DOCUMENTATIE

PROIECT MEDII SI INTRUMENTE DE PROGRAMARE

OSKY CHATBOT

REALIZAT DE

DUMITRASC CIPRIAN

10LF331

1. Introducere

Organizarea unui festival poate fi o sarcină complexă, cu multiple provocări, de la gestionarea artiștilor și biletelor până la oferirea unei experiențe memorabile pentru participanți. Într-o eră a digitalizării, soluțiile tradiționale devin tot mai ineficiente, iar utilizatorii se așteaptă la sisteme rapide, accesibile și intuitive. Proiectul nostru, **Festival Manager**, reprezintă un pas înainte în această direcție, fiind o aplicație completă care integrează gestionarea evenimentelor cu un chatbot interactiv, bazat pe procesarea limbajului natural (NLP).

Scopul aplicației este să transforme modul în care utilizatorii interacționează cu evenimentele, oferindu-le informații în timp real despre artiști, bilete și transport, totul printr-un asistent virtual simplu și eficient. De asemenea, aplicația facilitează administrarea festivalului printr-un panou de control dedicat organizatorilor.

Proiectul utilizează **tehnologii moderne** precum Java și Spring Boot pentru backend, Hibernate pentru accesul la bazele de date și PostgreSQL ca sistem de stocare a datelor. Partea de frontend este realizată cu ajutorul limbajelor HTML, CSS și JavaScript, asigurând o experiență vizuală atractivă și o interfață ușor de utilizat. În plus, aplicația integrează o componentă de **Procesare a Limbajului Natural (NLP)**, implementată prin clasa IntentRecognizer. Aceasta permite chatbotului să înțeleagă întrebările utilizatorilor, să recunoască intențiile acestora și să ofere răspunsuri adecvate în timp real.

Publicul țintă al aplicației este format din două categorii principale:

- **Participanții la festival**: Utilizatorii pot rezerva bilete, obține detalii despre artiști și orare, dar și informații despre transport, totul prin intermediul chatbotului virtual.
- **Organizatorii**: Aceștia beneficiază de un panou de administrare care permite gestionarea facilă a artiștilor și biletelor, precum și accesul la date centralizate.

Unul dintre elementele de bază ale aplicației este chatbotul inteligent, denumit **Osky**, care folosește NLP pentru a oferi o experiență conversațională naturală. Prin algoritmi specifici și modele de recunoaștere a intențiilor, Osky:

- Identifică întrebările despre bilete, prețuri și artiști.
- Extrage detalii relevante din inputul utilizatorului, cum ar fi tipul biletelor sau numele artistului.
- Gestionează contexte conversaționale pentru a răspunde întrebărilor de tip "dar" sau pentru a reține informații din interacțiunile anterioare.

Această integrare a NLP contribuie semnificativ la experiența utilizatorilor, eliminând barierele dintre aceștia și informațiile dorite. Spre exemplu, întrebări precum "Cât costă biletele VIP?" sau "Când cântă Dua Lipa?" sunt procesate inteligent de Osky, oferind răspunsuri rapide și precise.

Pe scurt, **Festival Manager** este o soluție completă pentru organizarea și administrarea festivalurilor moderne. Integrarea procesării limbajului natural transformă experiența utilizatorului, făcând interacțiunea mai fluidă, mai personalizată și mai

eficientă. Proiectul demonstrează cum tehnologia poate îmbunătăți industria evenimentelor, oferind o platformă unificată care aduce beneficii pentru toți cei implicați.

2. Arhitectura aplicației

1) Descriere generala

Aplicația Festival Manager este construită utilizând arhitectura Model-View-Controller (MVC), un design pattern bine cunoscut care separă logica aplicației în trei componente principale: Model, View și Controller. Această separare permite o dezvoltare mai modulară, ușurând întreținerea și scalarea aplicației.

- Model (M): Responsabil pentru gestionarea datelor și a logicii aplicației. În această aplicație, modelele sunt reprezentate de clasele Artist și Ticket, care reflectă structura datelor stocate în baza de date. Acestea sunt mapate cu ajutorul framework-ului Hibernate pentru interacțiunea cu baza de date PostgreSQL.
- View (V): Reprezintă interfața cu utilizatorul. Aceasta este construită folosind HTML, CSS și JavaScript, oferind o experiență vizuală atractivă și intuitivă. Două componente principale ale interfeței sunt:
    - Pagina principală (index.html): Utilizată de participanți pentru a interacționa cu chatbotul Osky.
    - Panoul de administrare (admin.html): Dedicat organizatorilor pentru gestionarea artiștilor și biletelor.
- Controller (C): Coordonează interacțiunile dintre utilizator și aplicație, gestionând cererile HTTP și conectându-se la model pentru a procesa datele. Controller-ele principale ale aplicației sunt ChatbotController și TransportController.

Fluxul de date al aplicației este simplificat astfel:

i. Utilizatorii sau organizatorii interacționează cu aplicația prin interfața web.
ii. Cererile sunt trimise către backend, unde sunt procesate de controller-e.
iii. Controller-ele accesează datele prin repository-uri și le trimit înapoi sub formă de răspunsuri JSON sau actualizări vizuale în interfața utilizatorului.

Această abordare modulară asigură o separare clară a responsabilităților, facilitând atât dezvoltarea, cât și extinderea funcționalităților aplicației.

2) Componentele Principale ale Arhitecturii

Aplicația Festival Manager este formată din mai multe componente, fiecare având un rol bine definit. Acestea colaborează pentru a oferi funcționalitățile necesare utilizatorilor finali și organizatorilor. Componentele principale ale aplicației sunt:

a) Backend (Java și Spring Boot)

Backend-ul reprezintă partea logică a aplicației, responsabilă pentru gestionarea datelor și procesarea cererilor utilizatorilor. Este construit utilizând framework-ul Spring Boot, care oferă un mediu ușor de configurat și scalabil pentru dezvoltarea aplicațiilor Java. Componentele backend sunt:

- **Controller-ele**:
  - **ChatbotController**:
    - Gestionează cererile primite de la chatbot.
    - Utilizează clasa IntentRecognizer pentru a identifica intențiile utilizatorilor și a răspunde în mod corespunzător.
    - Exemple de endpoint-uri:
      - /api/chat/send: Procesează mesaje și întoarce răspunsuri adecvate.
      - /api/chat/history/{sessionId}: Returnează istoricul conversațiilor pentru o sesiune specifică.
  - **TransportController**:
    - Răspunde la cererile privind transportul.
    - Interoghează baza de date pentru a găsi informații despre rutele disponibile.
    - Exemplu de endpoint: /api/transport/search.
- **Modelul de date**:
  - **Clasa Artist**:
    - Reprezintă artiștii din cadrul festivalului.
    - Este mapată în baza de date prin anotările Hibernate (@Entity, @Table).
    - Atribute principale: name, performanceDate, performanceTime.
  - **Clasa Ticket**:
    - Modelează biletele disponibile pentru festival.
    - Atribute principale: type, available, price.
- **Repository-uri**:
  - **ArtistRepository** și **TicketRepository**:
    - Interfețe care extind JpaRepository, simplificând interacțiunea cu baza de date.
    - Exemple de funcții utilizate:

        - findById(String id) pentru căutarea unui artist sau tip de bilet.
        - save(Entity entity) pentru salvarea sau actualizarea datelor.

❖ **Servicii**:

- **DataLoader**:
    - Populează baza de date cu date inițiale (artiști și bilete) atunci când aplicația este lansată.

b) Frontend (HTML, CSS, JavaScript)

Frontend-ul reprezintă interfața cu utilizatorul și este responsabil pentru interacțiunea vizuală. Este construit utilizând tehnologii web moderne:

- **Fișierele HTML**:
    - **index.html**:
        - Pagina principală, unde utilizatorii interacționează cu chatbotul Osky.
        - Include un câmp de introducere pentru mesaje, un buton pentru trimitere și opțiuni pentru autentificarea în panoul de administrare.
    - **admin.html**:
        - Panoul de administrare, care permite gestionarea artiștilor și biletelor.
        - Formulare pentru adăugarea artiștilor și biletelor, precum și o tabelă pentru listarea datelor existente.
- **Fișierele CSS**:
    - **style.css** și **admin.css**:
        - Definește stilurile vizuale ale aplicației.
        - Elemente de design responsive, animații și o experiență plăcută pentru utilizatori.
- **Fișierele JavaScript**:
    - **script.js**:
        - Garantează funcționarea chatbotului, trimițând cereri către backend și afișând răspunsurile.
    - **admin.js**:
        - Asigură interacțiunile din panoul de administrare, cum ar fi adăugarea și ștergerea artiștilor.

c) Baza de date (PostgreSQL)

Baza de date PostgreSQL este utilizată pentru stocarea persistentă a datelor aplicației. Structura sa include:

- Tabelul artists: Păstrează informații despre artiști (nume, dată și oră a performanței).
- Tabelul tickets: Păstrează tipurile de bilete, numărul disponibil și prețurile acestora.

Configurarea bazei de date este definită în fișierul application.yml:

```yaml
spring:
  datasource:
    url: jdbc:postgresql://localhost:5432/festival
    username: chatbot_user
    password: secure_password
  jpa:
    hibernate:
      ddl-auto: update
```

3) Dependențe

Proiectul Festival Manager folosește un set de dependențe configurate prin Maven, pentru a facilita dezvoltarea și integrarea cu diverse tehnologii. Aceste dependențe includ:

i. Spring Boot Starter Web:
   - Pentru crearea și gestionarea API-urilor REST.
   - Permite manipularea cererilor HTTP.

ii. Spring Boot Starter Data JPA:
   - Oferă suport pentru maparea obiect-relațională (ORM) utilizând Hibernate.
   - Simplifică accesul la baza de date.

iii. PostgreSQL Driver:
   - Asigură conectivitatea cu baza de date PostgreSQL.

iv. Spring Boot Starter Test:
   - Pentru scrierea și rularea testelor unitare și de integrare.

v. Apache Commons Lang:
   - Utilizat pentru manipulări avansate ale șirurilor de caractere sau alte operații auxiliare.

3. Arhitectura aplicației

Implementarea tehnică a aplicației **Festival Manager** cuprinde componentele cheie ale backend-ului, logica de gestionare a datelor și funcționalitățile principale. Vom detalia modul în care sunt implementate principalele funcționalități.

3.1. Gestionarea Artiștilor

Gestionarea artiștilor include operațiuni precum adăugarea, ștergerea și listarea artiștilor.

- **Clasa Artist**:
  - Este modelată ca o entitate JPA, mapată la tabelul artists din baza de date.
  - Atributele includ name, performanceDate și performanceTime.
  - Exemple de cod:

```java
@Entity  Ciprian D *
@Table(name = "artists")
public class Artist {
    @Id
    private String name;
    private String performanceDate;  2 usages
    private String performanceTime;  2 usages

    public Artist() {}  Ciprian D

    public Artist(String name, String performanceDate, String performanceTime) {  2 usages  Ciprian D
        this.name = name;
        this.performanceDate = performanceDate;
        this.performanceTime = performanceTime;
    }

    public String getName() { return name; }  4 usages  Ciprian D
    public String getPerformanceDate() { return performanceDate; }  1 usage  Ciprian D
    public String getPerformanceTime() { return performanceTime; }  no usages  Ciprian D
}
```

- **Repository-ul ArtistRepository**:
  - Extinde JpaRepository pentru a oferi operațiuni CRUD automate.
  - Exemplu de definiție:

```java
public interface ArtistRepository extends JpaRepository<Artist, String> {  8 usages  Ciprian D
}
```

- **Endpointuri din Admin Panel**:
  - Gestionate prin API-uri REST care permit adăugarea și ștergerea artiștilor.
  - Exemple de funcții JavaScript în admin.js:
    - Adăugare:

```
document.getElementById( elementId: 'addArtistForm').addEventListener( type: 'submit', listener: async function (event :SubmitEvent ) :Promise<void> {  Ciprian D
    event.preventDefault();
    const artist :{…} = {
        name: document.getElementById( elementId: 'artistName').value,
        performanceDate: document.getElementById( elementId: 'performanceDate').value,
        performanceTime: document.getElementById( elementId: 'performanceTime').value
    };

    try {
        const response :Response = await fetch( input: '/admin/artists', init: {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify(artist)
        });
        if (response.ok) fetchArtists();
    } catch (error) {
        alert('Eroare la adăugarea artistului.');
    }
});
```

- Stergere

```
document.getElementById( elementId: 'deleteArtistForm').addEventListener( type: 'submit', listener: async function (event :SubmitEvent ) :Promise<void> {  Ciprian
    event.preventDefault();
    const artistId = document.getElementById( elementId: 'artistSelect').value;

    if (!artistId) {
        alert('Te rog să selectezi un artist.');
        return;
    }

    try {
        const response :Response = await fetch( input: `/artists/${artistId}`, init: {
            method: 'DELETE'
        });

        if (!response.ok) throw new Error('Eroare la ștergerea artistului.');

        alert('Artistul a fost șters cu succes.');
        populateArtistSelect();
        fetchArtists();
    } catch (error) {
        alert(error.message);
    }
});
```

3.2. Gestionarea Biletelor

Biletele sunt gestionate în mod similar, cu operațiuni de adăugare, rezervare și interogare.

- Clasa Ticket:
  - Modelează biletele ca entități JPA, cu atribute precum type, available și price.
  - Exemple de cod:

```java
@Entity  Ciprian D
@Table(name = "tickets")
public class Ticket {
    @Id
    private String type;
    private int available;  2 usages
    private double price;  3 usages

    public Ticket() {}  Ciprian D

    public Ticket(String type, int available, double price) {  3 usages  Ciprian D
        this.type = type;
        this.available = available;
        this.price = price;
    }

    public String getType() { return type; }  no usages  Ciprian D
    public int getAvailable() { return available; }  3 usages  Ciprian D
    public double getPrice() { return price; }  3 usages  Ciprian D
    public void setPrice(double price) { this.price = price; }
}
```

- **Repository-ul TicketRepository**:
  - Permite interogarea și actualizarea datelor despre bilete.

- **Funcții în Chatbot**:
  - Răspunsuri la întrebări precum "Câte bilete VIP mai sunt disponibile?".
  - Cod în ChatbotController:

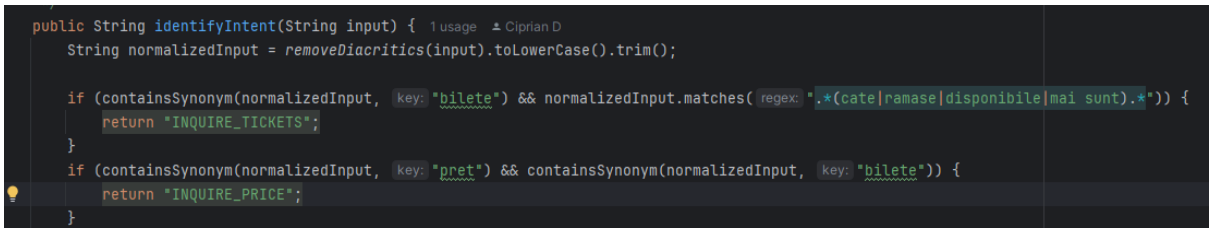
```java
private String getTicketAvailability(String ticketType, String sessionId) {  1 usage  Ciprian D
    Optional<Ticket> ticket = ticketRepository.findById(ticketType);
    if (ticket.isPresent()) {
        return sendResponse( response: "Avem " + ticket.get().getAvailable() + " bilete " + ticketType + " disponibile.", sessionId);
    }
    return sendResponse( response: "Nu am găsit bilete de tipul specificat.", sessionId);
}
```

3.3. Chatbot și Procesarea Limbajului Natural (NLP)

Componenta de NLP permite chatbotului să recunoască intențiile utilizatorilor și să răspundă contextului conversațional.

- **Clasa IntentRecognizer**:
  - Folosește sinonime și expresii pentru a identifica intențiile utilizatorilor.

- Exemple:



- **Controller-ul Chatbot**:
  - Coordonează răspunsurile bazate pe intențiile detectate.
  - Exemple de endpoint-uri:
    - /api/chat/send: Gestionează mesaje și răspunsuri.
    - /api/chat/history/{sessionId}: Returnează istoricul conversațiilor.

4. Concluzii și Direcții Viitoare

Proiectul **Festival Manager** reprezintă un exemplu solid de integrare a tehnologiilor moderne pentru automatizarea și simplificarea gestionării unui eveniment complex, precum un festival. Prin implementarea unui chatbot interactiv, a unui panou de administrare dedicat și a unor funcționalități avansate de gestionare a datelor, aplicația oferă o experiență eficientă atât utilizatorilor finali, cât și organizatorilor.

Beneficii ale aplicației

1. **Automatizare și eficiență**: Chatbotul bazat pe NLP reduce timpul necesar pentru a răspunde întrebărilor utilizatorilor, eliminând necesitatea interacțiunilor umane constante.
2. **Interfață intuitivă**: Pagina principală și panoul de administrare sunt construite cu o atenție deosebită la design și ușurința în utilizare.
3. **Modularitate**: Arhitectura MVC utilizată asigură o separare clară a componentelor, ceea ce facilitează extinderea și întreținerea proiectului.
4. **Scalabilitate**: Utilizarea Spring Boot și Hibernate permite aplicației să gestioneze creșteri semnificative în volumul de date și numărul de utilizatori.
5. **Interacțiune naturală**: Chatbotul Osky folosește procesarea limbajului natural pentru a răspunde în mod relevant și adaptiv la întrebările utilizatorilor.

Direcții de îmbunătățire

1. **Notificări și alerte**:
    - Integrarea unui sistem de notificări prin e-mail sau SMS pentru reamintirea performanțelor artiștilor sau a rezervărilor de bilete.
2. **Extinderea NLP**:
    - Adăugarea suportului pentru mai multe limbi și îmbunătățirea algoritmilor de recunoaștere a intențiilor.
3. **Integrare cu rețele sociale**:
    - Posibilitatea de a partaja informații despre eveniment direct pe platformele sociale.
4. **Statistici și rapoarte**:
    - Un modul de analiză a datelor pentru organizatori, care să includă vânzările de bilete, popularitatea artiștilor și feedback-ul utilizatorilor.
5. **Funcționalități avansate pentru transport**:
    - Adăugarea unor hărți interactive și a opțiunii de a rezerva transport direct din aplicație.

Impactul aplicației

Festival Manager este mai mult decât o simplă aplicație de gestionare; este o platformă completă care aduce împreună organizatorii și participanții, eliminând barierele de comunicare și optimizând procesele administrative. Acest proiect demonstrează cum tehnologia poate transforma experiența utilizatorilor, economisind timp și resurse, dar și cum poate sprijini organizatorii în luarea deciziilor bazate pe date concrete.