

LUCRARE DE LICENȚĂ

**Coordonator:**

**Absolvent:**

Chițu Raluca-Oana

Brașov

Februarie 2025



LUCRARE DE LICENȚĂ

Tehnologii web în realizarea aplicației Complex turistic Luana Nature

**Absolvent:** Chițu Raluca-Oana

**Coordonator:** Conf.Dr. Sângeorzean Livia

Brașov

Februarie 2025

CUPRINS

[Capitolul 1 Introducere 4](#_Toc187423107)

[1.1 Prezentarea lucrării 4](#_Toc187423108)

[1.2 Motivația alegerii temei 4](#_Toc187423109)

[1.3 Aplicații similare 4](#_Toc187423110)

[1.4. Structura lucrării 5](#_Toc187423111)

[Capitolul 2 Dezvoltarea aplicației 6](#_Toc187423112)

[2.1 Medii de dezvoltare și sisteme de gestionare a bazelor de date 6](#_Toc187423113)

[2.1.1 IntelliJ IDEA – liderul IDE pentru Java 6](#_Toc187423114)

[2.1.2 PostgreSQL – cea mai avansată bază de date relațională open-source din lume 7](#_Toc187423115)

[2.2. Dezvoltarea server-side (*Back-End)* 8](#_Toc187423116)

[2.2.1 Spring – Spring Boot 8](#_Toc187423117)

# Capitolul 1 Introducere

## 1.1 Prezentarea lucrării

Cine nu și-ar dori un spațiu în care munca de birou zilnică să fie combinată cu natura? Într-o lume a vitezei, în care alergăm după mici momente de respiro și relaxare, tânjim după ieșirea din lumea betonelor și a poluării care ne înconjoară.

Aplicația Luana Nature este în fapt un liant între complexul turistic cu același nume și cei care doresc o oază de relaxare în care pot gestiona activitățiile zilnice de birou, în liniște. În sine, această aplicație oferă utilizatorilor săi posibilitatea de a vizualiza toate beneficiile oferite de locația precedent menționată, precum și posibilitatea realizării unei rezervări online. Totodată, utilizatorii pot trimite mesaje diverse sau întrebări. De asemenea, acești pot evalua locația în sine prin intermediul opțiunii de review-uri.

O altă facilitate oferită de aplicație este aceea de achiziționare a produselor oferite de magazinul local aflat în complex. Fiecare utilizator își poate vizualiza comanda înainte de trimitere și totodată, poate vedea în timp real prețul aferent produselor.

Toate facilitățile oferite de aplicație (creeare unei rezervări, trimitere de review-uri sau mesaje, comandă de produse) se realizează pe bază de cont.

Aplicația oferă de asemenea acces utilizatorilor cu drept de administrator, aceștia având dreptul de a: vizualiza utilizatorii, review-urile, comenzile, rezervările.

## 1.2 Motivația alegerii temei

La baza alegerii temei pentru creeare aplicației a stat dorința de a pune în valoare un complex turistic care iese din tiparele cunoscute. Mediul online este mediul prin care oamenii au acces în cel mai facil și rapid mod la locuri și zone nou dezvoltate. Practic, prin realizarea acestui web site, se dorește promovarea complexului turistic Luana Nature și prezentarea lui în fața celor interesați de astfel de spații naturale.

## 1.3 Aplicații similare

În lumea online există numeroase site-uri web care promovează complexuri turistice. Există atât platforme de promovare, cât și pagini web specifice unei singure locații. Din prima categorie, poate cea mai cunoscută este Booking.com, iar pe lângă aceasta câteva exemple ar fi: DirectBooking.ro, TuristInfo.ro, TravelMinit.ro, Trivago.ro, Agoda.com. Din cea de-a doua categorie putem exemplifica prin următoarele: Cheile Grădiștei Fundata, Mona Caraiman, Serenity Resort, Europark Fundata.

Toate aceste pagini web, indiferent de categorie, au următoarele elemente cheie:

* Utilizatorii pot vizualiza facilitățiile oferite (cazare, restaurant, activități etc.)
* Există opțiunea de a realiza rezervări online
* Scurtă prezentare a obiectivului turistic
* Date de contact
* Galerie foto și video

## 1.4. Structura lucrării

Primul capitol prezintă, pe scurt, scopul aplicației, motivarea alegerii acestui tip de aplicație, precum și a altor aplicații asemănătoare, existente deja.

Cel de-al doilea capitol face referire la principalele elemente tehnologice utilizate în cadrul aplicației, atât tool-uri, cât și limbajele de programare utilizate.

În cel de-al treilea capitol este presentată specific partea de structură a aplicației, arhitecturam dar și partea de structurare a codului.

Capitolul patru prezintă, în fapt, ghidul aplicației, fiind prezentată partea grafică a acesteia și modul de utilizare.

În capitolul cinci, ultim, se regăsesc concluziile și prezentarea unor funcționalității viitoare.

# Capitolul 2 Dezvoltarea aplicației

## 2.1 Medii de dezvoltare și sisteme de gestionare a bazelor de date

### 2.1.1 IntelliJ IDEA – liderul IDE pentru Java

A black square with white letters on it

Description automatically generated

IntelliJ IDEA reprezintă un mediu de dezvoltare integrat (IDE) utilizat pentru crearea de aplicații software. A fost creat de compania JetBrains și este unul dintre cele mai populare IDE-uri, fiind preferat pentru programarea în limbaje precum Java, în primul rând, Kotlin, Groovy, Scala, în al doilea rând.

Principalele caracteristici ale IntelliJ IDEA sunt:

* **Oferă unul dintre cele mai puternice editoare de cod din industrie** – capacitate de înțelegere a codului ceea ce permite detectarea din timp a erorilor, sugerarea opțiunilor de completare a codului, refactorizarea acestuia și detectarea duplicatelor;
* **Capacitate de completarea a codului** - având un model avansat de învățare profundă, permite îmbunătățirea eficenței codării prin sugestii și prezicere a unor linii întregi de cod;
* **Asistent AI** – utilizat pentru scrierea automată a comentariilor, generare de mesaje de confirmare, sugestii de nume etc.;
* **Furnizează suport pentru cele mai populare framework-uri dedicate dezvoltării de aplicații moderne și microservicii** – oferă asistență pentru Spring și SpringBoot, Jakarta EE, JPA, Micronaut, Quarkus, Helidon etc.
* **Oferă instrumente de tip built**-**in și integrări –** debugger (util pentru urmărirea codului pas cu pas), profiler (pentru date privind timpul de execuție și alocare de memorie), suport pentru baze de date, HTTP client (utilizat pentru cerereri de tip HTTP în cadrul dezvoltării serviciilor de tip web);
* **Cloud și distribuție –** Kubernets (sistem open-source pentru implementarea, scalarea și gestionarea aplicațiilor continerizate), Docker (descărcare și construire de imagini, rulare și depanare aplicații), aplicații server (Tomcat, Jboss, Glassfish etc. – pornire și oprire servere locale, conectare la servere la distanță);
* **Dezvoltare web –** suport pentru o gamă largă de tehnologii frontend precum JavaScript, HTML, CSS, React, Node.js etc;
* **Version control -** colaborează cu cele mai cunoscute sisteme de control a versiunilor (GitHub, GitLub, Mercurial etc.) și, totodată, oferă posibilitatea de anulare a modificărilor și recuperarea fișierelor șterse accidental (Local History).

IntelliJ IDEA se comercializează atât în variant gratuită (*Community Edition)* , cât și în varianta plătită (*Ultimate*).

### 2.1.2 PostgreSQL – cea mai avansată bază de date relațională open-source din lume

A blue elephant with white outline

Description automatically generated

PostgreSQL este un sistem puternic de gestionare a bazelor de date care folosește și extinde limbajul SQL. Combină numeroase caracteristici care stochează și extind în siguranță cele mai complicate volume de date. Orginile sale datează încă din 1986 fiind dezvoltat de Universitatea California din Berkeley, Statele Unite ale Americii.

PostgreSQL rulează pe toate sistemele de operare majore, iar reputația sa se datorează arhitecturii sale, fiabilității, integrității datelor, extensibilității și setului robust de caracteristici. De asemenea, este dedicat comunității open-source, oferind constant soluții performante și inovatoare.

Totodată, este un sistem care încearcă să se conformeze standarului SQL și de aceea, începând cu versiunea 16 lansată în septembrie 2023, acesta este conform cu 170 din cele 177 caracteristici obilgatorii pentru conformitatea SQL:2023 Core.

Din punct de vedere al caracteristiciilor oferite, PostgreSQL oferă:

* **Tipuri de date -** primitive (integer, numeric, String, Boolean), structurate (dată/oră, interval/multi-interval), de tip document (JSON, XML), geometrice (punct, linie, cerc, poligon);
* **Integritate date -** chei primare, chei străine, constrângeri de excludere, evitare date nule, date unice, blocaje
* **Concurență și performanță –** userii pot lucra în mod simultan, fără alterarea datelor și cu performanță ridicată;
* **Fiabilitate și recuperare în caz de dezastru**
* **Securitate -**  inclusiv autentificare de tip multi-factor cu certificat;
* **Extensibilitate** – funcții stocate și proceduri, limbaje procedurale (PL/pgSQL, Phyton, Perl)
* **Internaționalizare** – suport pentru seturi internaționale de caractere, corelații care nu țin cont de majuscule, căutarea textului integral.

PostgreSQL are multe caracteristici dedicate să ajute dezvoltatorii să creeze aplicații, administratorii să protejeze integritatea datelor, indiferent de dimensiunea setului de date.

Pe lângă faptul că este un sistem gratuit, acesta este și foarte extensibil. Spre exemplu, se pot crea tipuri de date și funcții personalizate. De asemenea, oferă posibilitatea scrierii de cod în diferite limbaje de programare, fără să fie nevoie de reconstrucția bazei de date.

## 2.2. Dezvoltarea server-side (*Back-End)*

### 2.2.1 Spring – Spring Boot - introducere



Spring este cel mai popular framework folosit pentru a dezvolta aplicații pe baza limbajului Java. Este puternic și extrem de flexibil, fiind axat pe aplicații de tip web.

Unul din principalele avantaje ale Sprin este că se ocupă de majoritatea părților de nivel scăzut ale construirii unei aplicații, permițând focusul pe partea de bussines logic.

Spring este cunoscut pentru capacitățiile sale de Inversion of Control (IoC) și Dependency Injection (DI) care simplifică crearea de aplicații modulare și testabile. Caracteristicile cheie includ Spring MVC pentru dezvoltarea web, Spring Boot pentru configurarea rapidă a aplicațiilor și Spring Security pentru autentificare și autorizare robuste. Cu un ecosistem bogat care acoperă Spring Data pentru interacțiunile cu bazele de date și Spring Cloud pentru crearea de microservicii, Spring susține soluții de întreprindere scalabile și rezistente, făcându-l un cadru esențial pentru dezvoltatorii de toate nivelurile de experiență.

Din punc t de vedere al elementelor care fac Spring să strălucească, acestea ar fi:

* **Utilitate** – oferă multiple opțiuni de configurare și astfel este ușor de folosit de către dezvoltatori;
* **Modularitate** – există opțiunea de a utiliza întreg framework-ul Spring sau doar modulele necesare (Spring Security, Spring Web, Spring Data, Spring Boot, Spring Cloud etc.);
* **Aplicații web securizate** – oferă setări de securitate personalizate pentru autentificare de bază și previne vulnerabilitățiile;
* **Suportă modelul MVC (Model View Controller)** – acest model este folosit pentru aplicații web și permite separarea diferitelor aspecte ale acestora (interfață utilizator, data acces layer, business layer);
* **Comunicare ușoară cu baza de date**;
* **Permite integrarea cu alte framework-uri** – de exemplu Hibernate, operațiuniile de tip CRUD (Create, Read, Update, Delete) fiind mai ușor de implementat;
* **Inversion of control (IoC)** – permite transferul controlului asupra obiectelor către un framework;
* **Dependency injection** **(DI)** – modulele unui proiect pot fi utilizate în mod eficient în alt proiect (de exemplu, paginiile de logare și autentificare). Ideea de DI este aceea de a conecta obiectele între ele, astfel încât un obiect primește dependețele necesare de la o sursă externă în loc să le creeze el însuși;
* **Testare ușoară a unei aplicații**.



Spring Boot reprezină o extindere a framework-ului Spring și a fost dezvoltat pentru a simplifica procesul de configurare și de a permite crearea mai rapidă a aplicațiilor. În Spring Boot totul este auto-configurat.

Principalele caracteristici ale Spring Boot sunt:

* **Evitarea configurării grele prezentă în Spring, prin fișiere de tip XML:**
* **Incorporarea server-ului Apache Tomcat și altele asemenea (Jetty, Undertow)** – acesta reprezintă un server web care deservește aplicații de Java de tip web. Practic, este un program care ajută aplicațiile web Java să funcționeze și să fie accesibile utilizatorilor prin intermediul unui browser;
* **Util în arhitectura bazată pe microservicii** – fiecare serviciu care rulează într-o aplicație are propriul său proces, astfel serviciile fiind independente între ele. De exemplu, într-o aplicație de tip magazin online, microserviciile pot fi: serviciul de comandă, serviciul de gestionare useri, serviciul de vizualizare produse etc.;
* **Crește productivitatea și reduce timpul de dezvoltare a aplicațiilor;**
* **Integrare ușoară a altor module Spring –** precum Spring JDBC, Spring Security, Spring Web
* **Facilitează conectarea la baze de date -**  prin utilizarea unor simple adnotări (@Entity, @Table, @ Column), Spring Boot permite accesul și chiar crearea de tabele în tool-uri precum PostgreSQL, Oracle, MySQL etc.;
* **Permite comasarea adnotărilor -** de exemplu, @SpringBootApplication integrează adnotările din Spring: @Configuration, @ComponentScan, @EnableAutoConfiguration;
* **Oferă posibilitatea creării de endpoint-uri de tip REST (Representational State Transfer) –** practic construirea unei aplicații sau serviciu de tip web, prin care se creează comunicarea între un client și un server.

Spring Boot a fost creat în 2013 și lansat pentru prima dată în aprilie 2014. Versiunile sale pot fi vizualizate la adresa:

<https://mvnrepository.com/artifact/org.springframework.boot/spring-boot>

### 2.2.2 Spring Boot – arhitectură, elemente specifice și funcționare

1. **Arhitectura Spring Boot**

Are la bază patru nivele principale și anume:

* **Presentation layer**: reprezeintă partea de vizualizare sau, mai simplu spus, partea pe care o vede utilizatorul (*frontend*);
* **Data access layer**: reprezintă nivelul unde au loc operațiunile legate de baza de date – CRUD (*Create, Read, Update, Delete*)
* **Service Layer:** conține clasele de servicii și folosește serviciile furnizate de data access layer;
* **Integration layer:** se referă la diferite tipuri de servicii web, practic orice serviciu disponibil online și utilizează sistemul de mesagerie XML);

Pe lângă acestea, mai există clasele de utilitate, cele de validare și cele de vizualizare.

A diagram of a spring boot flow

Description automatically generated

Arhitectura Spring Boot

1. **Terminologie Spring Boot**

|  |  |
| --- | --- |
| TERMEN | DESCRIERERE |
| Actuator | Modul ce oferă funcționalități pregătite pentru producție, cum ar fi verificarea stării (health checks), metrici și puncte de monitorizare. |
| Bean | O componentă gestionată de containerul Spring, creată de obicei cu ajutorul anotărilor @Bean, @Component sau @Service. |
| Banner | Textul ASCII sau graficul afișat la pornirea aplicației, care poate fi configurat prin banner.txt. |
| Clasa SpringApplication | |  | | --- | |  |  |  | | --- | | Punctul de intrare pentru lansarea unei aplicații Spring Boot, folosită de obicei în metoda main. | |
| CommandLineRunner | O interfață utilizată pentru a rula cod specific după pornirea aplicației Spring Boot, implementată de obicei în bean-uri personalizate. |
| Controller | O clasă notată cu @Controller sau @RestController care gestionează cererile HTTP și răspunsurile. |
| DevTools | Un modul Spring Boot pentru productivitatea dezvoltatorilor, care permite reporniri automate și reîncărcări live în timpul dezvoltării. |
| Embedded Server | |  | | --- | |  |  |  | | --- | | Un server web integrat (cum ar fi Tomcat, Jetty sau Undertow) care elimină necesitatea unui server extern pentru a rula aplicația. | |
| Endpoint | |  | | --- | |  |  |  | | --- | | O cale URL (de obicei expusă de un controller) sau o funcționalitate a Actuator care oferă funcții specifice (ex. /api/v1/resource, /actuator/info). | |
| Entity | O clasă notată cu @Entity ce reprezintă un tabel din baza de date în JPA. |
| Fișier de proprietăți/YAML | Fișiere de configurare (application.properties sau application.yml) pentru setările aplicației și configurații specifice mediului. |
| Health Indicators | Funcționalități ale Spring Boot Actuator care raportează starea aplicației (ex. conectivitate la baza de date, spațiu pe disc). |
| Injecție de Dependențe (DI) | Procesul prin care Spring Boot furnizează automat bean-uri sau servicii către alte componente (prin @Autowired sau injecție prin constructor). |
| Logging | Gestionat prin SLF4J și Logback în mod implicit, configurabil prin application.properties. |
| Profile | Un mecanism pentru definirea configurațiilor specifice mediului, activate prin spring.profiles.active (ex. dev, test, prod). |
| Repository | O interfață notată cu @Repository, folosită pentru operații CRUD și interacțiuni cu baza de date, de obicei extinsă din JpaRepository. |
| Server încorporat | Un server web integrat (cum ar fi Tomcat, Jetty sau Undertow) care elimină necesitatea unui server extern pentru a rula aplicația. |
| Spring Boot Auto-Configuration | Configurare automată a bean-urilor Spring pe baza dependențelor și proprietăților existente în classpath. |
| Spring Boot CLI | Un instrument de linie de comandă pentru rularea aplicațiilor Spring Boot și scripturilor scrise în Groovy. |
| Spring Boot Initializr | Un instrument online pentru generarea de proiecte Spring Boot cu dependențe preconfigurate (disponibil la [start.spring.io](https://start.spring.io)). |
| Spring Boot Starter | Dependențe preconfigurate care simplifică gestionarea dependențelor (ex. spring-boot-starter-web, spring-boot-starter-data-jpa). |
| Spring Cloud | O suită de instrumente pentru construirea microserviciilor, cu funcții precum descoperirea serviciilor, configurarea și echilibrarea încărcării. |
| Spring Data JPA | Modul ce simplifică accesul și gestionarea bazelor de date prin intermediul repository-urilor JPA cu suport încorporat pentru operații CRUD. |
| Spring MVC | Un framework pentru construirea serviciilor RESTful sau a aplicațiilor web, utilizând anotări precum @Controller și @RestController. |
| Spring Security | Un modul Spring pentru securizarea aplicațiilor, oferind autentificare, autorizare și protecție CSRF. |
| Thymeleaf | Un motor de șabloane pe partea de server, utilizat frecvent în Spring Boot pentru a genera conținut HTML dinamic. |

**Tabel 1 – Terminologia specifică Spring Boot**

1. **Adnotări specifice Spring Boot**

O adnotare (în engleză: annotation) este un construct specific limbajului Java (și altor limbaje derivate), folosit pentru a atașa metadate (informații suplimentare) la codul sursă, cum ar fi clase, metode, câmpuri, sau parametri. Adnotările nu afectează direct logica aplicației, dar pot influența comportamentul programului prin intermediul procesării la rulare (runtime) sau la compilare.

|  |  |
| --- | --- |
| ADNOTARE | DESCRIERE |
| @Autowired | Permite injecția automată a unei dependențe gestionate de Spring. |
| @Bean | Definește un bean care va fi gestionat de Spring. |
| @Column | Specifică detalii despre o coloană din tabelul asociat cu o entitate JPA. |
| @Component | |  | | --- | |  |  |  | | --- | | Marchează o clasă ca fiind un bean gestionat de containerul Spring (componentă generică). | |
| @ComponentScan | Specifică pachetele care trebuie scanate pentru componente și bean-uri. |
| @Configuration | Marchează o clasă ca sursă de definiții pentru bean-uri Spring. |
| @DeleteMapping | Maparea unei metode la o cerere HTTP de tip DELETE. |
| @Entity | Definește o clasă ca fiind o entitate JPA, care va fi mapată la un tabel din baza de date. |
| @EnableAutoConfiguration | Activează mecanismul de configurare automată din Spring Boot. |
| @GeneratedValue | |  | | --- | |  |  |  | | --- | | Indică modul în care valoarea cheii primare este generată (ex. automat, secvențial etc.). | |
| @GetMapping | Maparea unei metode la o cerere HTTP de tip GET. |
| @Id | Marchează câmpul ca fiind cheia primară a unei entități JPA. |
| @PathVariable | Extrage o variabilă de cale din URL (ex. /api/{id}). |
| @PostMapping | Maparea unei metode la o cerere HTTP de tip POST. |
| @Primary | |  | | --- | |  |  |  | | --- | | Indică bean-ul implicit care să fie utilizat în cazul în care există mai multe potriviri. | |
| @Profile | |  | | --- | |  |  |  | | --- | | Specifică mediul în care un bean sau o configurație va fi activă (ex. dev, prod). | |
| @PutMapping | Maparea unei metode la o cerere HTTP de tip PUT. |
| @Qualifier | Specifică ce bean să fie injectat atunci când există mai multe candidaturi posibile. |
| @Repository | Marchează o clasă ca fiind responsabilă pentru interacțiunea cu baza de date. |
| @RequestBody | Convertește corpul unei cereri HTTP într-un obiect Java. |
| @RestController | Combina @Controller și @ResponseBody, returnând date JSON sau XML pentru cererile HTTP. |
| @RestTemplate | Creează un obiect de tip RestTemplate pentru a face apeluri HTTP către alte servicii web. |
| @Service | O specializare a @Component, utilizată pentru a marca clasele care conțin logica de business. |
| @SpringBootApplication | Indică punctul de intrare principal al unei aplicații Spring Boot. Combina @Configuration, @EnableAutoConfiguration și @ComponentScan. |
| @Table | Specifică numele tabelului asociat cu o entitate JPA. |
| @Transactional | Marchează o metodă sau o clasă pentru a executa operații într-o tranzacție. |
| @Value | Injecția unei valori din proprietăți externe (application.properties sau application.yml). |

**Tabel 2 – Adnotări specifice Spring Boot**

1. **Crearea unui proiect**

Initițializarea unui proiect care utilizează Spring Boot se poate face prin două metode. Prima metodă este cea de folosire a ceea ce se numește **spring initializr** (<https://start.spring.io/> ). Cea de-a două metodă se referă la crearea unui proiect direct din IDE-ul IntelliJ IDEA.

Referitor la prima metodă reprezintă un mod de a seta foarte rapid un proiect nou, fără a fi nevoie de configurare manuală.

A screenshot of a computer

Description automatically generated

Asftel, spring initializr este divizat în două parți. Partea din stânga se referă la setările privind tipul de proiect, limbajul de programare, versiunea de Spring Boot și datele de tip metadata:

* **Gradle – Groovy, Gradle – Kotlin, Maven** - sunt instrumente de automatizare a construirii proiectelor în ecosistemul Java. Ele ajută la gestionarea dependințelor, compilarea codului, rularea testelor și construirea aplicațiilor;
* **3.x.x / 3.x.x SNAPSHOT –** reprezintă versiunile cele mai recente ale Spring Boot, varianta snapshot fiind o versiune în dezvoltare;
* **Group** - este un identificator unic pentru grupul sau organizația care deține proiectul;
* **Artifact** - identificator unic pentru proiectul sau modulul efectiv;
* **Name –** denumirea proiectului;
* **Description** – scurtă descriere a proiectului ;
* **Package name** – pachetul în care sunt incluse clasele Java;
* **Packaging** - pecifică tipul de fișier care va fi creat în urma procesului de construcție. Indică ce tip de artefact va fi produs (JAR, WAR, EAR etc.) ;
* **Java** – versiunea de Java ce urmează a fi folosită.

Partea din dreapta oferă posibilitatea de a adăugare automată a dependințelor care se doresc a fi utilizate în proiect, cum ar fi: Spring Web, Thymeleaf, Spring Security, Spring Data Jpa etc.

A screenshot of a computer

Description automatically generated

În continuare, prin selectarea butonului A white rectangular sign with black text

Description automatically generated se poate vedea o pre-vizualizare a proiectului:

A screenshot of a computer

Description automatically generated

Butonul A black and white rectangle with black text

Description automatically generated este utilizat pentru generarea proiectului sub formă de arhiva .zip, acesta fiind ulterior deschisa în IntelliJ IDEA.