Universitatea Tehnică de Construcții din București
Facultatea de Hidrotehnică
Specializarea: Automatică și Informatică Aplicată

Daily Jounal GUI App

Coordonator proiect:
Drd. Ing. Olteanu Grabriela

Realizator:
Radulescu Ioan-Radu

București, 2025

CUPRINS

INTRODUCERE

Motivarea alegerii temei

Recent, a devenit un trend in cadrul comunitatiilor de dezvoltare personala, obiceiul de a scrie in jurnal. Fie ca este doar pentru a practica recunostiinta (gratitude journaling) sau poate pentru a-ti pune pe hartie ideelie pe care le-ai avut pe parcursul zilei, indiferent de motiv, eu consider ca este un hobby benefic mintii. Acest lucru combinat cu pasiunea mea petru acest obicei, m-a facut sa aleg realizarea acestei aplicatii.

Pe platformele online nu sunt prea multe aplicatii care sa ofere toate beneficile necesare pentru o platforma de journaling. In general ele vin cu o subscriptie si o gramada de feature-uri de care nu este nevoie si nici nu isi au locul intr-o astfel de aplicatie. Acesta este inca un motiv pentru care am decis sa dezvolt o astfel de aplicatie, cu scopul principal fiind, o aplicatie cat mai apropiata de sentimentul de a scrie direct cu pixul pe foaie, doar ca pe calculator. Ceva care sta la limita intre analog is digita.

Obiectivele propuse in cadrul lucrarii

Pentru dezvoltarea Daily Journal GUI App, mi-am propus sa ma inspir dupa putinele aplicatii care exista si care sunt utilizabile fara subscriptii, si sa adaug functii care le-as dori si care consider ca ar fi fost esentiale pentru experienta cat mai apropiata de real.

Cel mai important lucru, cat si cel mai greu, a fost realizarea pagini care afiseaza fiecare intrare/capitol din jurnal. Pentru asta am folosit un fisier de tip json care tine minte toate informatiile despre intrari, intr-un format usor de procesat si usor de intrgat in cod. Dupa asta a venit pagina de editare, adaugare si vizionare a unei intrari in jurnal. Cel mai important lucru aici a fost sitemul de salvare in fisiere json. Transformarea fiecarui camp de text, intr-un camp in fisierul json, cat si inversul in momentul afisarii este esential functionalitatii de baza a programului. Ultimul pas si nu cel din urma, o functie care in general era blocata in spatele subscriptiilor este cea de rutini. Rutinile sunt taskuri care apar zilnic, la fiecare intrare, de care trebuie sa tii cont. Exemple includ: Sala de forta, Skincare, Carte, Plimbare in parc, Medicamente, Progrmare, etc. In principal, ce am vrut sa adaug este o modalitate in care iti este usor sa tii minte ce ai de facut de pe o zi pe alta, chestii care ar trbui sa fie automate, dar nu chiar.

Pentru realizarea functionalitatiilor aplicatiei am folosit in principal doar JavaFX si Java pentru programare si logica din spate, SceneBuilder pentru pagini si combinat cu JavaFX pentru updatarea lor in timp real si organizarea pe grid, si o stilizare simpla CSS. Proiectul a durat aproximativ o luna pentru a fi realizat in doar 1000 de linii de cod.

Structura

Lucrarea este compusă din 5 capitole şi un capitol ce concretizează concluziile, contribuțiile şi perspectivele de dezvoltare viitoare. Proiectul prezintă contribuțiile cu privire la sistemele informatice pentru gestionarea disfuncţionalităţilor unei aplicaţii, semnalate de client, sau altfel spus, aplicatiile de ticheting.

In cadrul primului capitol am prezentat conceptul de ticketing, unde este intalnit, de ce este benefic, unde si cine il poate folosi si ce beneficii aduce folosirea acestuia in cadrul companiilor mici si nu numai.

In cel de al doilea capitol am descris succint tehnologiile pe care le-am folosit pentru dezvoltarea aplicatiei si limbajele de programare folosite in scrierea acesteia. Am folosit programe precum Visual Studio Community 2015 si SQL Server, dar si limbaje precum C#, JavaScript, CSS, HTML, JQuery.

In cadrul capitolului trei am prezentat modul de functionare al fiecarui modul cuprins in aplicatie, astfel explicand care este rolul modulelor si cum isi aduc acestea aportul pentru buna functionare a aplicatiei.

Capitolul patru este o scurta prezentare al modului in care conturile noi se creeaza in cadrul aplicatiei, cum sunt ele gestionate de catre administrator si rolurile pe care un utilizator le poate avea in aplicatie, roluri pe care este bazat si accesul in cadrul modulelor.

In ultimul capitol am descris aplicatia in sine, modul in care aceasta se configureaza, cum se foloseste si am prezentat succint un scenariu de crearea a unui utilizator nou, configurarea clientului pe care acesta va fi atribuit si modul in care utilizatorul introduce un tichet in sistem, este preluat mai departe de un alt utilizator helpdesk si solutioneaza tichetul.

1. ASPECTE TEORETICE REFERITOARE LA DAILY JOURNAL GUI APP

Acest capitol oferă o imagine clară și detaliată a proiectului **DailyJournal**, concentrându-se pe tehnologiile folosite, arhitectura generală, componentele principale, fluxurile de date și posibile direcții de îmbunătățire.

1.1 IntelliJ IDEA – mediul principal de dezvoltare

Proiectul a fost realizat folsind platforma de la JetBrainz, InteliJ IDEA, IDE-ul loc special conceput pentru programe java. Am ales acest mediu deoarece este cel mai folosit de pe piata si acest proiect a constat intr-un exercitiu personal de a ma obisnui cu aplicatia specifica pentru viitor.

Beneficiile InteliJ IDEA pentru aplicaite:

- **Verificare în timp real:** IDE-ul îți arată din start erorile de sintaxă, tipuri greșite, importuri nefolosite și chiar posibile bug-uri, fără să fie nevoie să rulezi aplicația.
- **Refactorizare sigură:** Dacă redenumești o clasă sau o metodă legată de FXML, IDE-ul poate actualiza automat toate referințele.
- **Integrare cu Maven:** IntelliJ citește automat dependențele din pom.xml, le sincronizează și îți arată structura lor. Poți și să rulezi direct task-uri Maven din IDE.
- **Debugging ușor:** Poți pune puncte de oprire în metodele controllerelor, cum ar fi cele legate de butoane sau inițializare, ca să vezi exact ce date sunt încărcate sau ce valori au câmpurile din UI.

1.2 Java – limbajul de programare

DailyJournal e scris în Java, folosind programarea orientată pe obiect (OOP), care ajută la o structură clară a datelor și a responsabilităților.

De ce am ales Java:

- **Portabilitate:** Aplicația poate rula pe orice sistem de operare care are Java instalat.
- **Claritate în organizare:** Modelul (ex. JournalEntry, RoutineStatus), salvarea datelor (JournalStorage, RoutineStorage) și partea de UI (controller-ele) sunt separate, ceea ce face codul mai ușor de întreținut.
- **Liste pentru rutine:** Pentru fiecare intrare din jurnal, este folosită o listă de rutine (List<RoutineStatus>), ceea ce este foarte natural în Java.

1.3 JavaFX – interfața grafică

JavaFX este framework-ul pentru UI folosit în acest proiect. În pom.xml sunt incluse:

- javafx-controls
- javafx-fxml

Cum funcționează JavaFX în DailyJournal:

- **Fereastra și elementele UI:** Aplicația are o fereastră (Stage), care conține o scenă (Scene), iar toate componentele UI sunt organizate într-un „arbore” (Scene Graph). Când încarci un fișier FXML, acest arbore se creează automat.
- **Separarea codului de design:** UI-ul este descris în fișiere .fxml, iar comportamentul este implementat în controller-ele Java.

Fișierele confirmate: main.fxml, entryCreation.fxml, routineManagement.fxml.

- **Conectarea elementelor din FXML cu controller-ul:** Componentele UI sunt injectate automat în controller, prin adnotarea @FXML.

Despre stilizare:

Proiectul folosete un fisier de tip CSS pentru stilizarea intreaga a aplicatie, pentru a face utilizarea ei mai placuta. JavaFX permite cu usurinta integrarea stilizarii direct in proiect, ca un fel de plating la mancare (poate fi adaugat direct la final si implementat usor, cand proiectul este deja gata).

2. ARHITECTURA PROIECTULUI – MODELUL MVC

DailyJournal urmează modelul **MVC** (Model-View-Controller):

- **Modelul**: obiectele de date, precum JournalEntry și RoutineStatus
- **View-ul**: fișierele .fxml care definesc interfața
- **Controller-ele**: clasele care gestionează interacțiunile (ex. MainController, EntryCreationController)

3. COMPONENTELE PRINCIPALE

3.1 Modelul – „inima datelor”

3.1.1 JournalEntry

Reprezintă o intrare în jurnal și conține:

- title: titlul intrării
- date: data (ca String formatat)
- notes: notițe
- routines: lista de rutine bifate pentru acea zi

Observație: Referitor la rutine, ele raman acelasi din ziua in care a fost postata intrarea in jurnal, indiferent daca a fost editat sau nu. Pagina de RoutineEdit (routineManagement.fxml) are rolul de a schimbat “presetul” de acum incolo al rutini, nu si cel anterior, pentru ca, realisting vorbind, nimeni nu mai tine cont de ce a facut in trecut, si cu atat mai mult, nu tin cont de ce trebuia sa fac

in trecut. Asta fost una dintre principale funcitonalitati pe care nu am vazut-o la nicio aplicatie de pe piata.

3.1.2 RoutineStatus

Reprezintă o rutină (numele ei) și dacă a fost completată sau nu (boolean completed).

Este bine separată de restul, ceea ce permite adăugarea ușoară a altor câmpuri în viitor (ex. observații, ora completării etc.).

3.2 Persistența – „memoria aplicației"

Datele sunt salvate în două fișiere:

- profile.json – pentru intrările din jurnal
- routines.txt – pentru lista generală de rutine

3.2.1 JournalStorage

Se ocupă de citirea și scrierea datelor în profile.json. Operațiile sunt făcute manual, adică fără biblioteci externe ca Gson sau Jackson.

Observatie: Proiectul nu foloseste nicio librarie externam de management al fisierelor de tip JSON deoarece, este menit ca un proiect mic. Nu am considerat ca ar fi nevoie de ceva mai extins, si cu ocazia asta am incercat sa invat logica manuala a fisierelor JSON. Pe langa asta, implementarea Gson (ceea ce am si incercat sa fac ulterior ca sa mai simplific codul) a dus la mai multe erori si mai multe probleme in cod decat asteptat, asa ca am decis sa raman la versiunea mai rudimentara.

Avantaje:

- înveți cum funcționează JSON;
- ai control total asupra formatului;
- nu depinzi de librării externe.

Dezavantaje:

- codul e mai fragil și mai greu de întreținut;
- riscul de erori crește, mai ales cu caractere speciale (diacritice, newline etc.);
- adăugarea unui nou câmp în model necesită modificarea manuală a codului de parsare.

3.2.2 RoutineStorage

Se ocupa cu gestionarea fisierului routines.txt, unde sunt stocate toate rutinele. Este un fisier simplu in care fiecare lini este o noua rutina.

3.3 Controllerele – clasele principale.

3.3.1 MainController (main.fxml)

Responsabilități:

- afișează lista de intrări;
- gestionează butoanele „New Entry” și „Edit Routines”;
- filtrează lista pe baza inputului dintr-un câmp de căutare.

MainController este practic, centrul de control al paginii principale: cea care contine intrarile, butonul de “New Entry”, “Edit Routines”, Searchbar-ul si lista cu intrari.

3.3.2 EntryCreationController (entryCreation.fxml)

Responsabilități:

- gestionează formularul de creare/editare a unei intrări;
- preia datele completate;
- salvează sau actualizează intrarea.

Moduri posibile:

- **Creare:** câmpurile sunt goale și se creează o nouă intrare.
- **Editare:** primește o intrare existentă, populează UI-ul și salvează modificările în același obiect.

3.3.3 RoutineEditController (routineManagement.fxml)

Resposabilitati:

- afiseaza rutinele actuale
- permite editarea lor individuala cat si stergelea lor

Rolul clasei este sa gestioneze operatiunile facute in pagina de rutini si managementul stocarii lor in fisierul routines.txt.

3.4 View-ul (FXML) – „fața aplicației”

Fișierele FXML definesc structura UI:

- main.fxml: lista intrărilor, câmpul de căutare, butoanele principale;
- entryCreation.fxml: formular pentru intrări;
- routineManagement.fxml: ecran pentru administrarea rutinelor.

Controller-ele asigură comportamentul, legând evenimentele (ex. apăsarea unui buton) de metodele definite în Java.

4. PUNCTUL DE PORNIRE

Clasa dailyjournal.Main este punctul de intrare. Inițializează aplicația JavaFX, încarcă main.fxml și afișează fereastra principală.

5. FLUXURI FUNCȚIONALE PRINCIPALE

- **Pornirea aplicației:** Main → MainController → ListView afișează datele din profile.json.
- **Crearea unei intrări:** click pe „New Entry” → se deschide formularul → se salvează în profile.json.
- **Editarea rutinelor:** click pe „Edit Routines” → modificări în listă → se salvează în routines.txt.

6. FORMATUL FIȘIERELOR

- **profile.json** – stochează date structurate despre intrările din jurnal.
- **routines.txt** – o listă simplă de rutine, câte una pe linie.

CONCLUZII ȘI CONTRIBUȚII PERSONALE

Concluzii

Prin dezvoltarea aplicației DailyJournal am atins obiectivele stabilite în etapa de planificare, urmărind în principal realizarea unei aplicații desktop funcționale, intuitive și ușor de utilizat. Cel mai important rezultat a fost implementarea modulului de gestionare a intrărilor de jurnal, prin care utilizatorul poate crea intrări noi, le poate actualiza ulterior și le poate vizualiza într-o listă centrală, cu suport pentru căutare/filtrare. Acest flux este susținut de separarea clară dintre interfață și logică, folosind fișiere FXML pentru partea de prezentare și controller-e dedicate pentru controlul interacțiunilor.

Un alt obiectiv esențial a fost implementarea unui sistem de rutine asociate fiecărei intrări, astfel încât aplicația să nu fie doar un spațiu de notițe, ci și un instrument de monitorizare a activităților zilnice. Gestionarea rutinelor printr-un ecran dedicat (routineManagement.fxml) și salvarea lor într-un fișier separat (routines.txt) oferă utilizatorului flexibilitate: lista globală de rutine poate fi modificată independent, iar la nivel de intrare se păstrează statusul completat/necompletat pentru ziua respectivă.

De asemenea, am considerat importantă implementarea persistenței locale a datelor, astfel încât aplicația să rămână utilizabilă pe termen lung fără dependențe externe. Intrările sunt salvate în profile.json, iar rutinele în routines.txt, ceea ce permite atât păstrarea unei structuri clare a datelor, cât și o verificare ușoară a fișierelor, inclusiv în afara aplicației. O componentă definitorie a proiectului este faptul că serializarea și parsarea JSON sunt realizate manual, ceea ce demonstrează înțelegerea formatului și a procesului de transformare dintre obiecte și reprezentarea lor persistentă.

Consider că, prin realizarea acestei aplicații, am construit o bază tehnică solidă pentru un produs de tip jurnal personal, cu arhitectură coerentă și cu posibilitatea de extindere ulterioară. Proiectul demonstrează utilizarea corectă a tehnologiilor JavaFX, integrarea UI–logică prin controller-e și organizarea componentelor în zona de model și stocare, într-un mod care poate fi menținut și dezvoltat progresiv.

Contribuții personale

În dezvoltarea aplicației DailyJournal, contribuția mea principală a constat în proiectarea și implementarea întregului flux de lucru al aplicației: structurarea interfețelor în fișiere FXML, realizarea controller-elor pentru navigare și acțiuni, modelarea datelor prin clase dedicate și implementarea mecanismelor de stocare locală. În mod specific, am realizat:

- un sistem de management al intrărilor care permite creare, vizualizare și actualizare, cu afișare într-o listă centrală și funcționalități de filtrare/căutare în ecranul principal;
- integrarea rutinelor în jurnal printr-o structură de tip RoutineStatus, astfel încât fiecare rutină să poată fi urmărită ca „completată” sau „necompletată” în contextul unei zile;
- implementarea persistenței locale în două formate: JSON pentru date complexe (intrări + rutine asociate) și text pentru lista globală de rutine, menținând astfel o separare logică între configurație și conținut;

- dezvoltarea mecanismului de salvare și actualizare a intrărilor prin metode dedicate (saveEntries, saveOrUpdateEntry, loadEntries), cu accent pe consistența datelor și pe menținerea unei structuri predictibile a fișierului de stocare.

Pe parcursul implementării, mi-am aprofundat cunoștințele în Java, JavaFX, lucrul cu FXML, organizarea aplicațiilor în stil MVC și gestionarea persistenței locale prin fișiere. De asemenea, am consolidat abilități practice de lucru în IntelliJ IDEA și de structurare a unui proiect Maven astfel încât să fie ușor de rulat și întreținut.

Perspective de viitor

Ca direcții de dezvoltare ulterioară, îmi propun să extind aplicația DailyJournal prin funcționalități care cresc utilitatea și robustețea ei în utilizare reală, cum ar fi:

- introducerea unei biblioteci standard pentru serializare (Gson/Jackson) pentru a elimina fragilitatea parsării manuale și pentru a crește stabilitatea la date complexe;
- adăugarea unui sistem de export/import (ex. export în PDF sau CSV pentru arhivare și analiză);
- extinderea modulului de rutine cu statistici (rate de completare, streak-uri, vizualizări săptămânale/lunare);
- îmbunătățiri de UI/UX: stilizare CSS, teme (light/dark), validări mai clare și feedback vizual la salvare;
- eventual, migrarea persistenței către o bază de date locală (ex. SQLite), dacă aplicația crește în volum și complexitate.

Prin aceste extinderi îmi doresc să transform aplicația dintr-un proiect funcțional într-un instrument complet, scalabil și robust, care să poată fi folosit constant și adaptat în timp pe baza feedback-ului și a cerințelor noi.

BIBLIOGRAFIE

[1] https://www.oracle.com/java/technologies/javase/javafx-docs.html
[2] https://openjfx.io/javadoc/21/javafx.graphics/javafx/scene/doc-files/cssref.html
[3] https://www.geeksforgeeks.org/javascript/json/
[4] https://www.youtube.com/playlist?list=PLZPZq0r_RZOM-8vJA3NQFZB7JroDcMwev
[5] https://www.youtube.com/watch?v=GpOO5iKzOmY
[6] https://en.wikipedia.org/wiki/IntelliJ_IDEA
[7] https://en.wikipedia.org/wiki/Java_(programming_language)