați pot vizualiza toate cererile de reparație înregistrate în aplicație printr-un raport centralizat. Aceste rapoarte pot fi filtrate în funcție de statusul reparației (de exemplu, „În așteptare”, „În curs de desfășurare”, „Finalizat”).

- Când o cerere este procesată, tehnicianul poate actualiza statusul acesteia în aplicație, iar schimbările sunt sincronizate în timp real cu baza de date.
- Utilizatorii autorizați pot, de asemenea, adăuga note suplimentare despre reparație, actualizând datele stocate și asigurându-se că toți participanții la proces au informații actualizate.

4. **Generarea rapoartelor finale**

După finalizarea unei reparații, aplicația poate genera un raport detaliat pentru fiecare client, care include informații despre reparațiile efectuate, piesele schimbate, costurile asociate și data finalizării lucrării. Aceste rapoarte sunt stocate și pot fi accesate de utilizatori pentru consultare ulterioară.

5.3. **Modul de gestionare a cererilor de reparație**

Gestionarea cererilor de reparație este un proces esențial în cadrul aplicației „Înregistrarea clientului cu probleme tehnice”. Aplicația oferă o metodologie bine structurată pentru a asigura că fiecare cerere este tratată corespunzător, iar statusul acestora este actualizat în mod eficient:

1. **Înregistrarea cererii**

Fiecare cerere de reparație este înregistrată în baza de date cu un set detaliat de informații. Aceste informații includ ID-ul clientului, tipul echipamentului, descrierea problemei și data solicitată pentru reparație. Fiecare cerere este asociată unui status inițial, de obicei „În așteptare”.

2. **Închiderea cererii**

După finalizarea reparației și la solicitarea tehnicianului, cererea de reparație este marcată ca fiind „Finalizată”. Aplicația poate trimite automat un raport de închidere a cererii, care include toate informațiile necesare despre lucrare, piesele schimbate și costurile aferente.

Prin implementarea unui astfel de flux de gestionare a cererilor de reparație, aplicația asigură o eficiență sporită, evitând pierderea datelor și facilitând comunicarea între departamentele implicate în procesul de reparație.

Hally KM - Gestionare Clienti
Fișier

Top
Instalare camere
Reparații calculator

ID	Nume	Prenume	Întrebare	Descriere	Telefon	Adresă	Data/Ora	Importanță	Servicii
2	makan	paran	nu stie sa serati	trebuie dahua banii	0404	hz	2025-03-03 11:43:57	Ridicat	Reparații calculat
3	Trion	companie	laptop	not bootable device	--	--	2025-03-04 11:16:12	Mediu	Reparații calculat
4	Nuca	Miric	printer	scoate intruna nu se o	069142073	-	2025-03-04 11:20:24	Mediu	Reparații calculat
5	primaria	Cucoara	printer	scoate pina la jumate	-	-	2025-03-04 11:21:03	Mediu	Reparații calculat
6	jenshina	v ciomom	cartuse	2 cartus incarcat , 1 d	-	-	2025-03-04 14:34:58	Mediu	Reparații calculat

Caută
Total clienți: 5
+ Adaugă Client
Sterge Client
Mod Intuneat

Fig.3.1

Implementarea funcționalităților

Aplicația „Înregistrarea clientului cu probleme tehnice” a fost dezvoltată pentru a îmbunătăți eficiența procesului de gestionare a cererilor de reparație și a datelor clienților, respectând cerințele companiei Halley KM. În această secțiune, vom descrie implementarea funcționalităților principale ale aplicației, inclusiv modulul de înregistrare a clienților, modulul de diagnosticare a problemelor tehnice, modulul de gestionare a reparațiilor și istoricul intervențiilor, precum și modulul de generare a rapoartelor.

6.1. Modulul de înregistrare a clienților

Modulul de înregistrare a clienților este un element esențial al aplicației, având rolul de a aduna și organiza informațiile de bază ale clienților care au nevoie de servicii de reparație. Scopul principal al acestui modul este de a crea o bază de date centralizată care să permită accesul rapid și ușor la informațiile clienților, pentru a putea iniția cu ușurință cereri de reparație.

Fluxul de funcționare al modului:

1. Introducerea datelor clientului

Utilizatorul (operator sau administrator) completează un formular în care introduce informații de bază ale clientului, cum ar fi:

- Numele complet
- Adresa de domiciliu
- Numărul de telefon
- Adresa de e-mail
- Detalii despre echipamentul care necesită reparație (tipul, marca, modelul).

2. Validarea datelor introduse

La completarea formularului, aplicația validează datele pentru a se asigura că informațiile sunt complete și corecte. Dacă anumite câmpuri sunt incomplete sau incorecte (de exemplu, un număr de telefon invalid), aplicația va afișa mesaje de eroare pentru a ghida utilizatorul să corecteze datele.

3. Stocarea datelor în baza de date

După validarea datelor, aplicația trimite informațiile clientului către baza de date SQLite. Datele sunt stocate într-o tabelă dedicată clienților, iar fiecare client primește un ID unic, care va fi folosit ulterior pentru a asocia cererile de reparație.

4. Actualizarea și căutarea clienților

Clienții pot fi căutați rapid în baza de date, folosind diverse filtre (nume, număr de telefon, echipament). Aceste funcționalități facilitează găsirea rapidă a datelor în cazul în care este necesar să fie modificat un set de informații sau să fie accesate cereri anterioare de reparație.

Beneficii:

- Centralizarea informațiilor despre clienți
- Acces rapid la datele clienților, esențial pentru un timp de răspuns mai rapid
- Reducerea erorilor și a timpului de procesare datorită validării automate a datelor

6.2. Modulul de diagnosticare a problemelor tehnice

Modulul de diagnosticare a problemelor tehnice este crucial pentru eficiența procesului de reparație. Acesta permite tehnicienilor și operatorilor să identifice rapid natura problemelor tehnice ale echipamentelor și să creeze o bază de informații clară pentru intervenția ulterioară. În această secțiune, tehnicienii pot adăuga detalii despre defecțiunile observate și posibilele soluții.

Fluxul de funcționare al modulului:

1. Înregistrarea detaliilor problemei

Atunci când un client adresează o cerere de reparație, technicianul completează un formular detaliat în care descrie problema tehnică observată la echipament. Formularul poate include:

- Descrierea simptomelor
- Posibile cauze ale defecțiunii
- Testele efectuate pentru diagnosticare
- Recomandări pentru soluționare (de exemplu, schimbarea unui anumit component, reinstalarea unui software, etc.).

2. Categorii de probleme tehnice

Pentru a eficientiza diagnosticarea, aplicația poate include un sistem de categorii pentru problemele tehnice. De exemplu, pentru imprimante, se poate include o categorie de „defecțiuni la nivel de hardware” și „defecțiuni la nivel de software”. Aceste categorii ajută technicianul să diagnosticheze mai rapid și să găsească soluții mai precise.

3. Interacțiunea cu istoricul clientului

Modulul permite technicianului să acceseze istoricul reparațiilor anterioare ale clientului, pentru a vedea dacă problema diagnosticată este una recurentă sau legată de o defecțiune mai veche. Acesta poate adăuga observații suplimentare în istoricul reparației, care vor fi utile în viitoarele intervenții.

Beneficii:

- Reducerea timpilor de intervenție datorită sistemului organizat de diagnosticare
- Acces rapid la istoricul reparațiilor pentru clienți
- asupra stării fiecărei reparații, îmbunătățind comunicarea internă

Testare și optimizare

Testarea și optimizarea sunt etape esențiale în procesul de dezvoltare a aplicațiilor software, deoarece acestea asigură că aplicația funcționează corect și eficient, răspunzând cerințelor utilizatorilor. În această secțiune, vom discuta scenariile de testare realizate pentru aplicația „Înregistrarea clientului cu probleme tehnice”, problemele întâmpinate în timpul procesului de dezvoltare și soluțiile aplicate, precum și pașii întreprinși pentru optimizarea performanței aplicației și a bazei de date.

7.1. Scenarii de testare

Testarea aplicației a fost realizată în mai multe etape, incluzând testele funcționale, testele de integrare și testele de performanță. Scenariile de testare au fost dezvoltate pentru a valida fiecare modul al aplicației, inclusiv înregistrarea clienților, diagnosticarea problemelor tehnice, gestionarea cererilor de reparație și generarea rapoartelor.

Testarea funcționalităților aplicației:

1. Testarea înregistrării clienților:

- **Scenariul 1:** Introducerea unui client cu toate datele corecte (nume, adresă, telefon, email) și verificarea că informațiile sunt salvate corect în baza de date.
- **Scenariul 2:** Introducerea unui client cu un număr de telefon invalid și verificarea faptului că aplicația va emite un mesaj de eroare.
- **Scenariul 3:** Verificarea că, în cazul în care toate câmpurile obligatorii nu sunt completate, utilizatorul este notificat corespunzător.

2. Testarea diagnosticării problemelor tehnice:

- **Scenariul 1:** Înregistrarea unei cereri de reparație și diagnosticarea unei probleme (de exemplu, „imprimanta nu mai imprimă”) și verificarea că statusul cererii de reparație se actualizează corect.
- **Scenariul 2:** Testarea completării formularului de diagnosticare cu date incomplete și verificarea că aplicația nu permite trimiterea formularului până la completarea tuturor câmpurilor obligatorii.
- **Scenariul 3:** Verificarea că tehnicianul poate adăuga observații suplimentare în istoricul reparațiilor.

3. Testarea gestionării cererilor de reparație:

- **Scenariul 1:** Verificarea procesului de alocare a unui tehnician la o cerere de reparație și actualizarea statusului cererii în timp real.
- **Scenariul 2:** Testarea fluxului de închidere a unei cereri de reparație și confirmarea că toate datele relevante sunt înregistrate în istoricul reparațiilor.

4. Testarea generării rapoartelor:

- **Scenariul 1:** Generarea unui raport cu toate cererile de reparație într-o perioadă de timp specificată și verificarea că datele sunt corect afișate (status, tip de problemă, nume client).
- **Scenariul 2:** Testarea generării unui raport al performanței tehnicienilor și verificarea că datele sunt corect sumarizate.

Testarea integrării bazei de date și interfața utilizatorului:

- Testarea integrării între aplicația frontend (interfața utilizatorului) și backend (baza de date) pentru a asigura că informațiile se transferă corect.
- Testarea interacțiunii dintre diferitele module ale aplicației, inclusiv procesul complet de înregistrare, diagnosticare și gestionare a reparațiilor, pentru a confirma că fluxul de date este coerent și consistent.

7.2. Probleme întâmpinate și soluții aplicate

În timpul testării aplicației, au fost întâmpinate mai multe probleme care au fost rezolvate pe parcursul procesului de dezvoltare. Aceste probleme au fost legate atât de logica aplicației, cât și de performanța bazei de date.

1. Probleme de validare a datelor:

- **Problemă:** Uneori, aplicația nu valida corect anumite câmpuri, cum ar fi numărul de telefon sau adresa de email. De asemenea, câmpurile obligatorii nu erau gestionate corespunzător.
- **Soluție:** A fost implementată o funcționalitate de validare riguroasă pentru fiecare câmp, folosind expresii regulate pentru a verifica formatul corect al datelor (de exemplu, numere de telefon sau emailuri valide). De asemenea, am adăugat mesaje de eroare clare pentru utilizatori, pentru a-i ghida în completarea corectă a formularului.

2. Probleme de sincronizare a datelor între frontend și backend:

- **Problemă:** Uneori, datele introduse în interfața utilizatorului nu se sincronizau corect cu baza de date, ceea ce ducea la inexactități.
- **Soluție:** Am optimizat codul de interacțiune între frontend și backend, asigurându-mă că datele sunt trimise corect prin interfețele API și că aplicația așteaptă confirmarea procesării datelor înainte de a trece la următoarea acțiune.

3. Probleme de performanță a bazei de date:

- **Problemă:** La volume mari de date, aplicația devenea lentă la interogarea bazei de date, mai ales în cazul generării rapoartelor.
- **Soluție:** Am optimizat interogările SQL prin adăugarea de indecși pentru câmpurile cele mai accesate și am simplificat structura tabelor pentru a reduce timpul de răspuns. De asemenea, am implementat funcționalități de cache pentru rapoarte frecvent utilizate, ceea ce a redus semnificativ timpul de generare a acestora.

4. Probleme de compatibilitate a interfeței cu diverse rezoluții ale ecranelor:

- **Problemă:** Interfața aplicației nu era complet adaptivă la diferite rezoluții de ecran, ceea ce ducea la probleme de vizualizare pe dispozitive mobile sau pe ecrane mai mici.

- **Soluție:** Am folosit framework-ul tkinter și ttkbootstrap pentru a crea o interfață mai flexibilă și mai adaptabilă la diverse dimensiuni de ecran, implementând elemente responsive care se ajustează automat.

7.3. Optimizarea performanței bazei de date și codului

Optimizarea performanței aplicației a fost o prioritate pentru a asigura o experiență de utilizator fluidă, mai ales în condițiile în care baza de date ar putea crește pe măsură ce numărul cererilor de reparație și clienților crește.

1. Optimizarea bazei de date:

- **Indexarea tabelor:** Am adăugat indici pe coloanele care sunt cel mai frecvent utilizate pentru căutare (de exemplu, nume_client, status_reparatie, data_reparatie). Aceasta a permis îmbunătățirea semnificativă a vitezei interogărilor de căutare și filtrare.
- **Normalizarea bazei de date:** Am revizuit structura tabelor pentru a elimina redundanțele și pentru a asigura că datele sunt organizate într-o formă normalizată, ceea ce a redus erorile și a îmbunătățit performanța.
- **Optimizarea interogărilor SQL:** Am refactorizat interogările complexe, folosind JOIN-uri eficiente și sub-interogări pentru a minimiza numărul de accesări ale bazei de date. În plus, am utilizat proceduri stocate pentru a executa mai multe operațiuni deodată într-un singur apel.

2. Optimizarea codului aplicației:

- **Refactorizarea funcțiilor:** Am revizuit și refactorizat funcțiile care erau folosite repetat în aplicație pentru a le face mai eficiente. De exemplu, am înlocuit buclele inutile și am simplificat logica condițională pentru a reduce timpul de execuție.
- **Utilizarea tehnologiilor de caching:** Am implementat caching pentru datele care nu se schimbă frecvent, cum ar fi lista de clienți sau tipurile de probleme tehnice, astfel încât aplicația să nu fie nevoită să facă interogări la baza de date pentru aceleași informații.
- **Îmbunătățirea performanței interfeței grafice:** Am optimizat interfața folosind elemente grafice mai ușoare și am redus utilizarea resurselor în timpul execuției aplicației, ceea ce a îmbunătățit performanța generală a aplicației.

Prin aceste măsuri, aplicația a devenit mai rapidă și mai eficientă, oferind o experiență de utilizare mai plăcută și facilitând gestionarea unui număr mare de cereri de reparație fără a afecta performanța aplicației.

Testarea și optimizarea sunt procese continue și iterative, iar îmbunătățirile făcute în timpul dezvoltării aplicației „Înregistrarea clientului cu probleme tehnice” au fost esențiale pentru asigurarea unei performanțe ridicate și a unei experiențe de utilizator fluide.

Concluzii și direcții de dezvoltare viitoare

În urma implementării aplicației „Înregistrarea clientului cu probleme tehnice” în cadrul companiei Halley KM, se poate concluziona că proiectul a avut un impact pozitiv asupra activității companiei, optimizând procesele interne și îmbunătățind gestionarea cererilor de reparație. Cu toate acestea, există posibilități de îmbunătățire și extindere a aplicației, ceea ce va contribui la creșterea eficienței companiei pe termen lung. În această secțiune, vom analiza evaluarea generală a proiectului, posibilele îmbunătățiri și extinderea funcționalităților, precum și perspectivele pentru integrarea aplicației cu alte sisteme.

8.1. Evaluarea generală a proiectului

Proiectul „Înregistrarea clientului cu probleme tehnice” a fost implementat cu succes în cadrul companiei Halley KM și a avut un impact semnificativ asupra modului în care compania gestionează cererile de reparație pentru imprimante și calculatoare. Aplicația a îndeplinit scopul principal de a digitaliza procesele administrative și operaționale, îmbunătățind eficiența și reducând timpul de procesare al cererilor.

1. **Automatizarea proceselor interne:** Unul dintre cele mai mari beneficii a fost automatizarea procesului de înregistrare a cererilor de reparație. În trecut, acest proces era realizat manual, iar acum, prin aplicație, a fost mult simplificat și rapidizat. Aceasta a dus la economisirea de timp prețios și a permis personalului să se concentreze pe sarcini mai importante, cum ar fi diagnosticarea problemelor tehnice și efectuarea reparațiilor.
2. **Gestionarea eficientă a datelor:** Aplicația a oferit un sistem eficient de gestionare a datelor clienților și a istoricului cererilor de reparație. Aceasta a eliminat riscul pierderii sau erorii înregistrărilor, îmbunătățind considerabil acuratețea informațiilor. Astfel, personalul a avut acces rapid și facil la datele necesare, ceea ce a contribuit la îmbunătățirea relației cu clienții.

Deși proiectul a fost un succes, există întotdeauna loc pentru îmbunătățire, iar în continuare vor fi implementate modificări pentru a răspunde cerințelor tot mai diverse ale utilizatorilor și pentru a face aplicația mai scalabilă.

8.2. Posibile îmbunătățiri și extinderea funcționalităților

În ciuda succesului obținut cu aplicația „Înregistrarea clientului cu probleme tehnice”, există mai multe direcții de îmbunătățire și extindere a funcționalităților, care pot contribui la creșterea performanței și adaptabilității aplicației pe termen lung.

1. **Integrarea cu platformele de notificare:** O îmbunătățire importantă ar fi integrarea aplicației cu platformele de notificare, pentru a permite trimiterea automată de alerte prin e-mail sau SMS clienților atunci când cererea lor de reparație este completă sau când intervin modificări ale statusului. Aceasta ar îmbunătăți comunicarea cu clienții și ar asigura o transparență mai mare în procesul de reparație.
2. **Aplicație mobilă pentru tehnicieni:** Deși aplicația este funcțională pe desktop, o variantă mobilă ar putea facilita accesul tehnicienilor la informațiile necesare direct de pe teren. Tehnicienii ar putea actualiza statusul reparațiilor și ar putea consulta istoricul clienților în timp real, ceea ce ar îmbunătăți semnificativ eficiența și viteza de răspuns. O aplicație mobilă ar adresa și nevoia de mobilitate și ar sprijini un flux de lucru mai rapid.
3. **Analize avansate ale datelor:** În viitor, compania Halley KM ar putea implementa un modul de analize avansate ale datelor, care să ajute la identificarea tendințelor și la generarea de rapoarte detaliate despre performanțele tehnicienilor, tipurile de reparații cele mai frecvente sau timpul mediu de reparație. Aceste rapoarte ar putea fi utilizate pentru optimizarea proceselor interne și pentru o gestionare mai bună a resurselor.
4. **Integrarea cu sistemele de management al stocurilor:** Un alt pas important în extinderea aplicației ar fi integrarea cu un sistem de management al stocurilor, care să permită urmărirea pieselor de schimb necesare pentru reparații și gestionarea stocurilor în timp real. Aceasta ar contribui la reducerea timpilor de așteptare pentru piesele de schimb și la optimizarea gestionării resurselor materiale.

În concluzie, aplicația „Înregistrarea clientului cu probleme tehnice” a avut un impact pozitiv semnificativ asupra proceselor interne ale companiei Halley KM, dar există o serie de îmbunătățiri și extinderi care pot fi implementate pentru a face aplicația mai scalabilă, mai eficientă și mai adaptată la nevoile în continuă schimbare ale companiei.

Caiet de sarcini Halley Soft

Nr. Ord.	Tematica lucrărilor preconizate	Termenele Planificate		Termenele Reale	
		Început / Sfârșit	Numărul de zile	Început / Sfârșit	Numărul de zile
1	Studierea C++ Builder	27.01 28.01	2	27.01 28.01	1
2	Inițierea în proiectul „ Administrarea Parbrizelor Auto,, Crearea bazei de date Fig.1	28.01 29.01	2	28.01 29.01	2
3	Crearea aplicației simple pentru desktop folosind C++ Builder Fig.2	30.01 05.02	5	30.01 05.02	5
4	Modificarea funcțiilor existente și adăugarea unor funcții noi pentru utilizare mai simplă	06.02 12.02	5	06.02 12.02	5
5	Importarea aplicației pe Android Fig.9	12.02 18.02	5	12.02 18.02	4

6	Adăugarea bazei de date și a fișierului API pe server	18.02 19.02	2	18.02 19.02	2
7	Crearea documentației a aplicației Creare modul log pentru evidența erorilor din aplicație	19.02 21.02	2	19.02 21.02	2

Agenda stagiului de practică Halley KM

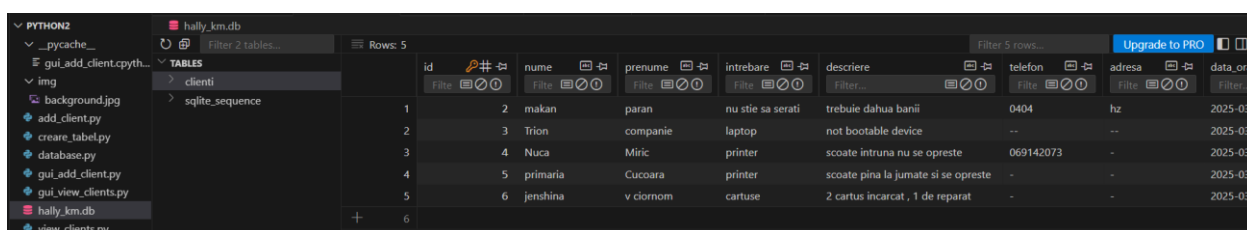
Data Executării lucrărilor	Descrierea concisă a lucrărilor efectuate pe parcursul stagerii	Observații și indicații ale conducătorilor practicii și ale stagiului
27.01	Am studiat mediul de dezvoltare Python, SQLite și Tkinter, familiarizându-mă cu componentele utilizate pentru dezvoltarea aplicațiilor.	
28.01	Am inițiat analiza cerințelor aplicației „Înregistrarea clientului cu probleme tehnice” și am început crearea bazei de date pentru gestionarea informațiilor despre clienți și reparații.	
29.01	Am continuat cu dezvoltarea bazei de date, definind tabelele necesare pentru gestionarea clienților, problemelor tehnice și istoricului reparațiilor. Fig.1	
30.01	Am început crearea aplicației cu interfața grafică utilizând Tkinter. Am creat feronerie de bază pentru gestionarea interacțiunii utilizatorului. Fig.2	
31.01	Am adăugat funcționalitatea de înregistrare a clientului și introducerea datelor de bază în baza de date. Fig.3	
03.02	Am implementat funcția de diagnosticare a problemelor tehnice și am legat interfața de baza de date pentru a stoca rezultatele.	
04.02	Am creat funcționalitatea de actualizare a statusului reparației și generare a istoricului pentru fiecare client. Fig.4	

05.02	Am implementat funcția de căutare și filtrare a clienților după diverse criterii (de ex., nume, problemă tehnică).	
06.02	Am creat un popupmenu pentru editarea datelor clientului și pentru actualizarea rapidă a informațiilor. Fig.5	

Agenda stagiului de practică Halley KM (Continuare)

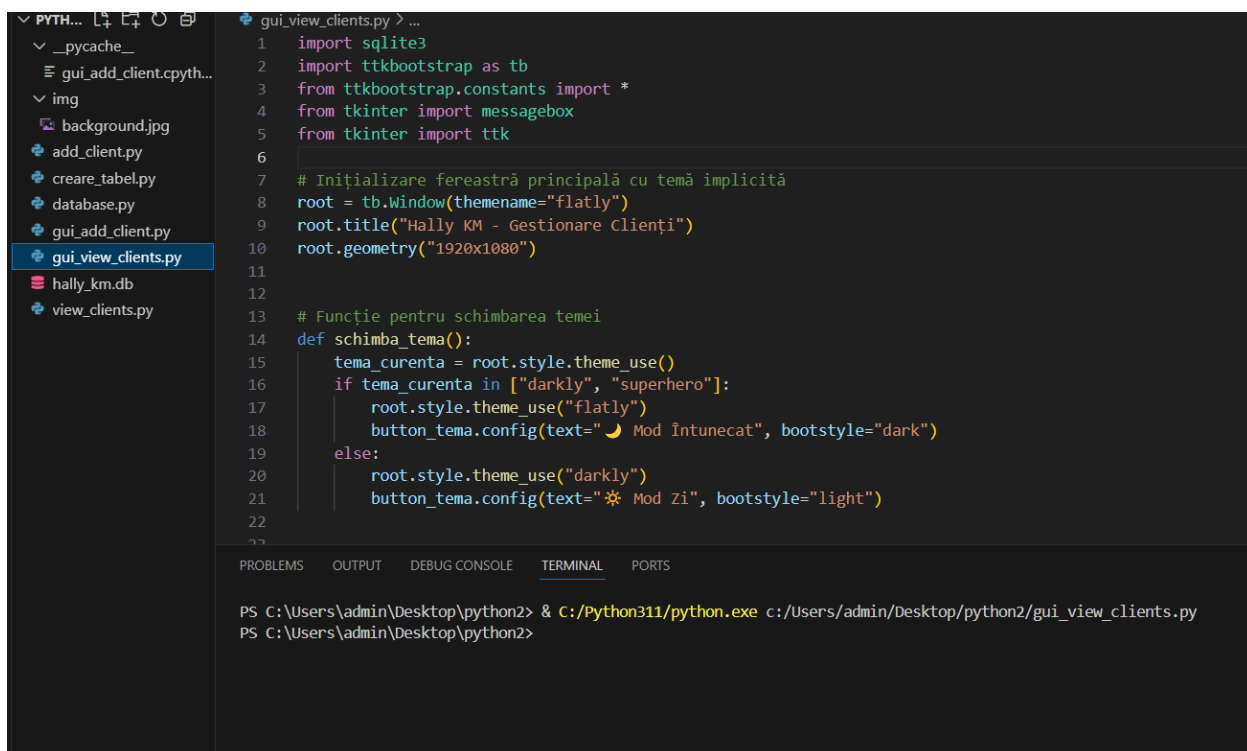
Data Executării lucrărilor		Observații și indicații ale conducătorilor practicii și ale stagiului
07.02	Am adăugat opțiunea de generare a rapoartelor pentru fiecare client în parte, cu detalii despre reparații și status.	
10.02	Am optimizat funcțiile existente pentru a îmbunătăți performanța aplicației.	
11.02	Am realizat teste pentru identificarea posibilelor erori și am început crearea documentației tehnice.	
12.02	Am început să adaug funcționalități suplimentare pentru gestionarea cererilor și reparațiilor multiple.	
13.02	Am adăugat funcția de export a rapoartelor într-un fișier CSV pentru o mai bună manipulare a datelor.	
14.02	Am optimizat aplicația, lucrând atât pe partea de back-end cât și pe partea de front-end pentru a îmbunătăți interacțiunea cu utilizatorul. fig.6	
17.02	Am testat aplicația pentru a preveni erori neprevăzute și am început procesul de integrare cu serverul companiei.	
20.02	Am adăugat baza de date și API-ul necesar pe server pentru a permite accesul de la distanță.	

18.02	Am continuat documentarea procesului de dezvoltare și am completat detalii tehnice pentru implementarea aplicației.	
19.02	Am adăugat un modul log pentru evidențierea erorilor și a evenimentelor din aplicație, pentru îmbunătățirea depanării.	
20.02	Am lucrat la crearea documentației pentru aceste 2 aplicații	
21.02	Creare modul log pentru evidența erorilor din aplicație	



id	nume	prenume	intrebare	descriere	telefon	adresa	data_ora
1	makan	paran	nu stie sa serati	trebuie dahua banii	0404	hz	2025-03-
2	Trion	companie	laptop	not bootable device	--	--	2025-03-
3	Nuca	Miric	printer	scoate intruna nu se opreste	069142073	-	2025-03-
4	primaria	Cucoara	printer	scoate pina la jumate si se opreste	-	-	2025-03-
5	jenshina	v ciornom	cartuse	2 cartus incarcat , 1 de reparat	-	-	2025-03-

Figură 1



```

1 import sqlite3
2 import ttkbootstrap as tb
3 from ttkbootstrap.constants import *
4 from tkinter import messagebox
5 from tkinter import ttk
6
7 # Inițializare fereastră principală cu temă implicită
8 root = tb.Window(themename="flatly")
9 root.title("Hally KM - Gestionare Clienți")
10 root.geometry("1920x1080")
11
12
13 # Funcție pentru schimbarea temei
14 def schimba_tema():
15     tema_curenta = root.style.theme_use()
16     if tema_curenta in ["darkly", "superhero"]:
17         root.style.theme_use("flatly")
18         button_tema.config(text="☾ Mod Întunecat", bootstyle="dark")
19     else:
20         root.style.theme_use("darkly")
21         button_tema.config(text="☀ Mod Zi", bootstyle="light")
22
23

```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

```

PS C:\Users\admin\Desktop\python2> & C:/Python311/python.exe c:/Users/admin/Desktop/python2/gui_view_clients.py
PS C:\Users\admin\Desktop\python2>

```

Figură 2

```

5 from datetime import datetime
6 from tkinter import messagebox
7
8 def adauga_client():
9     """Salvează datele clientului în baza de date."""
10    nume = entry_nume.get().strip()
11    prenume = entry_prenume.get().strip()
12    intrebare = entry_intrebare.get().strip()
13    descriere = entry_descriere.get("1.0", tk.END).strip()
14    telefon = entry_telefon.get().strip()
15    nivel_importanta = combobox_importanta.get()
16    adresa = entry_adresa.get().strip()
17    servicii = combobox_servicii.get().strip()
18
19    if not nume or not prenume or not intrebare or not descriere or not telefon or not adresa or not nivel_importanta or servicii == "Selectează":
20        messagebox.showwarning("Eroare", "Toate câmpurile trebuie completate!")
21        return
22
23    conn = sqlite3.connect("hally_km.db")
24    cursor = conn.cursor()
25
26    data_oră = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
27
28    cursor.execute("""
29        INSERT INTO clienti (nume, prenume, intrebare, descriere, telefon, nivel_importanta, adresa, servicii, data_oră)
30        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
31    """)
32    conn.commit()
33    conn.close()
34
35    messagebox.showinfo("Succes", "Clientul a fost adăugat cu succes!")
36
37    # Actualizare interfață
38    # ...

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\admin\Desktop\python2> & C:\Python311\python.exe c:/Users/admin/Desktop/python2/gui_view_clients.py
PS C:\Users\admin\Desktop\python2>

Figură 3

```

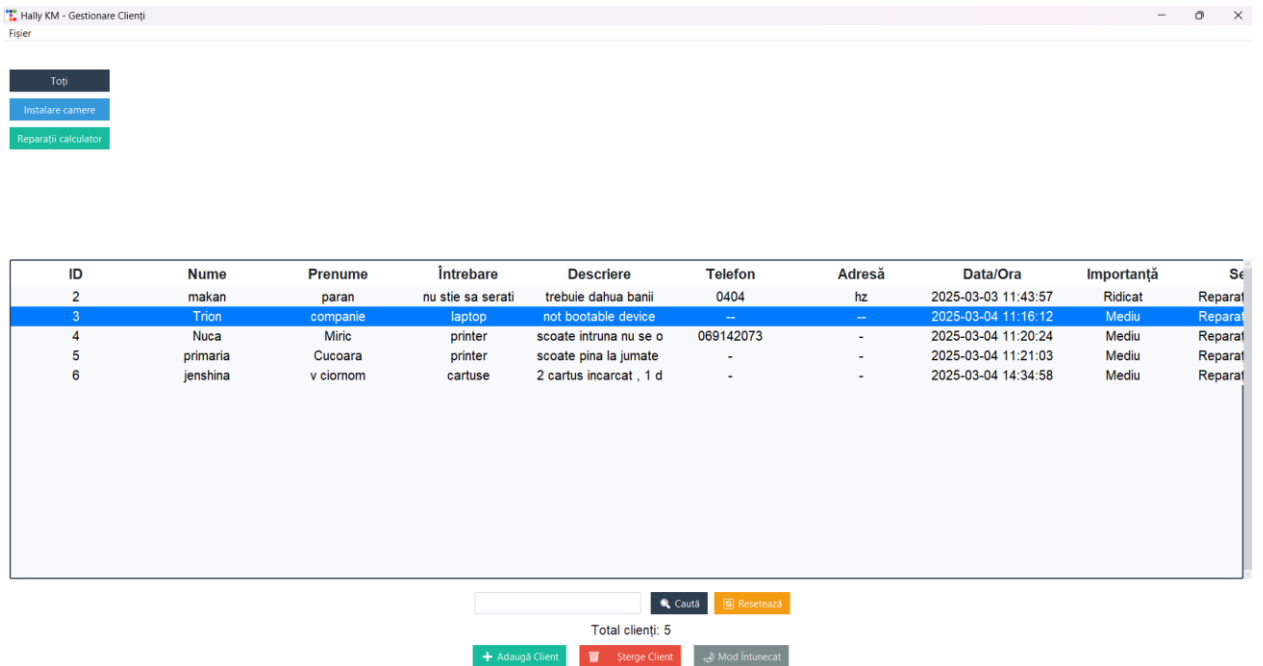
292 tree.bind("<Double-1>", afiseaza_detalii_client)
293
294 def afiseaza_detalii_client():
295     # ...
296
297 def editeaza_client(client):
298     fereastră_editare = tk.Toplevel(root)
299     fereastră_editare.title(f"Editează client - {client[1]} {client[2]}")
300     fereastră_editare.geometry("500x500")
301     fereastră_editare.configure(bg="#f8f9fa")
302
303     # Creare container scrollabil
304     container = tk.Frame(fereastră_editare)
305     canvas = tk.Canvas(container, height=450, bg="#f8f9fa")
306     scrollbar = tk.Scrollbar(container, orient="vertical", command=canvas.yview)
307     scroll_frame = tk.Frame(canvas)
308
309     scroll_frame.bind("<Configure>", lambda e: canvas.configure(scrollregion=canvas.bbox("all")))
310     canvas.create_window((0, 0), window=scroll_frame, anchor="nw")
311     canvas.configure(yscrollcommand=scrollbar.set)
312
313     container.pack(fill="both", expand=True, padx=10, pady=10)
314     canvas.pack(side="left", fill="both", expand=True)
315     scrollbar.pack(side="right", fill="y")
316
317     def on_mouse_wheel(event):
318         # ...
319
320     fereastră_editare.mainloop()

```

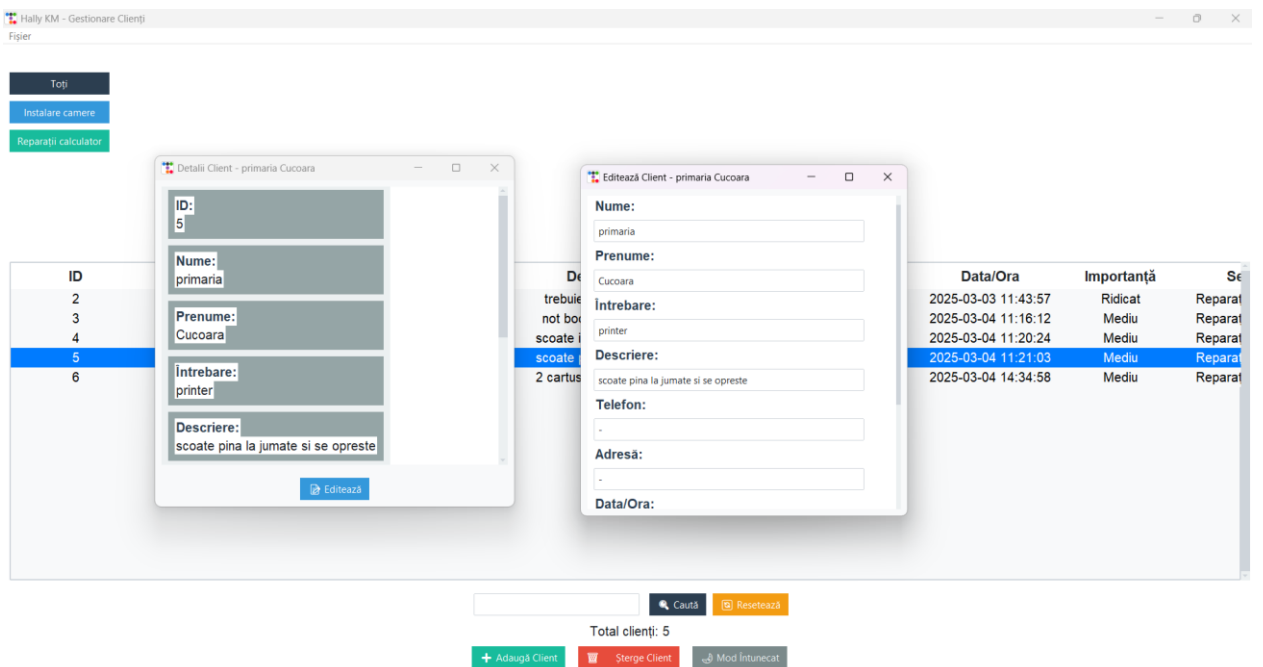
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\admin\Desktop\python2> & C:\Python311\python.exe c:/Users/admin/Desktop/python2/gui_view_clients.py
PS C:\Users\admin\Desktop\python2> & C:\Python311\python.exe c:/Users/admin/Desktop/python2/gui_view_clients.py

Figură 4



Figură 5



Figură 6