Licență

* **Romana:**

Aplicația WEB prezentată în această lucrare de licență se numește OfOps, aceasta având ca scop gestionarea spațiilor comune ale unei firme, astfel încât desfășurarea activităților de birou să fie optime.

Locurile uneori insuficiente pentru toți membrii echipei, goana de a ajunge mai devreme să prinzi biroul preferat sau chiar un loc de parcare sunt experiențe neplăcute, trăite, care au reprezentat baza acestei aplicații. Necesitatatea de a ști disponibilitățile birourilor, sălilor de ședință și al locurilor de parcare mi s-a părut un punct de plecare important pentru ca aplicația OfOps să ne scape de grijile venirii la birou.

Aplicația dispune de un calendar și un ceas pentru a selecta momentul în care se dorește o vizualizare a disponibilității în acel moment, dar și de hărți interactive. Acestea au rolul să faciliteze experiența navigării și, tot pe această hartă, se pot observa locurile colorate diferit în funcție de statusul rezervării lor. Paleta de culori folosită pentru a marca locurile este: verde – locul este liber –, roșu – ocupat – și, în cele din urmă, galben – rezervarea a fost făcută de utilizatorul autentificat în aplicație –.

Partea de rezervare se poate efectua printr-un simplu click pe locul dorit. Însă, în cazul în care mai există deja rezervări pentru acel loc, utilizatorul va fi anunțat printr-o notificare și nu va putea să efectueze rezervarea în cazul în care în intervalul dorit este programată altă rezervare. Pentru sălile de ședință, în schimb, user-ul va primi o recomandare de sală disponibilă pentru intervalul dorit în cazul în care sala nu este disponibilă în intervalul inițial.

Această lucrare va prezenta, în cele ce urmează, toate etapele care au stat la baza dezvoltării aplicației de la gândirea flow-ului inițial și baza sa de date până la API-urile folosite pentru integrarea funcționalităților sale.

* **Engleza:**

The WEB application presented in this bachelor's thesis is called OfOps, its purpose being the management of shared spaces within a company, so that office activities can be optimized.

The sometimes insufficient seating spaces for all team members, the rush to arrive early to catch the preferred desk or even a parking lot are unpleasant experiences that have served as the basis for this application. The need to know the availability of desks, meeting rooms, and parking lots was an important starting point for me to develop OfOps in order to relieve us of the worries of going to the office.

The application features a calendar and a clock to select the desired time of availability, as well as interactive maps. These maps are designed to enhance the navigation experience and, on these maps, places are color-coded depending on their reservation status. The color palette used to mark them is: green – available place –, red – occupied – and finally, yellow – reservation has been made by the authenticated user–.

The booking process can be done with a simple click on the desired spot. However, if there is already another reservation for that spot, the user will be notified about it and will not be able to make it if another reservation is scheduled for the wanted interval. For meeting rooms, on the other hand, the user will receive a recommendation for an available meeting room for the desired time if the room is not available in the initial interval.

This paper will present all the stages that have underpinned the development of the application, from the initial flow thinking and its database to the APIs used for integrating its functionalities.

* **Problemă:**

În cadrul firmei în care am efectuat practica vara trecută, prezența la birou mi s-a părut utilă pentru facilitarea colaborării în cadrul echipei de proiect, însă acest lucru venea și cu dezavantaje. Gândul că exista posibilitatea să nu găsesc un birou liber lângă cei de la care învățam mă neliniștea întrucât relaționarea și comunicarea nu mai erau oportune. Din cauza acestei probleme, încercam să ajung dimineața cât mai devreme astfel încât să găsesc un loc liber lângă cei cu care lucram în cadrul proiectului. Și nu numai eu aveam această problemă. Și colegii mei din departament se confruntau cu acest inconvenient. Deseori interveneau probleme legate de rezervarea spațiilor de lucru, dificultăți care se extindeau și în zona locurilor de parcare, ducând la un disconfort și o grijă inutilă pentru a începe o nouă zi de muncă.

* **Scop:**

Scopul OfOps este de a optimiza și de a utiliza eficient resursele comune existente la nivelul firmei, iar motivația realizării aplicației vizează îmbunătățirea desfășurării activităților angajaților la birou creând un mediu propice atingerii obiectivelor firmei.

* **Obiective:**

Obiectivele pe care și le propune aplicația OfOps să le îndeplinească sunt:

* Rezervarea optimă a birourilor și a locurilor de parcare, fără ca rezervările să se suprapună cu altele deja existente;
* Eficientizarea utilizării sălilor de ședințe
* Sugerarea unei alternative în cazul în care sala este ocupată în intervalul dorit;
* Utilizarea hărților interactive pentru a face mai ușoară experiența utilizatorului.
* **Tehnologii folosite:**

Aplicația WEB reprezintă un program de tip software stocat pe un server remote, devenind accesibil pentru o masă mare de utilizatori care doresc sa acceseze date rapid și ușor. Componentele sale pot comunica prin protocoale bazate pe internet cum ar fi HTTP (Hyper Text Transfer Protocol) sau HTTPS (Hyper Text Transfer Protocol Secure).\cite{citation1} Principalele avantaje ale aplicațiilor WEB le reprezintă accesibilitatea – acessarea lor se face folosind doar un browser WEB indiferent de circumstanțele externe (ora și loc) –, ușurința utilizatorului de a naviga pe platformă fără a avea nevoie de resurse suplimentare și scalabilitatea – poate gestiona un volum mare de date fără a se degrada –.

Aceasta are la bază arhitectura de tipul „client-server” în care clienții efectuează diferite operațiuni asupra serverului, iar răspunsurile serverului sunt primite prin intermediul internetului.

\begin{figure}[!htb]

\centering

\includegraphics[width=0.7\linewidth]{images/client-server-network.png}

\caption{Comunicarea client-server}

\label{fig:client-server}

\end{figure}

Aplicația OfOps a fost construită având partea de server construită cu ajutorul MySQL, Spring Boot și Java, iar partea de client a fost implementată folosind instrumnete specifice frontend-ului anume Angular, HTML (HyperText Markup Language), CSS (Cascading Style Sheets) și Typescript. Aceste tehnologii vor fi descrise în subcapitolele ce urmează.

* **Stocarea datelor:**

MySQL este un sistem pentru management-ul bazelor de date relaționale, adică baze de date care stochează date în tabele separate. 2! Structura sa este menită să fie un instrument flexibil pentru programare, întrucât poți să-ți setezi regulile după care să fie construită baza de date, relațiile între tabele (one-one, many-one, one-many, many-many), cheile primare sau străine, constrângeri etc. Utilizarea MySQL pentru stocarea bazei de date este una eficientă, întrucât modul în care această aplicație a fost construită nu permite inconsistența, duplicarea sau lipsa datelor.

Luând în considerare avantajele MySQL, am ales ca baza de date a aplicației OfOps să fie păstrată în MySQL Workbench versiunea 8.0 CE datorită volumul mare de date pe care îl suportă. Aceasta oferă o interfață intuitivă și ușor de utilizat, stocarea tabelelor este bine organizată și ai la dispoziție, printr-un singur click, detalii despre tabele, cât și datele păstrate în ele, făcând interacțiunea cu MySQL Workbench una facilă și rapidă.

* **Backend**

Pentru implementarea backend-ului am utilizat Java si Spring Boot.

* + Java – este un limbaj de programare orientat pe obiecte, high-level, creat in jurul anilor 1990. Până atunci, C și C++ erau cele mai răspândite limbaje de programare, însă utilizarea lor în masă largă era destul de restrânsă și costisitoare. Motivația dezvoltării Java a reprezentat-o nevoia unui limbaj de programare care să poată fi folosit pentru diferite dispozitive electronice cum ar fi cuptorul cu microunde sau chiar dispozitivele cu control remote. \cite{citation3}. Versatilitatea și simplitatea acestui limbaj l-au adus printre cele mai populare și răspândite modalități de a coda până în prezent.

De asemenea, Java joacă un rol important și în această licență, întrucât majoritatea codului dezvoltat are la bază acest limbaj de programare. Codul a necesitat instalarea în prealabil a unui JRE (Java Runtime Environment), astfel încât compilarea sa să se realizeze cu succes.

* + Spring Boot – Spring este un framework care vine în ajutorul dezvoltării rapide ale unor aplicații stand-alone, deoarece lucrează cu alte librării astfel încât construirea aplicației să nu mai aibă nevoie de de o configurare foarte mare în plus. \cite{citation4}.

Am ales Maven pentru adăugarea dependințelor proiectului, iar pe lângă cele cu care venea deja proiectul am mai adăugat: Spring-Boot-Starter-Data-JPA (interacțiune backend - baza de date), Spring-Boot-Starter-WEB (simplifică construirea aplicațiilor WEB cu Spring), MySQL-Connector-J (conexiunea cu baza de date), Lombok (generare de cod comun), Spring-Boot-Starter-Test (pentru testele unitare), Spring-Boot-Starter-Security și Java-JWT (pentru securitatea aplicației).

* **Frontend**
* **Asigurarea securității**

Securitatea OfOps se menține la două niveluri:

* Autentificarea propriu-zisă în aplicație printr-un sistem de tipul user-parolă
* Accesibilitatea datelor este limitată pentru fiecare rol, oferind doar anumite opțiuni de utilizare a aplicației în funcție de rolul atribuit

Aceste lucruri s-au realizat la nivel de implementare cu ajutorul JWT (JSON Web Token) care a rezolvat atat problema de autentificare, cât și pe cea de autorizare.

JWT este un standard bazat pe transmiterea în siguranță a unor date care, prin simplititatea, siguranța și versatilitatea sa, a ajuns să fie folosit în cele mai mari framework-uri web.\cite{citation6} Token-ul este format dintr-un header, un payload și o semnătură/ datele criptate. Primele două sunt obiecte de tip JSON cu o structură deja definită, însă ultima parte depinde în mod direct de tipul de criptare utilizat.

Modul în care se folosește acest token este următorul. Mai întâi, din browser, se trimite către backend, printr-o metodă HTTP de tipul POST pe endpoint-ul de /login, un username și o parolă pentru care server-ul crează un JWT. Acest token este trimis înapoi la frontend, moment în care acesta trimite înapoi header-ul pentru autorizare. Se verifică semnătura și se preiau detaliile utilizatorului și se trimite răspunsul, astfel, către client.

Folosirea token-ului pentru autentificare se face în modul următor: