Aplicatie informatica pentru gestionareaactivitatii din cadrul unei cafenele

# Documentul de proiectare

Cuprins

[1. Introducere 1](#_Toc160527836)

[1.1 Scopul documentului 1](#_Toc160527837)

[2. Prezentare generală și abordări de proiectare 2](#_Toc160527838)

[2.1 Prezentare generală 2](#_Toc160527839)

[2.2 Presupuneri/ Constrângeri/ Riscuri 2](#_Toc160527840)

[2.2.1 Presupuneri 2](#_Toc160527841)

[2.2.2 Constrângeri 2](#_Toc160527842)

[2.2.3 Riscuri 3](#_Toc160527843)

[3. Considerațiii de proiectare 4](#_Toc160527844)

[3.1 Obiective și linii directoare (ghiduri) 4](#_Toc160527845)

[3.2 Metode de dezvoltare 4](#_Toc160527846)

[3.3 Strategii de arhitectură 4](#_Toc160527847)

[4. Arhitectura Sistemului și Proiectarea Arhitecturii 6](#_Toc160527848)

[4.1 Vedere logică 6](#_Toc160527849)

[4.2 Arhitectură hardware 6](#_Toc160527850)

[4.3 Arhitectură software 6](#_Toc160527851)

[4.4 Arhitectura informațiilor 7](#_Toc160527852)

[4.5 Arhitectura de comunicații interne 7](#_Toc160527853)

[4.6 Diagrama de arhitectură a sistemului 8](#_Toc160527854)

[5. Proiectarea sistemului 9](#_Toc160527855)

[5.1 Proiectarea bazei de date 9](#_Toc160527856)

[5.1.1 Obiecte de date și structuri de date rezultante 9](#_Toc160527857)

[5.1.2 Fișiere și baze de date 9](#_Toc160527858)

[5.2 Conversii de date 9](#_Toc160527859)

[5.3 Interfețe utilizator 10](#_Toc160527860)

[5.3.1 Intrări 10](#_Toc160527861)

[5.3.2 Ieșiri 10](#_Toc160527862)

[5.4 Proiectarea interfețelor cu utilizatorul 10](#_Toc160527863)

[6. Scenarii de utilizare 11](#_Toc160527864)

[7. Proiectare de detaliu 12](#_Toc160527865)

[7.1 Proiectare hardware de detaliu 12](#_Toc160527866)

[7.2 Proiectare software de deatliu 12](#_Toc160527867)

[7.3 Proiectare detaliată de securitate 13](#_Toc160527868)

[7.4 Proiectare de detaliu pentru performanța sistemului 14](#_Toc160527869)

[7.5 Proiectare detaliată a comunicațiilor interne (între componente) 14](#_Toc160527870)

[8. Controale pentru verificarea integrității sistemului 15](#_Toc160527871)

[Anexa A: Gestiunea modificărilor documentului 16](#_Toc160527872)

[Anexa B: Acronime 17](#_Toc160527873)

[Anexa C Documente la care se face referire 18](#_Toc160527874)

## Introducere

Acest document reprezintă Documentul de Proiectare a Sistemului pentru aplicația web MysticCafe – o soluție software modernă destinată unei cafenele online, care oferă utilizatorilor posibilitatea de a comanda, personaliza și ridica băuturi direct din locație, cât și de a efectua plăți online.

Proiectul MysticCafe a fost dezvoltat utilizând tehnologii moderne precum Spring Boot pentru backend, HTML/CSS/JavaScript pentru frontend, și Oracle SQL Developer pentru gestiunea datelor. Documentul detaliază modul în care cerințele funcționale și non-funcționale identificate anterior sunt transpuse în specificații tehnice concrete și designul sistemului.

Securitatea datelor utilizatorilor este o prioritate, în special în contextul procesării plăților online prin Stripe. Datele sensibile precum parolele sunt criptate, iar toate comunicațiile sunt securizate prin HTTPS. Accesul la interfața de administrator este protejat și rezervat exclusiv personalului autorizat.

Documentul de Proiectare a Sistemului descrie modul în care cerințele funcționale și non-funcționale înregistrate în Documentul de Cerințe se transformă în specificații de proiectare a sistemului, mai tehnice, pe baza cărora se construiește sistemul. Se documentează proiectarea sistemului la un nivel înalt și specificațiile detaliate de proiectare.

Se descriu obiectivele și considerațiile de proiectare, se furnizează o prezentare generală la nivel înalt a arhitecturii sistemului și se descrie proiectarea datelor asociate cu sistemul, precum și interfața om-mașină și scenariile operaționale.

Proiectarea sistemului la nivel înalt este descompusă în continuare în specificații detaliate de proiectare pentru fiecare componentă a sistemului, inclusiv hardware, comunicațiile interne, software, controalele de integritate a sistemului și interfețele externe.

### Scopul documentului

Prin acest livrabil se documentează și urmăresc informațiile necesare pentru a defini eficient arhitectura și designul sistemului, în scopul de a oferi echipei de dezvoltare îndrumare asupra arhitecturii sistemului ce urmează să fie dezvoltat.

Documentele de proiectare sunt produse incremental și iterativ pe parcursul ciclului de viață al dezvoltării sistemului, în funcție de circumstanțele particulare ale proiectului de tehnologie informațională (IT) și de metodologia de dezvoltare a sistemului utilizată.

Publicul țintă este managerul de proiect, echipa de proiect și echipa de dezvoltare. Anumite părți ale acestui document, cum ar fi interfața cu utilizatorul (UI), pot fi împărtășite cu clientul/utilizatorul și cu alte părți interesate ale căror contribuții/aprobări sunt necesare în UI.

## Prezentare generală și abordări de proiectare

Această secțiune descrie principiile și strategiile care vor fi utilizate ca ghiduri în momentul proiectării și implementării sistemului.

### Prezentare generală

MysticCafe este o aplicație web dedicată unei cafenele care dorește digitalizarea procesului de vânzare și interacțiune cu clienții. Sistemul este organizat modular, cu componente separate pentru utilizatori, comenzi, produse, plăți și administrare.

Obiectivele de proiectare includ:o experiență fluentă și intuitivă pentru utilizatori, accesibilitate pe desktop, automatizarea procesului de fidelizare (1 băutură gratis la fiecare 5 cumpărate), interfață de administrare completă pentru gestionarea produselor, comenzilor și statisticilor.

### Presupuneri/ Constrângeri/ Riscuri

#### Presupuneri

Stripe este disponibil și funcțional în regiunea în care aplicația este folosită.

Datele utilizatorilor (conturi, comenzi, etc.) sunt stocate în Oracle DB.

Aplicația rulează local sau pe un server privat (nu în cloud).

#### Constrângeri

#### Riscuri

Stripe API poate suferi modificări ce necesită ajustări în cod.

Pierdere de date dacă baza de date nu este salvată periodic (backup).

## Considerații de proiectare

### Obiective și linii directoare (ghiduri)

### Metode de dezvoltare

Pentru dezvoltarea aplicației web „Cafenea Online”, a fost adoptată o abordare orientată pe obiecte, utilizând framework-ul Spring Boot pentru backend, iar pentru frontend s-au folosit HTML, CSS și JavaScript. S-a lucrat local în Visual Studio Code și s-a utilizat Oracle SQL Developer pentru gestionarea bazei de date.

Structura aplicației a fost proiectată folosind principiile arhitecturale MVC (Model-View-Controller), iar pentru modelarea logicii aplicației s-au folosit diagrame UML de tip caz de utilizare și diagramă de clase.

### Strategii de arhitectură

Arhitectura aplicației este de tip web-based client-server, împărțită logic în trei straturi: Frontend (View): HTML, CSS și JS pentru interfața grafică a utilizatorului, Backend (Controller + Business Logic): Spring Boot – gestionează logica de autentificare, procesarea comenzilor, sistemul de recompense etc., Persistență (Model): Oracle SQL Developer – pentru gestionarea datelor (utilizatori, produse, comenzi, statistici)

## Arhitectura Sistemului și Proiectarea Arhitecturii

Aplicația este structurată conform unui model **client-server**, utilizând o arhitectură **multi-tier** care separă clar responsabilitățile între front-end, back-end și sistemul de gestionare a datelor.

**Front-end este d**ezvoltat cu **HTML, CSS și JavaScript**, reprezintă interfața utilizatorului și este responsabil de colectarea input-ului utilizatorului, afișarea datelor și transmiterea cererilor către server.

**Back-end este i**mplementat în **Java** folosind **Spring Boot**, gestionează logica aplicației, prelucrează cererile clientului și comunică cu baza de date. Expune un set de **servicii REST API** utilizate de front-end pentru schimbul de date.

**Baza de date folosita este Oracle Database si** este utilizată pentru stocarea datelor persistente. Serverul de back-end interacționează cu baza de date folosind framework-uri de persistență standard (ex: JPA/Hibernate).

**Interacțiunea între componente:** Front-end-ul transmite cereri HTTP către back-end folosind metodele standard REST (GET, POST, PUT, DELETE). Back-end-ul procesează aceste cereri, execută logica de business, realizează operații asupra bazei de date și răspunde clientului cu date structurate în format **JSON**.

**Justificarea particionării sistemului:**

Alegerea separării în trei straturi (client, server, bază de date) permite o **separare clară a responsabilităților**, **scalabilitate** și **întreținere ușoară** a aplicației.

Această structură suportă o dezvoltare independentă a fiecărei componente și permite extinderea sistemului fără modificări majore ale back-end-ului sau bazei de date.

**Tipare de proiectare utilizate: MVC (Model-View-Controller)**: utilizat implicit în structura Spring Boot pentru organizarea codului serverului (Controller - gestionează cererile, Service - implementează logica de business, Repository - interacționează cu baza de date).

**Alternative luate în considerare și respinse:**

S-a evaluat folosirea unui framework SPA (Single Page Application) dedicat (ex: React, Angular), dar s-a ales soluția simplificată HTML/JS/CSS datorită cerințelor proiectului și a complexității reduse a interfeței.

Alte baze de date (MySQL, PostgreSQL) au fost luate în considerare, însă s-a optat pentru Oracle datorită cerințelor de integrare cu alte sisteme enterprise existente.

### Vedere logică

Instrucțiuni: Introduceți orice vederi logice relevante sau furnizați o referință către locul în care sunt stocate.

Aplicația urmează modelul MVC:

Model – clasele Java corespunzătoare entităților (User, Order, Product etc.)

View – paginile HTML cu CSS/JS

Controller – rutele și logica din Spring Boot

UML:

A diagram of a diagram

AI-generated content may be incorrect.

### Arhitectură hardware

Sistemul este centralizat, testat și rulând local pe o stație de lucru (localhost).

Este necesar accesul la internet pentru plățile online via Stripe.

### Arhitectură software

Aplicația "MysticCafe" a fost dezvoltată utilizând o arhitectură software de tip trei straturi, care include: stratul de prezentare (frontend), stratul de aplicație (backend) și stratul de date (baza de date). Această separare clară permite o dezvoltare modulară, flexibilă și scalabilă a aplicației.

Pentru realizarea interfeței utilizatorului am folosit HTML5, CSS3 și JavaScript, tehnologii standard în dezvoltarea web, care permit crearea unor pagini web responsive și interactive.

Backend-ul aplicației a fost realizat în limbajul Java, utilizând framework-ul open-source Spring Boot. Acesta permite o configurare rapidă a aplicației și integrarea facilă a diferitelor module, precum gestionarea cererilor HTTP prin Spring Web, accesarea și manipularea datelor cu ajutorul Spring Data JPA, și protecția operațiunilor sensibile prin Spring Security. Baza de date relațională utilizată este Oracle Database 21c, administrată și modelată cu ajutorul platformei Oracle SQL Developer.

În ceea ce privește instrumentele utilizate în procesul de dezvoltare, am folosit Visual Studio Code ca mediu principal de programare

Aplicația integrează mai multe module software esențiale: autentificare și înregistrare utilizatori, catalog de produse, coș de cumpărături, programare a ridicării comenzilor, plăți online, recenzii, quiz interactiv pentru recomandarea băuturilor potrivite, sistem de fidelizare prin care utilizatorul primește o băutură gratuită după achiziționarea a cinci produse, precum și o interfață de administrator pentru gestionarea comenzilor, utilizatorilor și vizualizarea statisticilor și a profitului.

Din punct de vedere al fluxului de date, aplicația funcționează printr-un ciclu continuu între browserul utilizatorului, serverul Spring Boot și baza de date Oracle. Datele utilizatorului sunt prelucrate într-un mod sigur și eficient, asigurând o experiență fluentă și plăcută.

### Arhitectura informațiilor

Date gestionate: date utilizatori: nume, email, parolă (hash), produse: denumire, tip, preț, stoc, comenzi: produse comandate, oră ridicare, metodă plată, statistici: total profit, top produse, comenzi pe zi.

Datele sunt introduse electronic de către utilizator sau administrator și sunt salvate în baza de date Oracle.

### Arhitectura de comunicații interne

Comunicația între componentele aplicației se face astfel:

* **Frontend ↔ Backend**: HTTP (REST API)
* **Backend ↔ Bază de date**: JDBC – conexiune securizată
* **Backend ↔ Stripe**: API extern prin HTTPS

Rețeaua locală permite testarea pe localhost.

### Diagrama de arhitectură a sistemului

[Browser Utilizator]

↓

[Frontend HTML/CSS/JS]

↓ REST API

[Java Backend + Springboot framework]

↓ JDBC

[Oracle DB]

## Proiectarea sistemului

### Proiectarea bazei de date

Tabele și structuri de date:

* Utilizatori:

user\_id (INT, PK, AUTO\_INCREMENT): Identificator unic al utilizatorului.

username (VARCHAR(255), NOT NULL): Numele de utilizator.

password (VARCHAR(255), NOT NULL): Parola criptată.

email (VARCHAR(255), NOT NULL): Adresa de email.

#### Obiecte de date și structuri de date rezultante

Instrucțiuni: Pentru fiecare obiect de date funcțional, specificați structura/ile de date care va/vor fi utilizată/e pentru a stoca și procesa datele. Descrieți orice structuri de date care sunt o parte importantă a sistemului, inclusiv structurile de date majore care sunt transmise între componente.

Acestea sunt descrierea tabelelor și relațiile dintre ele, de exemplu:

**Frontend (JavaScript):** Structură: Obiect JavaScript { id: number, username: string, email: string }

**Backend (Java, Spring Boot):** Structură: Entitate User Tip de date: Clasă Java

public class User{

private Long id;

private String username;

private String email;

}

**Transfer între Frontend și Backend:**Format: JSON

{

"id": 1,

"username": "john\_doe",

"email": "john@example.com"}

· **Baza de date (Oracle):**Structură: Tabel Users cu coloane:

ID (NUMBER),

USERNAME (VARCHAR2(50)),

EMAIL (VARCHAR2(100))

#### Fișiere și baze de date

Aplicația folosește o bază de date relațională Oracle SQL Developer.

Structura bazei de date va fi un set de tabele relaționale, cu relații între acestea pentru a menține integritatea datelor.

Actualizările bazei de date se fac prin query-uri SQL si JPA Repository (INSERT, UPDATE, DELETE).

##### Baze de date

A screenshot of a data table

AI-generated content may be incorrect.

##### Fișiere non-DBMS

Exemplu de fisier non-DBMS:

Lista de produse extrase din baza de date de catre JPA Repository in backend si trimise catre front-end sub forma de JSON(non-DBMS):  
  
[

{

"id": 26,

"productName": "Esspresso",

"productDescription": "lactoza, nuci, alune, gluten, oua si produse dervivate",

"productAllergies": "nuci, oua",

"imagePath": "http://localhost:8600/mystic-cafe/uploads/images/products/menu-3.png",

"price": 4.99,

"quantity": -1,

"packaged": false

},

{

"id": 21,

"productName": "Nicaragua",

"productDescription": "lactoza, nuci, alune, gluten, oua si produse dervivate",

"productAllergies": "lactoza, nuci, alune, gluten, oua si produse dervivate",

"imagePath": "http://localhost:8600/mystic-cafe/uploads/images/products/product-1.png",

"price": 17.99,

"quantity": 25,

"packaged": true

}]

### Conversii de date

### Interfețe utilizator

Utilizatori (Clients): Pot plasa comenzi, se pot loga, adăuga produse în coș, plăti online și vizualiza istoricul comenzii.

Administratori: Pot vizualiza comenzile, gestionarea stocurilor și promoțiile, și pot obține statistici despre vânzări.

#### Intrări

Datele vor fi introduse prin interfețele de utilizator (formulare HTML) pentru înregistrare, login, plasarea comenzii, și completarea plății.

#### Ieșiri

Ieșirile vor include rapoarte pentru administratori (statistici de vânzări, profituri, produse populare etc.), iar pentru clienți, confirmări de comandă, notificări de plată

### Proiectarea interfețelor cu utilizatorul

.

## Scenarii de utilizare

**Crearea unui cont/logare utilizator**:

Eveniment: Utilizatorul acceseaza site-ul cafenelei si doreste sa se logheze sau sa isi faca un cont.

Actiuni: acceseaza formularul de logare/inregisrare, completeaza campurile necesare apoi apasa pe butonul de logare/inregistrare.

Sistemul validează informațiile (verifică dacă adresa de e-mail nu este deja înregistrată) și trimite un e-mail de confirmare.

Sistemul salvează datele utilizatorului în baza de date.

**Plasarea unei comenzi:**

Eveniment: Utilizatorul doreste sa plaseze o comanda.

Actiuni: Utilizatorul acceseaza pagica cu produse, adauga in cos produse, apasa butonul de plaseaza comanda, completeaza informatiile necesare.

Sistemul validează produsele, calculează prețul total și transmite cererea de plată către procesatorul de plăți (Stripe).

Detaliile comenzii sunt salvate în baza de date.

Dacă plata este reușită, comanda este finalizată și utilizatorul primește un e-mail.

## Proiectare de detaliu

Instrucțiuni: Furnizați informațiile necesare echipei de dezvoltare a sistemului pentru a construi și integra efectiv componentele hardware, pentru a programa și integra componentele software, și pentru a interconecta segmentele hardware și software într-un produs funcțional.

### Proiectare hardware de detaliu

În cadrul acestui proiect, nu sunt implicate componente hardware complexe, întrucât aplicația este o soluție software bazată pe web.

### Proiectare software de detaliu

Serviciul **UsersService** este un serviciu de business logic responsabil pentru gestionarea utilizatorilor în sistem, inclusiv operații precum obținerea listei tuturor utilizatorilor, crearea unui utilizator nou și autentificarea acestuia. Acest serviciu interacționează cu repository-urile **UsersRepository** și **ShoppingCartsRepository** pentru accesul și manipularea datelor stocate în baza de date **Oracle**. Pentru operația de creare a unui utilizator, serviciul inițiază un coș de cumpărături gol, îl salvează și îl asociază utilizatorului înainte de a persista obiectul **User** în baza de date, asigurând astfel integritatea relațiilor între entități. În procesul de autentificare, serviciul caută utilizatorul după email și verifică potrivirea parolei în mod direct, returnând utilizatorul dacă datele sunt valide sau null în caz contrar. Structurile de date interne folosite sunt entitățile **User** și **ShoppingCart**. Printre constrângerile serviciului se numără validarea parolei și gestionarea corectă a asocierii între utilizator și coșul de cumpărături. Serviciul colaborează cu nivelul de acces la date prin repository-uri și cu controller-ele **REST** care expun aceste funcționalități către frontend. Metodele oferite sunt simple și eficiente, iar procesele sunt realizate sincron, fără concurență explicită, cu focus pe manipularea corectă a stării în baza de date. Interfețele exportate sunt funcțiile getAll(), createUser(User user) și login(User user), care permit extragerea datelor, crearea și autentificarea utilizatorilor în sistem.

### Proiectare detaliată de securitate

### Proiectare de detaliu pentru performanța sistemului

Cerințe/estimări de capacitate: Sistemul trebuie să gestioneze simultan cel puțin 1000 de utilizatori activi.

Așteptări de performanță: Timpi de răspuns sub 2 secunde pentru majoritatea cererilor.

### Proiectare detaliată a comunicațiilor interne (între componente)

Sistemul este format din trei componente principale care rulează local pe aceeași mașină sau în rețea locală: frontend-ul (JavaScript, HTML, CSS), backend-ul Spring Boot și baza de date Oracle. Frontend-ul rulează pe un server local sau direct în browser, accesând backend-ul prin apeluri REST API pe un port local specific (ex: 8080 pentru backend). Backend-ul comunică cu baza de date Oracle care este instalată local sau în rețeaua internă, folosind JDBC pentru accesul la date. Numărul de servere este unul sau mai multe, în funcție de configurarea dezvoltării și testării, iar clienții sunt utilizatori care accesează aplicația pe mașina locală sau în rețea LAN. Comunicarea între componente se face prin protocoale locale (HTTP pentru frontend-backend, SQL pentru backend-bază de date), iar datele transferate sunt în format JSON. Sincronizarea este simplă, fără control complex al busului, deoarece sistemul rulează local și cererile HTTP sunt stateless. Topologia rețelei LAN este simplă, adesea stea, în care toate componentele sunt conectate printr-un switch sau direct pe aceeași mașină, iar distanțele fizice sunt reduse, eliminând astfel probleme majore de latență sau securitate. Această arhitectură locală simplifică comunicarea și permite un ciclu rapid de dezvoltare și testare.

## Controale pentru verificarea integrității sistemului

Instrucțiuni: Furnizați specificații de proiectare pentru următoarele nivele minime de control și orice controale suplimentare adecvate sau necesare:

* Securitate internă pentru a restricționa accesul la datele critice doar pentru acele tipuri de acces necesare de către utilizatori/operatori
* Proceduri de audit pentru a îndeplini cerințele de control, raportare și perioade de reținere pentru rapoartele operaționale și de management
* Piste de auditare a aplicațiilor pentru a audita dinamic accesul la recuperare la datele critice designate
* Tabele standard care urmează să fie utilizate sau solicitate pentru validarea câmpurilor de date
* Procese de verificare pentru adăugarea, ștergerea sau actualizarea datelor critice
* Capacitatea de a identifica toate informațiile de auditare prin identificarea utilizatorului, identificarea terminalului de rețea, dată, oră și datele accesate sau modificate.

Anexa A: Gestiunea modificărilor documentului

Instrucțiuni: Furnizați informații despre modul în care dezvoltarea și distribuția documentului va fi controlată și urmărită. Utilizați tabelul de mai jos pentru a furniza numărul de versiune, data versiunii, autorul/deținătorul versiunii și o scurtă descriere a motivului pentru crearea versiunii revizuite.

Tabel 1 – Înregistrarea modificărilor asupreaa documentului curent

| versiune | Data | Autorul/Deținătorul | Descriere |
| --- | --- | --- | --- |
| <X.X> | <ZZ/LL/AAAA> | <nume autor> | <Descrierea modificării> |
| <X.X> | <ZZ/LL/AAAA> | <nume autor> | <Descrierea modificării> |
| <X.X> | <ZZ/LL/AAAA> | <nume autor> | <Descrierea modificării> |

Anexa B: Acronime

*Instrucțiuni: Furnizați o listă de acronime și traduceri literale asociate utilizate în cadrul documentului. Enumerați acronimele în ordine alfabetică folosind un format tabular, așa cum este ilustrat mai jos.*

Tabel 2 - Acronime

| Acronim | Forma completă |
| --- | --- |
| <Acronim> | <Forma completă> |
| <Acronim> | <Forma completă> |
| <Acronim> | <Forma completă> |

Anexa C Documente la care se face referire

*Instrucțiuni: Sintetizați relația acestui document cu alte documente relevante. Furnizați informații de identificare pentru toate documentele folosite pentru a ajunge la și/sau referite în acest document (de exemplu, documente conexe și/sau asociate, documente prealabile, documentație tehnică relevantă, etc.).*

Tabel 3 – Documente la care se facce referire

| Nume document | Locație sau URL | Dată emitere document |
| --- | --- | --- |
| < Nume document > | <Locație sau URL> | <ZZ/LL/AAAA> |
| < Nume document > | <Locație sau URL> | <ZZ/LL/AAAA> |
| < Nume document > | <Locație sau URL> | <ZZ/LL/AAAA> |