

**MedCare**

**Nume**: Jarda Adina-Ionela

**Grupa**: 30232

**Cuprins**

1. Specificatiile Proiectului………………………………………….

2. Functionalitatile Proiectului………………………………………

3. Use Case Model…………………………………………………..

3.1 Identificarea Unui Use Case…………………………………

3.2 Diagarama Use Case………………………………………..

4. Diagrama De Clase…………………………………………………

5. Specificatii Suplimentare………………………………………….

1. **Specificatiile Proiectului**

Proiectul propus consta intr-o aplicatie pentru gestionarea programarilor pacientilor de la clinica MedCare. Administratorul clinicii doreste o aplicatie desktop care sa permita inregistrarea si gestionarea consultaiilor realizate de medici.

Proiectul implementeaza o aplicatie destktop pentru gestionarea acestei clinici, dezvoltata in Java, SpringBoot si Java Swing pentru interfata utilizator. Aplicatia este structurata pe arhitectura stratificata(Layers).

* Presentation Layer: Interfata utilizator dezvoltata in Java Swing
* Business Logic Layer: Servicii care implementeaza logica de business
* Data Access Layer: Repository-uri care faciliteaza accesul la date
* Data Layer: Entitatile de baza si modelul de date

Sistemul foloseste o baza de date MySQL pentru persistenta datelor si implementeaza mecanisme de securitate pentru autentificarea utilizatorilor si criptarea parolelor.

1. **Functionalitatile Proiectului**

Aplicatia ofera utmatoarele functionalitati:

1. **Autentificare si Autorizare**

 Sistemul permite accesul pe baza de login

 Exista doua tipuri de utilizatori: administratori și receptionisti

 Parolele sunt stocate criptat in baza de date

1. **Functionalitati pentru Administratori**

 Gestionarea conturilor pentru receptionisti (creare, vizualizare)

 Gestionarea listei de medici (adaugare, editare, stergere)

 Gestionarea serviciilor medicale oferite de clinica

 Generarea si vizualizarea rapoartelor si statisticilor

 Exportarea rapoartelor in format CSV sau XML

1. **Functionalitati pentru Receptionisti**

 Gestionarea programarilor pacientilor (adaugare, editare, stergere)

 Verificarea disponibilitatii medicilor

 Actualizarea statusului programarilor

 Cautarea si filtrarea programarilor

1. **Gestionarea Medicilor**

* Inregistrarea informatiilor despre medici (nume, specializare)
* Definirea programului de lucru pentru fiecare medic, pe zile ale saptamanii
* Verificarea disponibilitatii medicilor pentru programari

1. **Gestionarea Serviciilor Medicale**

* Inregistrarea serviciilor medicale oferite de clinica
* Definirea pretului si duratei fiecarui serviciu

1. **Gestionarea Programarilor**

* Inregistrarea programarilor pacientilor
* Verificarea disponibilitatii medicilor
* Evitarea suprapunerilor in programari
* Actualizarea statusului programarilor (nou, in desfasurare, finalizat)

1. **Rapoarte si Statistici**

* Generarea rapoartelor cu programari intre doua date calendaristice
* Statistici privind cei mai solicitați medici
* Statistici privind cele mai solicitate servicii medicale
* Exportarea rapoartelor in format CSV sau XML

1. **Use Case Model** 
   1. **Identificarea Unui Use Case**

**Use case**: Crearea unei programari noi

**Actor principal**: Receptionist

**Prediction**: Receptionistul este autentificat

**Main Flow**:

Receptionistul selecteaza optiunea “New Appointment” din interfata

Sistemul afiseaza un formula pentru introducerea detaliilor

Receptionistul introduce numele pacientului

Receptionistul selecteaza medical dorit din lista disponibila

Receptionistul selecteaza serviciul medical dorit

Receptionistul selecteaza data si ora dorita pentru programare

Sistemul verifica dispoonibilitatea medicului la ora si data selectata

Receptionistul confirma programarea

Sistemul salveaza programarea cu statusul “NEW”

**Alternative Flows**:

Daca medical nu este disponibil la data si ora selectata, sistemul afiseaza un mesaj de eroare si receptionistul trebuie sa selecteze alta ora/data

Daca exista deja o programare pentru medical respectiv in intervalul selectat, sostemul afiseaza un measaj de eroare

* 1. **Diagrama Use Case**

O imagine care conține text, captură de ecran, Font, număr

Conținutul generat de inteligența artificială poate fi incorect.

1. **Diagrama De Clasa**

O imagine care conține text, captură de ecran, diagramă, Font

Conținutul generat de inteligența artificială poate fi incorect.

O imagine care conține text, captură de ecran, Font, scris de mână

Conținutul generat de inteligența artificială poate fi incorect.

1. **Specificatii Suplimentare**

**Tehnologii Utilizate:**

Backend: Java 17, SpringBoot 3.4.4

Frontend: Java Swing

Persistenta: Spring Data JPA, Hibernate, MySQL

Securitate: Spring Security, BCrypt pentru criptarea parolelor

**Structura proiectului:** Proiectul este organizat in urmatoarele pachete

**-com.medcare.model**: Entitatile de business

**-com.medcare.repository**: Interfete pentru accesul la date

**-com.medcare.service**: Servicii care implementeaza logica de business

**-com.medcare.ui**: Componente de interfata utilizator

**-com.medcare.config**: Configurari Spring

**Persistența datelor:**

Entitatile sunt mapeate la tabele in baza de date folosind JPA

Se foloseste relatia de mostenire SINGLE\_TABLE pentru entitatile utilizator

Se folosește relatia many-to-one pentru a modela legatura intre programari si medici/servicii

Programul de lucru al medicilor este stocat ca o colectie de entitati embedded

**Inițializarea datelor:**

La prima rulare, aplicatia creeaza automat conturi pentru un administrator si un receptionist

De asemenea, se creeaza cativa medici si servicii medicale implicite pentru testare

**Imbunatatiri viitoare:**

Implementarea unui modul pentru gestionarea pacientilor

Adaugarea unui modul de facturare

Implementarea unui sistem de notificari pentru programari