

Challenge - Backend Web Developer

2025-04-08

corrado.campisano@gmail.com

Indice generale

1 – Introduzione.....	2
1.1 – Strumenti utilizzati.....	2
1.2 – Scelte implementative (versione “vanilla”).....	3
2 – Diagrammi UML (versione “vanilla”).....	4
2.1 – Use Cases.....	4
2.2 – Diagramma delle classi.....	5
2.3 – Call sequence diagram.....	6
3 – Implementazione dei Test (versione “vanilla”).....	7
4 – Struttura repository e sorgenti (versione “vanilla”).....	8
5 – Screenshots del frontend (versione “vanilla”).....	9
5.1 – Home Page.....	9
5.2 – Storico Programmazione.....	10
5.3 – Film in Programmazione (lista completa).....	11
5.4 – Film in Programmazione (lista filtrata).....	12
6 – Oltre la versione “vanilla”: 0.0.3 “gestione sale”.....	13
6.1 - Diagrammi UML: nuovo diagramma delle classi.....	14
6.2 - Implementazione dei Test.....	15
6.3 – Test coverage.....	16
6.4 - Struttura repository e sorgenti.....	17
6.5 - Screenshots del frontend.....	18

1 – Introduzione

Il presente documento e' a corredo delle attivita' svolte per la “[Challenge – Backend Web Developer](#)” e riepiloga il materiale di documentazione del progetto.

I sorgenti ed il materiale di documentazione (README.mds, diagrammi UML, screenshots) sono disponibili sul repository GitHub “[Challenge – Backend Developer repository](#)”, nella cartella “[docs](#)”.

1.1 – Strumenti utilizzati

Per lo sviluppo, sono stati utilizzati i seguenti strumenti:

- Sistema operativo: Debian GNU/Linux 12 (da cui il vincolo a Java 17 ed Angular 17.3)
- IDE backend: Spring Tools Studio (STS);
- IDE frontend: Eclipse-Php (con estensioni per JS e TS);
- Dia, per i diagrammi UML;
- OpenOffice Write, per il documento corrente.

1.2 – Scelte implementative (versione “vanilla”)

Come documentato nel README.md, al fine di restare entro le 8 ore indicativamente ammesse per la realizzazione del progetto, si e' scelto di realizzare innanzitutto una implementazione di base, nel branch “vanilla”, rinunciando alla gestione delle “sale di proiezione” come entita' autonoma, da associare ai “film”, tramite una relazione @OneToMany e @ManyToOne.

Di seguito si riporta la screenshot del file README.md generale, per questa prima versione:

Challenge - Readme generale

Requirements

Implementazione del backend in Java/SpringBoot per la challenge [Backend Web Developer](#)

Version 0.0.1_vanilla

In questa implementazione di base non e' gestita la relazione tra i film in programmazione e le relative sale, che in effetti non e' richiesta esplicitamente nella challenge.

Per questo, in prima battuta, la "sala di proiezione" e' un semplice attributo dell'entity "Film".

Questa implementazione di base e' contenuta nel branch "vanilla", che viene implementato per primo, non potendo stimare il tempo necessario ad un'implementazione completa, nelle prime fasi di sviluppo, rispetto al tempo concesso per completare la challenge.

Sempre per motivi di tempo, in prima battuta:

- non vengono incluse le validazioni di base sui campi (quelle con "org.hibernate.validator") e relativi test cases;
- i test cases implementati si limitano alle restrizioni sulla durata delle programmazioni ed agli use cases ("lista film in proiezione" e "lista storico film passati");
- non viene validata l'eventuale sovrapposizione di piu' film in una sala di proiezione, non essendo gestita la relazione tra film e sale.

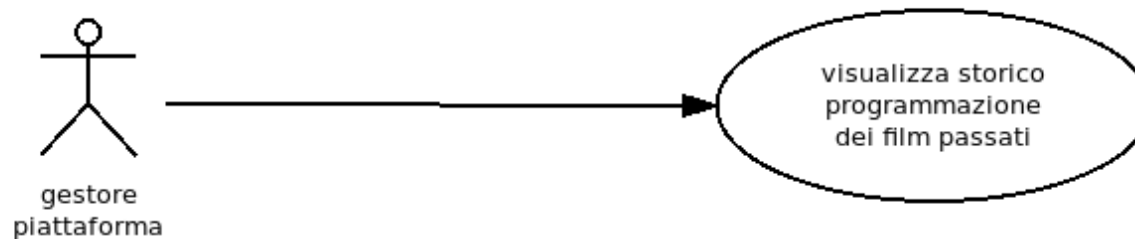
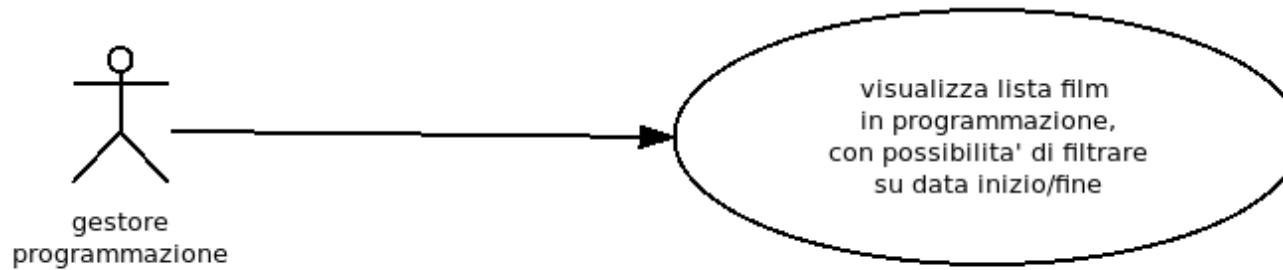
Vedere il branch "gestione_sale" per una implementazione piu' completa, che sara' realizzata compatibilmente con le restrizioni sulla data di consegna (vedere "negotiating requirements", nel contesto della metodologia agile).

2 – Diagrammi UML (versione “vanilla”)

Per la documentazione del progetto, sono stati creati i seguenti diagrammi UML.

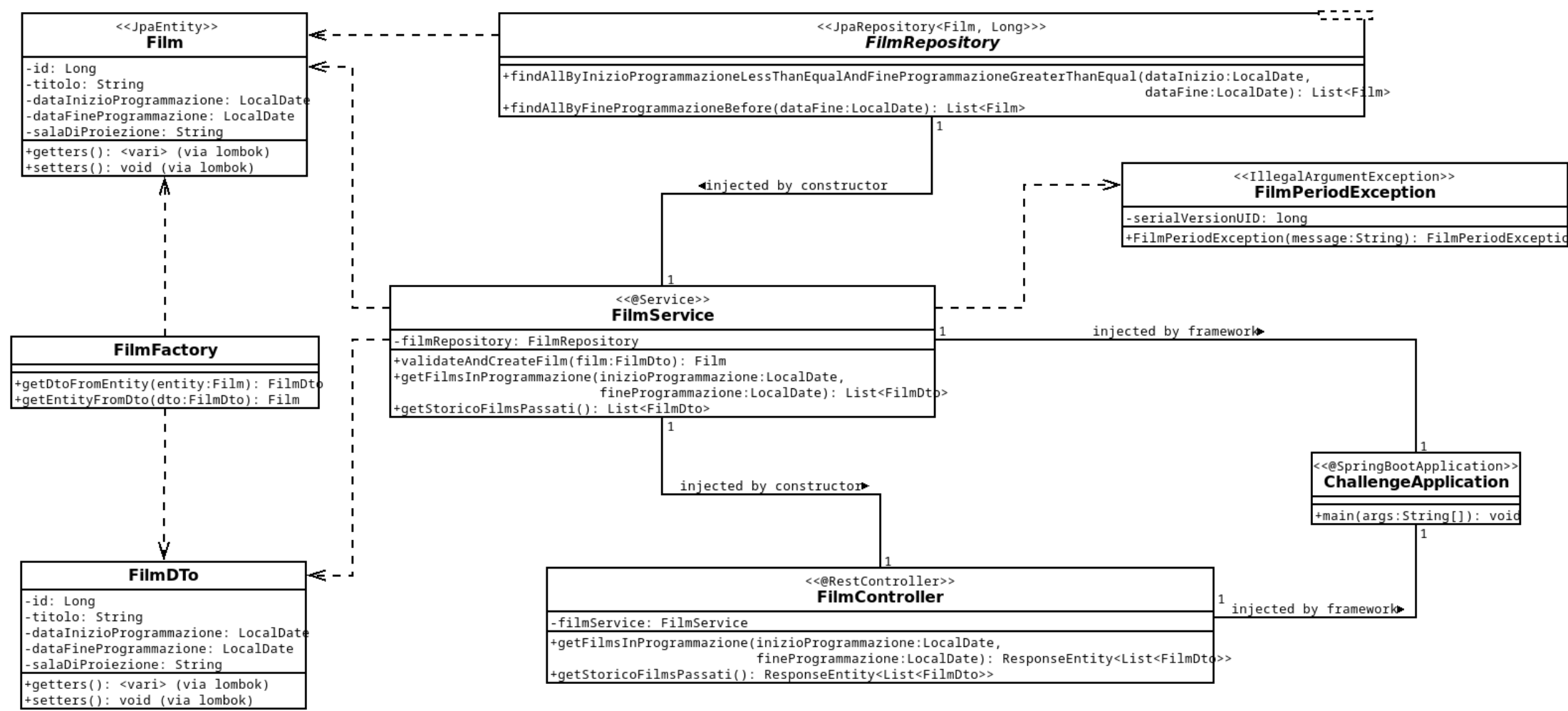
2.1 – Use Cases

I due use cases qui sotto riportati rappresentano i requisiti funzionali RF1: “Elenco Film” e RF2: “Storico”:



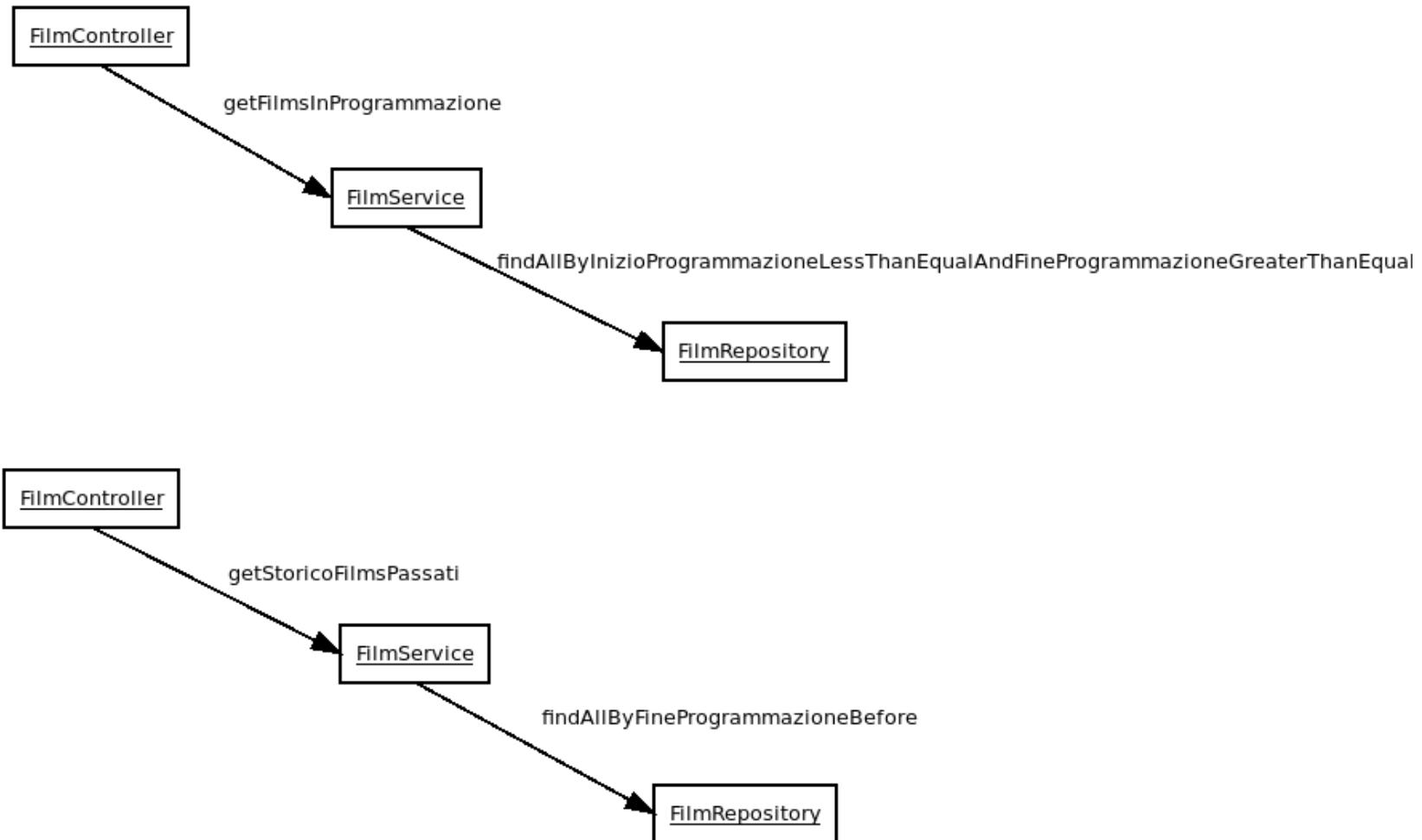
2.2 – Diagramma delle classi

Il diagramma delle classi sotto riportato rappresenta la struttura dei sorgenti del backend, escluso il package di supporto “config”:



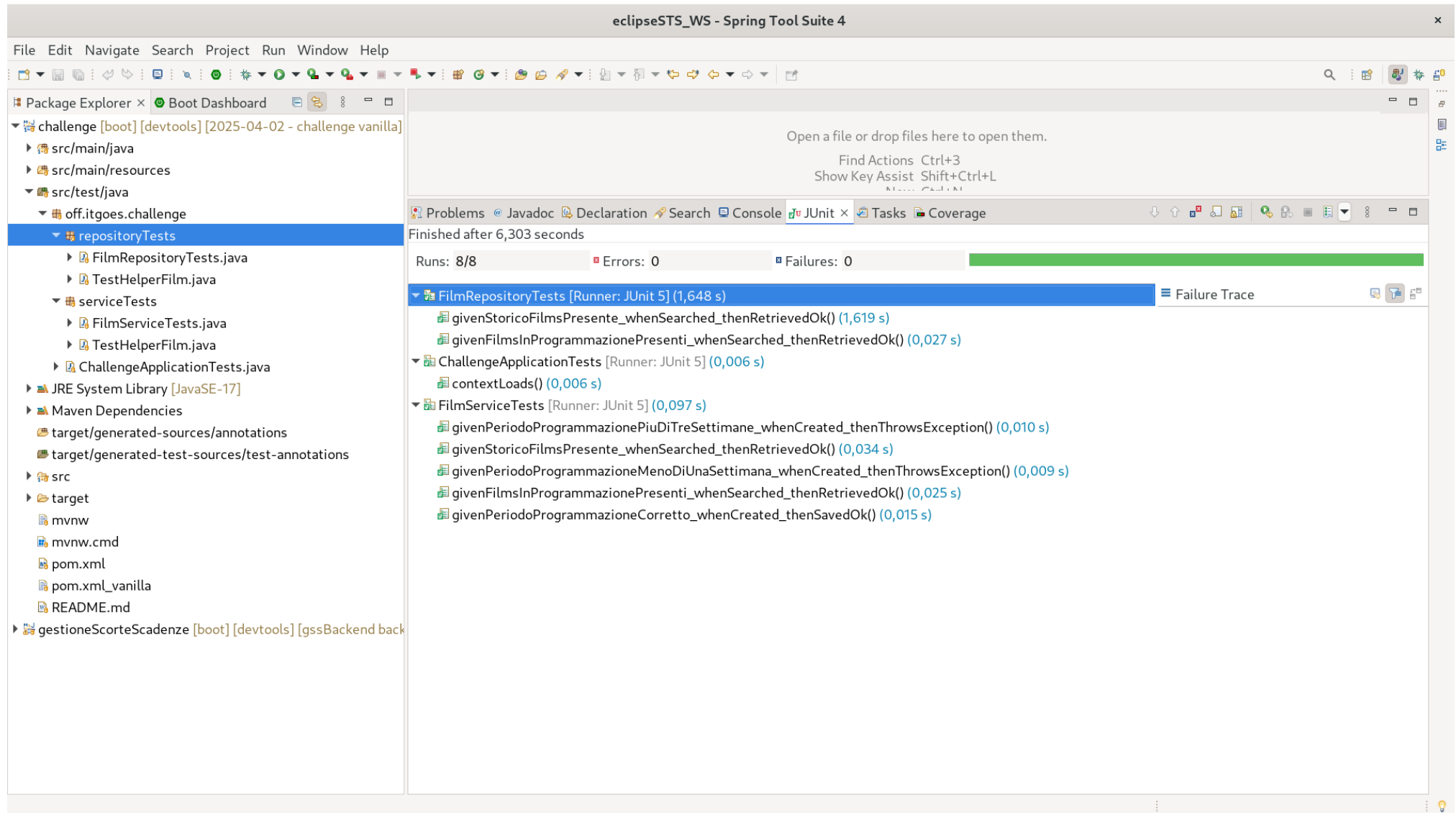
2.3 – Call sequence diagram

Il seguente diagramma rappresenta, nei limiti di quanto possibile con Dia, la sequenza delle chiamate, a partire dai due endpoint REST del controller “FilmController”:



3 – Implementazione dei Test (versione “vanilla”)



Per il progetto backend sono stati implementati solo test di integrazione sia a livello di repository che a livello di service (che contiene una logica di business piuttosto semplice), come evidenziato dalla shot sottostante:



4 – Struttura repository e sorgenti (versione “vanilla”)

Stante il legame tra backend e frontend, che risultano “tightly coupled” a causa del “contratto” definito dall’API REST, si e’ scelto di usare un solo repository sia per il codice del backend, che del frontend, nonche’ per la documentazione ed i file di supporto comuni.

Di seguito si riportano le screenshots con la struttura dei sorgenti del backend e del frontend:

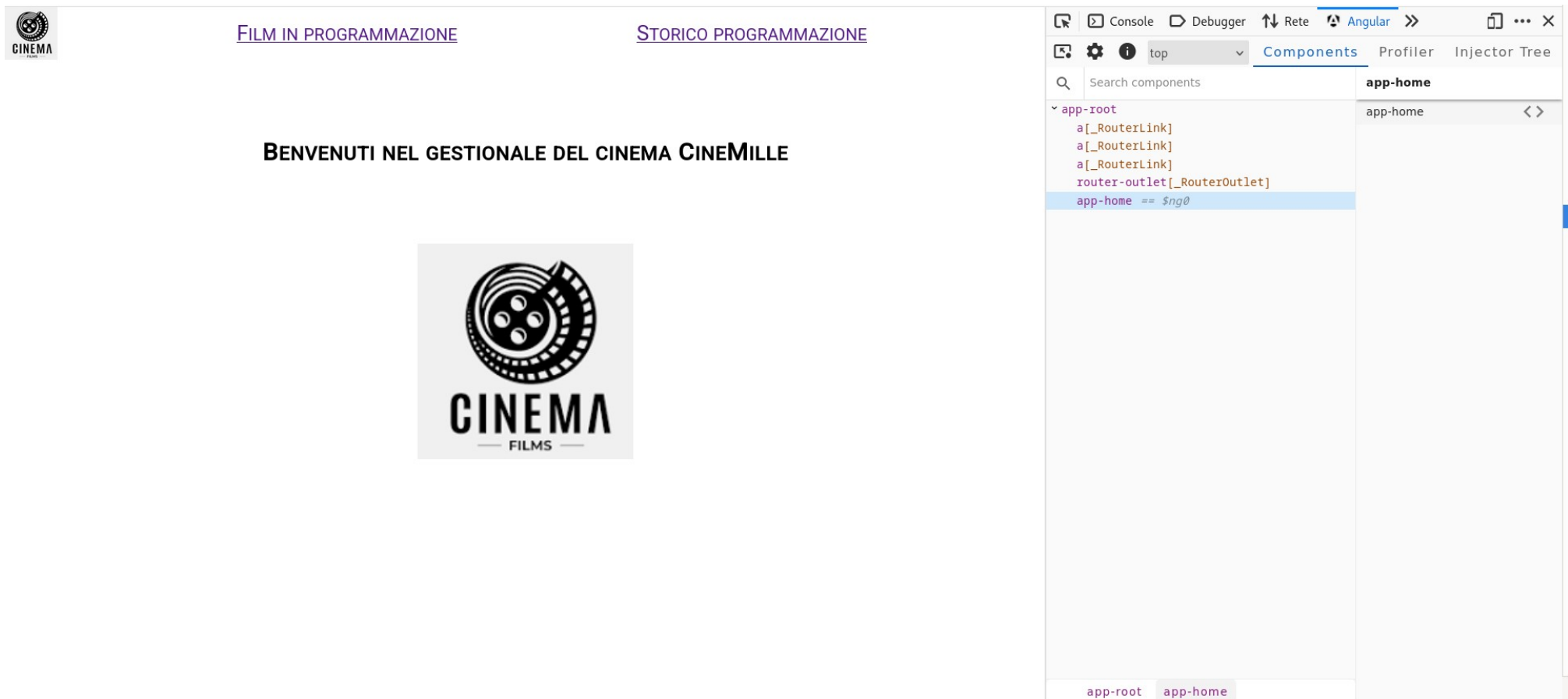
Struttura sorgenti backend	Struttura sorgenti frontend
	

5 – Screenshots del frontend (versione “vanilla”)

Di seguito si riportano le screenshots del frontend, ad evidenziare l’implementazione degli use cases.


5.1 – Home Page

Semplice home page di benvenuto, con un minimo di formattazione e di grafica, con in evidenza i componenti Angular:



5.2 – Storico Programmazione

Semplice pagina con la tabella della programmazione dei film passati, con in evidenza i componenti Angular:



[FILM IN PROGRAMMAZIONE](#)

[STORICO PROGRAMMAZIONE](#)

STORICO PROGRAMMAZIONE CINEMILLE

Titolo	Inizio Programmazione	Fine Programmazione	Sala di Proiezione
Il dottor Stranamore	2025-03-12	2025-03-25	sala 1
2001: odissea nello spazio	2025-03-19	2025-04-01	sala 2
Barry Lyndon	2025-03-26	2025-04-01	sala 1

ConsoleDebuggerReteAngular>>

ComponentsProfilerInjector Tree

Search components

app-root

app-root == \$ng0

a[_RouterLink]

a[_RouterLink]

a[_RouterLink]

router-outlet[_RouterOutlet]

app-history

table

tr

th[_MatHeaderCell]

th[_MatHeaderCell]

th[_MatHeaderCell]

th[_MatHeaderCell]

tr

td[_MatCell]

td[_MatCell]

td[_MatCell]

td[_MatCell]

tr

td[_MatCell]

td[_MatCell]

td[_MatCell]

td[_MatCell]

tr

td[_MatCell]

td[_MatCell]

td[_MatCell]

td[_MatCell]


Properties

title: Challenge

app-root

5.3 – Film in Programmazione (lista completa)

Semplice pagina con la tabella dei film in programmazione, con tutti i risultati (non filtrati), con in evidenza i componenti Angular:



[FILM IN PROGRAMMAZIONE](#)

[STORICO PROGRAMMAZIONE](#)

PROGRAMMAZIONE CORRENTE CINEMILLE

Enter a date range

format: MM/DD/YYYY – MM/DD/YYYY

Filtra per data di inizio/data fine

Cerca tutti

Titolo	Inizio Programmazione	Fine Programmazione	Sala di Proiezione
Arancia meccanica	2025-03-19	2025-04-08	sala 3
Full Metal Jacket	2025-03-26	2025-04-08	sala 4

Console Debugger Rete Angular

Components Profiler Injector Tree

Search components

app-root

app-root == \$ng0

a[_RouterLink]

a[_RouterLink]

a[_RouterLink]

router-outlet[_RouterOutlet]

app-current

app-date-range-picker

mat-form-field OnPush

label[_MatFormFieldFloatingLabel]

mat-label[_MatLabel]

mat-date-range-input[_NgControlSta]

div[_CdkMonitorFocus]

input[_MatStartDate, _DefaultV]

input[_MatEndDate, _DefaultVa]

mat-date-range-picker OnPush

mat-datepicker-toggle[_MatSuffix]

button OnPush

div[_MatFormFieldLineRipple]

mat-hint[_MatHint]

table

tr

th[_MatHeaderCell]

th[_MatHeaderCell]

th[_MatHeaderCell]

th[_MatHeaderCell]

tr

td[_MatCell]

td[_MatCell]

td[_MatCell]

td[_MatCell]

tr

td[_MatCell]

app-root


app-root

Properties

title: Challenge

5.4 – Film in Programmazione (lista filtrata)

Semplice pagina con la tabella dei film in programmazione, con i risultati filtrati, con in evidenza i componenti Angular:



[FILM IN PROGRAMMAZIONE](#)

[STORICO PROGRAMMAZIONE](#)

PROGRAMMAZIONE CORRENTE CINEMILLE

Enter a date range
3/20/2025 – 4/7/2025
format: MM/DD/YYYY – MM/DD/YYYY

Filtra per data di inizio/data fine

Cerca tutti

Titolo	Inizio Programmazione	Fine Programmazione	Sala di Proiezione
Arancia meccanica	2025-03-19	2025-04-08	sala 3

app-root

app-root == \$ng0

a[_RouterLink]

a[_RouterLink]

a[_RouterLink]

router-outlet[_RouterOutlet]

app-current

app-date-range-picker

mat-form-field OnPush

label[_MatFormFieldFloatingLabel]

mat-label[_MatLabel]

mat-date-range-input[_NgControlSt

div[_CdkMonitorFocus]

input[_MatStartDate, _Default'

input[_MatEndDate, _DefaultVa

mat-date-range-picker OnPush

mat-datepicker-toggle[_MatSuffix]

button OnPush

div[_MatFormFieldLineRipple]

mat-hint[_MatHint]

table

tr

th[_MatHeaderCell]

th[_MatHeaderCell]

th[_MatHeaderCell]

th[_MatHeaderCell]

tr

td[_MatCell]

td[_MatCell]

td[_MatCell]

td[_MatCell]

app-root

app-root

Properties

title: Challenge

6 – Oltre la versione “vanilla”: 0.0.3 “gestione sale”

Avendo impiegato circa altre 8 ore di tempo, si e’ proceduto a realizzare una versione piu’ completa, in particolare per il backend, lavorando sul branch “gestione_sale”, di cui e’ stato poi fatto il merge sul main.

Di seguito si riporta la screenshot del file README.md generale aggiornato alla versione 0.0.3 “gestione sale”:

Challenge - Readme generale

Requirements

Implementazione del backend in Java/SpringBoot per la challenge [Backend Web Developer](#)

Version 0.0.3 "gestione sale"

In questa versione e' gestita la relazione tra i film in programmazione e le relative sale, anche se non e' richiesta esplicitamente nella challenge, utilizzando le relazioni @OneToMany e @ManyToOne di JPA.

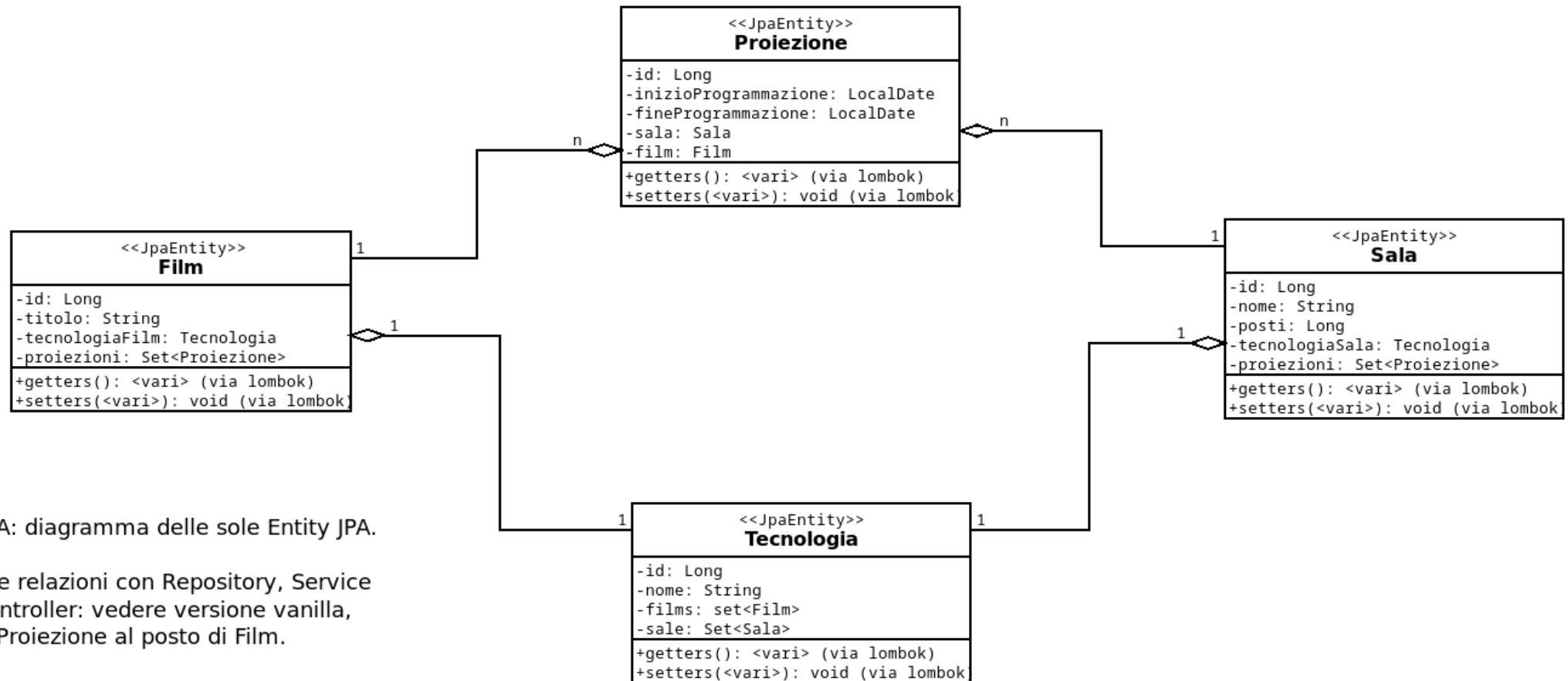
Sono stati inoltre aggiunti e/o migliorati i controlli di validazione, vedere readme del backend per i dettagli.

Per testare tali controlli di validazione, sono state ristrutturate le classi di test, sia a livello di database/entity che di servizi/business logic.

In questa versione non e' cambiato molto sul frontend, se non a livello dell'interface "film.type", che e' stata riallineata alla ProiezioneDto del backend, con un semplice cambiamento nella nomenclatura dei campi.

6.1 - Diagrammi UML: nuovo diagramma delle classi

Dei diagrammi UML e' stato aggiornato solo il diagramma delle classi, semplificato ad evidenziare le relazioni tra le entity Tecnologia, Sala, Film e Proiezione

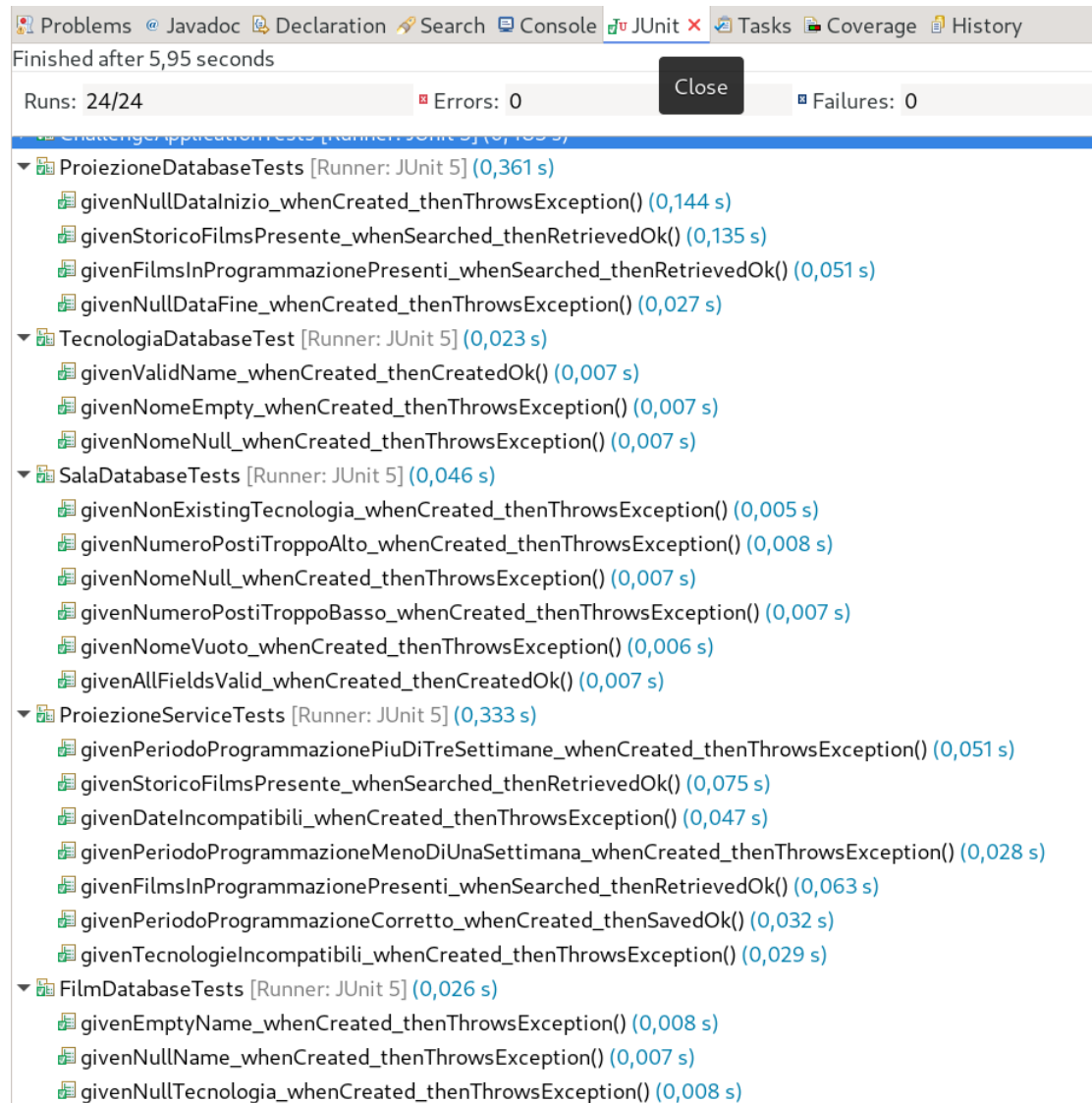


NOTA: diagramma delle sole Entity JPA.

Per le relazioni con Repository, Service e Controller: vedere versione vanilla, con Proiezione al posto di Film.

6.2 - Implementazione dei Test

In questa versione, per il progetto backend sono stati implementati solo test di integrazione sia a livello di repository che a livello di service (che contiene una logica di business relativamente piu' complessa), come evidenziato dalla shot sottostante:



6.3 – Test coverage

In questa versione, per il progetto backend si e' ottenuta una test coverage accettabile, come evidenziato dalla shot sottostante:

The screenshot displays an IDE with a Java file named `ProiezioneFactory.java` open. The code includes logic for finding a film by ID and throwing an exception if it's not found. Below the code editor, the `Coverage` tab is active, showing a detailed report for the project `off.itgoes.challenge` as of 8 apr 2025 17:29:26.

Element	Coverage	Covered Instr	Missed Inst	Total Instructions
challenge	90,8 %	1.538	156	1.694
src/main/java	80,1 %	331	82	413
off.itgoes.challenge.programmazione.proiezione	79,0 %	282	75	357
ProiezioneController.java	0,0 %	0	48	48
ProiezioneFactory.java	73,6 %	64	23	87
ProiezioneService.java	94,6 %	70	4	74
Proiezione.java	100,0 %	3	0	3
ProiezioneBusinessLogic.java	100,0 %	142	0	142
ProiezioneDto.java	100,0 %	3	0	3
off.itgoes.challenge	37,5 %	3	5	8
ChallengeApplication.java	37,5 %	3	5	8
off.itgoes.challenge.config	93,5 %	29	2	31
MariaDbCollateDialect.java	60,0 %	3	2	5
CorsConfig.java	100,0 %	26	0	26
off.itgoes.challenge.programmazione.film	100,0 %	3	0	3
off.itgoes.challenge.programmazione.proiezione.exceptions	100,0 %	8	0	8
off.itgoes.challenge.programmazione.sala	100,0 %	3	0	3
off.itgoes.challenge.programmazione.tecnologia	100,0 %	3	0	3
src/test/java	94,2 %	1.207	74	1.281

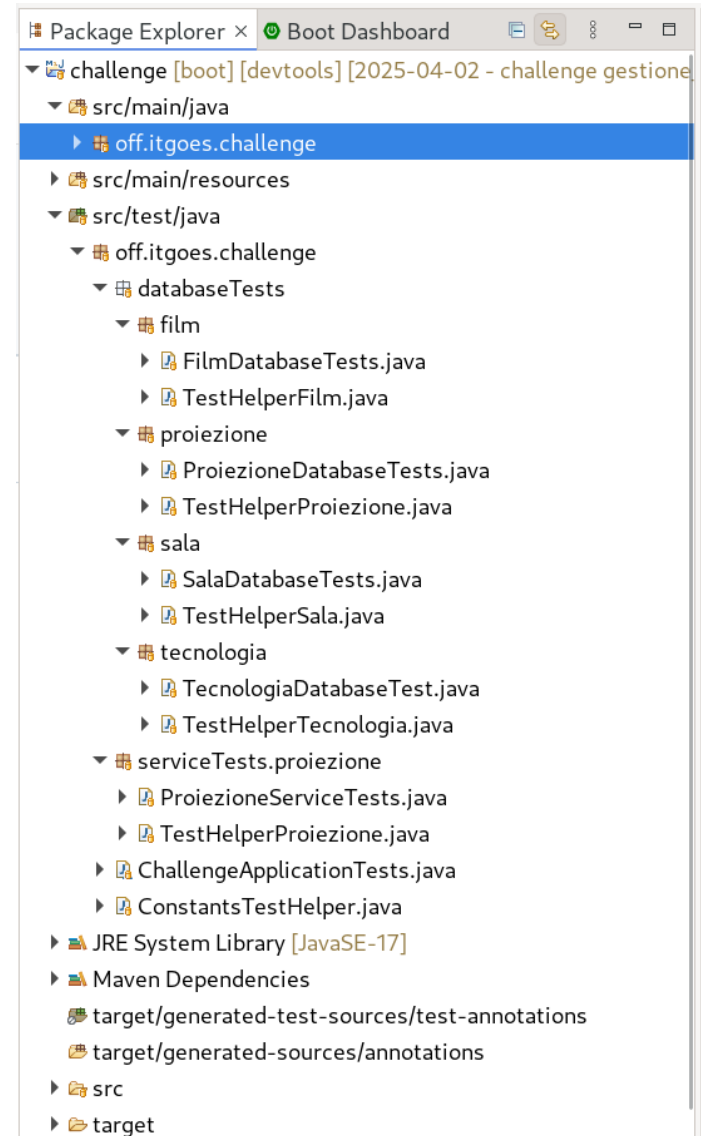
6.4 - Struttura repository e sorgenti

La struttura dei sorgenti del frontend non è cambiata, in questa versione.

Struttura sorgenti backend – produzione




Struttura sorgenti backend – test



6.5 - Screenshots del frontend

Non essendo cambiato molto sul frontend, si riporta solo la pagina con la tabella dei film in programmazione, con tutti i risultati (non filtrati), con in evidenza la nuova struttura dell’interface/type “film.type”:



[FILM IN PROGRAMMAZIONE](#)[STORICO PROGRAMMAZIONE](#)

PROGRAMMAZIONE CORRENTE CINEMILLE

Enter a date range

format: MM/DD/YYYY – MM/DD/YYYY

Filtra per range di date

Cerca per oggi

Titolo	Inizio Programmazione	Fine Programmazione	Sala di Proiezione
Arancia meccanica	2025-03-25	2025-04-14	sala 3
Full Metal Jacket	2025-04-01	2025-04-14	sala 4

ConsoleDebuggerReteEditor stiliPrestazioniMemoria

Filtra URL

TuttiHTMLCSSJSXHRCaratteriImmaginiMediaWSAltro

Stato	Metodo	Dominio	File	Iniziatore	Tipo	Trasferito	D...
200	GET	localhost:...	filmsInProgrammazione?inizioProgrammazi	polyfills.js:22...	json	612 B	3...

Una richiesta321 B di 612 B trasferitiCompletato: 16 ms

HeaderCookieRichiestaRispostaTempiAnalisi dello stack

Filtra proprietà

JSONNon elaborata (raw)

0: Object { id: 109, inizioProgrammazione: "2025-03-25", fineProgrammazione: "2025-04-14", ... }

id: 109

inizioProgrammazione: "2025-03-25"

fineProgrammazione: "2025-04-14"

nomeSala: "sala 3"

idSala: 149

titoloFilm: "Arancia meccanica"

idFilm: 434

1: Object { id: 110, inizioProgrammazione: "2025-04-01", fineProgrammazione: "2025-04-14", ... }