**Titolo:** Sistema di Gestione Programmazione Cinema

**1: Introduzione**

* Obiettivo: Creazione di un'applicazione web per la gestione e visualizzazione della programmazione cinematografica.
* Utenti:
  + **Pubblico**: Visualizzazione film in programmazione.
  + **Amministratori**: Gestione completa dei film (inserimento, storico, ricerca avanzata).

**2: Architettura dell'applicazione**

* **Frontend:** Angular (SPA)
* **Backend:** Spring Boot (REST API)
* **Database:** PostgreSQL
* **Containerizzazione:** Docker + Docker Compose

**3: Funzionalità principali**

* Visualizzazione pubblica film in programmazione
* Accesso amministrativo tramite login (Basic Auth)
* Inserimento film con:
  + Titolo, Descrizione, Data inizio/fine, Orario, Sala, Genere
* Visualizzazione dello storico completo
* Ricerca avanzata per data o anno (solo per admin)

**4: Modello dati (UML semplificato)**

* **Film – Entità principale del dominio, corrispondente ad un film.**
  + id: Long -> identificativo univoco del film
  + titolo: String -> titolo del film
  + descrizione: String (max 255) -> descrizione del film
  + dataInizio: LocalDate -> data di inizio proiezione
  + dataFine: LocalDate -> data di fine proiezione
  + orario: String -> orario di inizio del film
  + sala: String -> sala della proiezione
  + genere: String -> genere del film

**5: Backend - Classi principali**

* **FilmController**: Gestione delle rotte REST
  + GET /api/films/current -> ritorna I film attualmente in programmazione
  + GET /api/films/past -> ritorna I film passati (accesso admin)
  + POST /api/films -> aggiunge un nuovo film (solo admin)
  + GET /api/films/range -> ritorna I film compresi in un certo intervallo di date
  + GET /api/films/range/past -> ritorna I film passati in un intervallo (ricerca annuale, storico admin)
* **FilmService**: Logica di business
  + getAllFilms() -> ritorna tutti i film nel DB
  + getCurrentFilms() -> film in programmazione
  + getPastFilms() -> film il cui periodo di programmazione è finito
  + getFilmsByDateRange() -> film già usciti ma compresi in un intervallo di date
  + saveFilm(Film film) -> salva un nuovo film nel DB
* **Interfaccia FilmRepository**: Accesso al database

Estende JpaRepository<Film, Long> e offre metodi personalizzati:

**List<Film> findByDataInizioBetween(LocalDate start, LocalDate end)**

* + Ritorna film la cui programmazione inizia tra due date.

**List<Film> findByDataFineBefore(LocalDate date)**

* + Ritorna film terminati prima di una certa data (storico).

**List<Film> findByDataInizioBeforeAndDataFineAfter(LocalDate end, LocalDate start)**

* + Ritorna i film attualmente in programmazione.

**6: Frontend - Componenti principali**

* **FilmListComponent**: Pagina pubblica -> mostra film in programmazione
* **LoginComponent**: Autenticazione admin -> autenticazione, reindirizza a pannello admin
* **AdminPanelComponent**: Inserimento film e storico -> aggiunta nuovi film, logout, visualizzazione storico, ricerca per anno film passati, ritorna a storico.
* **RicercaFilmComponent**: Ricerca avanzata per data -> ricerca lato utente per intervallo date

**7: Sicurezza e autenticazione**

* **Spring Security**
  + Basic Auth per le chiamate admin
  + Accesso controllato a rotte e API
* **Angular Interceptor**: Aggiunta header Authorization automatica

**8: Containerizzazione**

* **Docker Compose:**
  + db: PostgreSQL
  + backend: Spring Boot
  + frontend: Angular static site (con Node.js server)
* Separazione dei servizi per ambiente di produzione

**9: Conclusioni**

* App stabile, scalabile, containerizzata
* Possibilità di estensioni future:
  + JWT Auth
  + Prenotazioni
  + Interfaccia responsive
* Utilizzabile da un cinema reale o piccola catena