Università degli Studi di Milano Bicocca

**Scuola di Scienze**

**Dipartimento di Informatica, Sistemistica e Comunicazione**

**Corso di laurea in Informatica**

**CINEMA MULTISALA**

**Realizzazione di una web app per la gestione di un cinema**

**Relatore**: *Dott.ssa* *Daniela Micucci*

**Co-relatore**: *Dott.ssa Annalisa Marra (Sync Lab S.r.l)*

**Relazione della prova finale di:**

*Daniele Scalco*

*Matricola 844762*

**Anno Accademico 2020-2021**

***INDICE***

Introduzione

1. Sync Lab
2. Descrizione del progetto
   1. Requisiti funzionali
   2. Diagramma dei casi d’uso
   3. Descrizione dettagliata dei casi d’uso
   4. Diagrammi di attività
3. Framework e tecnologie utilizzate per lo sviluppo
   1. Spring
      1. Spring MVC
      2. Spring Boot
      3. Spring Security
   2. Hibernate
   3. Thymeleaf
   4. Bootstrap
4. Base dati
   1. Modello Entity-Relationship
   2. Database MySql
5. Struttura del progetto java
6. Dettagli dell’interfaccia utente
7. Conclusioni e possibilità di sviluppo futuri

***INTRODUZIONE***

***1) SYNC LAB***

Sync Lab è un’azienda nata come Software house e, nel corso degli anni dopo aver acquisito più competenze tecnologiche, si è tramutata in System Integrator e attualmente possiede cinque sedi in tutta Italia.

Grazie agli ottimi prodotti realizzati nel loro laboratorio di ricerca e sviluppo sono riusciti a conquistare fette di mercato nei settori mobile, videosorveglianza e sicurezza informatica delle infrastrutture informatiche aziendali e fra i loro clienti possono vantare alcuni colossi come Unicredit, H&M, Sky, Fastweb e Eni.

***2) DESCRIZIONE DEL PROGETTO***

Il progetto assegnatomi consiste nella realizzazione di un’applicazione web per la gestione di un ipotetico cinema realizzata utilizzando il framework di Spring e Hibernate (questi elementi verranno spiegati meglio nel capitolo successivo).

Le persone che interagiscono con il sistema possono avere i seguenti ruoli:

* Utente
* Dipendente
* Amministratore

L’utente è il cliente interessato al cinema in questione. Può visitare il sito liberamente e vedere i vari film in programmazione disponibili e se desidera effettuare una prenotazione è obbligato a dover effettuare prima la registrazione.

Il dipendente è colui che lavora per il cinema e si occupa della gestione della programmazione dei film.

L’amministratore può fare le stesse cose di un dipendente ed inoltre si occupa anche della gestione degli account registrati.

***2.1) Requisiti funzionali***

Di seguito vengono mostrare i requisiti funzionali, ovvero i servizi che il sistema deve (o non deve) offrire, suddivisi per ognuno dei ruoli menzionati precedentemente.

Utente:

* Può registrarsi nel sistema ed in seguito effettuare login/logout;
* Può visualizzare la programmazione dei film disponibili nell’arco di una settimana;
* Per ogni film in programmazione può visualizzare la relativa scheda con tutti i dati in dettaglio (titolo, immagine locandina, regista, cast, sala, descrizione, orari);
* Può effettuare una prenotazione per uno dei film disponibili se ci sono posti disponibili in sala.

Dipendente:

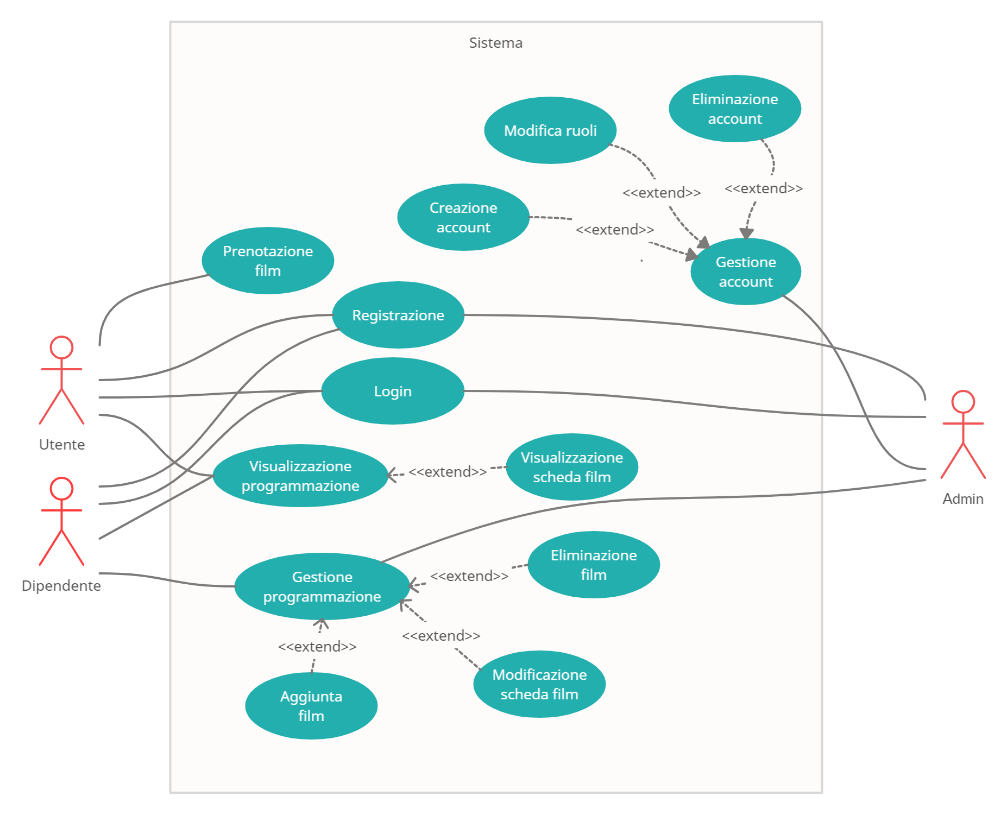
* Può effettuare il login/logout;
* Può aggiungere un nuovo film nel database specificando tutti i dati necessari;
* Può eliminare un film dal database (sia un singolo orario di riproduzione che tutti quelli con un determinato titolo);
* Può modificare i dati di dettaglio di un film e aggiungere nuovi orari.

Amministratore:

* Può effettuare il login/logout;
* Può aggiungere un nuovo film nel database specificando tutti i dati necessari;
* Può eliminare un film dal database (sia un singolo orario di riproduzione che tutti quelli con un determinato titolo);
* Può modificare i dati di dettaglio di un film e aggiungere nuovi orari;
* Può visualizzare la lista degli account registrati nel sistema;
* Può modificare i ruoli assegnati ad un account;
* Può eliminare un account;
* Può creare un nuovo account fornendogli un qualunque ruolo.

***2.2) Diagramma dei casi d’uso***

Di seguito viene mostrato il diagramma dei casi d’uso che mostra i vari attori con i casi d’uso associati e come questi interagiscono con il sistema in questione.



*Figura 1: Diagramma dei casi d’uso*

***2.3) Descrizione dettagliata dei casi d’uso***

In questa sezione vengono riportate le descrizioni dettagliate dei casi d’uso previsti, in cui vengono mostrati gli *happy flow* ovvero gli scenari di successo.

***REGISTRAZIONE***

**Livello:** sottofunzione

**Attori Primari:** utente

**Parti interessate e interessi:** l’utente vuole registrare il proprio account per poter poi usare l’applicazione sfruttando le sue funzionalità al massimo.

**Pre-condizioni:** nessuna

**Garanzia di successo:** l’utente viene registrato correttamente nel sistema

**Scenario principale di successo:**

1. L’utente seleziona “REGISTRAZIONE” nel menù in alto a destra dell’homepage
2. Il sistema mostra la pagina contente la form per registrarsi
3. L’utente compila tutti i campi (“e-mail”, “password” e “conferma password”) e seleziona il bottone “Registrati”
4. Il sistema registra l’utente e ritorna nella Homepage mostrando un messaggio di conferma avvenuta registrazione

***LOGIN***

**Livello:** sottofunzione

**Attori Primari:** utente/dipendente/amministratore

**Parti interessate e interessi:** l’utilizzatore vuole effettuare l’accesso al proprio account

**Pre-condizioni:** l’utilizzatore dev’essersi precedentemente registrato nel sistema

**Garanzia di successo:** l’utilizzatore effettua il login

**Scenario principale di successo:**

1. L’utente seleziona “LOGIN” nel menù in alto a destra dell’homepage
2. Il sistema mostra la pagina contente la form per effettuare il login
3. L’utente compila tutti i campi (E-mail e password) e seleziona il bottone “Login”
4. L’utente viene loggato correttamente e ritorna nella Homepage.

***VISUALIZZAZIONE PROGRAMMAZIONE***

**Livello:** obiettivo utente

**Attori Primari:** utente

**Parti interessate e interessi:** l’utente vuole vedere i film che saranno presenti in sala nei prossimi sette giorni

**Pre-condizioni:** nessuna

**Garanzia di successo:** l’utente visualizza la lista di film in programmazione

**Scenario principale di successo:**

1. L’utente seleziona “PROGRAMMAZIONE” nel menù in alto a sinistra dell’homepage
2. Il sistema mostra la pagina contenente tutti i film in programmazione nei prossimi sette giorni partendo dalla data attuale

***PRENOTAZIONE***

**Livello:** obiettivo utente

**Attori Primari:** utente

**Parti interessate e interessi:** l’utente vuole prenotare dei posti per il film, la data e l’orari da lui scelti

**Pre-condizioni:** l’utente dev’essersi loggato nel sistema

**Garanzia di successo:** l’utente effettua la prenotazione con successo

**Scenario principale di successo:**

1. L’utente seleziona “PROGRAMMAZIONE” nel menù in alto a sinistra dell’homepage
2. Il sistema mostra la pagina con i film presenti in programmazione
3. L’utente seleziona la data e l’orario che preferisce
4. Il sistema mostra la pagina con lo stato attuale delle prenotazioni per il film scelto dall’utente
5. L’utente seleziona i posti che desidera fra quelli segnalati come liberi e clicca sul bottone “Prenota”
6. Il sistema registra la prenotazione nel database mostra una pagina di riepilogo con tutti i dati della prenotazione

***GESTIONE PROGRAMMAZIONE***

**Livello:** obiettivo utente

**Attori Primari:** dipendente/amministratore

**Parti interessate e interessi:** l’utilizzatore vuole gestire la programmazione aggiungendo, rimuovendo o modificando i film

**Pre-condizioni:** l’utilizzatore dev’essersi loggato nel sistema

**Garanzia di successo:** l’utilizzatore effettua i cambiamenti che desidera alla programmazione

**Scenario principale di successo:**

Se si vuole aggiungere un film:

1. Il dipendente seleziona “AGGIUNGI FILM” nel menù in alto a sinistra dell’homepage
2. Il sistema mostra la pagina con una form in cui inserire tutti i dati riguardanti il film da aggiungere
3. Il dipendente immette i dati del film (titolo, regista, cast, descrizione, immagine locandina, data, orario, sala) e preme su “Aggiungi”
4. Il sistema torna alla pagina di aggiunta film mostrando un messaggio di avvenuta aggiunta

Se si vuole modificare/eliminare un film:

1. Il dipendente seleziona “PROGRAMMAZIONE” nel menù in alto a sinistra dell’homepage
2. Il sistema mostra i film in programmazione

Se si vuole eliminare un orario di un film:

* 1. Il dipendente seleziona il bottone relativo all’orario che vuole eliminare
  2. Il sistema elimina il film con l’orario selezionato e ritorna nella schermata della programmazione mostrando un messaggio di conferma

Se si vuole eliminare un film completamente con tutti gli orari:

* 1. Il dipendente seleziona il bottone “Elimina film” relativo ad un titolo presente
  2. Il sistema elimina ogni film con quel titolo e ritorna alla schermata della programmazione mostrando un messaggio di conferma

Se si vuole aggiungere un orario per un film:

* 1. Il dipendente seleziona il bottone “Aggiungi orario” relativo ad un titolo presente
  2. Il sistema mostra all’utilizzatore una nuova pagina in cui è possibile immettere data, ora e numero sala
  3. Il dipendente compila tutti i campi correttamente e seleziona “Aggiungi orario”
  4. Il sistema registra il nuovo orario e, rimanendo sulla schermata di aggiunta orario per permettere eventualmente di inserirne un altro, mostra un messaggio di conferma

Se si vuole modificare i dati di dettaglio di un film:

* 1. Il dipendente seleziona il bottone “Modifica film” relativo ad un titolo presente
  2. Il sistema mostra una schermata con i dati di dettaglio attuali del film, ovvero titolo, regista, cast, descrizione e immagine locandina
  3. Il dipendente modifica tutti i campi che desidera cambiare e seleziona “MODIFICA”
  4. Il sistema ritorna alla schermata di programmazione mostrando un messaggio di conferma avvenuta modifiche

***GESTIONE ACCOUNT***

**Livello:** obiettivo utente

**Attori Primari:** amministratore

**Parti interessate e interessi:** l’amministratore vuole visualizzare e/o modificare gli account registrati nel sistema

**Pre-condizioni:** l’amministratore dev’essere loggato nel sistema

**Garanzia di successo:** l’amministratore porta a compimento tutte le azioni da lui richieste

**Scenario principale di successo:**

1. L’amministratore clicca in alto a sinistra nel menu “GESTIONE UTENTI”
2. Il sistema mostra una pagina con una lista contenente tutti gli account registrati in ordine alfabetico per indirizzo e-mail

Se si vuole aggiungere un nuovo account:

* 1. L’amministratore seleziona il bottone “Aggiungi utente”
  2. Il sistema mostra una nuova pagina con una form per aggiungere i dati relativi al nuovo account
  3. L’amministratore compila i campi “e-mail”, “password” e “conferma password” e seleziona i ruoli e seleziona “Crea account”
  4. Il sistema ritorna alla schermata con la lista degli utenti aggiornata e mostra un messaggio di avvenuta registrazione

Se si vuole eliminare un account:

* 1. L’amministrazione seleziona il bottone “Elimina” relativo alla riga dell’account che vuole eliminare
  2. Il sistema mostra un alert per confermare l’eliminazione dell’account selezionato
  3. L’amministratore conferma la scelta
  4. Il sistema elimina l’account e mostra un messaggio di conferma eliminazione

Se si vuole modificare i ruoli che possiede un account:

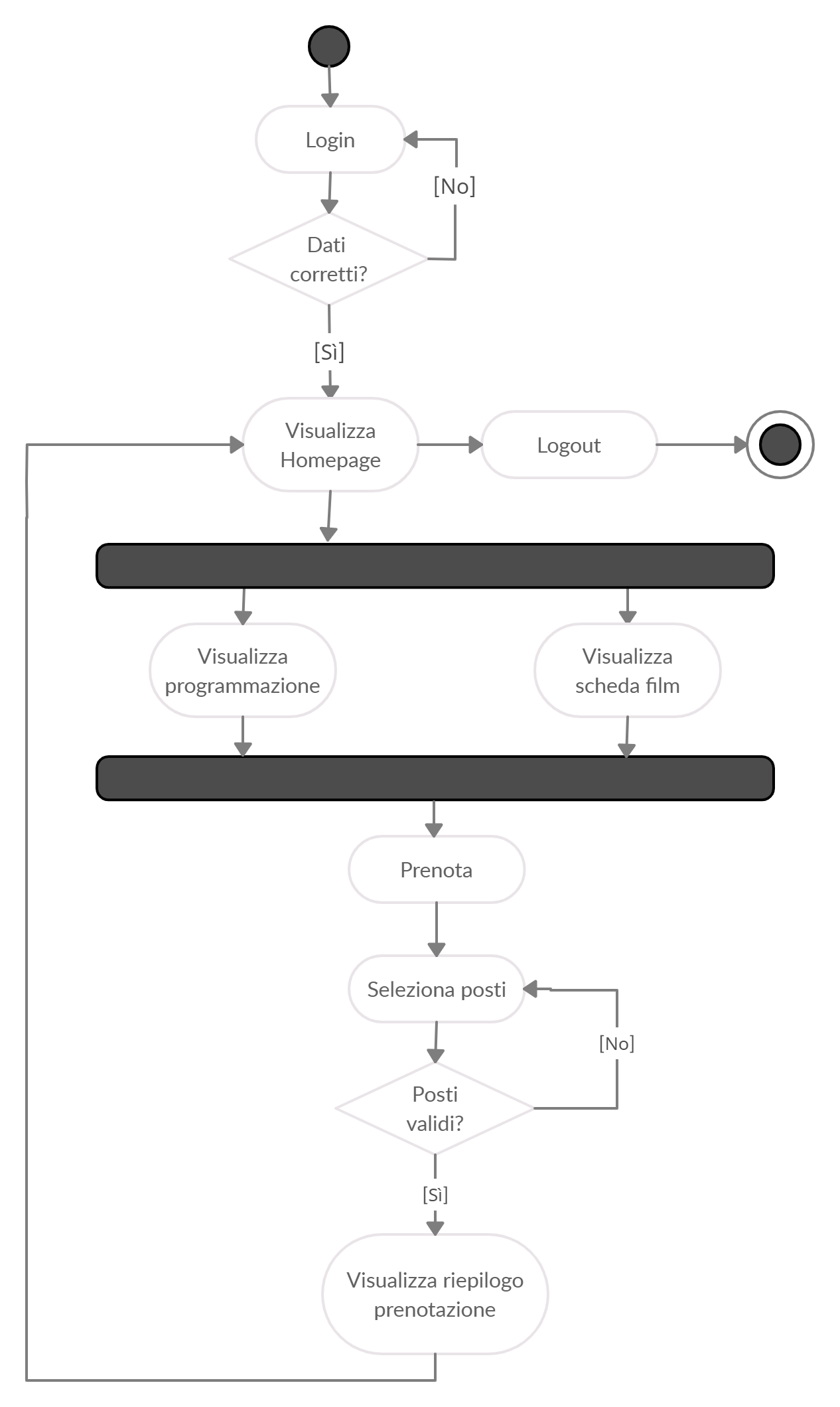
* 1. L’amministrazione seleziona il bottone “Aggiorna” relativo alla riga dell’account che vuole modificare
  2. Il sistema mostra una nuova pagina in cui si vede la mail dell’account selezionato e si possono scegliere i ruoli
  3. L’amministratori aggiunge e/o elimina i ruoli che desidera e seleziona “Applica modifiche”
  4. Il sistema ritorna alla schermata con la lista degli utenti aggiornata e mostra un messaggio di conferma delle modifiche

***2.4) DIAGRAMMI DI ATTIVITA’***

In questo paragrafo verranno mostrato i diagrammi di attività che modellano gli aspetti dinamici e sono molto utili a capire il flusso delle azioni che eseguono i vari attori che interagiscono con il sistema.

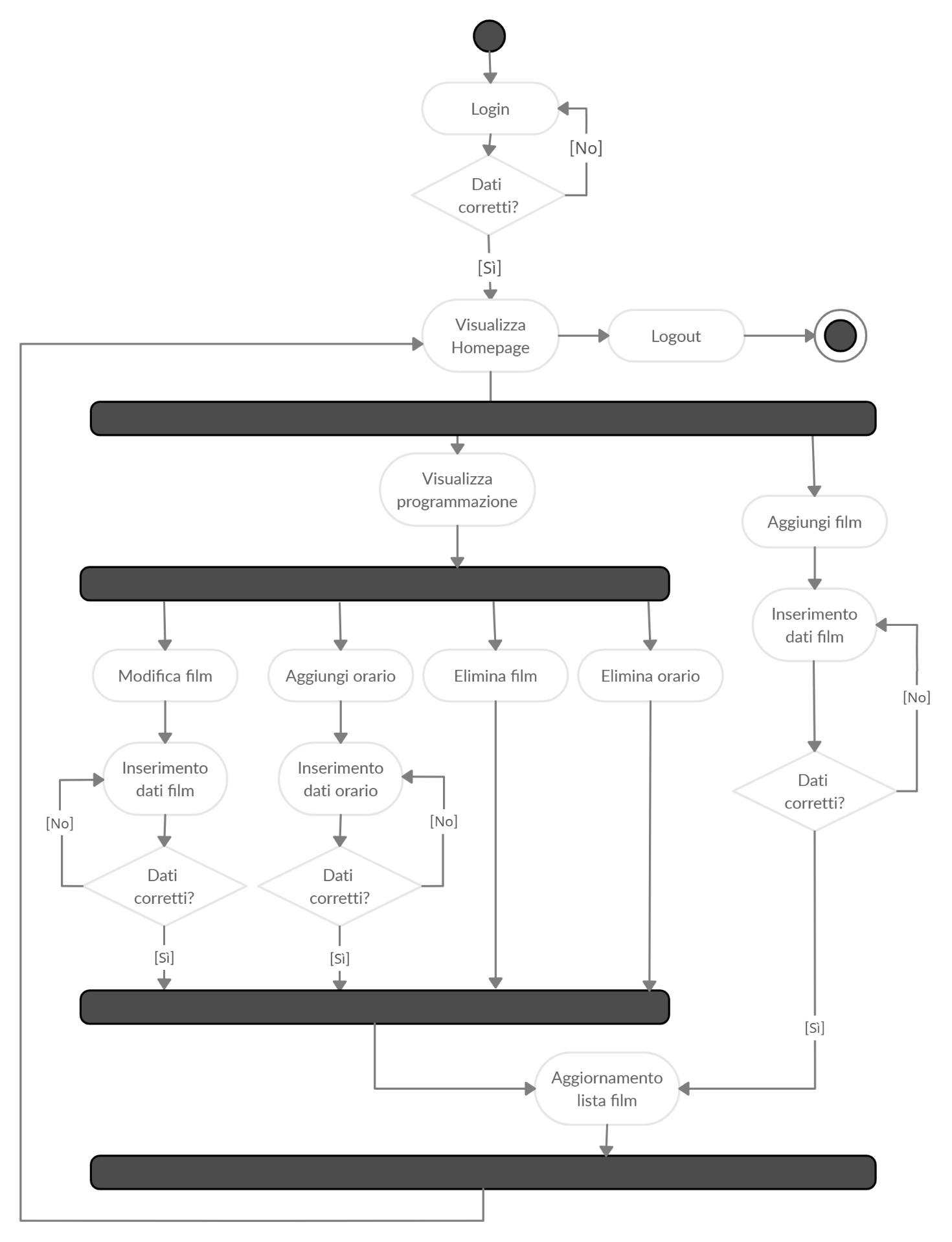
Di seguito vengono mostrati tre diagrammi, uno per ruolo, in cui vengono mostrate le attività che ciascuno può eseguire. Convenzionalmente la “fork” è una barra di sincronizzazione che divide il flusso in due parti che avvengono in parallelo, qui si è deciso che invece rappresenta le possibili azioni differenti (esclusive, o una o l’altra) che l’attore coinvolto può decidere di eseguire.

**UTENTE**



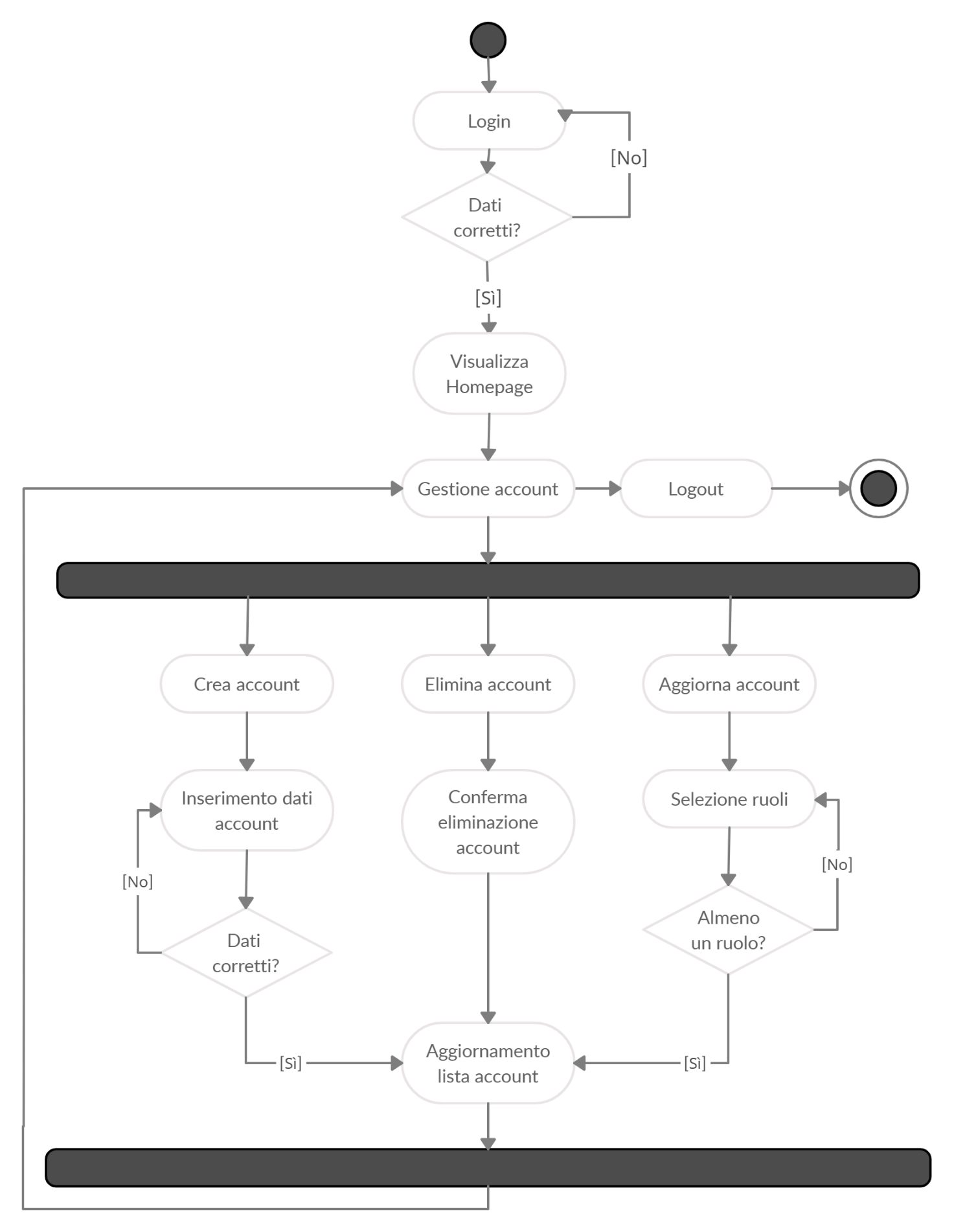
*Figura 2.1: Diagramma di attività - utente*

**DIPENDENTE**



*Figura 2.2: Diagramma di attività - dipendente*

**AMMINISTRATORE**



*Figura 2.3: Diagramma di attività – amministratore*

***3) FRAMEWORK E TECNOLOGIE UTILIZZATE PER LO SVILUPPO***

Per lo sviluppo dell’applicativo sono stati utilizzati diversi framework e piattaforme, ognuno con le sue peculiarità e funzionalità che verranno descritte in questo capitolo.

***3.1) Spring***

Spring è un framework opensource utile per lo sviluppo di applicazioni enterprise scritte in java. È nato come alternativa a *J2EE* (Java 2 Enterprise Edition) proponendo un modello più semplice e leggero per lo sviluppo di entità di business dato che la sua architettura è modulare permettendone così l’utilizzo sia nella sua interezza che solo in parte.

Spring si basa sui concetti di *Inversion of Control* (IoC) e *Dependecy Injection* (DI):

IoC:

nella logica tradizionale è lo sviluppatore che si occupa della creazione, inizializzazione ed invocazione dei metodi di un oggetto, con questo principio architetturale invece è il framework che si preoccupa di gestire tutto ciò.

DI:

è un pattern per implementare l’IoC che permette di collegare gli oggetti tra loro attraverso un assembler che si occupa di iniettare le dipendenze. Le dipendenze possono essere iniettate in tre modi:

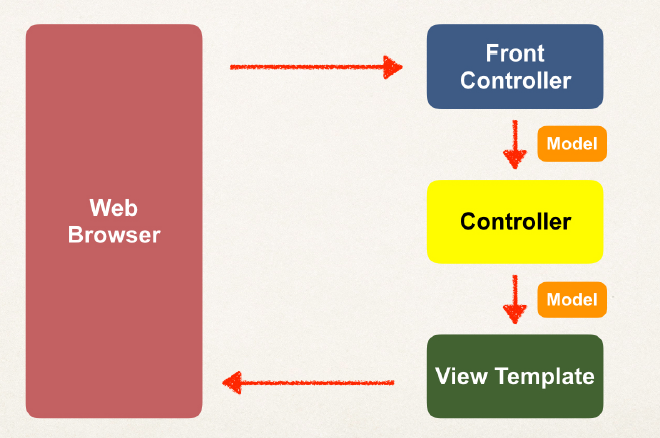
1. Constructor-based DI
2. Setter-based DI
3. Field-based DI (utilizzata in questo progetto)

***3.1.1) Spring MVC***

Spring MVC è un web framework di Spring basato sul paradigma Model-View-Controller (MVC), un pattern architetturale che separa la logica di business dalla logica di presentazione. Ogni strato ha la sua funzione:

* Model: contiene i metodi di accesso ai dati
* View: permette di visualizzare i dati ed è il mezzo con il quale l’utente interagisce col sistema
* Controller: si occupa della gestione delle altre due componenti. Riceve gli input dall’utente per mezzo della View, elabora i dati eseguendo operazioni sul Model e porta ad un cambiamento di stato della View.

Spring MVC è dotata di una classe *DispatcherServlet* (Front Controller) che riceve gli URI in arrivo e li inoltra alle classi Controller corrispondenti.



*Figura 3: Spring MVC dietro le quinte*

***3.1.2) Spring Boot***

Spring Boot è un framework che permette la semplificazione dello sviluppo di un’applicazione Spring standalone, riducendo le configurazioni manuali che deve effettuare lo sviluppatore e il codice boilerplate. Spring Boot, infatti, configura in automatico l’applicazione basandosi sulle dipendenze che sono state aggiunte e fornisce inoltre anche un server Tomcat incorporato.

La creazione di un progetto può essere facilmente eseguita utilizzando Spring Initializr in cui bisogna solamente selezionare il tipo di progetto (Maven o Gradle), il linguaggio utilizzato e le dipendenze che servono.

***3.1.3) Spring Security***

Spring Security è un framework di supporto di Spring per le funzionalità di autenticazione, autorizzazione e controllo degli accessi. In questo modo è possibile salvare nella base dati gli account che vengono registrati e far accedere alle pagine web solo gli utenti con determinati ruoli per garantire la sicurezza dell’applicativo. Per il salvataggio delle password nel progetto si è fatto uso della classe *BCryptPasswordEncoder* che fanno uso di bcrypt, una funzione di hashing di password, in modo da salvare sul database i dati in modo sicuro. Una funzione di hashing è una funzione *one-way*, ovvero non è possibile invertirla e in questo modo i dati sensibili non possono essere decrittati.

***3.2) Hibernate***

Hibernate è una piattaforma middleware open source per lo sviluppo di applicazioni java che offre numerosi servizi. Il servizio più usato è quello di *Object-Relational Mapping* (ORM) che permette di mappare degli oggetti java su un database relazione in modo da gestire correttamente la persistenza e riducendo il codice *Java Database Connectivity* (JDBC) da dover inserire. Hibernate implementa le specifiche della *Java Persistence Api* (JPA) per la persistenza dei dati. Un altro servizio che fornisce che è stato utilizzato nel progetto è Hibernate Validator, utilizzato per la validazione dei dati degli account degli utenti che si vogliono registrare.

***3.3) Thymeleaf***

Thymeleaf è un *template engine* opensource che si può utilizzare sia in ambienti web che non web. È pensato per essere utilizzato nello strato di *View* in un paradigma *Model-View-Controller* (MVC) mirando ad essere un sostituto delle *Java Server Pages* (JSP). Offre una completa integrazione con Spring.

***3.4) Bootstrap***

Bootstrap è un framework per la creazione di siti e applicazioni web. È nato con lo scopo di uniformare e accelerare lo sviluppo delle interfacce web e dalla versione 2.0 permette di rendere i siti web responsivi, permettendo così di adattarsi alle miriade di schermi di misure differenti presenti oggigiorno. È una raccolta di strumenti grafici, stilistici e di impaginazione che permettono di avere a disposizione una gran quantità di funzionalità ed è compatibile con tutte le ultime versioni dei principali browser.

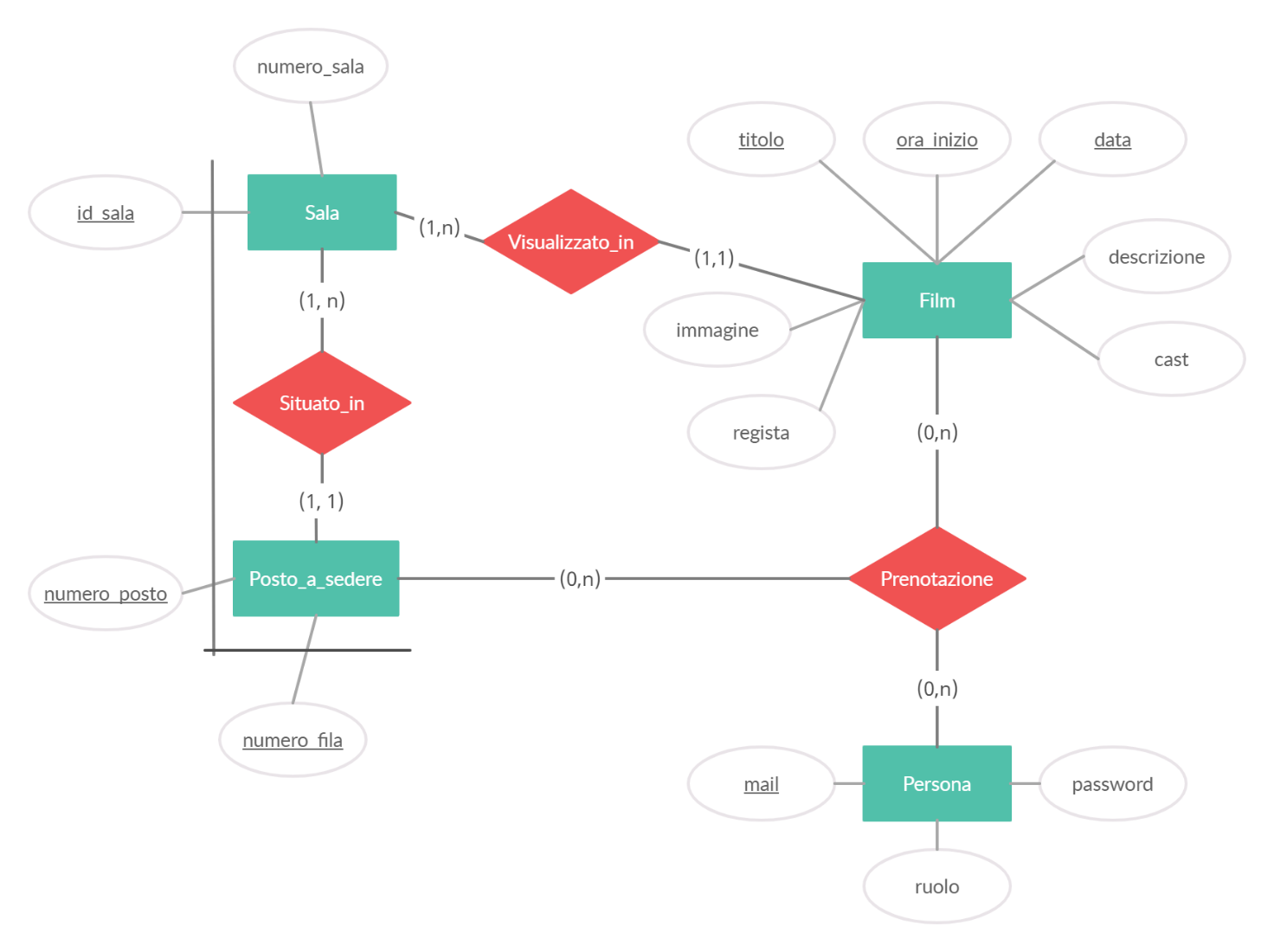
***4) BASE DATI***

La base dati è una componente necessaria per ogni progetto che ha bisogno di far persistere dei dati e in questo capitolo verranno mostrati i passi che hanno portato alla sua creazione.

***4.1) Modello Entity-Relationship***

La prima fase nella progettazione di una base dati è la fase di progettazione logica in cui, dopo un attento studio del modello di dominio dell’applicativo, si analizza “che cosa” dev’essere rappresentato. L’output di questa fase è il modello *Entity-Relationship* (ER), uno schema concettuale, composto principalmente da entità, relazioni e attributi.

* Entità: elementi che rappresentano gli oggetti del mondo reale che hanno esistenza autonoma
* Relazione: elemento che rappresenta come le entità sono collegate logicamente tra loro
* Attributi: rappresentano delle qualità che possono avere le entità o le relazioni

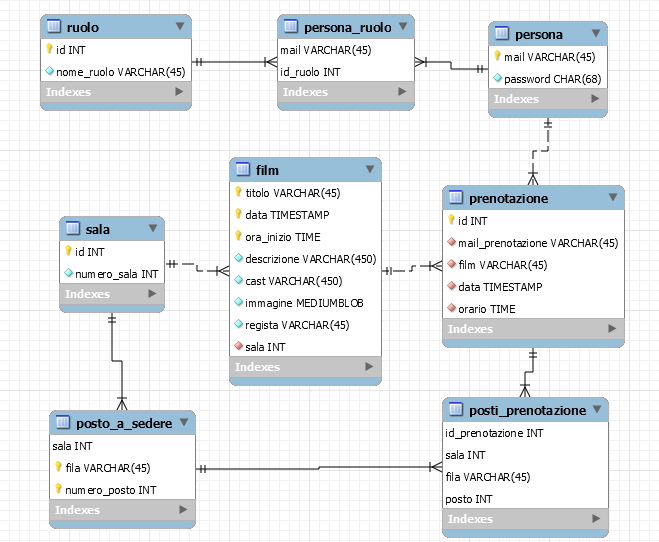


*Figura 4.1: Modello ER*

***4.2) Database MySql***

In seguito alla creazione del modello ER si è passati alla creazione del database vero e proprio utilizzando MySql, un sistema open source di gestione di database relazionali SQL (Structured Query Language).

Un database relazionale è composto da un insieme di relazioni, ovvero delle tabelle, che contengono tutti i dati. Queste tabelle rappresentano le entità o le relazioni e ognuna di esse ha delle chiavi primarie, ovvero un insieme di uno o più attributi che permettono di identificare univocamente una riga di una tabella. I collegamenti tra le tabelle avvengono per mezzo di queste chiavi primarie per evitare la ripetizione dei dati all’interno del database.

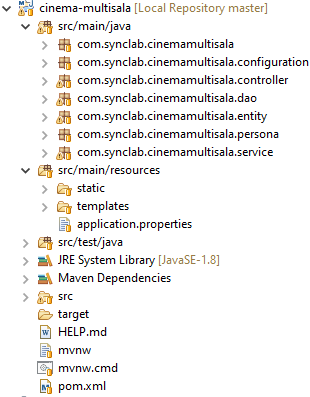


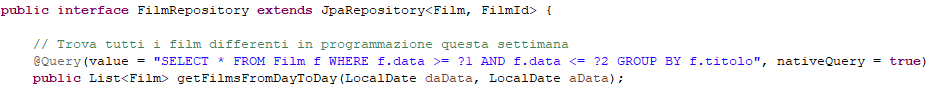
*Figura 4.2: Database MySql*

***5) STRUTTURA DEL PROGETTO JAVA***

Di seguito vengono analizzati i packages in cui è stato diviso il codice java.

Tutti i packages sono contenuti nel package principale *com.synclab.cinemamultisala* che contiene anche la classe *CinemaMultisalaApplication* che contiene il metodo main per far partire l’applicazione.



* *configuration*: contiene solamente due classi, *SecurityConfig* e *SecurityWebApplicationInitializr*, e si occupa di tutta la gestione della sicurezza impostando l’*authentication provider* e il password encoder di bcrytp e configurando i permessi in base al ruolo dell’account per i vari url dell’applicazione
* *controller*: questo package contiene tutti i controller del modello MVC. Sono presenti due classi *AmministratoreController* e *DipendenteController* che si occupano della gestione delle azioni che possono eseguire i rispettivi ruoli e gli altri controller si occupano ciascuno di una sezione dell’applicazione
* *dao*: dao significa Data Access Object, quindi questo package contiene tutte le interfacce che servono per interagire con il database. Ognuna di esse estende l’interfaccia *JPARepository<entity, primaryKey>* che supporta le operazioni CRUD (Create, Read, Update e Delete) e contiene eventuali query aggiuntive. Di seguito un esempio
* *entity*: in questo package si trovano le classi che rappresentano gli elementi del modello di dominio che sono salvati nel database. Ogni classe possiede i metodi getter e setter e per mezzo delle annotazioni sono mappate con le rispettive tabelle di MySql

*Figura 5: Struttura progetto*

***6) DETTAGLI DELL’INTERFACCIA UTENTE***

***7) CONCLUSIONI E POSSIBILITA’ DI SVILUPPI FUTURI***

Pagamenti, rimozione logica e non fisica dei film, aggiunta sezione per accedere ad eventuali altre sedi, aggiunta di una pagina con le informazioni riguardanti il cinema (es orario/indirizzo), conferma prenotazione con invio e-mail, sviluppo dell’applicazione mobile.