



UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA

CORSO DI LAUREA IN INFORMATICA

TESI DI LAUREA IN INGEGNERIA DEL SOFTWARE

**“PROGRESO”: LA SOLUZIONE PER LA GESTIONE
INTEGRATA DEI PROGETTI SOFTWARE**

Relatore:

Prof.ssa

Rita Francese

Candidato:

Emanuele Setaro

Matricola: 0512107414

ANNO ACCADEMICO 24/25



Sommario

| | | |
|-------|--|----|
| 1. | Introduzione | 3 |
| 1.1 | Contesto e motivazione del progetto | 3 |
| 1.1.1 | Contesto generale | 3 |
| 1.1.2 | Motivazione del progetto | 3 |
| 1.2 | Obiettivi della tesi | 4 |
| 1.3 | Stato dell'arte | 5 |
| 1.4 | Definizioni, acronimi e abbreviazioni | 6 |
| 1.4.1 | Definizioni..... | 6 |
| 1.4.2 | Acronimi e abbreviazioni | 6 |
| 1.5 | Struttura della tesi..... | 7 |
| 2. | Analisi dei Requisiti | 9 |
| 2.1 | Requisiti funzionali | 9 |
| 2.1.1 | Gestione dei progetti | 9 |
| 2.1.2 | Gestione dei task | 9 |
| 2.1.3 | Gestione dei team | 10 |
| 2.1.4 | Gestione dei commenti..... | 10 |
| 2.1.5 | Gestione degli utenti..... | 10 |
| 2.1.6 | Gestione dell'autenticazione | 11 |
| 2.2 | Requisiti non funzionali (FURPS+) | 12 |
| 2.2.1 | Functionality..... | 12 |
| 2.2.2 | Usability | 12 |
| 2.2.3 | Reliability | 12 |
| 2.2.4 | Performance | 12 |
| 2.2.5 | Supportability | 13 |
| 2.3 | Modello dei casi d'uso | 14 |
| 2.3.1 | Diagramma dei casi d'uso | 14 |
| 2.3.2 | Descrizione dei principali casi d'uso..... | 18 |
| 2.4 | Scelte tecnologiche..... | 25 |
| 2.4.1 | Back-end: <i>Spring Boot, Maven, JPA/Hibernate</i> | 25 |
| 2.4.2 | Front-end: <i>Angular</i> | 25 |
| 3. | Progettazione del Sistema | 27 |
| 3.1 | Architettura generale del sistema | 27 |



| | | |
|-------|---|-----|
| 3.1.1 | Separazione di back-end e front-end..... | 27 |
| 3.1.2 | Comunicazione tra front-end e back-end | 27 |
| 3.1.3 | Comunicazione tra back-end e database tramite <i>JPA/Hibernate</i> | 28 |
| 3.1.4 | Panoramica completa delle interazioni tra back-end e front-end..... | 28 |
| 3.2 | Decomposizione in sottosistemi..... | 30 |
| 3.2.1 | Diagramma dei sottosistemi | 31 |
| 3.2.2 | Spiegazione delle dipendenze dei sottosistemi | 31 |
| 3.3 | Gestione dei dati persistenti | 33 |
| 3.3.1 | Entity class diagram | 33 |
| 4. | Implementazione | 35 |
| 4.1 | Sviluppo del back-end..... | 35 |
| 4.1.1 | Struttura dei package..... | 35 |
| 4.2 | Sviluppo del front-end..... | 48 |
| 4.2.1 | Struttura delle cartelle | 48 |
| 4.2.2 | Interfaccia utente | 52 |
| 5. | Testing e validazione | 75 |
| 5.1 | Test unitari..... | 75 |
| 5.2 | Test di interfaccia utente..... | 76 |
| 6. | Conclusioni..... | 77 |
| 6.1 | Sintesi dei risultati ottenuti..... | 77 |
| 6.2 | Possibili miglioramenti futuri..... | 77 |
| 7. | Appendici | 79 |
| 7.1 | Appendice A..... | 79 |
| 7.1.1 | Appendice A.1: Descrizione delle relazioni | 79 |
| 7.1.2 | Appendice A.2: Schema logico | 80 |
| 7.1.3 | Appendice A.3: Dizionario dei dati | 81 |
| 7.2 | Appendice B | 84 |
| 7.2.1 | Appendice B.1: Descrizione approfondita dei package | 84 |
| 7.3 | Appendice C | 109 |
| 7.3.1 | Appendice C.1: Documentazione dei test case unitari | 109 |
| 7.3.2 | Appendice C.2: Documentazione dei test case dell’interfaccia utente | 117 |



1. Introduzione

1.1 Contesto e motivazione del progetto

1.1.1 Contesto generale

Nei luoghi di lavoro moderni, i team affrontano sfide legate all'organizzazione dei progetti, alla delega dei task e alla comunicazione interna. Con il crescente utilizzo di ambienti di lavoro remoti e ibridi, la richiesta di soluzioni digitali che migliorino direttamente la gestione dei progetti è in costante aumento. Si tratta di garantire che determinati progetti siano assegnati alle persone giuste, stabilire scadenze per assicurarsi che tutte le attività individuali vengano completate in tempo e massimizzare la produttività del progetto.

1.1.2 Motivazione del progetto

Nel processo di sviluppo del sistema “Progresso”, è emersa la necessità di affrontare alcune criticità legate al lavoro in team, quali:

- **Frammentazione degli strumenti:** la decentralizzazione nella gestione delle attività porta i team a utilizzare strumenti differenti come e-mail, fogli elettronici e piattaforme di messaggistica, generando frammentazione e confusione.
- **Difficoltà nel monitoraggio dei progressi:** il monitoraggio dei progressi risulta problematico in assenza di un sistema di tracciamento aperto. Senza la possibilità di tenere traccia delle scadenze e dei task da completare, diventa difficile valutare lo stato di avanzamento dei progetti, individuare eventuali ritardi e intervenire tempestivamente per ripristinare il corretto flusso di lavoro. Questo comporta non solo una minore trasparenza, ma anche un aumento del rischio di incomprensioni e inefficienze operative che possono compromettere il raggiungimento degli obiettivi stabiliti.



1.2 Obiettivi della tesi

I principali obiettivi della tesi sono:

- **Piattaforma centralizzata per la gestione dei progetti:** è fondamentale sviluppare una piattaforma centralizzata per la gestione dei progetti software al fine di integrare tutti gli aspetti cruciali. Questo sistema facilita la comunicazione istantanea e l'assegnazione dei task, con l'obiettivo di risolvere i problemi derivanti dall'uso di strumenti obsoleti. Si fornisce così ai team una piattaforma unificata per gestire tutti i requisiti dei progetti in modo efficiente.
- **Interfaccia utente intuitiva e reattiva:** per garantire un'esperienza utente di qualità, è fondamentale progettare un'interfaccia intuitiva e reattiva che risulti facile da navigare e utilizzare per gli utenti. L'applicazione includerà un dashboard moderno e strumenti interattivi per la visualizzazione dei dati dei progetti.
- **Scalabilità e ottimizzazione delle prestazioni:** per garantire un'efficace gestione di elevati volumi di dati e richieste, assicurando un'esperienza utente fluida anche in situazioni di elevata domanda, si propone l'adozione di un'architettura ad alte prestazioni basata su *Spring Boot* e *Angular*.
- **Sicurezza e integrità dei dati:** garantire la sicurezza e l'integrità dei dati è fondamentale per proteggere le informazioni sensibili del progetto. È essenziale implementare misure di sicurezza robuste, come stringenti controlli di accesso, crittografia dei dati e regolari controlli di sicurezza per assicurare che i dati siano protetti e non siano soggetti a utilizzi non autorizzati.
- **Testing approfondito e validazione del sistema:** per garantire che l'applicazione soddisfi tutti i requisiti funzionali e non funzionali, è importante eseguire test approfonditi. Questo assicura l'affidabilità, l'accuratezza e la stabilità complessiva del sistema, rivelando anche possibili aree di miglioramento per il futuro.



1.3 Stato dell'arte

Negli ultimi anni sono emerse numerose soluzioni digitali nel settore della gestione dei progetti software. *Asana*, *Jira*, *Trello* e *Microsoft Project* sono alcuni dei programmi più popolari. Questi programmi includono funzioni avanzate per la pianificazione, la collaborazione e il reporting. Tuttavia, queste soluzioni presentano alcune criticità:

- **Complessità di utilizzo:** strumenti come *Jira*, sebbene potenti, richiedono una formazione specifica e una configurazione accurata, rendendoli meno accessibili per team con competenze tecniche limitate.
- **Costi e dipendenza da servizi esterni:** molte piattaforme operano su modelli di abbonamento e sono basate su soluzioni cloud, comportando costi elevati e una dipendenza da provider esterni, con implicazioni sulla sicurezza e sul controllo dei dati.
- **Flessibilità e personalizzazione limitate:** le soluzioni commerciali, pur offrendo un ampio ventaglio di funzionalità, spesso non permettono una personalizzazione profonda, limitando l'adattamento a esigenze specifiche di determinati contesti lavorativi.

Alla luce di queste considerazioni, il sistema “Progresso”, proposto in questa tesi, mira a colmare queste lacune attraverso un software scalabile, personalizzabile e *self-hosted*¹, che offre maggiore controllo sui dati e un’esperienza d’uso semplificata.

¹ Il termine *self-hosted* indica un sistema software che viene installato e gestito direttamente sui server dell’organizzazione o in un ambiente privato, anziché utilizzare soluzioni cloud gestite da terze parti. Questa modalità consente un maggiore controllo sui dati, una personalizzazione più approfondita e una gestione diretta della sicurezza e della configurazione del sistema.



UNIVERSITÀ DEGLI STUDI
DI SALERNO

1.4 Definizioni, acronimi e abbreviazioni

1.4.1 Definizioni

| Sigla/Termine | Definizione |
|------------------------|--|
| Admin | Utente con privilegi avanzati per gestire e configurare il sistema. |
| Project Manager | Utente responsabile della pianificazione, esecuzione e gestione di un progetto. |
| Team Member | Utente che partecipa ad un progetto, con accesso ai task e alle risorse assegnate. |
| Progetto | Insieme di task organizzate con un obiettivo specifico, risorse definite e una scadenza. |
| Task | Un'attività o un compito specifico all'interno di un progetto, solitamente con una scadenza e un responsabile assegnato. |
| Team | Un gruppo di persone che collaborano insieme per raggiungere obiettivi comuni all'interno di un progetto. |
| Commento | Una nota o osservazione scritta aggiunta ad un progetto, utilizzata per fornire feedback, condividere informazioni o risolvere problemi relativi a specifici aspetti del lavoro. |

1.4.2 Acronimi e abbreviazioni

- **AM** = Admin
- **PM** = Project Manager
- **TM** = Team Member
- **RF** = Requisito funzionale
- **NFR** = Requisito non funzionale
- **FURPS** = Funcitonality (Funzionalità), Usability (Usabilità), Reliability (Affidabilità), Performance, Supportability (Supportabilità)
- **FU** = Funcitonality
- **US** = Usability
- **RL** = Reliability
- **PF** = Performance
- **SU** = Supportability
- **UC** = Use case
- **UCD** = Use case diagram
- **TC** = Test case



1.5 Struttura della tesi

La presente tesi è strutturata in modo da guidare progressivamente il lettore attraverso il contesto, l'analisi, la progettazione, l'implementazione e la validazione del sistema "Progresso".

- **Capitolo 2 – Analisi dei Requisiti:** in questo capitolo vengono definiti e analizzati nel dettaglio i requisiti funzionali e non funzionali del sistema, illustrando le funzionalità chiave (gestione di progetti, task, team, commenti, utenti e autenticazione), il modello dei casi d'uso che guida il comportamento dell'applicazione e le scelte tecnologiche adottate per soddisfare gli obiettivi progettuali.
- **Capitolo 3 – Progettazione del sistema:** in questo capitolo viene definita l'architettura complessiva del sistema, evidenziando la chiara separazione tra back-end e front-end e illustrando le modalità di comunicazione, sia tra il front-end e il back-end che tra il back-end ed il database. Inoltre, il capitolo presenta la decomposizione del sistema in sottosistemi, supportata da diagrammi che illustrano le dipendenze e le interazioni tra i vari moduli, e termina con la gestione dei dati persistenti, illustrata tramite l'entity class diagram.
- **Capitolo 4 – Implementazione:** in questo capitolo viene illustrato lo sviluppo pratico del sistema. Vengono presentate le soluzioni adottate per la realizzazione del back-end, evidenziando come la struttura dei package organizzi il codice e gestisca la comunicazione con il database. Viene inoltre descritto lo sviluppo del front-end, in cui la struttura delle cartelle è stata definita per garantire una chiara separazione delle responsabilità, e l'interfaccia utente viene esposta attraverso screenshot che ne evidenziano i principali flussi operativi.
- **Capitolo 5 – Testing e validazione:** in questo capitolo vengono illustrate le attività di testing e validazione effettuate per garantire la qualità e l'affidabilità del sistema. Sono stati eseguiti test unitari e test di validazione dell'interfaccia utente, al fine di verificare il corretto funzionamento delle principali funzionalità di ciascun sottosistema.
- **Capitolo 6 – Conclusioni:** questo capitolo sintetizza i risultati raggiunti durante lo sviluppo del sistema "Progresso" e analizza le prospettive future. La prima parte offre una sintesi dei risultati ottenuti, evidenziando come gli obiettivi siano stati raggiunti grazie alle tecnologie e metodologie adottate. La seconda parte esplora possibili miglioramenti futuri, suggerendo ulteriori sviluppi e ottimizzazioni del sistema per ampliare le funzionalità e migliorare l'efficienza operativa.
- **Capitolo 7 – Appendici:** Questo capitolo raccoglie la documentazione tecnica di supporto alla tesi. In particolare:
 - **Appendice A:** fornisce la descrizione delle relazioni, lo schema logico e il dizionario dei dati, con lo scopo di chiarire la struttura informativa del sistema.
 - **Appendice B:** offre una descrizione approfondita dei package, illustrando in dettaglio l'organizzazione e le responsabilità dei vari sottosistemi.



UNIVERSITÀ DEGLI STUDI
DI SALERNO

- **Appendice C:** contiene la documentazione completa dei test case, sia unitari che dell’interfaccia utente, garantendo una visione esaustiva delle attività di validazione.



2. Analisi dei Requisiti

2.1 Requisiti funzionali

2.1.1 Gestione dei progetti

| ID | Descrizione | Priorità |
|---------------|--|----------|
| RF_AM_1 | L'admin deve poter visualizzare tutti i progetti anche tramite filtri di ricerca. | Alta |
| RF_AM_2 | L'admin deve poter creare un nuovo progetto nel sistema fornendo tutte le informazioni necessarie per il progetto. | Alta |
| RF_AM_3 | L'admin deve poter aggiornare il project manager assegnato a un progetto. | Alta |
| RF_PM_1 | Il project manager deve poter visualizzare i progetti che gestisce tramite anche filtri di ricerca. | Alta |
| RF_AM_PM_1 | L'admin e il project manager devono poter aggiornare le informazioni di un progetto esistente. | Alta |
| RF_AM_PM_2 | L'admin e il project manager devono poter segnare un progetto come cancellato. | Alta |
| RF_AM_PM_3 | L'admin e il project manager devono poter assegnare un team a un progetto esistente. | Alta |
| RF_AM_PM_4 | L'admin e il project manager devono poter riassegnare un team a un progetto esistente. | Alta |
| RF_AM_PM_5 | L'admin e il project manager devono poter segnare un progetto come completato. | Alta |
| RF_TM_1 | Il team member deve poter visualizzare i progetti a cui sta partecipando. | Alta |
| RF_AM_PM_TM_1 | L'admin, il project manager e il team member devono poter visualizzare la percentuale di completamento di un progetto, la quale si basa sui task completati rispetto ai task totali. | Alta |
| RF_AM_PM_TM_2 | L'admin, il project manager e il team member devono poter visualizzare i dettagli di uno specifico progetto. | Alta |

2.1.2 Gestione dei task

| ID | Descrizione | Priorità |
|---------------|---|----------|
| RF_AM_PM_6 | L'admin e il project manager devono poter creare un nuovo task e assegnarlo ad un team member. | Alta |
| RF_AM_PM_7 | L'admin e il project manager devono poter riassegnare un task ad un team member diverso da quello precedente. | Alta |
| RF_AM_PM_8 | L'admin e il project manager devono poter aggiornare le informazioni di un task esistente. | Alta |
| RF_AM_PM_9 | L'admin e il project manager devono poter segnare un task come completato. | Alta |
| RF_AM_PM_10 | L'admin e il project manager devono poter rimuovere un task da un progetto. | Alta |
| RF_AM_PM_TM_3 | L'admin, il project manager e il team member devono poter visualizzare i task di un progetto utilizzando anche i filtri di ricerca. | Alta |



2.1.3 Gestione dei team

| ID | Descrizione | Priorità |
|-------------|--|----------|
| RF_AM_4 | L'admin deve poter visualizzare tutti i team, applicando anche filtri di ricerca. | Alta |
| RF_AM_5 | L'admin deve poter disattivare un team. | |
| RF_AM_PM_11 | L'admin e il project manager devono poter visualizzare i team disponibili, ossia i team senza progetti attivi. | Alta |
| RF_AM_PM_12 | L'admin e il project manager devono poter creare un nuovo team fornendo tutte le informazioni necessarie. | Alta |
| RF_AM_PM_13 | L'admin e il project manager devono poter aggiornare il nome di un team esistente. | Alta |
| RF_AM_PM_14 | L'admin e il project manager devono poter aggiungere team member a un team. | Alta |
| RF_AM_PM_15 | L'admin e il project manager devono poter rimuovere team member da un team. | Alta |

2.1.4 Gestione dei commenti

| ID | Descrizione | Priorità |
|---------------|---|----------|
| RF_AM_PM_TM_4 | L'admin, il project manager e il team member devono poter visualizzare tutti i commenti associati ad un progetto specifico. | Alta |
| RF_AM_PM_TM_5 | L'admin, il project manager e il team member devono poter aggiungere commenti a un progetto specifico. | Alta |
| RF_AM_PM_TM_6 | L'admin, il project manager e il team member devono poter modificare i propri commenti. | Alta |
| RF_AM_PM_TM_7 | L'admin deve poter eliminare qualsiasi commento all'interno di un progetto con lo scopo di moderare la sezione commenti, mentre il project manager e il team member possono eliminare solo i propri commenti. | Alta |

2.1.5 Gestione degli utenti

| ID | Descrizione | Priorità |
|---------------|---|----------|
| RF_AM_6 | L'admin deve poter visualizzare tutti gli utenti presenti nel sistema, con la possibilità di applicare filtri di ricerca. | Alta |
| RF_AM_7 | L'admin deve poter visualizzare i project manager disponibili, ossia quelli con meno di cinque progetti attivi, con la possibilità di applicare filtri di ricerca. | Alta |
| RF_AM_PM_16 | L'admin e il project manager devono potere visualizzare i team member disponibili, ossia quelli che non sono assegnati a team con progetti attivi, con la possibilità di applicare filtri di ricerca. | Alta |
| RF_AM_PM_TM_8 | L'admin, il project manager e il team member devono poter visualizzare i team member di un team specifico con la possibilità di applicare filtri di ricerca. | Alta |



UNIVERSITÀ DEGLI STUDI
DI SALERNO

2.1.6 Gestione dell'autenticazione

| ID | Descrizione | Priorità |
|-----------------------|--|----------|
| RF_AM_8 | L'admin deve poter registrare nuovi utenti nel sistema, fornendo le informazioni necessarie per la creazione dell'account. | Alta |
| RF_AM_9 | L'admin deve poter disattivare un utente, impedendogli l'accesso al sistema. | Alta |
| RF_AM_10 | L'admin deve poter riattivare un utente disattivato, consentendogli di accedere nuovamente al sistema. | Alta |
| RF_AM_PM_TM_9 | L'admin, il project manager e il team member devono poter effettuare il login al sistema, fornendo le credenziali necessarie per l'autenticazione. | Alta |
| RF_AM_PM_TM_10 | L'admin, il project manager e il team member devono poter eseguire il logout dal sistema, terminando la loro sessione attiva. | Alta |



UNIVERSITÀ DEGLI STUDI
DI SALERNO

2.2 Requisiti non funzionali (FURPS+)

2.2.1 Functionality

| ID | Descrizione | Priorità |
|----------|--|----------|
| NFR_FU_1 | Il sistema deve proteggere gli endpoint sensibili garantendo che solo gli utenti con i ruoli appropriati possano accedere alle risorse, riducendo il rischio di accessi non autorizzati, mediante un adeguato controllo degli accessi basato sui ruoli e sulle autorizzazioni. | Alta |

2.2.2 Usability

| ID | Descrizione | Priorità |
|----------|--|----------|
| NFR_US_1 | Il sistema deve garantire che tutte le risposte siano uniformi e ben strutturate, migliorando l'esperienza d'uso e facilitando la gestione degli errori e l'integrazione con i client. | Alta |
| NFR_US_2 | Il sistema deve garantire che gli endpoint siano progettati in modo chiaro e coerente, favorendo una comprensione intuitiva dell'API. | Alta |

2.2.3 Reliability

| ID | Descrizione | Priorità |
|----------|---|----------|
| NFR_RL_1 | Il sistema deve garantire che i dati in ingresso siano validati correttamente, assicurando che vengano ricevuti solo dati conformi. | Alta |
| NFR_RL_2 | Il sistema deve includere chiamate a logger in punti critici per tracciare le operazioni, semplificando il debugging e aumentando la trasparenza per la manutenzione e il monitoraggio del sistema. | Alta |

2.2.4 Performance

| ID | Descrizione | Priorità |
|----------|---|----------|
| NFR_PF_1 | Il sistema deve gestire grandi volumi di dati in modo efficiente, riducendo il carico sul sistema e migliorando i tempi di risposta, rendendo l'API scalabile per dataset di grandi dimensioni. | Alta |
| NFR_PF_2 | Il sistema deve adottare un'architettura stateless per garantire che ogni richiesta contenga tutte le informazioni necessarie, riducendo l'overhead e facilitando la scalabilità orizzontale, migliorando così le prestazioni sotto carico elevato. | Alta |



UNIVERSITÀ DEGLI STUDI
DI SALERNO

2.2.5 Supportability

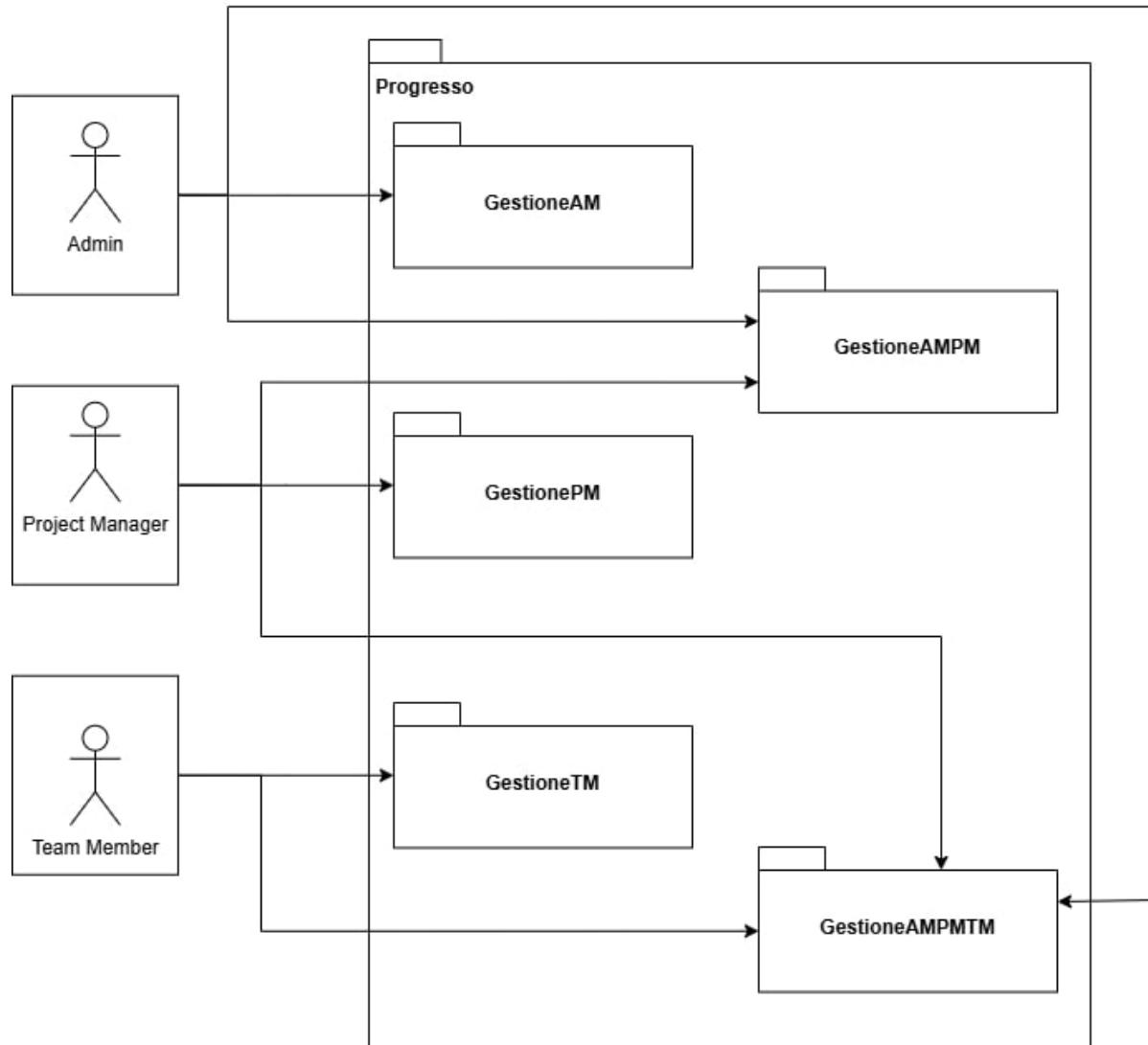
| ID | Descrizione | Priorità |
|----------|--|----------|
| NFR_SU_1 | Il sistema deve adottare un'architettura modulare che separi la logica di business dalla gestione delle richieste, migliorando la manutenibilità e la flessibilità. | Alta |
| NFR_SU_2 | Il sistema deve adottare convenzioni REST standardizzate e utilizzare oggetti dedicati per il trasferimento dei dati, garantendo interoperabilità con altri sistemi, semplificando l'integrazione con servizi esterni. | Alta |
| NFR_SU_3 | Il sistema deve supportare un deployment flessibile su diverse infrastrutture, sia cloud che on-premise, garantendo scalabilità ed efficienza negli ambienti di produzione. | Alta |



2.3 Modello dei casi d'uso

2.3.1 Diagramma dei casi d'uso

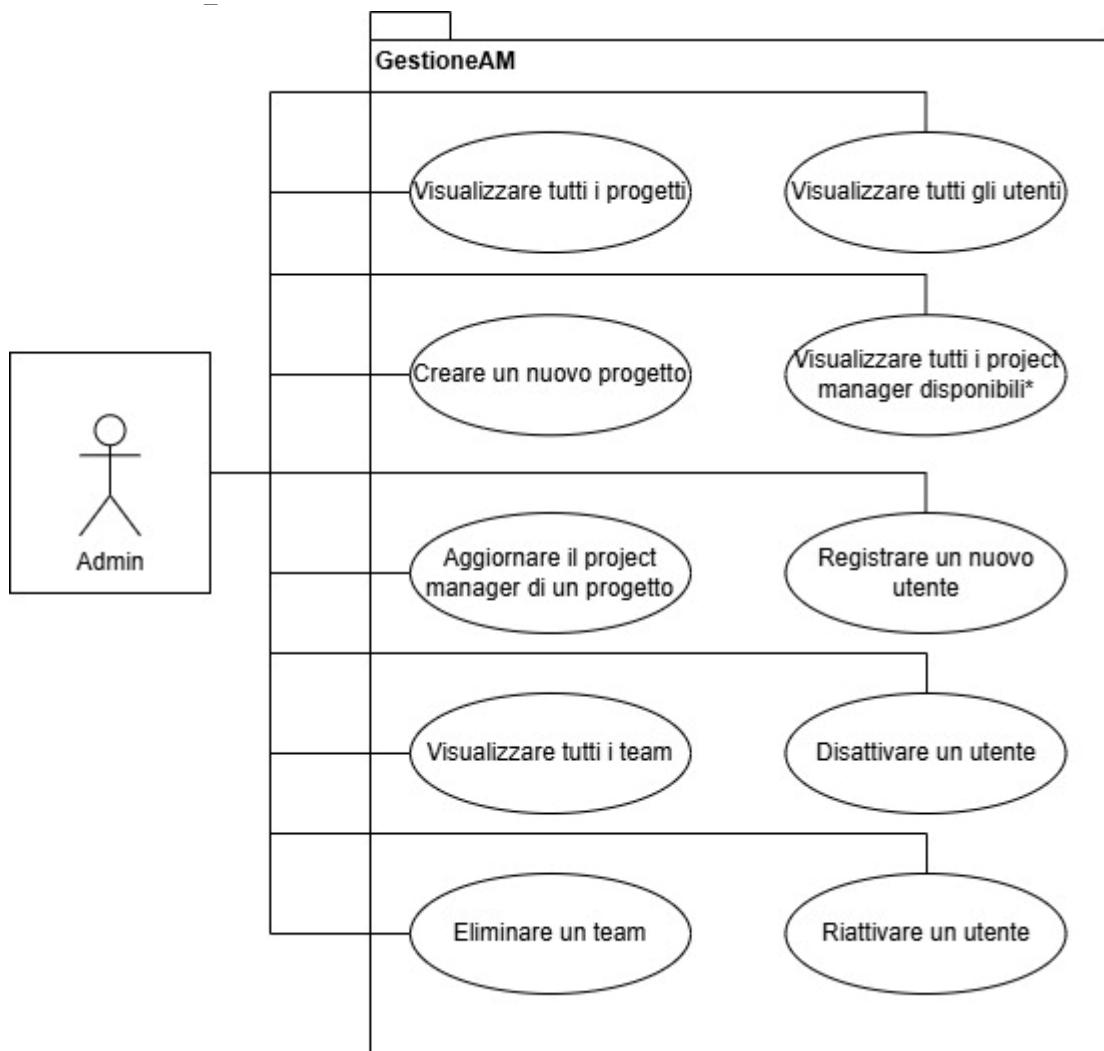
2.3.1.1 UCD_Progresso





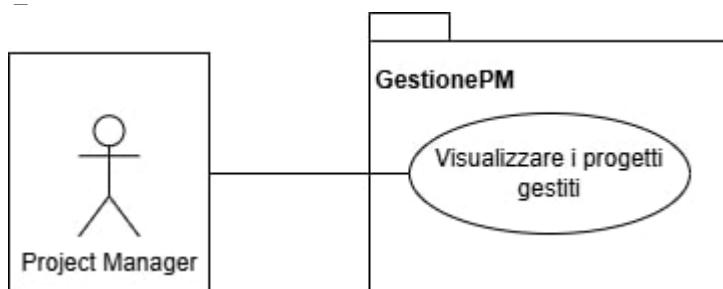
UNIVERSITÀ DEGLI STUDI
DI SALERNO

2.3.1.2 UCD_GestioneAM



*i project manager che gestiscono meno di cinque progetti attivi

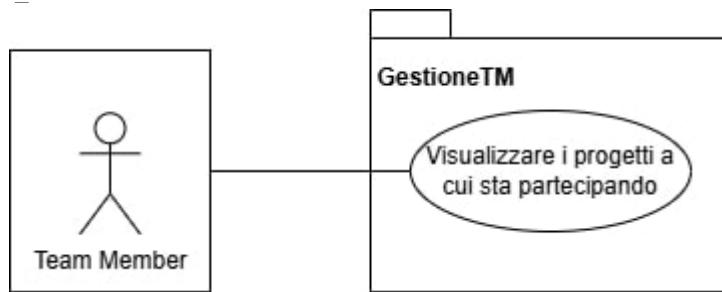
2.3.1.3 UCD_GestionePM



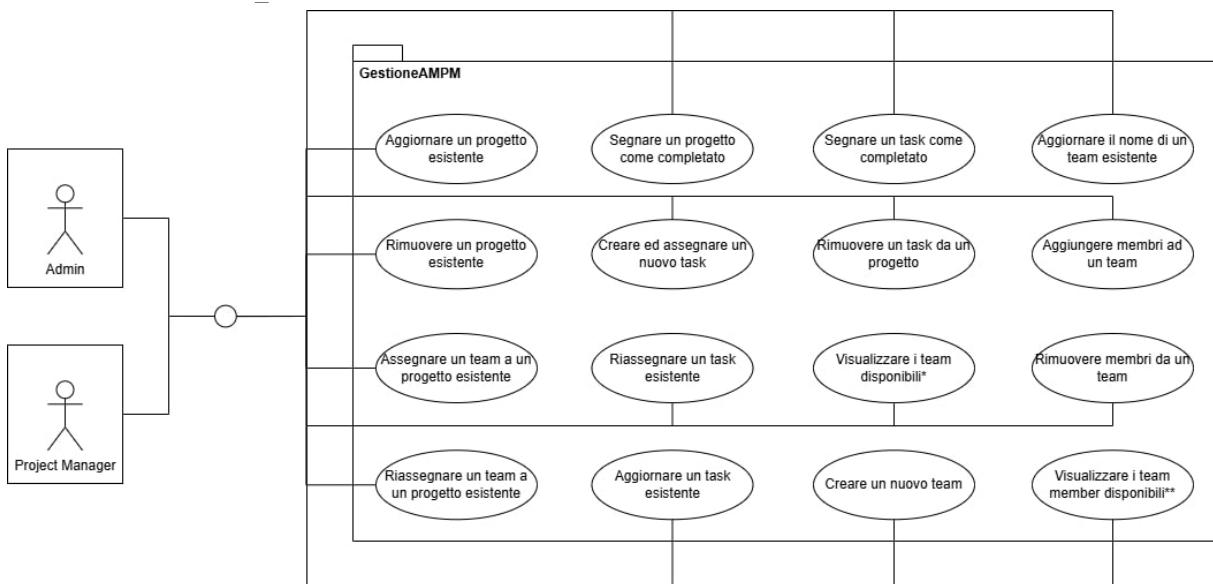


UNIVERSITÀ DEGLI STUDI
DI SALERNO

2.3.1.4 UCD_GestioneTM



2.3.1.5 UCD_GestioneAMPM

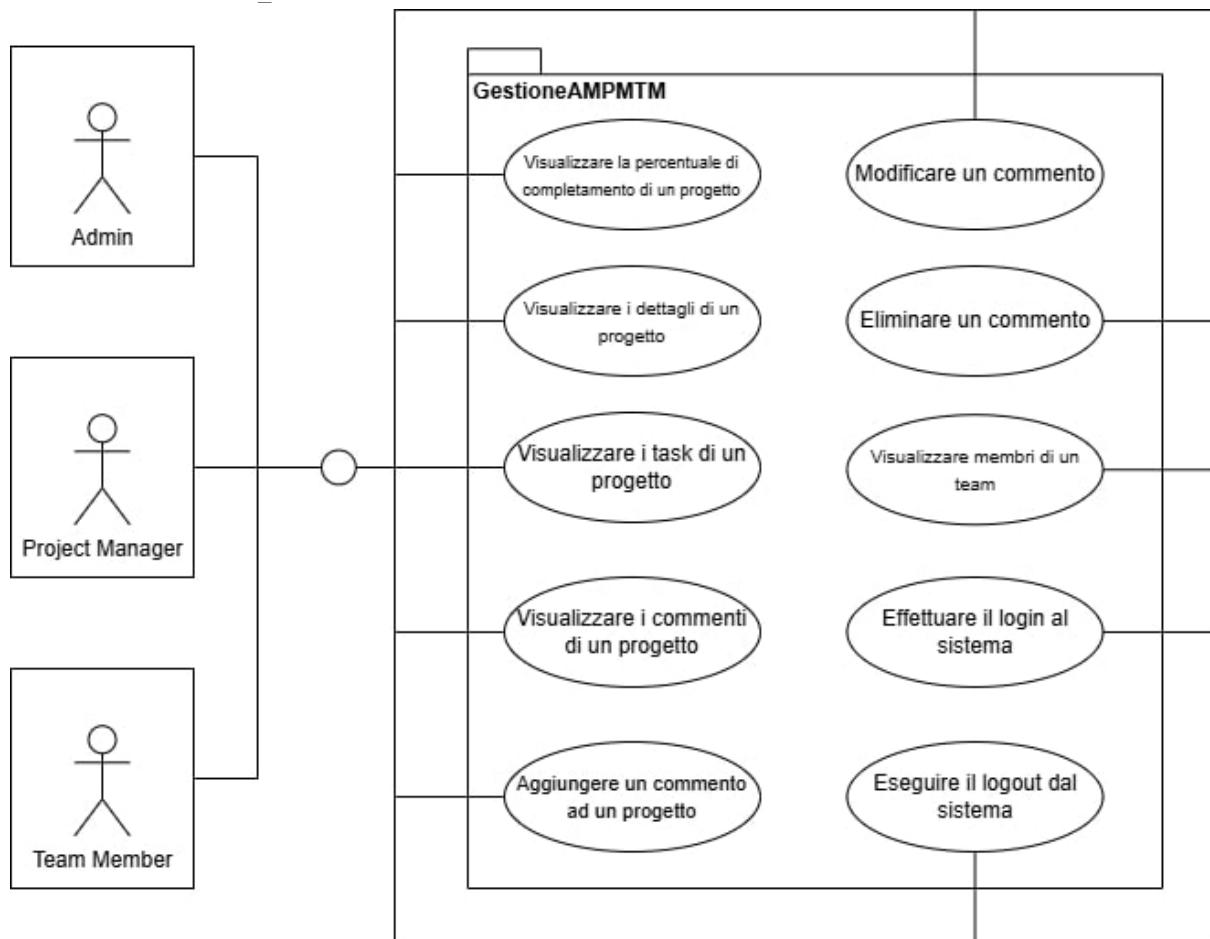


*i team che non stanno lavorando su progetti attivi

**i team member che non appartengono a nessun team



2.3.1.6 UCD_GestioneAMPMTM





2.3.2 Descrizione dei principali casi d'uso

2.3.2.1 UC_GestioneAM_1 - Creare un nuovo progetto

| | |
|--|--|
| Identificativo: UC_GestioneAM_1 | |
| Descrizione | <i>Questo UC illustra il processo attraverso il quale un admin effettua la creazione di un progetto.</i> |
| Attore principale | Admin L'admin può effettuare la creazione di un progetto. |
| Attori secondari | Nessuno |
| Entry condition | ❖ L'admin è autenticato. ❖ L'admin si trova nella homepage della piattaforma. |
| Exit condition | Il progetto viene creato e reso disponibile sulla piattaforma. |
| On success | |
| Exit condition | Il progetto non viene creato e l'operazione termina con errore. |
| On failure | |
| Rilevanza/User priority | Alta |
| Frequenza stimata | 3 usi/giorno |
| Extension point | Nessuno |
| Generalization of | Nessuno |
| FLUSSO DI EVENTI PRINCIPALE/MAIN SCENARIO | |
| 1 | Admin: L'admin utilizza il comando "Create". |
| 2 | Sistema: Il sistema mostra una finestra di dialogo contenente uno stepper. Il primo step, chiamato "Project Info", contiene un form con i seguenti campi: ❖ Name: stringa di caratteri alfanumerici. ❖ Description: stringa di caratteri alfanumerici. ❖ Start Date <ul style="list-style-type: none">○ YYYY: una lista di valori per selezionare l'anno.○ MM: una lista di valori per selezionare il mese.○ DD: una lista di valori per selezionare il giorno. ❖ Due Date <ul style="list-style-type: none">○ YYYY: una lista di valori per selezionare l'anno.○ MM: una lista di valori per selezionare il mese.○ DD: una lista di valori per selezionare il giorno. |
| 3 | Admin: L'admin compila il form. |
| 4 | Sistema: Il sistema controlla i dati inseriti: ❖ Controlla che il campo "Name" sia valido. ❖ Controlla che il campo "Description" sia valido. ❖ Controlla che il campo "Start Date" sia valido. ❖ Controlla che il campo "Due Date" sia valido. |
| 5 | Admin: L'admin utilizza il comando "Next", il quale è stato reso disponibile dopo la validazione del form, per spostarsi al prossimo step. |



| | | |
|---|-----------------|--|
| 6 | Sistema: | Il sistema mostra il secondo step, chiamato “Assign Project Manager”, contenente una tabella che mostra tutti i project manager disponibili, ossia quelli che gestiscono meno di cinque progetti attivi. |
| 7 | Admin: | L’admin seleziona il project manager desiderato cliccando sulla riga della tabella corrispondente. |
| 8 | Sistema: | Il sistema verifica se è stata fatta una selezione. |
| 9 | Admin: | L’admin utilizza il comando “Next”, il quale è stato reso disponibile dopo la selezione del project manager, per spostarsi al prossimo step. |
| 10 | Sistema: | Il sistema mostra il terzo step, chiamato “Assign Team”, contenente tre pulsanti, ciascuno corrispondente alle opzioni disponibili per l’admin nell’assegnazione del team: <ul style="list-style-type: none">❖ Assign Existing Team: per selezionare un team esistente.❖ Create New Team: per creare un nuovo team.❖ Assign Team Later: per rimandare l’assegnazione del team a un momento successivo. |
| 11 | Admin: | L’admin utilizza il comando “Assign Team Later”. |
| 12 | Sistema: | Il sistema salva il progetto e mostra un toast contenente un messaggio di successo. Infine, rende disponibile il nuovo progetto sulla piattaforma. |
| I Scenario/Flusso di eventi alternativo: L’admin non ha compilato correttamente il form del primo step | | |
| 4.1 | Sistema: | Il sistema mostra un messaggio di errore specifico per ogni campo non valido. |
| 4.2 | Sistema: | Il sistema attende che il form sia valido. |
| I Scenario/Flusso di eventi di errore: Il sistema non riesce a salvare il progetto | | |
| 11.1 | Sistema: | Il sistema visualizza un messaggio di errore ed invita l’admin a riprovare in un secondo momento. |
| 11.2 | Sistema: | Il sistema termina l’operazione con insuccesso. |



| Identificativo: UC_GestioneAMPM_1 | |
|--|--|
| Descrizione | <i>Questo UC illustra il processo attraverso il quale un admin può creare un nuovo team per un progetto. Questo caso d'uso è un'estensione di UC_GestioneAM_1 durante lo step di assegnazione del team.</i> |
| Attore principale | Admin L'admin può effettuare la creazione di un team. |
| Attori secondari | Nessuno |
| Entry condition | ❖ L'admin è autenticato. ❖ L'admin si trova nel flusso di creazione di un progetto. |
| Exit condition | Il team viene creato e assegnato al progetto. |
| On success | |
| Exit condition | Il team non viene creato e l'operazione termina con errore. |
| On failure | |
| Rilevanza/User priority | Alta |
| Frequenza stimata | 2 usi/giorno |
| Extension point | UC_GestioneAM_1, passo 10 |
| Generalization of | Nessuno |
| FLUSSO DI EVENTI PRINCIPALE/MAIN SCENARIO | |
| 1 | Admin: L'admin utilizza il comando “Create New Team”. |
| 2 | Sistema: Il sistema mostra una finestra di dialogo contenente uno stepper. Il primo step, chiamato “Enter Team Name”, contiene un form con i seguenti campi: ❖ Name: stringa di caratteri alfanumerici. |
| 3 | Admin: L'admin compila il form. |
| 4 | Sistema: Il sistema controlla i dati inseriti: ❖ Controlla che il campo “Name” sia valido. |
| 5 | Admin: L'admin utilizza il comando “Next”, il quale è stato reso disponibile dopo la validazione del form, per spostarsi al prossimo step. |
| 6 | Sistema: Il sistema mostra il secondo step, chiamato “Select Team Members”, contenente una tabella che mostra tutti i team member disponibili, ossia quelli che non appartengono a nessun team. |
| 7 | Admin: L'admin seleziona i team member desiderati cliccando sulle righe della tabella corrispondenti. |
| 8 | Sistema: Il sistema verifica se è stata fatta una selezione. |
| 9 | Admin: L'admin utilizza il comando “Complete”, il quale è stato reso disponibile dopo la selezione dei team member, per creare il team. |
| 10 | Sistema: Il sistema salva il progetto, salva e assegna il team, e mostra tre toast: uno per l'avvenuta creazione del progetto, uno per la creazione del team e uno per l'assegnazione del team al |



UNIVERSITÀ DEGLI STUDI
DI SALERNO

| | | |
|---|-----------------|---|
| | | progetto. Infine, il progetto viene reso disponibile sulla piattaforma. |
| I Scenario/Flusso di eventi alternativo: L'admin non ha compilato correttamente il form del primo step | | |
| 4.1 | Sistema: | Il sistema mostra un messaggio di errore specifico per il campo non valido. |
| 4.2 | Sistema: | Il sistema attende che il form sia valido. |
| I Scenario/Flusso di eventi di errore: Il sistema non riesce a salvare il team | | |
| 9.1 | Sistema: | Il sistema visualizza un messaggio di errore ed invita l'admin a riprovare in un secondo momento. |
| 9.2 | Sistema: | Il sistema termina l'operazione con insuccesso. |



| | |
|--|--|
| Identificativo: UC_GestioneAMPM_2 | |
| Descrizione | <i>Questo UC illustra il processo attraverso il quale un project manager può creare ed assegnare un nuovo task.</i> |
| Attore principale | Project Manager Il project manager può effettuare la creazione e assegnazione di un task. |
| Attori secondari | Nessuno |
| Entry condition | <ul style="list-style-type: none">❖ Il project manager è autenticato.❖ Il project manager si trova nella pagina del progetto in cui desidera creare il task.❖ Il progetto selezionato deve avere un team assegnato. |
| Exit condition On success | Il task viene creato e assegnato. |
| Exit condition On failure | Il task non viene creato e l'operazione termina con errore. |
| Rilevanza/User priority | Alta |
| Frequenza stimata | 5 usi/giorno |
| Extension point | Nessuno |
| Generalization of | Nessuno |
| FLUSSO DI EVENTI PRINCIPALE/MAIN SCENARIO | |
| 1 | Project Manager: Il project manager utilizza il comando “Add Task”. |
| 2 | Sistema: Il sistema mostra una finestra di dialogo contenente uno stepper. Il primo step, chiamato “Task Info”, contiene un form con i seguenti campi: <ul style="list-style-type: none">❖ Name: stringa di caratteri alfanumerici.❖ Description: stringa di caratteri alfanumerici.❖ Priority: una lista composta da tre opzioni: “High”, “Medium”, “Low”.❖ Start Date<ul style="list-style-type: none">○ YYYY: una lista di valori per selezionare l’anno.○ MM: una lista di valori per selezionare il mese.○ DD: una lista di valori per selezionare il giorno.❖ Due Date<ul style="list-style-type: none">○ YYYY: una lista di valori per selezionare l’anno.○ MM: una lista di valori per selezionare il mese.○ DD: una lista di valori per selezionare il giorno. |
| 3 | Project Manager: Il project manager compila il form. |
| 4 | Sistema: Il sistema controlla i dati inseriti: <ul style="list-style-type: none">❖ Controlla che il campo “Name” sia valido.❖ Controlla che il campo “Description” sia valido.❖ Controlla che il campo “Priority” sia valido.❖ Controlla che il campo “Start Date” sia valido.❖ Controlla che il campo “Due Date” sia valido. |



| | | |
|--|-------------------------|--|
| 5 | Project Manager: | Il project manager utilizza il comando “Next”, il quale è stato reso disponibile dopo la validazione del form, per spostarsi al prossimo step. |
| 6 | Sistema: | Il sistema mostra il secondo step, chiamato “Assign User”, contenente una tabella che mostra tutti i team member del team assegnato al progetto. |
| 7 | Project Manager: | Il project manager seleziona il team member desiderato cliccando sulla riga della tabella corrispondente. |
| 8 | Sistema: | Il sistema verifica se è stata fatta una selezione. |
| 9 | Project manager: | Il project manager utilizza il comando “Complete”, il quale è stato reso disponibile dopo la selezione del team member, per creare ed assegnare il task. |
| 10 | Sistema | Il sistema salva il task e mostra un toast contenente un messaggio di successo. Infine, rende disponibile il nuovo task sulla piattaforma. |
| I Scenario/Flusso di eventi alternativo: Il project manager non ha compilato correttamente il form del primo step | | |
| 4.1 | Sistema: | Il sistema mostra un messaggio di errore specifico per i campi non validi. |
| 4.2 | Sistema: | Il sistema attende che il form sia valido. |
| I Scenario/Flusso di eventi di errore: Il sistema non riesce a salvare il task | | |
| 9.1 | Sistema: | Il sistema visualizza un messaggio di errore ed invita il project manager a riprovare in un secondo momento. |
| 9.2 | Sistema: | Il sistema termina l’operazione con insuccesso. |



| Identificativo: UC_GestioneAMPMTM_1 | |
|---|---|
| Descrizione | <i>Questo UC illustra il processo attraverso il quale un team member può aggiungere un commento ad un progetto.</i> |
| Attore principale | Team Member Il team member può aggiungere un commento ad un progetto. |
| Attori secondari | Nessuno |
| Entry condition | <ul style="list-style-type: none">❖ Il team member è autenticato.❖ Il team member si trova nella pagina del progetto in cui desidera aggiungere il commento.❖ Il progetto selezionato deve essere attivo. |
| Exit condition On success | Il commento viene aggiunto. |
| Exit condition On failure | Il commento non viene aggiunto e l'operazione termina con errore. |
| Rilevanza/User priority | Alta |
| Frequenza stimata | 3 usi/giorno |
| Extension point | Nessuno |
| Generalization of | Nessuno |
| FLUSSO DI EVENTI PRINCIPALE/MAIN SCENARIO | |
| 1 | Sistema: |
| | Il sistema mostra un form contenente la casella di testo per l'inserimento del commento. |
| 2 | Team Member: |
| | Il team member inserisce il commento nella casella di testo. |
| 3 | Sistema: |
| | Il sistema verifica che il commento sia valido. |
| 4 | Team Member: |
| | Il team member utilizza l'apposito pulsante per inviare il commento, il quale è stato reso disponibile dopo la validazione del form. |
| 5 | Sistema: |
| | Il sistema salva il commento e lo aggiunge alla sezione dei commenti del progetto. |
| I Scenario/Flusso di eventi alternativo: Il team member non ha compilato correttamente il form | |
| 3.1 | Sistema: |
| | Il sistema mostra un messaggio di errore specifico per il campo non valido. |
| 3.2 | Sistema: |
| | Il sistema attende che il form sia valido. |
| I Scenario/Flusso di eventi di errore: Il sistema non riesce a salvare il commento | |
| 4.1 | Sistema: |
| | Il sistema visualizza un messaggio di errore ed invita il team member a riprovare in un secondo momento. |
| 4.2 | Sistema: |
| | Il sistema termina l'operazione con insuccesso. |



2.4 Scelte tecnologiche

2.4.1 Back-end: *Spring Boot, Maven, JPA/Hibernate*

- **Spring Boot:** *Spring Boot* è stato scelto per la sua resilienza e adattabilità nello sviluppo di applicazioni web. Con le sue caratteristiche di auto-configurazione, *Spring Boot* renderà lo sviluppo più semplice, eliminando il codice ripetitivo e accelerando le procedure di installazione.
- **Maven:** *Maven* è un tool progettato per automatizzare il processo di build, gestire le dipendenze e assicurare coerenza in un progetto. Aiuta ad uniformare la compilazione e a gestire l'integrazione delle librerie, garantendo così un'efficace gestione delle dipendenze.
- **JPA/Hibernate:** sfruttando l'*Object-Relational Mapping (ORM)*, *JPA/Hibernate* è stato selezionato come soluzione per la persistenza dei dati. Con questo approccio, lo sviluppo sarà effettuato utilizzando oggetti *Java* di alto livello invece di istruzioni *SQL* di basso livello, riducendo la complessità delle interazioni con il database. Come provider, *JPA/Hibernate* fornirà funzionalità avanzate come *caching, lazy loading* e *auto schema generation*, che renderanno la gestione dei dati più efficiente e manutenibile.

Pertanto, l'integrazione di *Spring Boot, Maven* e *JPA/Hibernate* porterà ad un architettura back-end scalabile, sostenibile e stabile. Questa combinazione tecnologica robusta non solo accelererà lo sviluppo, ma assurerà anche che il sistema sarà in grado di gestire carichi crescenti, preparandolo per futuri miglioramenti.

2.4.2 Front-end: *Angular*

Basandosi sulla potenza del framework nella produzione di interfacce utente dinamiche, reattive e scalabili, si è deciso di costruire il front-end utilizzando *Angular*. Il framework open-source codificato utilizzando *TypeScript* e mantenuto da Google trova ampio uso per progettare applicazioni a pagina singola (*SPA*). Di seguito sono riportate alcune caratteristiche importanti di *Angular*:

- **Component-based architecture:** *Angular* si basa sul meccanismo di *component-based architecture*, che consente agli sviluppatori di implementare componenti di interfaccia utente autonomi e riutilizzabili. Ogni componente si occupa del proprio *HTML, CSS* e *TypeScript*, promuovendo così la modularità e la manutenibilità.
- **Two-Way Data Binding:** questa funzione assicura che qualsiasi modifica apportata all'interfaccia utente si riflette immediatamente nel modello di dati del componente e viceversa. Ciò si ottiene sincronizzando automaticamente la vista e il modello del componente garantendo un aggiornamento istantaneo dell'interfaccia utente.
- **Dependency Injection:** la dipendenza dei componenti dai servizi è gestita correttamente dal meccanismo di *dependency-injection* integrato in *Angular*. Grazie alla modularità promossa da questo framework, la base di codice diventa più facile da collaudare.



UNIVERSITÀ DEGLI STUDI
DI SALERNO

- **Directives:** le *directives* sono proprietà uniche sugli elementi che forniscono indicazioni al compilatore di *Angular* su come gestire specifici elementi o persino su come trasformarli insieme ai loro elementi figlio. Grazie a questa caratteristica, *Angular* arricchisce le capacità dell'*HTML* e potenzia la flessibilità e la dinamicità dell'interfaccia utente.
- **Routing:** con il supporto per definire molteplici viste e cambiare tra di loro facilmente, il *router* nativo di *Angular* rende semplice lo sviluppo delle applicazioni a pagina singola (*SPA*). Grazie a questa funzionalità, si ottiene un'interfaccia reattiva che migliora l'esperienza dell'utente.
- **Angular Command Line Interface (Angular CLI):** l'*Angular Command Line Interface (Angular CLI)* semplifica il processo di sviluppo fornendo risorse applicative, di sviluppo e test. Questo strumento accelera il lavoro, favorisce l'adozione delle best practices e automatizza le attività ripetitive.

Sfruttando le molteplici funzionalità di *Angular*, si garantirà un'esperienza utente completa che sia veloce, intuitiva e in grado di adattarsi alle continue richieste dei partner esterni e degli utenti finali.



3. Progettazione del Sistema

3.1 Architettura generale del sistema

3.1.1 Separazione di back-end e front-end

Il back-end e il front-end sono i due principali livelli che formeranno l'architettura del sistema. Questa divisione esplicita ha lo scopo di rendere il sistema più scalabile e di accelerare il processo di sviluppo.

- **Livello front-end:** il livello front-end sarà realizzato con *Angular* per gestire il rendering e migliorare l'interfaccia utente. Sarà responsabile di offrire un'interfaccia utente dinamica e reattiva, utilizzando il *two-way data binding* e la progettazione basata su componenti per gestire le interazioni con il client. Questo strato si concentrerà esclusivamente sull'esperienza utente e il rendering, senza coinvolgere la logica di business.
- **Comunicazione tramite API REST:** le *API REST* forniranno l'unico mezzo di comunicazione tra il back-end e l'app *Angular*. Assicurandosi che ogni richiesta proveniente dal front-end abbia tutti i dati necessari per l'elaborazione, questo paradigma di comunicazione stateless scollega l'interfaccia utente dai servizi sottostanti. Di conseguenza, il front-end rimarrà flessibile e potrà cambiare in maniera indipendente senza richiedere modifiche significative nel modo in cui accederà ai dati.
- **Livello back-end:** il livello back-end verrà implementato utilizzando *Spring Boot* e sarà incaricato della persistenza dei dati e della gestione centralizzata della logica di business. Il sistema offrirà una solida piattaforma per gestire le richieste degli utenti e comunicare con il database tramite un *Object Relational Mapping (ORM)*, imponendo standard di sicurezza incapsulando tutte le attività di trattamento e gestione dei dati all'interno di questo livello.
- **Integrazione ed estensibilità:** le *API REST* vengono utilizzate per integrare il front-end *Angular* e il back-end *Spring Boot*. Grazie alla strategia di integrazione modulare, è possibile espandere il back-end senza compromettere il front-end. La struttura modulare del sistema permette la scalabilità consentendo l'aggiunta di nuove funzionalità in modo indipendente e garantire che il sistema possa crescere senza interruzioni o degrado del servizio.

Questa divisione architetturale garantirà che il back-end gestisca i dati in modo affidabile, mentre il front-end rimarrà interamente focalizzato sull'esperienza utente. La comunicazione *REST* senza stato tra i due livelli garantirà efficienza e manutenibilità nel software, facilitando l'espansione e la scalabilità del sistema in futuro.

3.1.2 Comunicazione tra front-end e back-end

- **Richieste HTTP tramite API REST:** *Angular* utilizza il modulo *HttpClient* per gestire le comunicazioni *HTTP* con il back-end. Nei progetti che combinano *Angular* e *Spring Boot*, quest'ultimo fornisce endpoint *REST* che possono essere consumati dal front-end per scambiare dati in modo efficiente e asincrono. Ogni volta che un utente interagisce con l'interfaccia utente (ad esempio, compila un form, clicca un pulsante per accedere



ai dettagli dei progetti, ecc.), Angular invia una chiamata *HTTP* all'*API* appropriata. Dati, parametri e intestazioni (come i token di autenticazione) possono essere inclusi nel payload *JSON*. Dopo aver eseguito le *query*, il back-end fornisce al front-end una risposta *JSON*, che *Angular* utilizza per aggiornare l'interfaccia utente.

- **Autenticazione e autorizzazione:** importanti misure di sicurezza che proteggono la comunicazione tra front-end e back-end sono l'autenticazione e l'autorizzazione. È possibile scegliere tra l'autenticazione basata su sessione o l'utilizzo di *JSON Web Token (JWT)* per garantire la sicurezza. Il back-end fornisce un token o ID di sessione durante la procedura di login dell'utente, che viene quindi salvato da *Angular* (solitamente nei *cookies* o nel *localStorage*). Questo token viene incluso in ogni successiva richiesta *HTTP* ed è verificato da *Spring Security*², ad esempio, utilizzando annotazioni di autorizzazione a livello di metodo, come `@PreAuthorize`³. Questa procedura garantisce che solo gli utenti autenticati e autorizzati possano accedere agli endpoint *API* sensibili, aggiungendo un ulteriore livello di sicurezza del sistema.

3.1.3 Comunicazione tra back-end e database tramite *JPA/Hibernate*

- **Astrazione della persistenza dei dati:** *JPA/Hibernate* traduce le operazioni sugli oggetti *Java* in istruzioni *SQL*, semplificando notevolmente il processo di interazione con il database. Questa astrazione riduce la complessità del codice e facilita la gestione delle operazioni di *CRUD (Create, Read, Update, Delete)*.
- **Funzionalità avanzate:** il provider *Hibernate* offre funzionalità avanzate come il *caching*, che riduce il tempo di accessi ripetitivi al database; il *lazy loading*, che consente di archiviare i dati solo quando è assolutamente necessario; l'*auto schema generation*, che semplifica l'aggiornamento della struttura del database nel corso del ciclo di vita dell'applicazione.
- **Efficienza e manutenibilità:** l'utilizzo di questi meccanismi consente di ottimizzare l'efficienza nell'accesso e nella gestione dei dati, garantendo al contempo una maggiore manutenibilità del codice. La capacità di operare a livello di oggetti riduce l'impatto degli errori tipici delle operazioni *SQL*, contribuendo a rendere il sistema più robusto e affidabile.

3.1.4 Panoramica completa delle interazioni tra back-end e front-end

- **Ciclo di vita delle richieste:** ogni volta che un'azione viene eseguita dall'utente sul front-end di *Angular*, viene inviata una richiesta *HTTP* da *HttpClient* ad un endpoint *REST API* specifico. Questa richiesta include le informazioni necessarie (in formato *JSON*) e i token di autenticazione per garantire la sicurezza. Sul back-end, *Spring Boot* riceve e gestisce questa richiesta, passandola al relativo metodo del *controller*. Il

² *Spring Security* è un framework di sicurezza per applicazioni *Java* basate su *Spring*. Fornisce autorizzazione, autenticazione e protezione contro attacchi comuni. È altamente configurabile e supporta varie forme di autenticazione come *JWT* e *OAuth2*.

³ `@PreAuthorize` è un'annotazione di *Spring Security* che consente di definire regole di autorizzazione a livello di metodo. Utilizza espressioni *SpEL (Spring Expression Language)* per specificare i permessi richiesti prima dell'esecuzione di un metodo, garantendo un controllo granulare degli accessi.



UNIVERSITÀ DEGLI STUDI
DI SALERNO

controller a sua volta delega la logica di business ai *services*, i quali utilizzano *JPA/Hibernate* per interagire con il database, eseguire *query* o salvare i dati. Una volta completata l'elaborazione, *Angular* utilizza l'oggetto *JSON* restituito dal back-end, incluso in una *ResponseEntity*, per aggiornare l'interfaccia utente.

- **Implementazione della sicurezza:** *Spring Security* si collega al back-end per applicare controlli di pre-autorizzazione agli endpoint sensibili. Questo garantisce che soltanto gli utenti con token validi e ruoli adeguati (come ADMIN, PROJECTMANAGER o TEAMMEMBER) siano in grado di eseguire azioni, proteggendo il sistema da accessi non autorizzati.

In conclusione, *Angular* semplifica la comunicazione con un back-end *Spring Boot* gestendo i *binding* dei dati lato client e inviando richieste a un'*API REST* stateless. Il back-end utilizza *JPA/Hibernate* per comunicare con il database in modo efficace. La sicurezza è garantita da procedure di autenticazione e autorizzazione robuste, assicurando una comunicazione affidabile tra i componenti. Il binding dei dati bidirezionale permette ad *Angular* di mantenere la coerenza tra l'interfaccia utente e il modello di dati in tempo reale. Questo approccio integrato non solo migliora la scalabilità e la versatilità del sistema, ma offre anche un'esperienza utente superiore e prestazioni di sistema robuste.



3.2 Decomposizione in sottosistemi

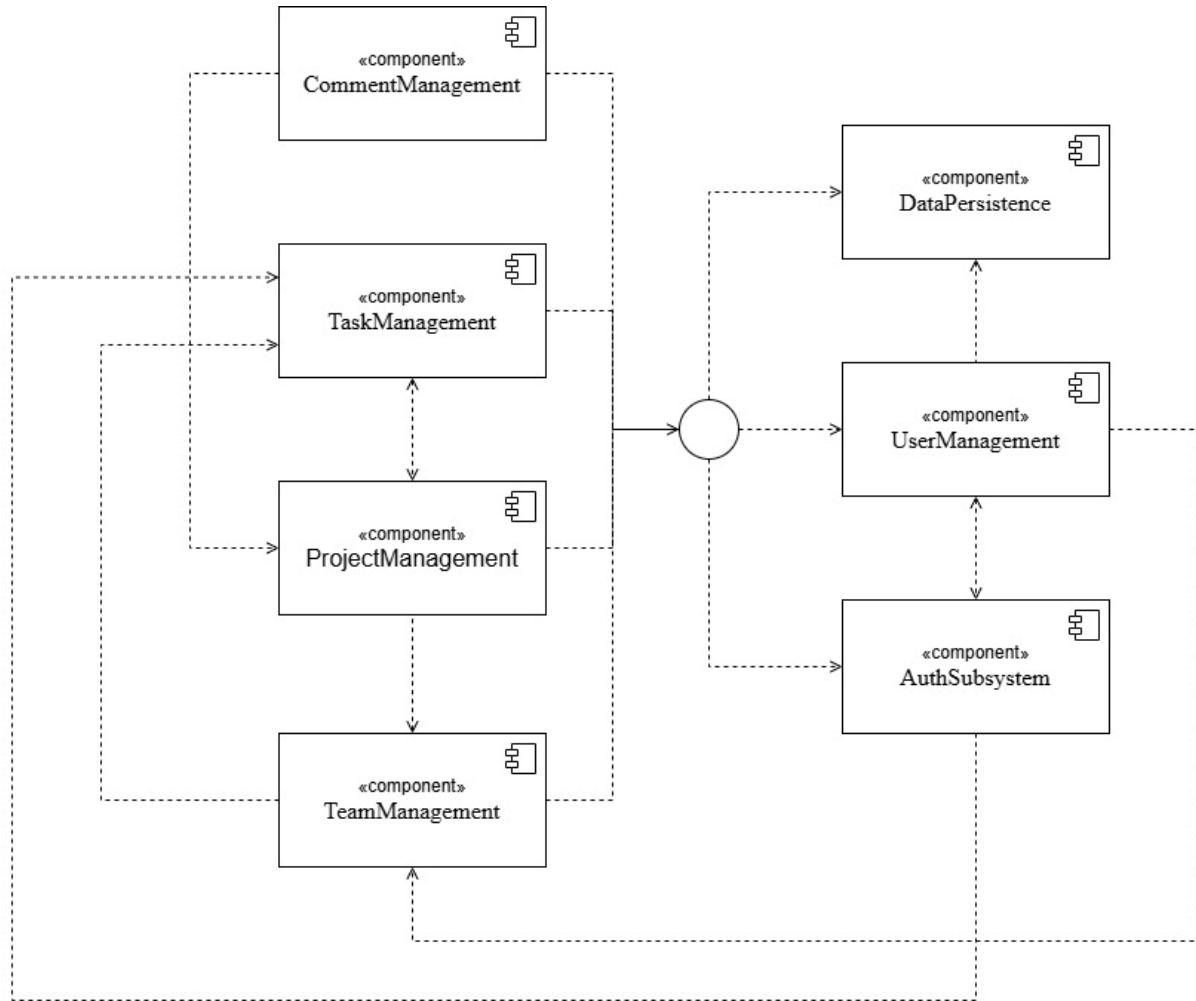
Per riflettere le funzionalità del sistema e l'architettura proposta, il server dedicato al front-end ospita le interfacce grafiche del sistema, mentre il server, destinato al back-end, incorpora tutte le funzionalità suddivise nei livelli di model, service e control.

Il sistema viene suddiviso nei seguenti sottosistemi:

- **ProjectManagement:** questo sottosistema si occupa della creazione, modifica e cancellazione dei progetti all'interno della piattaforma. Permette di associare un team e un project manager ad un progetto, filtrare e visualizzare i progetti disponibili e calcolarne lo stato di avanzamento.
- **TaskManagement:** questo sottosistema gestisce la creazione, assegnazione, modifica e completamento dei task all'interno di un progetto. Permette inoltre di filtrare i task in base a stato e priorità, riassegnarli a diversi team member e rimuoverli se necessario.
- **TeamManagement:** questo sottosistema permette la creazione, modifica e gestione dei team, inclusa l'aggiunta e rimozione di team member. Inoltre, supporta il filtraggio dei team disponibili.
- **CommentManagement:** questo sottosistema consente agli utenti di aggiungere, modificare ed eliminare commenti associati ai progetti.
- **UserManagement:** questo sottosistema fornisce funzionalità per visualizzare informazioni sugli utenti, filtrarli in base a ruolo e stato, e recuperare team member disponibili per progetti e team.
- **AuthSubsystem:** questo sottosistema gestisce l'autenticazione e le autorizzazioni degli utenti, la registrazione, il logout e l'attivazione/disattivazione degli utenti.
- **DataPersistence:** questo sottosistema si occupa della gestione della persistenza dei dati tramite l'utilizzo di un database.



3.2.1 Diagramma dei sottosistemi



3.2.2 Spiegazione delle dipendenze dei sottosistemi

- **CommentManagement:** dipende da ProjectManagement in quanto ogni commento appartiene ad un progetto.
- **TaskManagement:** dipende da ProjectManagement poiché ogni task appartiene ad un progetto.
- **ProjectManagement:**
 - dipende da TeamManagement in quanto ogni progetto può essere assegnato ad un team.
 - dipende da TaskManagement poiché le informazioni sui task contribuiscono alla percentuale di completamento di un progetto.
- **TeamManagement:** dipende da TaskManagement in quanto i task degli utenti devono essere aggiornati quando i membri vengono rimossi da un team.
- **UserManagement:** dipende da TeamManagement per effettuare controlli sullo stato e sull'appartenenza degli utenti ai team.



UNIVERSITÀ DEGLI STUDI
DI SALERNO

➤ **AuthSubsystem:**

- dipende da TaskManagement per aggiornare lo stato dei task assegnati all'utente durante operazioni come la disattivazione di quest'ultimo, in quanto i task in corso devono essere aggiornati e gestiti correttamente.
- dipende da UserManagement perché la gestione dell'autenticazione e dell'autorizzazione richiede informazioni sugli utenti registrati nel sistema.

Tutti i sottosistemi (CommentManagement, TaskManagement, ProjectManagement, TeamManagement) dipendono da:

- **UserManagement:** perché tutti i sottosistemi interagiscono con gli utenti per gestire le loro informazioni e operazioni correlate.
- **DataPersistence:** per garantire la persistenza dei dati attraverso i vari sottosistemi.
- **AuthSubsystem:** per garantire che solo gli utenti autorizzati possano eseguire le operazioni che offrono i vari sottosistemi.

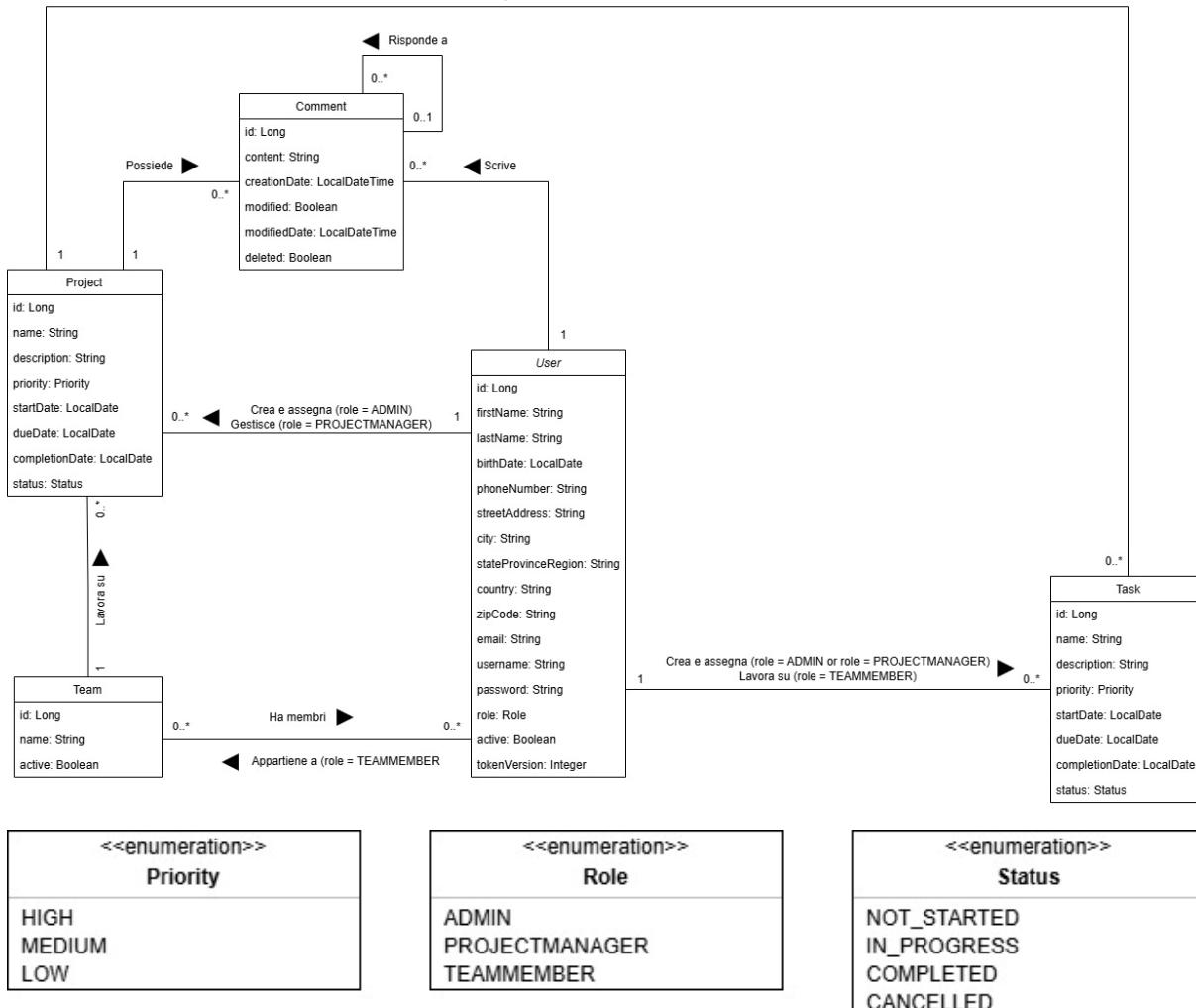


3.3 Gestione dei dati persistenti

3.3.1 Entity class diagram

L'entity class diagram rappresenta in modo chiaro il modello concettuale fondamentale del sistema. Mostra come il database supporti i requisiti funzionali necessari per gestire task, progetti, team, commenti e utenti. In particolare, il diagramma mira a mettere in evidenza le seguenti necessità:

- **Gestione dei progetti:** l'entità *Project* comprende attributi di base come nome, descrizione, le date (inizio, scadenza, completamento) e stato. Include anche riferimenti ad altre entità correlate per facilitare la creazione, l'aggiornamento, il monitoraggio e l'eliminazione dei progetti, assicurando una visione completa dello stato di avanzamento.
- **Gestione dei task:** l'entità *Task* include dati come il nome, la descrizione, la priorità, le date e lo stato. Queste informazioni permettono di svolgere varie operazioni come la creazione, l'aggiunta, l'assegnazione, il completamento e la rimozione. Grazie alla struttura del modello, ogni task è associato a un progetto specifico, garantendo il rispetto dei criteri per il controllo del flusso di lavoro.
- **Gestione dei team:** l'entità *Team* raccoglie le informazioni relative ai gruppi di lavoro. Questo approccio sostiene la gestione dinamica dei team, agevolando la creazione di nuovi gruppi e migliorando la selezione dei team member per adeguarsi alla necessità di collegare risorse e monitorare i progetti internamente.
- **Gestione dei commenti:** l'entità *Comment* permette di tenere traccia e gestire i commenti associati ai progetti.
- **Gestione degli utenti:** l'entità *User* memorizza i dati anagrafici, le credenziali e i ruoli, supportando operazioni di registrazione, visualizzazione e filtraggio. Il modello garantisce la corretta implementazione dei requisiti di *UserManagement* e si integra con il sottosistema *AuthSubsystem* per assicurare l'autenticazione e l'autorizzazione.



Per una descrizione dettagliata delle scelte progettuali relative alle relazioni tra le entità, per visionare lo schema logico e avere una panoramica completa del dizionario dei dati, si rimanda all'**Appendice A**, nella quale sono riportate:

- [*Appendice A.1: Descrizione delle relazioni*](#)
- [*Appendice A.2: Schema logico*](#)
- [*Appendice A.3: Dizionario dei dati*](#)



4. Implementazione

4.1 Sviluppo del back-end

4.1.1 Struttura dei package

I package sono stati strutturati seguendo il design pattern architetturale *Repository-Service-Controller*. Questo modello divide in modo chiaro le responsabilità tra la gestione dei dati persistenti, la logica di business e l'esposizione delle API. Ogni package rappresenta un sottosistema e i loro nomi sono stati scelti in linea con i relativi sottosistemi per garantire un'organizzazione chiara e una manutenzione del codice semplificata. Inoltre, oltre ai package dei sottosistemi, sono stati aggiunti anche i seguenti package:

- **security**: contiene le classi e le configurazioni per la gestione della sicurezza, come la configurazione dei token *JWT*, i filtri per l'autenticazione e le impostazioni *CORS*⁴.
- **validation**: include le annotazioni di validazione personalizzate e i relativi *validator*, che assicurano che i dati in ingresso rispettino i vincoli definiti.
- **dto**: racchiude gli oggetti di trasferimento dati usati per lo scambio di informazioni tra i vari livelli dell'applicazione.
- **entity**: contiene le classi di entità che mappano le tabelle del database e rappresentano la struttura dei dati persistenti.
- **enumeration**: comprende le enumerazioni utilizzate all'interno del sistema.
- **exception**: contiene le eccezioni personalizzate e il gestore globale delle eccezioni, centralizzando la gestione degli errori e migliorando la manutenibilità del codice.

I repository, che si trovano nel package di ogni sottosistema, estendono l'interfaccia *JpaRepository* di *Spring Data JPA*, la quale fornisce metodi predefiniti per la gestione delle operazioni *CRUD (Create, Read, Update, Delete)* sui dati persistenti. Grazie a questa estensione, i repository possono eseguire query complesse senza dover scrivere manualmente la logica *SQL*, semplificando notevolmente la gestione dei dati. Prima di tutto, sarà mostrato il package principale **com.progresso.backend**, che contiene al suo interno tutti gli altri package e, successivamente, verranno illustrate le principali funzionalità dei package basati sui sottosistemi.

Per la visione completa di ogni package, si faccia riferimento all'**Appendice B**, in particolare:

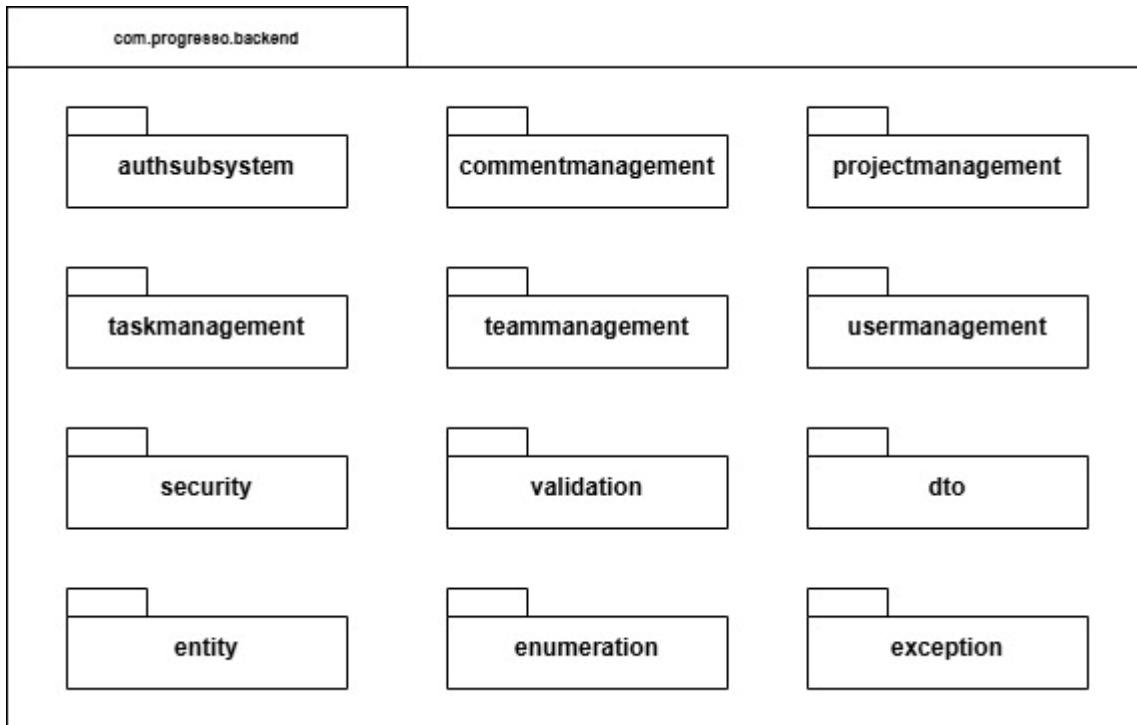
[Appendice B.1: Descrizione approfondita dei package](#)

⁴ *CORS (Cross-Origin Resource Sharing)* è una politica di sicurezza che controlla l'accesso a risorse tra domini diversi, consentendo l'interazione solo se configurata correttamente.

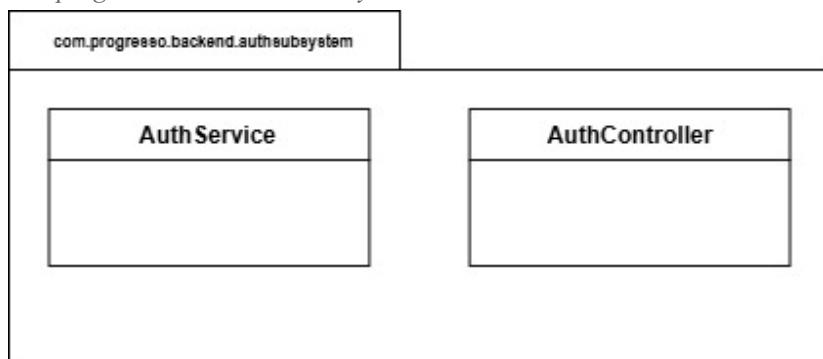


UNIVERSITÀ DEGLI STUDI
DI SALERNO

4.1.1.1 com.progresso.backend



4.1.1.2 com.progresso.backend.authsubsystem



| AuthService | | | |
|------------------|----------------------|---|---|
| Metodo | Tipo di ritorno | Parametri | Descrizione |
| registerUser | UserResponseDto | UserRegistrationDto <i>userRegistrationDto</i> | Registra un nuovo utente verificando l'unicità dell'e-mail, generando username e password, e salvando l'utente. |
| authenticateUser | UserLoginResponseDto | UserLoginDto <i>loginDto</i> | Autentica l'utente verificando le credenziali e, in caso di successo, genera un token JWT. |
| logout | UserResponseDto | | Effettua il logout dell'utente aggiornando il <i>tokenVersion</i> per invalidare i token attivi. |



UNIVERSITÀ DEGLI STUDI
DI SALERNO

| AuthController /api/auth | | | | |
|-----------------------------|-----------|-------------|---|---------------------------|
| Metodo | Endpoint | Http Method | Parametri | Sicurezza / PreAuthorize |
| login | /login | POST | RequestBody: UserLoginDto <i>userLoginDto</i> (validato) | |
| register | /register | POST | RequestBody: UserRegistrationDto <i>userRegistrationDto</i> (validato) | hasAuthority ('ADMIN') |
| logout | /logout | POST | | |

4.1.1.3 com.progresso.backend.commentmanagement

com.progresso.backend.commentmanagement

CommentRepository

CommentService

CommentController

CommentRepository

| Metodo | Tipo di ritorno | Parametri |
|--------------------------|-----------------|---|
| findByIdProjectId | Page<Comment> | Long <i>projectId</i> , Pageable <i>pageable</i> |

CommentService

| Metodo | Tipo di ritorno | Parametri | Descrizione |
|--------------------------|------------------|---|---|
| findByIdProjectId | Page<CommentDto> | Long <i>projectId</i> , Pageable <i>pageable</i> | Recupera in modalità paginata tutti i commenti associati al progetto identificato da <i>projectId</i> , convertendoli in <i>CommentDto</i> . |
| createComment | CommentDto | CommentDto <i>commentDto</i> | Crea un nuovo commento per un progetto: valida l'esistenza e lo stato attivo dell'utente e del progetto, gestisce eventuali risposte (commento padre) e salva il commento, restituendo il relativo DTO. |
| updateComment | CommentDto | Long <i>commentId</i> , String <i>newContent</i> | Aggiorna il contenuto di un commento esistente, impostando il flag di modifica e aggiornando la data di modifica, a condizione che il commento non sia già eliminato e il progetto sia attivo. |
| deleteComment | CommentDto | Long <i>commentId</i> | "Elimina" un commento segnalandolo come cancellato (impostando un |



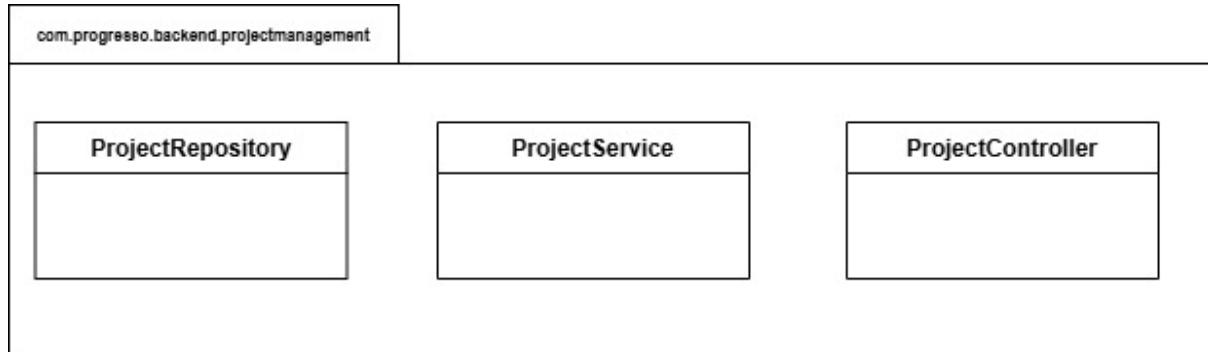
UNIVERSITÀ DEGLI STUDI
DI SALERNO

| | | | |
|--|--|--|---|
| | | | messaggio standard e il flag di eliminazione), se il progetto è attivo e il commento non è già stato eliminato. |
|--|--|--|---|

| CommentController <i>/api/comments</i> | | | | |
|---|-------------------------------|-------------|---|---|
| Metodo | Endpoint | Http Method | Parametri | Sicurezza / PreAuthorize |
| findByIdProjectId | /project/{projectId}/comments | GET | PathVariable: Long <i>projectId</i> Pageable <i>pageable</i> | hasAuthority('ADMIN') or @commentService.isManagerOrMemberOfProject(#projectId, authentication.name) |
| createComment | | POST | RequestBody: CommentDto <i>commentDto</i> (validato) | hasAuthority('ADMIN') or @commentService.isManagerOrMemberOfProject(#commentDto.projectId, authentication.name) |
| updateComment | /{id} | PUT | PathVariable: Long <i>id</i> RequestParam: String <i>newContent</i> (non vuoto, massimo 500 caratteri) | @commentService.isCommentOwner(#id, authentication.name) |
| deleteComment | /{id} | DELETE | PathVariable: Long <i>id</i> | hasAuthority('ADMIN') or @commentService.isCommentOwner(#id, authentication.name) |



4.1.1.4 com.progresso.backend.projectmanagement



| ProjectRepository | | | |
|---|-----------------|---|--|
| Metodo | Tipo di ritorno | Parametri | |
| findAllWithFilters | Page<Project> | Status <i>status</i> , Priority <i>priority</i> , String <i>name</i> , Pageable <i>pageable</i> | |
| findByProjectManagerUsernameAndFilters | Page<Project> | String <i>managerUsername</i> , Status <i>status</i> , Priority <i>priority</i> , String <i>name</i> , Pageable <i>pageable</i> | |
| findActiveProjectsByTeamMemberUsername | Page<Project> | String <i>teamMemberUsername</i> , Pageable <i>pageable</i> | |

| ProjectService | | | |
|---|------------------|---|--|
| Metodo | Tipo di ritorno | Parametri | Descrizione |
| findAllProjectsWithFilters | Page<ProjectDto> | String <i>status</i> , String <i>priority</i> , String <i>name</i> , Pageable <i>pageable</i> | Recupera in modalità paginata i progetti che rispettano i filtri opzionali su <i>status</i> , <i>priority</i> e <i>name</i> , convertendoli in DTO. |
| findProjectsByProjectManagerUsernameAndFilters | Page<ProjectDto> | String <i>managerUsername</i> , String <i>status</i> , String <i>priority</i> , String <i>name</i> , Pageable <i>pageable</i> | Recupera in modalità paginata i progetti gestiti da un project manager specifico, applicando filtri su <i>status</i> , <i>priority</i> e <i>name</i> , convertendoli in DTO. |
| findActiveProjectsByTeamMemberUsername | Page<ProjectDto> | String <i>teamMemberUsername</i> , Pageable <i>pageable</i> | Recupera in modalità paginata i progetti attivi (con status IN_PROGRESS o NOT_STARTED) in cui è coinvolto un team member specificato. |
| createProject | ProjectDto | ProjectDto <i>projectDto</i> | Crea un nuovo progetto dopo aver validato le date (start e due), verificato la validità del project manager (ruolo, stato attivo, numero massimo di progetti attivi) e generato un nome univoco per il progetto. |
| updateProject | ProjectDto | Long <i>projectId</i> , ProjectDto <i>projectDto</i> | Aggiorna un progetto esistente dopo aver eseguito varie validazioni (date, priorità, task, team, commenti) e controllato lo stato del progetto. |



UNIVERSITÀ DEGLI STUDI
DI SALERNO

| | | | |
|----------------------------|------------|---|--|
| assignTeamToProject | ProjectDto | Long <i>projectId</i> , Long <i>teamId</i> | Assegna un team ad un progetto, dopo aver verificato che il progetto non sia già assegnato, che il team sia attivo, che il progetto non sia completato/cancellato e che il team non sia già impegnato in un altro progetto attivo. |
| completeProject | ProjectDto | Long <i>projectId</i> | Completa un progetto verificando che non vi siano task in corso; se tutti i task sono completati, aggiorna lo stato a COMPLETED, imposta la data di completamento e regola la priorità a LOW. |
| removeProject | ProjectDto | Long <i>projectId</i> | "Rimuove" un progetto impostandolo come CANCELLED, rimuovendo le associazioni dei task collegati al progetto e aggiornando il relativo stato. |

| ProjectController <i>api/projects</i> | | | | |
|--|---|-------------|--|--|
| Metodo | Endpoint | Http Method | Parametri | Sicurezza / PreAuthorize |
| getAllProjectsByFilters | | GET | RequestParam (required=false): String <i>status</i> , String <i>priority</i> , String <i>name</i> , Pageable <i>pageable</i> | hasAuthority ('ADMIN') |
| getProjectsByManagerAndFilters | /manager /{managerUsername} | GET | PathVariable: String <i>managerUsername</i> , RequestParam (required=false): String <i>status</i> , String <i>priority</i> , String <i>name</i> Pageable <i>pageable</i> | hasAuthority ('ADMIN') or (hasAuthority ('PROJECTMANAGER') and #managerUsername == authentication.name)) |
| getActiveProjectsByTeamMemberUsername | /active /teamMember /{teamMemberUsername} | GET | PathVariable: String <i>teamMemberUsername</i> Pageable <i>pageable</i> | hasAuthority ('ADMIN') or (hasAuthority ('TEAMMEMBER') and #teamMemberUsername == authentication.name)) |
| createProject | | POST | RequestBody: ProjectDto <i>projectDto</i> (validato) | hasAuthority ('ADMIN') |
| updateProject | /{projectId} | PUT | PathVariable: | hasAuthority |

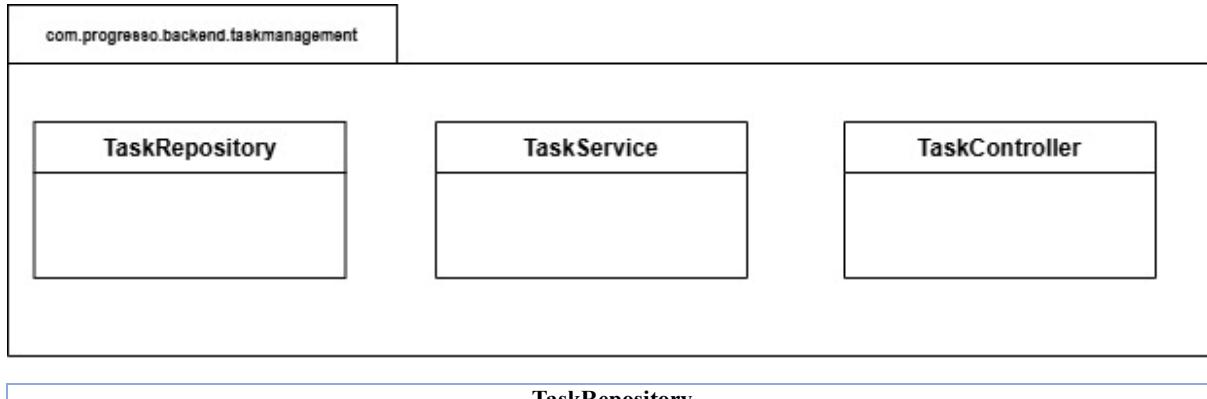


UNIVERSITÀ DEGLI STUDI
DI SALERNO

| | | | | |
|---------------------------------|---|-----|--|--|
| | | | Long <i>projectId</i> RequestBody: ProjectDto <i>projectDto</i> (validato) | ('ADMIN') or (hasAuthority ('PROJECTMANAG ER')) and @projectService .isManagerOfProject (# <i>projectId</i> , authentication.name)) |
| removeProject | /{projectId} /remove | PUT | PathVariable: Long <i>projectId</i> | hasAuthority ('ADMIN') or (hasAuthority ('PROJECTMANAG ER')) and @projectService .isManagerOfProject (# <i>projectId</i> , authentication.name)) |
| assignTeamToProjec t | /{projectId} /assign-team /{teamId} | PUT | PathVariable: Long <i>projectId</i> , Long <i>teamId</i> | hasAuthority ('ADMIN') or (hasAuthority ('PROJECTMANAG ER')) and @projectService .isManagerOfProject (# <i>projectId</i> , authentication.name)) |
| completeProject | /{projectId} /complete | PUT | PathVariable: Long <i>projectId</i> | hasAuthority ('ADMIN') or (hasAuthority ('PROJECTMANAG ER')) and @projectService .isManagerOfProject (# <i>projectId</i> , authentication.name)) |



4.1.1.5 com.progresso.backend.taskmanagement



| TaskRepository | | |
|---|-----------------|---|
| Metodo | Tipo di ritorno | Parametri |
| findByProjectIdAndStatus AndPriority | Page<Task> | Long <i>projectId</i> , Status <i>status</i> , Priority <i>priority</i> , Pageable <i>pageable</i> |

| TaskService | | | |
|---|-----------------|--|--|
| Metodo | Tipo di ritorno | Parametri | Descrizione |
| findByProjectIdAndStatus AndPriority | Page<TaskDto> | Long <i>projectId</i> , String <i>status</i> , String <i>priority</i> , Pageable <i>pageable</i> | Recupera in modalità paginata i task associati al progetto identificato da <i>projectId</i> , applicando filtri opzionali su <i>status</i> e <i>priority</i> . |
| createAndAssignTask | TaskDto | TaskDto <i>taskDto</i> , Long <i>userId</i> | Crea un nuovo task, assegnandolo a un utente specificato. Il metodo esegue varie validazioni su date, stato del progetto e del team, e il ruolo dell'utente prima di salvare il task e aggiornare le assegnazioni. |
| updateTask | TaskDto | Long <i>taskId</i> , TaskDto <i>taskDto</i> | Aggiorna un task esistente. Effettua controlli per impedire modifiche a campi non ammessi (ad es. completion date, status, utente assegnato) e aggiorna il nome se necessario, garantendo la coerenza con le regole di business. |
| completeTask | TaskDto | Long <i>taskId</i> | Segna un task come completato, aggiornando lo status a COMPLETED e impostando la data di completamento. Prima di completare il task, vengono verificate le condizioni relative alla data e allo stato del task e del progetto. |
| removeTaskFromProject | TaskDto | Long <i>projectId</i> , Long <i>taskId</i> | Cancella un task associato a un progetto, impostando lo status a CANCELLED e aggiornando la priorità a LOW. Il metodo esegue controlli sullo stato del |



| | | | |
|--|--|--|--|
| | | | progetto e del task per garantire che l'operazione sia consentita. |
|--|--|--|--|

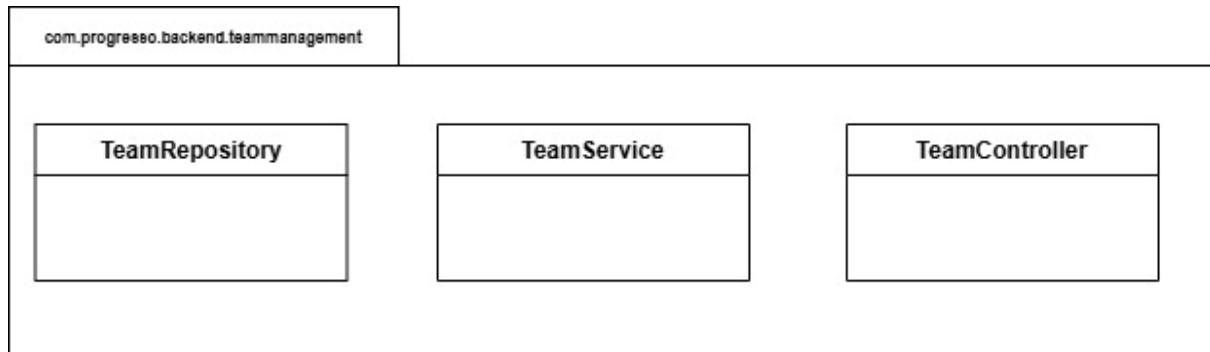
| TaskController <i>/api/tasks</i> | | | | |
|--------------------------------------|----------------------|-------------|---|---|
| Metodo | Endpoint | Http Method | Parametri | Sicurezza / PreAuthorize |
| getTasksByProjectIdAndFilters | /project/{projectId} | GET | PathVariable: Long <i>projectId</i> RequestParam (required=false): String <i>status</i> , String <i>priority</i> Pageable <i>pageable</i> | hasAuthority ('ADMIN') or (hasAuthority ('PROJECTMANAGER') and @projectService.isManagerOfProject (#projectId, authentication.name)) or (hasAuthority ('TEAMMEMBER') and @projectService.isTeamMemberOfProject (#projectId, authentication.name)) |
| createAndAssignTask | | POST | RequestBody: TaskDto <i>taskDto</i> (validato) RequestParam: Long <i>userId</i> | hasAuthority ('ADMIN') or (hasAuthority ('PROJECTMANAGER') and @projectService.isManagerOfProject (#taskDto.projectId, authentication.name)) |
| updateTask | /{taskId} | PUT | PathVariable: Long <i>taskId</i> RequestBody: TaskDto <i>taskDto</i> (validato) | hasAuthority ('ADMIN') or (hasAuthority ('PROJECTMANAGER') and @projectService.isManagerOfProject (#taskDto.projectId, authentication.name)) |
| completeTask | /{taskId}/complete | PATCH | PathVariable: Long <i>taskId</i> | hasAuthority ('ADMIN') or (hasAuthority ('PROJECTMANAGER') and @projectService.isManagerOfProject (@taskService.getProjectIdByTaskId) |



UNIVERSITÀ DEGLI STUDI
DI SALERNO

| | | | | |
|-----------------------|------------------------------------|--------|---|---|
| | | | | (#taskId), authentication.name)) |
| removeTaskFromProject | /project/{projectId}/task/{taskId} | DELETE | PathVariable: Long projectId, Long taskId | hasAuthority('ADMIN') or (hasAuthority('PROJECTMANAGER') and @projectService.isManagerOfProject(#projectId, authentication.name)) |

4.1.1.6 com.progresso.backend.teammanagement



| TeamRepository | | | |
|--------------------------------|-----------------|---|--|
| Metodo | Tipo di ritorno | Parametri | |
| findAllTeamsWithFilters | Page<Team> | Boolean active, String searchTerm, Pageable pageable | |
| findTeamsWithoutActiveProjects | Page<Team> | List<Status> activeStatuses, String searchTerm, Pageable pageable | |

| TeamService | | | |
|-------------------------------|-----------------|--------------------------------------|--|
| Metodo | Tipo di ritorno | Parametri | Descrizione |
| getAllTeamsWithFilters | Page<TeamDto> | Pageable pageable, String searchTerm | Recupera in modalità paginata i team attivi che non sono associati a progetti attivi, filtrando opzionalmente per nome. |
| getTeamsWithoutActiveProjects | Page<TeamDto> | Pageable pageable, String searchTerm | Recupera in modalità paginata i team attivi che non sono associati a progetti attivi, filtrando opzionalmente per nome. |
| createTeam | TeamDto | String teamName | Crea un nuovo team verificando che il nome sia valido e univoco (aggiungendo un suffisso se necessario), imposta lo stato attivo e restituisce il DTO del team creato. |
| updateTeam | TeamDto | Long id, String newName | Aggiorna il nome del team con l'ID specificato, dopo aver verificato che il nuovo nome sia valido e non sia |



| | | | |
|------------------------------|---------|---|--|
| | | | già in uso da altri team; salva le modifiche e restituisce il DTO aggiornato. |
| addMembersToTeam | TeamDto | Long <i>teamId</i> , List<Long> <i>userIds</i> | Aggiunge gli utenti, identificati dagli ID in <i>userIds</i> , al team specificato; verifica che gli utenti esistano, siano attivi, abbiano il ruolo corretto e non siano già membri di un altro team attivo; aggiorna le associazioni bidirezionali e restituisce il DTO aggiornato del team. |
| removeMembersFromTeam | TeamDto | Long <i>teamId</i> , List<Long> <i>userIds</i> | Rimuove dal team gli utenti indicati in <i>userIds</i> dopo aver verificato che siano membri attivi; aggiorna le assegnazioni dei task per questi utenti e restituisce il DTO del team aggiornato. |
| deleteTeam | TeamDto | Long <i>teamId</i> | Disattiva il team con l'ID specificato, verificando che il team non abbia progetti attivi; salva lo stato aggiornato e restituisce il DTO del team disattivato. |

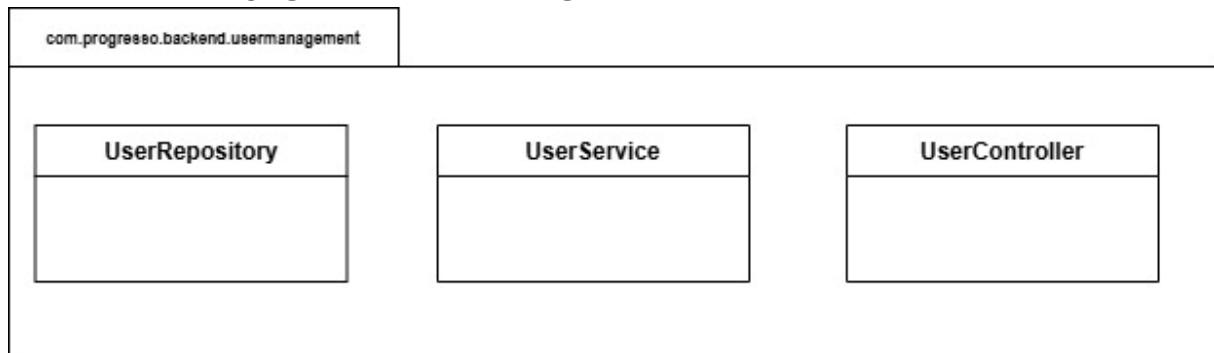
| TeamController <i>/api/teams</i> | | | | |
|-------------------------------------|-------------------|-------------|--|---|
| Metodo | Endpoint | Http Method | Parametri | Sicurezza / PreAuthorize |
| getAllTeams | | GET | RequestParam (required = false): Boolean active, String searchTerm Pageable pageable | hasAuthority ('ADMIN') |
| getAvailableTeams | /availableTeams | GET | RequestParam (required = false): String searchTerm Pageable pageable | hasAuthority ('ADMIN') or hasAuthority ('PROJECTMANAGER') |
| createTeam | | POST | RequestParam: String <i>teamName</i> (non vuoto, massimo 100 caratteri) | hasAuthority ('ADMIN') or hasAuthority ('PROJECTMANAGER') |
| updateTeam | / <i>{teamId}</i> | PUT | PathVariable: Long <i>teamId</i> RequestParam: String <i>newName</i> (non vuoto, massimo 100 caratteri) | hasAuthority ('ADMIN') or (hasAuthority ('PROJECTMANAGER') and @teamService.isProjectManagerOf TeamProjects (#teamId, authentication.name)) |



UNIVERSITÀ DEGLI STUDI
DI SALERNO

| | | | | |
|------------------------------|----------------------------|--------|--|--|
| addMembersToTeam | /{{teamId}}/members | POST | PathVariable: Long <i>teamId</i> RequestBody: List<Long> <i>userIds</i> | hasAuthority ('ADMIN') or (hasAuthority ('PROJECTMANAGER') and @teamService .isProjectManagerOf TeamProjects (#teamId, authentication.name)) |
| removeMembersFromTeam | /{{teamId}}/remove-members | POST | PathVariable: Long <i>teamId</i> RequestBody: List<Long> <i>userIds</i> | hasAuthority ('ADMIN') or (hasAuthority ('PROJECTMANAGER') and @teamService .isProjectManagerOf TeamProjects (#teamId, authentication.name)) |
| deleteTeam | /{{teamId}} | DELETE | PathVariable: Long <i>teamId</i> | hasAuthority ('ADMIN') |

4.1.1.7 com.progresso.backend.usermanagement



| UserRepository | | | |
|-------------------------------------|-----------------|--|--|
| Metodo | Tipo di ritorno | Parametri | |
| findAvailableProjectManagers | Page<User> | String <i>searchTerm</i> , Pageable <i>pageable</i> | |
| findAvailableTeamMembers | Page<User> | String <i>searchTerm</i> , Pageable <i>pageable</i> | |
| findUsersByTeamId | Page<User> | Long <i>teamId</i> , String <i>searchTerm</i> , Pageable <i>pageable</i> | |

| UserService | | | |
|------------------------------------|-----------------------|--|---|
| Metodo | Tipo di ritorno | Parametri | Descrizione |
| getAvailableProjectManagers | Page<UserResponseDto> | Pageable <i>pageable</i> , String <i>searchTerm</i> | Recupera i Project Manager disponibili con filtro di ricerca. |
| getAvailableTeamMembers | Page<UserResponseDto> | Pageable <i>pageable</i> , String <i>searchTerm</i> | Recupera i team member disponibili con filtro di ricerca. |



UNIVERSITÀ DEGLI STUDI
DI SALERNO

| | | | |
|-------------------------|-----------------------|--|---|
| getUsersByTeamId | Page<UserResponseDto> | Long <i>teamId</i> , Pageable <i>pageable</i> , String <i>searchTerm</i> | Recupera gli utenti appartenenti a un team specifico. |
|-------------------------|-----------------------|--|---|

| UserController <i>/api/users</i> | | | | |
|-------------------------------------|------------------------------|-------------|---|---|
| Metodo | Endpoint | Http Method | Parametri | Sicurezza / PreAuthorize |
| getAvailableProjectManagers | /available-project-managers | GET | Pageable <i>pageable</i> RequestParam (required = false): String <i>searchTerm</i> | hasAuthority ('ADMIN') |
| getAvailableTeamMembers | /available-team-members | GET | RequestParam (required = false): String <i>searchTerm</i> Pageable <i>pageable</i> | hasAuthority ('ADMIN') or hasAuthority ('PROJECTMANAGER') |
| getUsersByTeamId | /teams/{teamId}/team-members | GET | PathVariable: Long <i>teamId</i> RequestParam (required = false): String <i>searchTerm</i> Pageable <i>pageable</i> | hasAuthority ('ADMIN') or hasAuthority ('PROJECTMANAGER') or (hasAuthority ('TEAMMEMBER') and @teamService.isTeamMemberOfTeam (#teamId, authentication.name)) |



4.2 Sviluppo del front-end

4.2.1 Struttura delle cartelle

La disposizione delle cartelle è stata organizzata in modo da assicurare una chiara separazione delle responsabilità e una migliore manutenibilità del codice. La cartella principale, denominata **app**, rappresenta il nucleo dell'applicazione *Angular* e contiene tutte le risorse necessarie per gestire l'interfaccia utente e l'interazione con il back-end. All'interno della cartella **app** si trova il componente radice, chiamato *AppComponent*, che funge da punto di ingresso per l'interfaccia utente e si occupa dell'inizializzazione e del caricamento dei vari moduli e componenti.

Tutte le risorse della cartella **app** sono state organizzate nelle seguenti sottocartelle:

- **component**: contiene i componenti dell'interfaccia utente, ognuno organizzato in una propria cartella che include i file relativi: *HTML*, *CSS*, *TypeScript*.
- **dto**: include le interfacce utilizzate per modellare i dati trasferiti tra il front-end e il back-end, garantendo una struttura chiara e tipizzata.
- **guard**: contiene le guardie di navigazione di *Angular*, utilizzate per controllare l'accesso a determinate *routes* dell'applicazione in base allo stato di autenticazione e autorizzazione dell'utente.
- **interceptor**: comprende gli *interceptor HTTP*, i quali permettono di modificare o gestire globalmente le richieste e le risposte *HTTP*, per aggiungere token di autenticazione e gestire errori globali.
- **service**: contiene i servizi che comunicano con il back-end tramite chiamate *HTTP REST*, permettendo il recupero, l'invio e la manipolazione dei dati necessari all'applicazione.

Oltre a queste cartelle, nella cartella **app** sono presenti alcuni file di configurazione fondamentali:

- **app.component.ts/.html/.css**: file principali del componente radice dell'applicazione, responsabile della struttura generale dell'interfaccia.
- **app.config.ts**: contiene configurazioni globali dell'applicazione.
- **app.routes.ts**: definisce le *routes* dell'applicazione, specificando i componenti associati a ciascun percorso di navigazione.
- **material.module.ts**: modulo che importa e organizza i componenti *Angular Material*⁵ utilizzati nell'interfaccia grafica⁶.

⁵ *Angular Material* è una libreria di componenti UI per *Angular* che implementa le linee guida di design *Material* di *Google*. Fornisce una serie di componenti predefiniti e stilizzati, come pulsanti, form e tavole, che facilitano lo sviluppo di interfacce utente moderne e responsive.

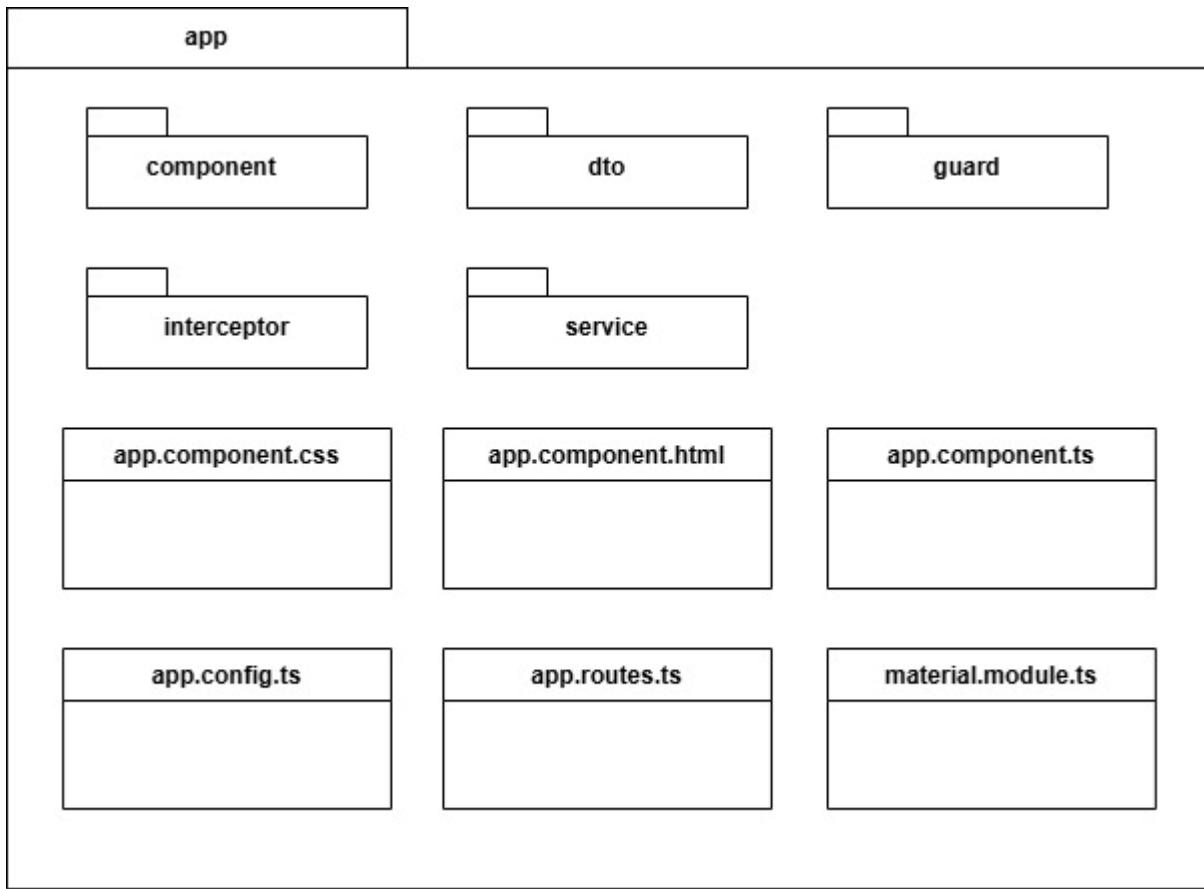
⁶ Oltre ad *Angular Material*, sono stati utilizzati anche il framework *Bootstrap* e il modulo *NgbModule*, parte della libreria *NgBootstrap* per *Angular*.



UNIVERSITÀ DEGLI STUDI
DI SALERNO

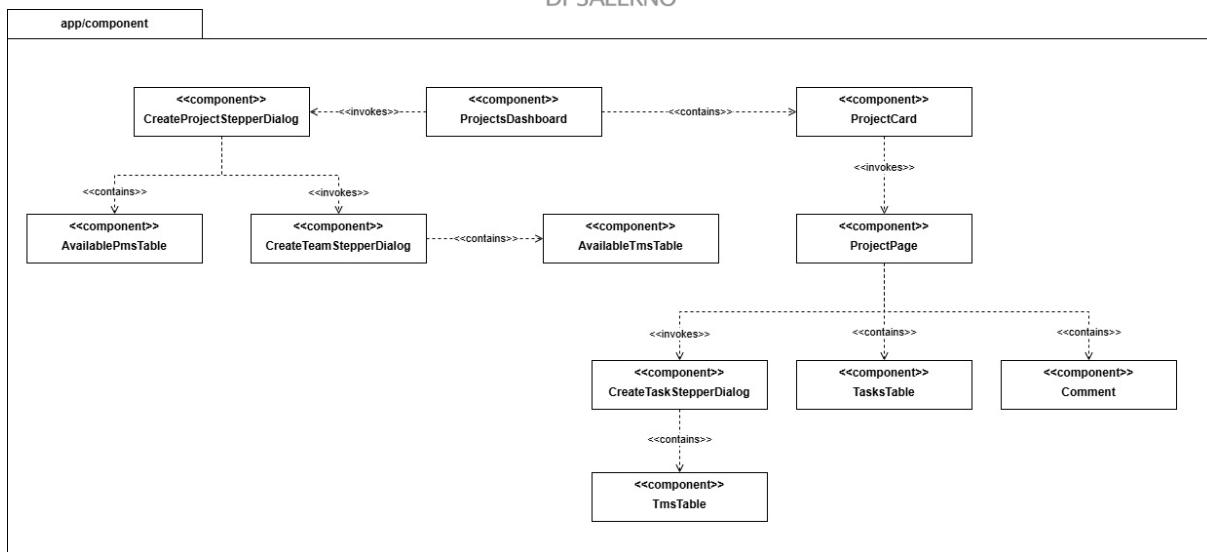
Per quanto riguarda il back-end, la struttura dei package è stata descritta in modo dettagliato, poiché questa parte dell'applicazione è generalmente più stabile e meno soggetta a modifiche rispetto al front-end. Di conseguenza, è stato utile fornire una panoramica completa di tutti i package coinvolti, per garantire una comprensione approfondita del funzionamento del sistema. Al contrario, per il front-end, dove la struttura è più dinamica e soggetta a cambiamenti frequenti, si è scelto di descrivere esclusivamente i service, i componenti e le altre parti del sistema che sono direttamente rilevanti per le principali operazioni messe a disposizione dalla piattaforma.

4.2.1.1 app



4.2.1.2 app/component

Il seguente diagramma si concentra sui componenti *Angular* direttamente utilizzati nelle operazioni principali offerte dal sistema e alcuni di essi sono stati progettati per essere riutilizzati in contesti differenti. Ad esempio, il componente *TmsTable* viene anche utilizzato in maniera diretta all'interno del componente *ProjectPage* per visualizzare i team member appartenenti al team assegnato al progetto. Inoltre, nella figura, la relazione <<contains>> indica che un componente include (o è “padre di”) un altro componente nel suo template, mentre <<invokes>> rappresenta un’interazione funzionale tra due componenti, in cui un componente richiama un altro tramite un evento.



Di seguito viene fornito un breve riepilogo dei componenti rappresentati nel diagramma, con una descrizione del loro ruolo e delle loro funzioni nelle operazioni principali dell'applicazione.

| Nome del componente | Ruolo | Interazioni |
|-----------------------------------|--|---|
| ProjectsDashboard | Rappresenta la pagina principale dell'applicazione. | Invoca il componente <i>CreateProjectStepperDialog</i> tramite un pulsante. |
| ProjectCard | Visualizza l'anteprima dei dati più significativi di un progetto. | Invoca il componente <i>ProjectPage</i> al click. |
| ProjectPage | Mostra tutti i dettagli del progetto selezionato, inclusa la sezione commenti e la lista dei task. | Invoca il componente <i>CreateTaskStepperDialog</i> tramite un pulsante. |
| CreateProjectStepperDialog | Gestisce il processo step-by-step per la creazione di un progetto. | Invoca il componente <i>CreateTeamStepperDialog</i> tramite un pulsante. |
| TmsTable | Visualizza tutti i membri di un team. | |
| CreateTaskStepperDialog | Gestisce il processo step-by-step per la creazione di un task. | |
| TasksTable | Mostra l'elenco dei task associati ad un progetto. | |
| Comment | Gestisce la visualizzazione e il caricamento dei commenti per un progetto. | |
| AvailablePmsTable | Mostra l'elenco dei project manager disponibili, ossia quelli che gestiscono meno di cinque progetti attivi. | |



| | | |
|--------------------------------|--|--|
| CreateTeamStepperDialog | Gestisce il processo step-by-step per la creazione di un team. | |
| AvailableTmsTable | Mostra l'elenco dei team member disponibili, ossia quelli che non appartengono a nessun team attivo. | |

4.2.1.3 app/dto

La cartella **dto** contiene le interfacce che definiscono la struttura dei dati trasferiti tra il front-end e il back-end. I *DTO* qui presenti sono gli stessi utilizzati dal back-end, garantendo coerenza e uniformità nel modello dei dati. Per una descrizione dettagliata dei *DTO* si rimanda all'**Appendice B**, in particolare:

[Appendice B.1: Descrizione approfondita dei package – com.progresso.backend.dto](#)

4.2.1.4 app/guard

La cartella **guard** contiene le guardie di navigazione di *Angular*, utilizzate per controllare l'accesso alle *routes*, definite nel file di configurazione principale **app.routes.ts**, in base allo stato di autenticazione e autorizzazione dell'utente. Queste guardie assicurano che solo gli utenti appropriati possano accedere alle aree protette dell'applicazione. Le guardie di navigazione utilizzate sono le seguenti:

- **AuthGuard:** Controlla l'accesso alle *routes* protette, verificando l'autenticazione e assicurando che l'utente possieda le autorizzazioni richieste.
- **LoginGuard:** Impedisce l'accesso alla pagina di login per gli utenti già autenticati.

4.2.1.5 app/interceptor

La cartella **interceptor** contiene gli *interceptor HTTP*, utilizzati per gestire le richieste e le risposte *HTTP* dell'applicazione *Angular* in modo centralizzato. In particolare, sono stati implementati i seguenti *interceptor*:

- **AuthInterceptor:** questo *interceptor* si occupa di includere il token di autenticazione nelle intestazioni di ogni richiesta *HTTP*, garantendo che l'utente sia autorizzato ad accedere alle risorse protette del sistema.
- **ErrorInterceptor:** questo *interceptor* intercetta le risposte *HTTP* che contengono errori (ad esempio, status 400, 403, 404, 500), fornendo una gestione centralizzata degli errori del client o del server. Genera messaggi di errore chiari e leggibili per l'utente e logga i dettagli tecnici per facilitare il debugging.

4.2.1.6 app/service

La cartella **service** contiene i servizi *Angular*, il cui ruolo principale è gestire la comunicazione tra il front-end e il back-end. Questi servizi fungono da intermediari, implementando metodi che inviano richieste *HTTP* agli endpoint definiti nei *controller* del back-end e restituiscono i dati elaborati ai componenti front-end che li utilizzano. Per ogni controller del back-end è stato realizzato un corrispondente servizio Angular per gestire la comunicazione. Di seguito viene



UNIVERSITÀ DEGLI STUDI
DI SALERNO

fornita una tabella dove vengono mostrati i servizi front-end implementati e i controller associati, ai quali si può accedere tramite collegamento apposito inserito nelle righe della tabella che rimandano all'**Appendice B.1 – Descrizione approfondita dei package**.

| Nome service | Controller associato |
|----------------|-----------------------------------|
| AuthService | AuthController |
| CommentService | CommentController |
| ProjectService | ProjectController |
| TaskService | TaskController |
| TeamService | TeamController |
| UserService | UserController |

4.2.2 Interfaccia utente

In questo paragrafo vengono presentati gli screenshot dell’interfaccia utente dell’applicazione, organizzati in base ai principali casi d’uso descritti nel paragrafo:

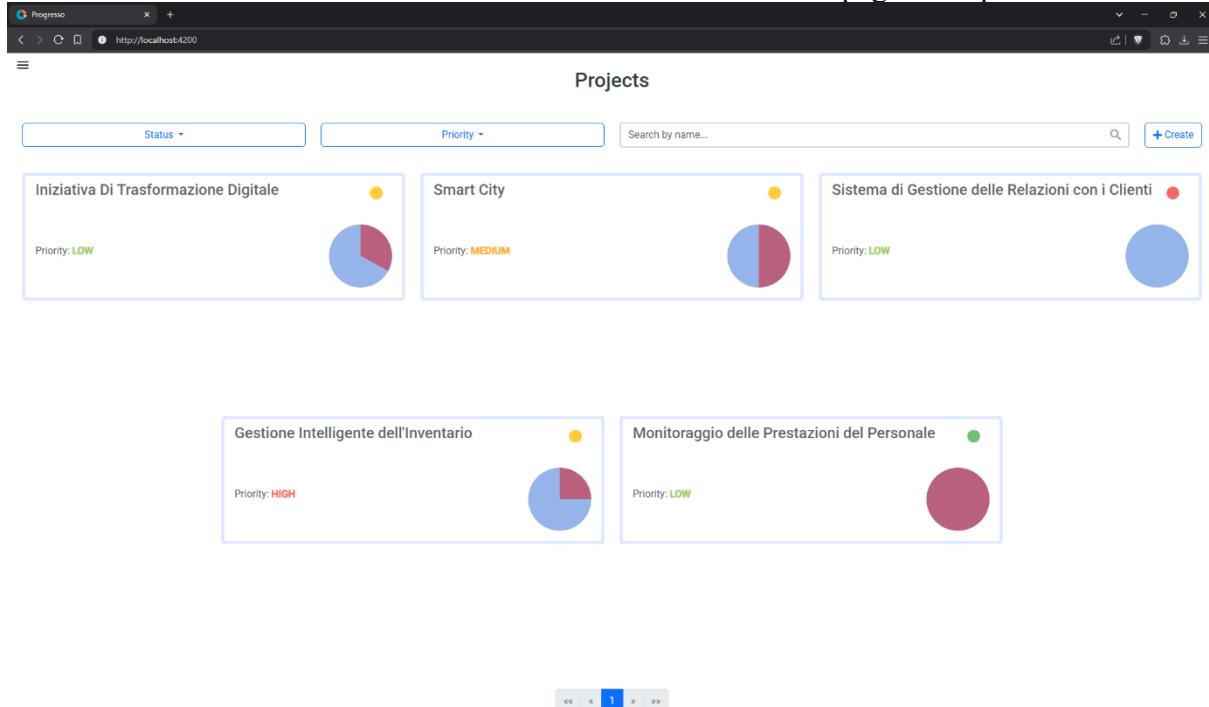
2.3.2 Descrizione dei principali casi d’uso

Gli screenshot sono disposti in sequenza e suddivisi in sezioni dedicate a ciascun flusso operativo, dove per ogni caso d’uso viene illustrato il percorso dell’utente e le interazioni con l’applicazione. Per comodità, saranno riportati le entry condition e i passi del flusso di eventi principale di ciascun caso d’uso, in corrispondenza con le schermate dell’applicazione.

4.2.2.1 UC_GestioneAM_1 – Creare un nuovo progetto

Entry condition

- ❖ L’admin è autenticato.
- ❖ L’admin si trova nella homepage della piattaforma.





UNIVERSITÀ DEGLI STUDI
DI SALERNO

1 Admin:

L'admin utilizza il comando “Create”.

Progresso

Status ▾ Priority ▾ Search by name... + Create

Iniziativa Di Trasformazione Digitale

Priority: LOW

Smart City

Priority: MEDIUM

Sistema di Gestione delle Relazioni con i Clienti

Priority: HIGH

Gestione Intelligente dell'Inventory

Priority: HIGH

Monitoraggio delle Prestazioni del Personale

Priority: LOW

1 / 1



UNIVERSITÀ DEGLI STUDI
DI SALERNO

2 Sistema:

Il sistema mostra una finestra di dialogo contenente uno stepper. Il primo step, chiamato “Project Info”, contiene un form con i seguenti campi:

- ❖ Name: stringa di caratteri alfanumerici.
- ❖ Description: stringa di caratteri alfanumerici.
- ❖ Start Date
 - YYYY: una lista di valori per selezionare l'anno.
 - MM: una lista di valori per selezionare il mese.
 - DD: una lista di valori per selezionare il giorno.
- ❖ Due Date
 - YYYY: una lista di valori per selezionare l'anno.
 - MM: una lista di valori per selezionare il mese.
 - DD: una lista di valori per selezionare il giorno.

The screenshot shows a web browser window titled "Progresso" with the URL "http://localhost:4200". The main page is titled "Projects" and displays several project cards. One card for "Iniziativa Di Trasformazione Digitale" has "Priority: LOW". Another card for "Gestione Intelligente" has "Priority: HIGH". A third card for "Smart City" is currently selected, showing its details. A modal dialog box is open over the Smart City card, titled "Project Info". It contains fields for "Name" (with placeholder "Please enter the project name.") and "Description" (with placeholder "Please enter the project description."). Below these are dropdown menus for "Start Date" (with placeholder "Please select a start date.") and "Due Date" (with placeholder "Please select a due date."). At the bottom of the dialog are "Close" and "Next" buttons, and a navigation bar at the bottom of the page includes buttons like "««", "< >", "»»", and "1".



UNIVERSITÀ DEGLI STUDI
DI SALERNO

3 Admin:

L'admin compila il form.

4 Sistema:

Il sistema controlla i dati inseriti:

- ❖ Controlla che il campo “Name” sia valido.
- ❖ Controlla che il campo “Description” sia valido.
- ❖ Controlla che il campo “Start Date” sia valido.
- ❖ Controlla che il campo “Due Date” sia valido.

5 Admin:

L'admin utilizza il comando “Next”, il quale è stato reso disponibile dopo la validazione del form, per spostarsi al prossimo step.

The screenshot shows a web browser window with a title bar "Progetto" and address bar "http://localhost:4200". The main content area is titled "Projects" and displays several project cards. One card for "Iniziativa Di Trasformazione Digitale" has "Priority: LOW". Another card for "Gestione Intelligente" has "Priority: HIGH". A third card for "Sistema di Gestione delle Relazioni con i Clienti" has a red error icon and "Priority: LOW". A modal dialog box is open over the cards, titled "Smart City". It contains three tabs: 1 Project info, 2 Assign Project Manager, and 3 Assign Team. The first tab is active, showing fields for "Name" (filled with "Progresso") and "Description" (filled with "Piattaforma per la gestione centralizzata dei progetti software."). Below these fields are dropdowns for "Start Date" (set to 2025-03-24) and "Due Date" (set to 2025-05-24). Validation messages "Project name is valid." and "Description is valid." are displayed next to their respective fields. At the bottom of the modal are "Close" and "Next" buttons, with "Next" being highlighted by a mouse cursor. The footer of the page shows navigation icons like back, forward, and search.



UNIVERSITÀ DEGLI STUDI
DI SALERNO

6 Sistema:

Il sistema mostra il secondo step, chiamato “Assign Project Manager”, contenente una tabella che mostra tutti i project manager disponibili, ossia quelli che gestiscono meno di cinque progetti attivi.

The screenshot shows a web application interface titled "Projects". At the top, there are filters for "Status" and "Priority", a search bar, and a "Create" button. Below the header, several project cards are displayed, each with a title, a progress bar, and a priority indicator (yellow for low, red for high). A modal window is open in the center, titled "Assign Project Manager". It contains a search bar and a table with four rows of data:

| ID | First Name | Last Name | Username |
|----|------------|------------|--------------------------------|
| 68 | Giuseppe | Rossi | g.rossi.pm1@progresso.com |
| 88 | Carlo | Vanvitelli | c.vanvitelli.pm2@progresso.com |
| 89 | Giovanna | Zante | g.zante.pm3@progresso.com |
| 90 | Carmela | Pedicini | c.pedicini.pm4@progresso.com |

At the bottom of the modal, there are buttons for "Close", "Back", and "Next".



UNIVERSITÀ DEGLI STUDI
DI SALERNO

- 7 **Admin:** L'admin seleziona il project manager desiderato cliccando sulla riga della tabella corrispondente.
- 8 **Sistema:** Il sistema verifica se è stata fatta una selezione.
- 9 **Admin:** L'admin utilizza il comando “Next”, il quale è stato reso disponibile dopo la selezione del project manager, per spostarsi al prossimo step.

| ID | First Name | Last Name | Username |
|----|------------|------------|--------------------------------|
| 68 | Giuseppe | Rossi | g.rossi.pm1@progresso.com |
| 88 | Carlo | Vanvitelli | c.vanvitelli.pm2@progresso.com |
| 89 | Giovanna | Zante | g.zante.pm3@progresso.com |
| 90 | Carmela | Pedicini | c.pedicini.pm4@progresso.com |

1 – 4 of 4 < >

Close Back Next



UNIVERSITÀ DEGLI STUDI
DI SALERNO

10 Sistema:

Il sistema mostra il terzo step, chiamato “Assign Team”, contenente tre pulsanti, ciascuno corrispondente alle opzioni disponibili per l’admin nell’assegnazione del team:

- ❖ Assign Existing Team: per selezionare un team esistente.
- ❖ Create New Team: per creare un nuovo team.
- ❖ Assign Team Later: per rimandare l’assegnazione del team a un momento successivo.

11 Admin:

L’admin utilizza il comando “Assign Team Later”.

The figure consists of two screenshots of a web application interface titled "Projects".
The top screenshot shows a modal dialog box centered over the "Assign Team" step. The dialog contains three buttons: "Assign Existing Team", "or", "Create New Team", "or", and "Assign Team Later". The "Assign Team Later" button is highlighted with a mouse cursor.
The bottom screenshot shows the same modal dialog, but the "Assign Team Later" button has been clicked, as indicated by a small white circle on its surface. The other buttons are now dimmed.
Both screenshots show four project cards in the background: "Iniziativa Di Trasformazione Digitale" (Priority: LOW), "Smart City" (Priority: MEDIUM), "Gestione Intelligente" (Priority: HIGH), and "Sistema di Gestione delle Relazioni con i Clienti" (Priority: LOW). Each card has a pie chart icon and a colored dot indicating its priority level.



UNIVERSITÀ DEGLI STUDI
DI SALERNO

12 Sistema:

Il sistema salva il progetto e mostra un toast contenente un messaggio di successo. Infine, rende disponibile il nuovo progetto sulla piattaforma.

The screenshot shows a web browser window titled "Progresso" with the URL "http://localhost:4200". The page is titled "Projects" and displays six project cards. From top-left to bottom-right, the projects are:

- Iniziativa Di Trasformazione Digitale (Priority: LOW)
- Smart City (Priority: MEDIUM)
- Sistema di Gestione delle Relazioni con i Clienti (Priority: HIGH)
- Gestione Intelligente dell'Inventario (Priority: HIGH)
- Monitoraggio delle Prestazioni del Personale (Priority: LOW)
- Progresso (Priority: LOW, highlighted with a blue border and a cursor icon)

A green toast notification at the top right of the page says "Project created successfully!".



UNIVERSITÀ DEGLI STUDI
DI SALERNO

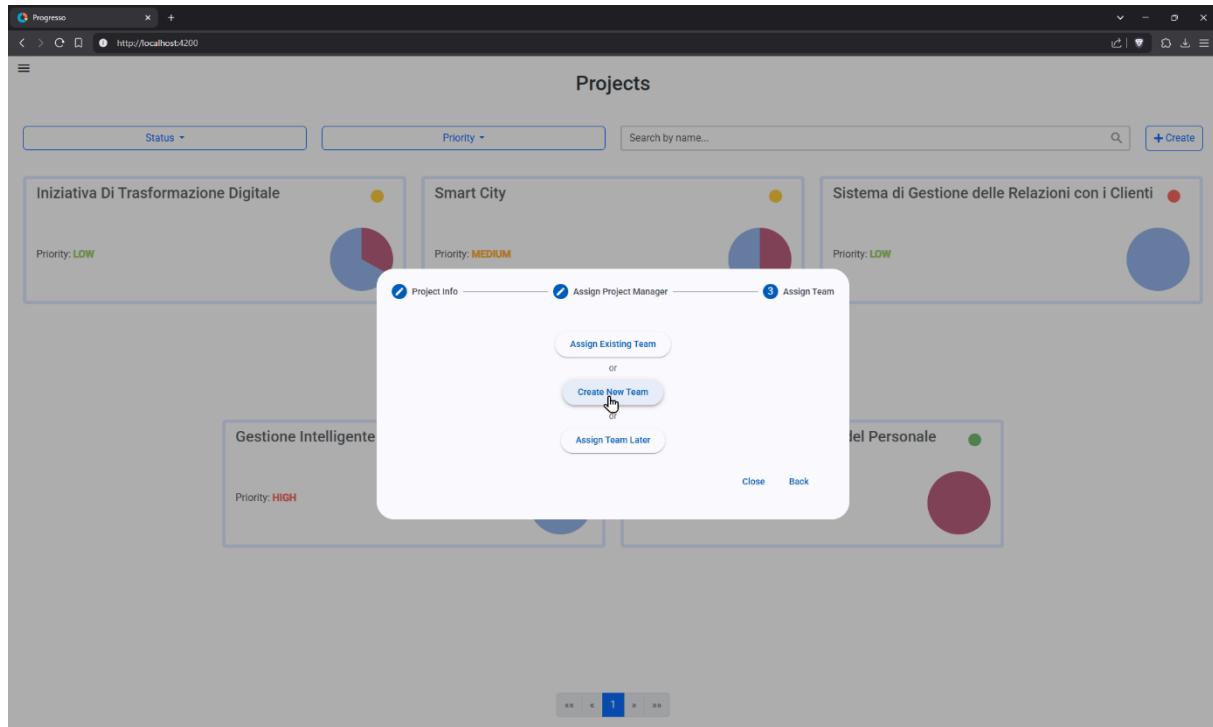
4.2.2.2 UC_GestioneAMPM_1 – Creare un nuovo team

Entry condition

- ❖ L'admin è autenticato
- ❖ L'admin si trova nel flusso di creazione di un progetto.

1 Admin:

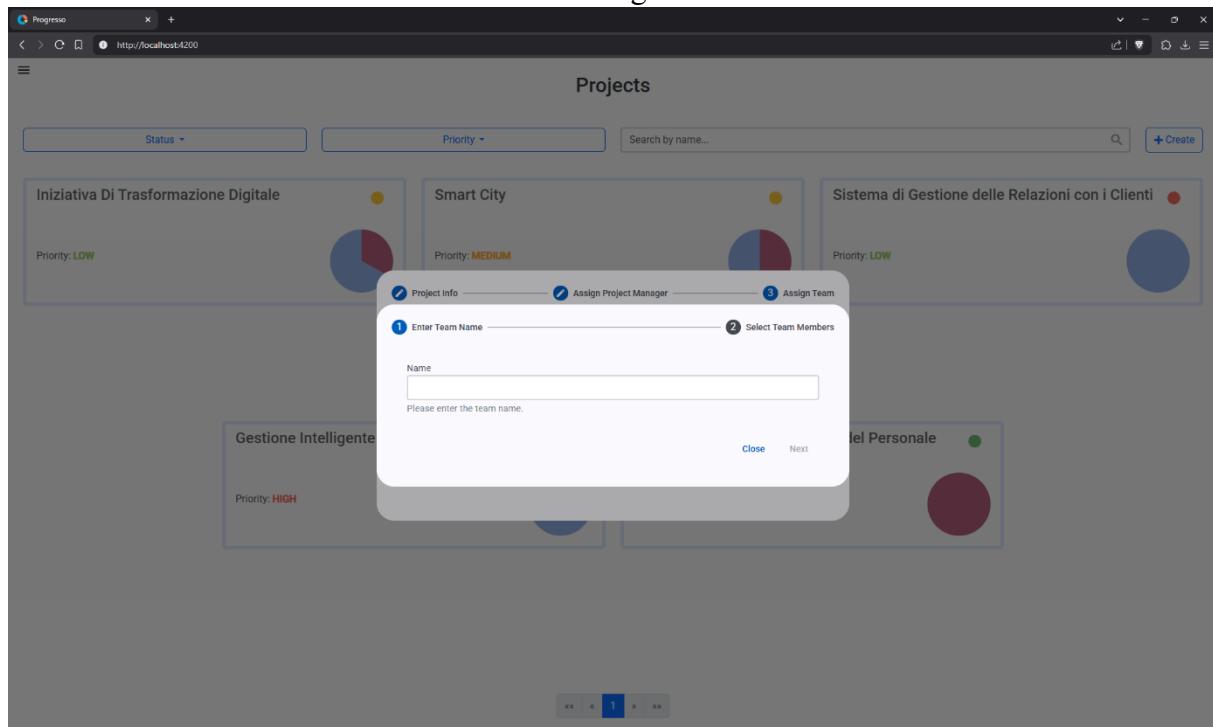
L'admin utilizza il comando “Create New Team”.



2 Sistema:

Il sistema mostra una finestra di dialogo contenente uno stepper. Il primo step, chiamato “Enter Team Name”, contiene un form con i seguenti campi:

- ❖ Name: stringa di caratteri alfanumerici.





UNIVERSITÀ DEGLI STUDI
DI SALERNO

- 3 **Admin:** L'admin compila il form.
- 4 **Sistema:** Il sistema controlla i dati inseriti:
❖ Controlla che il campo "Name" sia valido.
- 5 **Admin:** L'admin utilizza il comando "Next", il quale è stato reso disponibile dopo la validazione del form, per spostarsi al prossimo step.

The screenshot shows a web browser window titled 'Progresso' with the URL 'http://localhost:4200'. The main page is titled 'Projects' and displays several project cards: 'Iniziativa Di Trasformazione Digitale' (Priority: LOW), 'Smart City' (Priority: MEDIUM), 'Sistema di Gestione delle Relazioni con i Clienti' (Priority: LOW), 'Gestione Intelligente' (Priority: HIGH), and 'Gestione del Personale' (Priority: HIGH). A modal dialog is open in the center, divided into two tabs: '1 Enter Team Name' and '2 Select Team Members'. The 'Enter Team Name' tab shows a text input field with 'AlphaTeam' and a validation message '✓ Team name is valid.' Below the tabs are 'Close' and 'Next' buttons, with 'Next' being the one currently highlighted by a cursor. At the bottom of the page, there is a navigation bar with icons for back, forward, and search.



UNIVERSITÀ DEGLI STUDI
DI SALERNO

6 Sistema:

Il sistema mostra il secondo step, chiamato “Select Team Members”, contenente una tabella che mostra tutti i team member disponibili, ossia quelli che non appartengono a nessun team.

The screenshot shows a web browser window with the URL <http://localhost:4200>. The main page has a header "Projects" and filters for "Status" and "Priority". Below the header are four project cards: "Iniziativa Di Trasformazione Digitale" (Priority: LOW), "Smart City" (Priority: HIGH), "Sistema di Gestione delle Relazioni con i Clienti" (Priority: LOW), and "Gestione Intelligente" (Priority: HIGH). A modal dialog box titled "Select Team Members" is open in the center. It contains a search bar and a table with the following data:

| ID | First Name | Last Name | Username |
|----|------------|------------|--------------------------------|
| 84 | Giacomo | Giorgio | g.giorgio.tm6@progresso.com |
| 85 | Matteo | Mattei | m.mattei.tm7@progresso.com |
| 86 | Lorenzo | Prattico | l.prattico.tm8@progresso.com |
| 87 | Michele | Bartolomei | m.bartolomei.tm9@progresso.com |

At the bottom of the modal are buttons for "Close", "Back", and "Complete".



UNIVERSITÀ DEGLI STUDI
DI SALERNO

- 7 **Admin:** L'admin seleziona i team member desiderati cliccando sulle righe della tabella corrispondenti.
- 8 **Sistema:** Il sistema verifica se è stata fatta una selezione.
- 9 **Admin:** L'admin utilizza il comando “Complete”, il quale è stato reso disponibile dopo la selezione dei team member, per creare il team.

The screenshot shows a web application interface titled "Projects". In the center, a modal dialog is open for creating a team. The dialog has two main sections: "Enter Team Name" and "Select Team Members". The "Select Team Members" section contains a search bar and a table with four rows of data:

| ID | First Name | Last Name | Username |
|----|------------|------------|--------------------------------|
| 84 | Giacomo | Giorgio | g.giorgio.tm6@progresso.com |
| 85 | Matteo | Mattel | m.mattel.tm7@progresso.com |
| 86 | Lorenzo | Prattico | l.prattico.tm8@progresso.com |
| 87 | Michele | Bartolomei | m.bartolomei.tm9@progresso.com |

At the bottom of the modal, there are three buttons: "Close", "Back", and "Complete". A hand cursor is hovering over the "Complete" button. The background of the application shows several project cards with titles like "Iniziativa Di Trasformazione Digitale", "Smart City", "Gestione Intelligente", and "Sistema di Gestione delle Relazioni con i Clienti".



UNIVERSITÀ DEGLI STUDI
DI SALERNO

10 Sistema:

Il sistema salva il progetto, salva e assegna il team, e mostra tre toast: uno per l'avvenuta creazione del progetto, uno per la creazione del team e uno per l'assegnazione del team al progetto. Infine, il progetto viene reso disponibile sulla piattaforma.

The screenshot displays a web-based project management interface. At the top, there's a header bar with the title 'Progresso' and a URL 'http://localhost:4200'. Below the header, a search bar says 'Search by name...'. The main area is titled 'Projects' and contains five cards, each representing a different project:

- Iniziativa Di Trasformazione Digitale**: Priority: LOW. Pie chart shows approximately 75% blue and 25% red.
- Smart City**: Priority: MEDIUM. Pie chart shows approximately 50% blue and 50% red.
- Sistema di Gestione delle Relazioni con i Clienti**: Priority: LOW. Pie chart is mostly blue with a small red segment.
- Gestione Intelligente dell'Inventario**: Priority: HIGH. Pie chart shows approximately 70% blue and 30% red.
- Monitoraggio delle Prestazioni del Personale**: Priority: LOW. Pie chart is mostly red with a small blue segment.

A progress bar at the bottom indicates 'Progresso' at 100%, with a hand cursor icon over it. Above the progress bar, three green toast notifications appear sequentially:

- Team created successfully!
- Project created successfully!
- Team assigned to project successfully!



UNIVERSITÀ DEGLI STUDI
DI SALERNO

4.2.2.3 UC_GestioneAMPM_2 – Creare ed assegnare un nuovo task

Entry condition

- ❖ Il project manager è autenticato.
- ❖ Il project manager si trova nella pagina del progetto in cui desidera creare il task.
- ❖ Il progetto selezionato deve avere un team assegnato.

The screenshot shows the Progresso web application interface. At the top, it displays the project details: Start Date: 2025-03-24, Due Date: 2025-05-24, Priority: LOW, Project Manager: Giovanna Zante (g.zante.pm3@progresso.com), and Team: AlphaTeam. Below this, there is a search bar and a table listing team members:

| ID | First Name | Last Name | Username |
|----|------------|------------|--------------------------------|
| 84 | Giacomo | Giorgio | g.giorgio.tn6@progresso.com |
| 85 | Matteo | Mattel | m.mattel.tn7@progresso.com |
| 87 | Michele | Bartolomei | m.bartolomei.tn9@progresso.com |

On the right side, there is a large blue circle representing the project progress at 0% complete. Below the progress bar, there is a "Task List" section with a table header:

| Name | Start Date | Due Date | Completion Date | Priority | Status | Owner | Description | Actions |
|------|------------|----------|-----------------|----------|--------|-------|-------------|---------|
|------|------------|----------|-----------------|----------|--------|-------|-------------|---------|

A message "No tasks found." is displayed below the table. At the bottom, there is a "Comments" section where Giovanna Zante has written "Write a comment..." and a blue send icon.

1 Project Manager: Il project manager utilizza il comando “Add Task”.

The screenshot shows the Progresso web application interface, identical to the previous one but with a focus on the "Add Task" button. The "Task List" table header now includes a blue "+ Add Task" button. The rest of the interface, including the team list, progress bar, and comments section, remains the same.



UNIVERSITÀ DEGLI STUDI
DI SALERNO

2 Sistema:

Il sistema mostra una finestra di dialogo contenente uno stepper. Il primo step, chiamato “Task Info”, contiene un form con i seguenti campi:

- ❖ Name: stringa di caratteri alfanumerici.
- ❖ Description: stringa di caratteri alfanumerici.
- ❖ Priority: una lista composta da tre opzioni: “High”, “Medium”, “Low”.
- ❖ Start Date
 - YYYY: una lista di valori per selezionare l’anno.
 - MM: una lista di valori per selezionare il mese.
 - DD: una lista di valori per selezionare il giorno.
- ❖ Due Date
 - YYYY: una lista di valori per selezionare l’anno.
 - MM: una lista di valori per selezionare il mese.
 - DD: una lista di valori per selezionare il giorno

The screenshot shows a web browser window for the 'Progresso' application at the URL <http://localhost:4200/project/196/Progresso>. The main interface displays project details: Start Date: 2025-03-24, Due Date: 2025-05-24, Priority: LOW. It also shows the Project Manager (Giovanna Zante) and the Team (AlphaTeam). A search bar is present. On the right, there's a large blue circle indicating 0% complete progress. Below the main area, there's a table with columns for Name, Start Date, and Due Date, and a comments section where Giovanna Zante has commented. A 'Task Info' dialog box is open in the center, prompting for task details like Name, Description, Priority, and dates.

Start Date: 2025-03-24 Due Date: 2025-05-24 Priority: LOW

Project Manager: Giovanna Zante (g.zante.pm3@progresso.com)

Team: AlphaTeam

Search...

| ID | First Name | Last Name | Username |
|----|------------|------------|---------------|
| 84 | Giacomo | Giorgio | g.giorgio.tm6 |
| 85 | Matteo | Mattel | m.mattel.tm7 |
| 87 | Michele | Bartolomei | m.bartolomei |

Comments
Giovanna Zante (g.zante.pm3@progresso.com)
Write a comment...

Task Info

Name:
Please enter the task name.

Description:
Please enter the task description.

Priority: Select priority

Please select the task priority.

Start Date: YYYY MM DD Due Date: YYYY MM DD

Please select a start date. Please select a due date.

Assign User

Owner:
Description:
Actions:

Close **Next**



UNIVERSITÀ DEGLI STUDI
DI SALERNO

3 Project Manager: Il project manager compila il form.

4 Sistema: Il sistema controlla i dati inseriti:

- ❖ Controlla che il campo “Name” sia valido.
- ❖ Controlla che il campo “Description” sia valido.
- ❖ Controlla che il campo “Priority” sia valido.
- ❖ Controlla che il campo “Start Date” sia valido.
- ❖ Controlla che il campo “Due Date” sia valido.

5 Project Manager: Il project manager utilizza il comando “Next”, il quale è stato reso disponibile dopo la validazione del form, per spostarsi al prossimo step.

The screenshot shows a web-based application titled "Progresso" running on a local host. The main page displays a table of team members (AlphaTeam) with columns for ID, First Name, Last Name, and Username. A modal dialog box is open, titled "Task Info", containing fields for "Name" (set to "Implementazione autenticazione utente"), "Description" (set to "Realizzare un modulo di autenticazione che Integri Spring Security e JWT per gestire login, logout e la validazione del token"), and "Priority" (set to "HIGH"). Validation messages indicate that the task name is valid and the priority is valid. Below the dialog, a large blue circle represents the task's progress at 0% complete. At the bottom right of the dialog, there are "Close" and "Next" buttons, with "Next" being highlighted with a cursor icon. The overall interface is clean and modern, using a light gray color scheme.



UNIVERSITÀ DEGLI STUDI
DI SALERNO

6 Sistema:

Il sistema mostra il secondo step, chiamato “Assign User”, contenente una tabella che mostra tutti i team member del team assegnato al progetto.

The screenshot shows a web-based project management application named "Progresso". The title bar indicates the URL is <http://localhost:4200/project/196/Progresso>. The main header says "Progresso" and "Piattaforma per la gestione centralizzata di progetti software." Below this, there's a summary section with "Start Date: 2025-03-24", "Due Date: 2025-05-24", and "Priority: LOW". It lists the "Project Manager: Giovanna Zante (g.zante.pm3@progresso.com)" and the "Team: AlphaTeam". A search bar is present above a table titled "Task Info". The table has columns: ID, First Name, Last Name, and Username. It contains three rows of data:

| ID | First Name | Last Name | Username |
|----|------------|------------|------------------|
| 84 | Giacomo | Giorgio | g.giorgio.tm6 |
| 85 | Matteo | Mattel | m.mattel.tm7 |
| 87 | Michele | Bartolomei | m.bartolomei.tm9 |

To the right of the table is a large blue circle with a progress bar at 0% complete. Below the table, there are buttons for "Close", "Back", and "Complete". At the bottom of the page, there's a "Comments" section with a message from "Giovanna Zante (g.zante.pm3@progresso.com)" and a text input field for "Write a comment...".



UNIVERSITÀ DEGLI STUDI
DI SALERNO

- 7 **Project Manager:** Il project manager seleziona il team member desiderato cliccando sulla riga della tabella corrispondente.
- 8 **Sistema:** Il sistema verifica se è stata fatta una selezione.
- 9 **Project manager:** Il project manager utilizza il comando “Complete”, il quale è stato reso disponibile dopo la selezione del team member, per creare ed assegnare il task.

The screenshot shows the Progresso software interface. At the top, it displays the project details: Start Date: 2025-03-24, Due Date: 2025-05-24, Priority: LOW, Project Manager: Giovanna Zante (g.zante.pm3@progresso.com), and Team: AlphaTeam. Below this, there is a search bar and a table listing team members:

| ID | First Name | Last Name | Username |
|----|------------|------------|------------------|
| 84 | Giacomo | Giorgio | g.giorgio.tm6 |
| 85 | Matteo | Mattei | m.mattei.tm7 |
| 87 | Michele | Bartolomei | m.bartolomei.tm9 |

A modal dialog box titled "Task Info" is open, showing the same table with the first row (Giacomo) selected. A hand cursor is hovering over the "Complete" button at the bottom right of the dialog. To the right of the dialog, a large blue circle indicates "0% complete".

Below the table, there is a section for "Comments" with a placeholder "Write a comment..." and a send icon.

The main interface below the dialog shows a table with columns: Name, Start Date, Due Date, Owner, Description, and Actions. A message "No tasks found." is displayed.

The second screenshot shows the same interface after a task has been completed. The "Complete" button in the dialog is now highlighted with a blue background, indicating it has been clicked. The blue circle on the right still shows "0% complete".



UNIVERSITÀ DEGLI STUDI
DI SALERNO

10 Sistema

Il sistema salva il task e mostra un toast contenente un messaggio di successo. Infine, rende disponibile il nuovo task sulla piattaforma.

The screenshot shows a web browser window for the 'Progresso' platform. At the top, there's a header bar with the URL 'http://localhost:4200/project/196/Progresso'. A green toast notification on the right says 'Task created successfully!'. Below the header, the page title is 'Progresso' with the subtitle 'Piattaforma per la gestione centralizzata di progetti software.' and a blue edit icon. On the left, there's a table titled 'Team' with columns: ID, First Name, Last Name, and Username. It lists three team members: Giacomo (ID 84), Giorgio (ID 85), and Bartolomei (ID 87). On the right, there's a large blue circle with a yellow dot and the text '0% complete'. Below this is a 'Task List' section with a table header: Name, Start Date, Due Date, Completion Date, Priority, Status, Owner, Description, and Actions. One task is listed: 'Implementaz...' with a start date of '2025-03-24', due date of '2025-04-01', priority 'HIGH', status 'Not Completed', owner 'g.giorgio.tm6@progresso.com', and description 'Realizzare un m...'. At the bottom left, there's a 'Comments' section with a comment from 'Giovanna Zante (g.zante.pm3@progresso.com)'. A text input field says 'Write a comment...' with a blue send icon on the right. Navigation icons for back, forward, and search are at the very bottom.



UNIVERSITÀ DEGLI STUDI DI SALERNO

Nota: è possibile visualizzare il nome completo del task e la sua descrizione completa passando il mouse sopra alle voci corrispondenti nella tabella.

Screenshot of the Progresso software interface showing a project dashboard and task list.

Project Overview:

- Start Date: 2025-03-24
- Due Date: 2025-05-24
- Priority: LOW
- Project Manager: Giovanna Zante (g.zante.pm3@progresso.com)
- Team: AlphaTeam

Task List:

| Name | Start Date | Due Date | Completion Date | Priority | Status | Owner | Description | Actions |
|---------------------------------------|------------|------------|-----------------|----------|---------|-----------------------------|--------------------|---------|
| Implementazione autenticazione utente | 2025-03-24 | 2025-04-01 | Not Completed | HIGH | Pending | g.giorgio.tn6@progresso.com | Realizzare un m... | |

Comments:

Giovanna Zante (g.zante.pm3@progresso.com)

Write a comment...

Task Details:

Task Name: Implementazione autenticazione utente

Start Date: 2025-03-24

Due Date: 2025-04-01

Completion Date: Not Completed

Priority: HIGH

Status: Pending

Owner: g.giorgio.tn6@progresso.com

Description: Realizzare un m...

Actions:

Progress Indicators:

- A large blue circle indicates 0% complete.
- A smaller yellow dot is positioned near the top right of the blue circle.



UNIVERSITÀ DEGLI STUDI
DI SALERNO

4.2.2.4 UC_GestioneAMPMTM_I – Aggiungere un commento ad un progetto

Entry condition

- ❖ Il team member è autenticato.
- ❖ Il team member si trova nella pagina del progetto in cui desidera aggiungere il commento.
- ❖ Il progetto selezionato deve essere attivo.

1 Sistema:

Il sistema mostra un form contenente la casella di testo per l'inserimento del commento.

The screenshot shows a web-based project management interface. At the top, there's a header bar with the title 'Progresso' and a URL 'http://localhost:4200/project/1%6/Progresso'. Below the header, project details are displayed: Start Date: 2025-03-24, Due Date: 2025-05-24, Priority: LOW, Project Manager: Giovanna Zante (g.zante.pm3@progresso.com), and Team: AlphaTeam. A search bar is present. Below these details is a table listing team members:

| ID | First Name | Last Name | Username |
|----|------------|------------|--------------------------------|
| 84 | Giacomo | Giorgio | g.giorgio.tm6@progresso.com |
| 85 | Matteo | Mattei | m.mattei.tm7@progresso.com |
| 87 | Michele | Bartolomei | m.bartolomei.tm9@progresso.com |

Below the table is a large blue circle representing a progress bar with the text '0% complete' next to it. The main content area is titled 'Task List' and contains a single task entry:

| Name | Start Date | Due Date | Completion Date | Priority | Status | Owner | Description |
|-------------------|------------|------------|-----------------|----------|--------|-----------------------------|-----------------|
| Implementazion... | 2025-03-24 | 2025-04-01 | Not Completed | HIGH | ● | g.giorgio.tm6@progresso.com | Realizzare m... |

Below the task list is a section titled 'Comments' which includes a list of comments from Giacomo Giorgio (g.giorgio.tm6@progresso.com). A text input field for adding a new comment is shown, along with a message indicating 'No comments found for this project.' at the bottom.



UNIVERSITÀ DEGLI STUDI
DI SALERNO

- 2 Team Member:** Il team member inserisce il commento nella casella di testo.
- 3 Sistema:** Il sistema verifica che il commento sia valido.
- 4 Team Member:** Il team member utilizza l'apposito pulsante per inviare il commento, il quale è stato reso disponibile dopo la validazione del form.

Screenshot of a web application interface showing a project details page and a task list page.

Project Details Page:

- Start Date: 2025-03-24
- Due Date: 2025-05-24
- Priority: LOW
- Project Manager: Giovanna Zante (g.zante.pm3@progresso.com)
- Team: AlphaTeam

A large blue circle progress bar indicates 0% complete.

Task List Page:

| Name | Start Date | Due Date | Completion Date | Priority | Status | Owner | Description |
|-------------------|------------|------------|-----------------|----------|---------------------------------------|-----------------------------|--------------------|
| Implementazion... | 2025-03-24 | 2025-04-01 | Not Completed | HIGH | ● | g.giorgio.tm6@progresso.com | Realizzare un m... |

No comments found for this project.

Comments Section:

Giacomo Giorgio (g.giorgio.tm6@progresso.com)

Buonasera Giovanna, ho terminato la task "Implementazione autenticazione utente". Potresti dare un'occhiata? Grazie!

Comment is valid.

Send button icon.

Task List Page (Second View):

| Name | Start Date | Due Date | Completion Date | Priority | Status | Owner | Description |
|-------------------|------------|------------|-----------------|----------|---------------------------------------|-----------------------------|--------------------|
| Implementazion... | 2025-03-24 | 2025-04-01 | Not Completed | HIGH | ● | g.giorgio.tm6@progresso.com | Realizzare un m... |

No comments found for this project.

Comments Section (Second View):

Giacomo Giorgio (g.giorgio.tm6@progresso.com)

Buonasera Giovanna, ho terminato la task "Implementazione autenticazione utente". Potresti dare un'occhiata? Grazie!

Comment is valid.

Send button icon.



UNIVERSITÀ DEGLI STUDI
DI SALERNO

5 Sistema:

Il sistema salva il commento e lo aggiunge alla sezione dei commenti del progetto.

The screenshot shows a web-based project management application. At the top, there is a header with the title "Progresso" and a URL "http://localhost:4200/project/1%6/Progresso". Below the header, there is a table listing users:

| ID | First Name | Last Name | Username |
|----|------------|------------|--------------------------------|
| 84 | Giacomo | Giorgio | g.giorgio.tm6@progresso.com |
| 85 | Matteo | Mattei | m.mattei.tm7@progresso.com |
| 87 | Michele | Bartolomei | m.bartolomei.tm9@progresso.com |

Below the user list is a large blue circular progress bar with the text "0% complete" next to it. To the right of the progress bar is a vertical blue bar.

Further down the page is a "Task List" table:

| Name | Start Date | Due Date | Completion Date | Priority | Status | Owner | Description |
|-------------------|------------|------------|-----------------|----------|---------|-----------------------------|--------------------|
| Implementazion... | 2025-03-24 | 2025-04-01 | Not Completed | HIGH | Pending | g.giorgio.tm6@progresso.com | Realizzare un m... |

At the bottom of the page is a "Comments" section. It shows a comment from Giacomo Giorgio:

Giacomo Giorgio (g.giorgio.tm6@progresso.com)
Buonasera Giovanna, ho terminato la task "Implementazione autenticazione utente". Potresti dare un'occhiata? Grazie!

There are icons for reply, edit, and delete at the end of the comment.



5. Testing e validazione

5.1 Test unitari

Per effettuare i test unitari sono stati impiegati strumenti quali *Mockito*⁷ e *JUnit*⁸, adottando l'approccio del *Category Partition*. Tale approccio, che consiste nel suddividere le condizioni di input in categorie significative al fine di coprire sistematicamente tutti i comportamenti attesi (comprese le eccezioni sollevate dai singoli metodi), ha permesso di definire metodi di test specifici per ogni caso critico. La strategia di testing si è focalizzata esclusivamente sui metodi che implementano le funzionalità principali del sistema, in particolare quelli presenti nei *service*, appartenenti ai package del back-end illustrati in precedenza. Per agevolare la consultazione, di seguito sono elencati i *service* interessati e i rispettivi metodi testati:

- **AuthService**: *registerUser*.
- **CommentService**: *createComment*.
- **ProjectService**: *createProject*.
- **TaskService**: *createAndAssignTask*.
- **TeamService**: *createTeam*.

Ogni funzionalità testata è stata incapsulata in una classe di test dedicata, organizzata in un package speculare a quello della classe originale, e denominata aggiungendo il suffisso "Test" (ad esempio, *ProjectServiceTest*).

Per visionare tutti i test case individuati, si faccia riferimento all'**Appendice C**, in particolare: [Appendice C.1: Documentazione dei test case unitari](#)

Per ciascun metodo testato, la sezione dedicata nell'appendice offrirà una struttura organizzata che include: una tabella sintetica sul metodo analizzato, una tabella degli attributi significativi dei *DTO* utilizzati nei test case, e una spiegazione dettagliata dei test case implementati come metodi all'interno delle rispettive classi di test.

⁷ *Mockito* è un framework di mocking per Java open source che permette di creare oggetti finti (*mock*) per simulare il comportamento delle dipendenze in fase di test, facilitando l'isolamento delle unità di codice.

⁸ *JUnit* è un framework di test unitari per Java che offre annotazioni e strumenti per definire ed eseguire automaticamente i test, supportando così il rilevamento precoce di errori nel codice.



5.2 Test di interfaccia utente

Per effettuare i test di validazione dell’interfaccia utente, sono stati verificati i form presenti nell’interfaccia front-end delle funzionalità principali della piattaforma, utilizzando i *validator* di *Angular* per garantire la correttezza dei dati inviati dal front-end verso il back-end. I *validator* sono regole di validazione predefinite che controllano la correttezza dei dati immessi in un campo, assicurandosi che rispettino criteri specifici. Tra i principali *validator* utilizzati vi sono:

- **required**, per i campi obbligatori.
- **minLength** e **maxLength**, per i limiti di lunghezza dell’input.
- **pattern**, per la validazione del formato tramite espressioni regolari.

Questi *validator* impediscono l’inserimento di dati errati o malformati, fornendo all’utente un riscontro immediato tramite messaggi di errore visualizzati direttamente sotto i campi interessati. I test sono stati automatizzati utilizzando *Selenium IDE*⁹, che ha consentito di registrare e riprodurre scenari di interazione con l’interfaccia utente. In particolare, sono stati simulati comportamenti come clic, inserimento di dati nei form e invio del form, verificando la corretta visualizzazione dei messaggi di errore. Infine, è stato validato il comportamento positivo, in cui tutti i dati rispettano le regole definite dai *validator* e il form viene inviato con successo. Per una visione completa di tutti i test case effettuati, si rimanda all’**Appendice C**, in particolare:

[Appendice C.2: Documentazione dei test case dell’interfaccia utente](#)

⁹ *Selenium IDE* è uno strumento che consente di registrare le azioni effettuate su un browser e di riprodurlle automaticamente per testare l’interfaccia di un’applicazione web, assicurandosi che tutti gli elementi funzionino correttamente.



6. Conclusioni

6.1 Sintesi dei risultati ottenuti

I risultati del lavoro svolto evidenziano il successo dell'approccio modulare adottato nello sviluppo della piattaforma “Progresso”. In particolare, l’implementazione dei package e dei sottosistemi del back-end ha garantito una logica ben strutturata e scalabile, mentre l’integrazione dei *repository*, *service* e *controller* per le funzionalità offerte dalla piattaforma ha facilitato la manutenzione e l'estendibilità del sistema. Sul fronte del testing, l’adozione dei test unitari (realizzati con *Mockito* e *JUnit*) ha permesso di validare in modo sistematico ogni metodo critico, assicurando la robustezza del codice. Inoltre, il testing dell’interfaccia utente, automatizzato tramite *Selenium IDE*, ha dimostrato l’efficacia dei *validator* di *Angular* nel bloccare l’inserimento di dati errati e nel fornire un riscontro immediato all’utente. La documentazione completa dei test case, riportata in appendice, conferma che ogni aspetto del sistema è stato verificato con attenzione, senza privilegiare alcuni test case rispetto ad altri. In sintesi, il progetto “Progresso” ha raggiunto un elevato livello di affidabilità e qualità, ponendo solide basi per future implementazioni e miglioramenti, e confermando il suo potenziale nel facilitare l’organizzazione e la gestione di progetti software nel contesto di ambienti di lavoro moderni.

6.2 Possibili miglioramenti futuri

Tra i possibili sviluppi futuri, si evidenzia l’opportunità di implementare un’area notifiche dedicata, capace di gestire in modo più efficace gli avvisi e i messaggi di sistema, migliorando l’interazione e la tempestività della comunicazione con l’utente. Inoltre, un’altra miglioria da effettuare al sistema potrebbe essere l’integrazione di un servizio e-mail, incaricato di notificare gli utenti riguardo azioni significative, includendo, ad esempio, link per il recupero delle credenziali, che faciliterebbe la gestione degli accessi e rafforzerebbe la sicurezza complessiva dell’applicazione.



7. Appendici

7.1 Appendice A

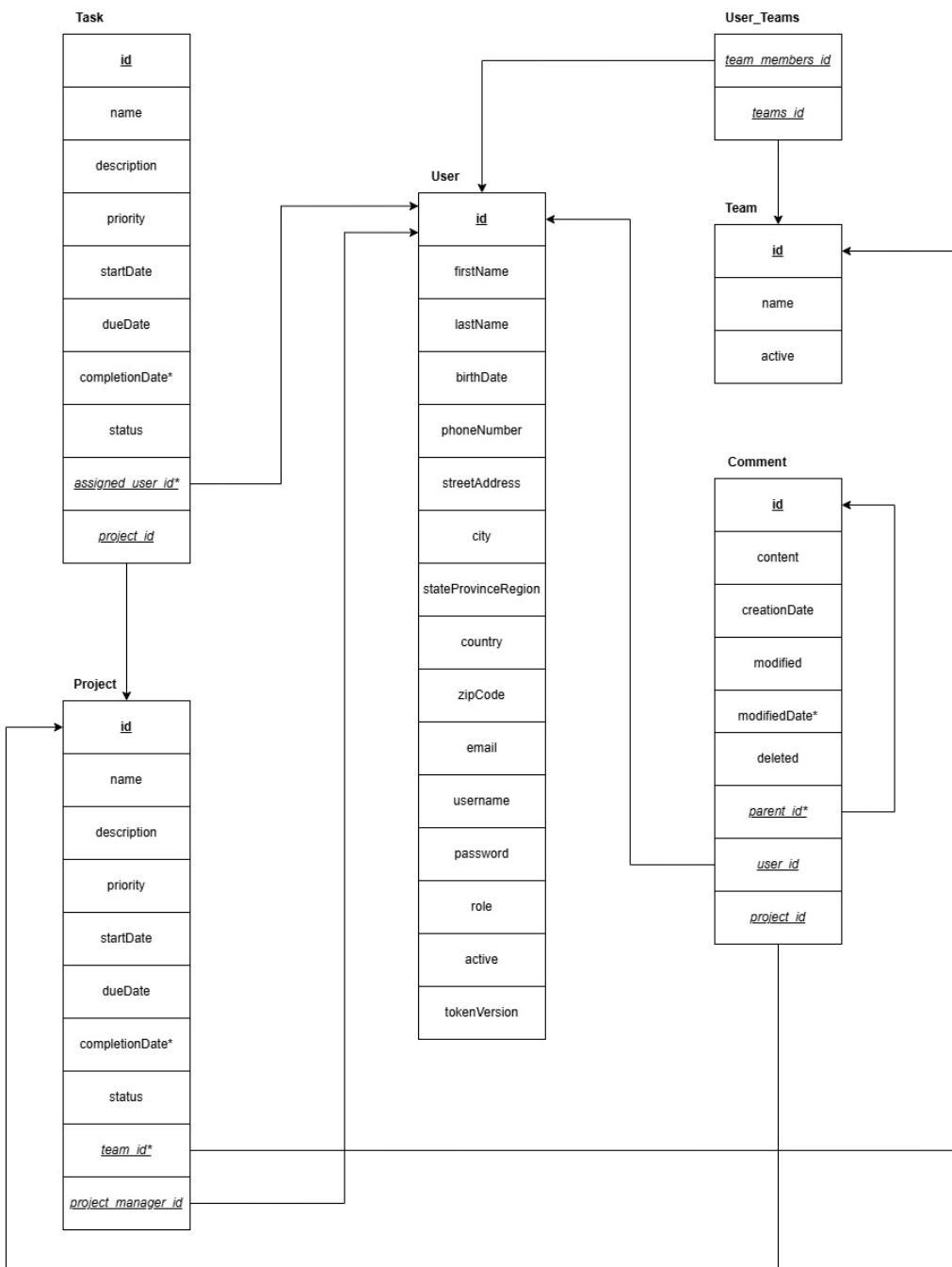
7.1.1 Appendice A.1: Descrizione delle relazioni

- L'entità *Project* è in una relazione *Possiede* di tipo uno-a-molti con cardinalità $1 - 0..*$ con l'entità *Comment* perché un progetto può possedere nessuno o molti commenti.
- L'entità *Project* è in una relazione *Contiene* di tipo uno-a-molti con cardinalità $1 - 0..*$ con l'entità *Task* in quanto un progetto può contenere nessuno o molti task.
- L'entità *Comment* è collegata a sé stessa mediante la relazione *Risponde a*. In questa relazione, un commento può rispondere a un solo commento (cardinalità $0..1$), mentre un commento può ricevere risposte da molti altri commenti (cardinalità $0..*$).
- L'entità *Team* è in una relazione *Lavora su* di tipo uno-a-molti con l'entità *Project*, con cardinalità $1 - 0..*$. Sebbene un team possa lavorare su diversi progetti nel tempo, il sistema impone che non sia assegnato a più progetti contemporaneamente, per garantire una gestione ottimale delle risorse.
- L'entità *User* è in una relazione:
 - *Crea e assegna* (*role = ADMIN*) oppure *Gestisce* (*role = PROJECTMANAGER*) con l'entità *Project* di tipo uno-a-molti, con cardinalità $1 - 0..*$, in quanto un admin può creare e assegnare nessuno o molti progetti a un project manager e quest'ultimo può gestire nessuno o molti progetti, con il vincolo che può gestire massimo cinque progetti attivi contemporaneamente.
 - *Scrive* con l'entità *Comment* di tipo uno-a-molti, con cardinalità $1 - 0..*$, perché un utente può scrivere nessuno o molti commenti.
 - *Crea e assegna* (*role = ADMIN or role = PROJECTMANAGER*) oppure *Lavora su* (*role = TEAMMEMBER*) con l'entità *Task* di tipo uno-a-molti, con cardinalità $1 - 0..*$, in quanto un admin o un project manager possono creare e assegnare nessuno o molti task ad un team member e quest'ultimo può lavorare su nessuno o molti task.
- Le entità *User* e *Team* si trovano in una relazione *Appartiene a* (partendo da *User* con *role = TEAMMEMBER*) e *Ha membri* (partendo da *Team*) di tipo molti-a-molti, con cardinalità $0..* - 0..*$, in quanto un utente può appartenere a nessuno o molti team e un team può avere nessuno o molti membri, con il vincolo che un team member può appartenere ad un solo team attivo alla volta.



UNIVERSITÀ DEGLI STUDI
DI SALERNO

7.1.2 Appendice A.2: Schema logico





UNIVERSITÀ DEGLI STUDI
DI SALERNO

7.1.3 Appendice A.3: Dizionario dei dati

7.1.3.1 User

| Nome entità | User | | |
|----------------------------|--|-------------------|------------------|
| Nome Campo | Tipo | Vincolo di chiave | Altri vincoli |
| id | bigint | primary key | auto increment |
| firstName | varchar (255) | | not null |
| lastName | varchar (255) | | not null |
| birthDate | date | | not null |
| phoneNumber | varchar (255) | | not null |
| streetAddress | varchar (255) | | not null |
| city | varchar (255) | | not null |
| stateProvinceRegion | varchar (255) | | not null |
| country | varchar (255) | | not null |
| zipCode | varchar (255) | | not null |
| email | varchar (255) | | unique, not null |
| username | varchar (255) | | unique, not null |
| password | varchar (255) | | not null |
| role | enum ('ADMIN', 'PROJECTMANAGER', 'TEAMMEMBER') | | not null |
| active | bit | | not null |
| tokenVersion | int | | not null |

7.1.3.2 Team

| Nome entità | Team | | |
|---------------|---------------|-------------------|----------------|
| Nome Campo | Tipo | Vincolo di chiave | Altri vincoli |
| id | bigint | primary key | auto increment |
| name | varchar (255) | | not null |
| active | bit | | not null |

7.1.3.3 User_Teams

| Nome entità | User_Teams | | |
|------------------------|------------|----------------------------------|---------------|
| Nome Campo | Tipo | Vincolo di chiave | Altri vincoli |
| team_members_id | bigint | foreign key references user (id) | not null |
| teams_id | bigint | foreign key references team (id) | not null |



7.1.3.4 Task

| Nome entità | Task | | |
|-------------------------|---|-------------------------------------|----------------|
| Descrizione | Contiene dati relativi ad un task. | | |
| Nome Campo | Tipo | Vincolo di chiave | Altri vincoli |
| id | bigint | primary key | auto_increment |
| name | varchar (255) | | not null |
| description | varchar (255) | | not null |
| priority | enum ('HIGH', 'LOW', 'MEDIUM') | | not null |
| startDate | date | | not null |
| dueDate | date | | not null |
| completionDate | date | | null |
| status | enum ('CANCELLED', 'COMPLETED', 'IN_PROGRESS', 'NOT_STARTED') | | not null |
| assigned_user_id | bigint | foreign key references user (id) | null |
| project_id | bigint | foreign key references project (id) | not null |

7.1.3.5 Project

| Nome entità | Project | | |
|---------------------------|---|----------------------------------|----------------|
| Descrizione | Contiene dati relativi ad un progetto. | | |
| Nome Campo | Tipo | Vincolo di chiave | Altri vincoli |
| id | bigint | primary key | auto_increment |
| name | varchar (255) | | not null |
| description | varchar (255) | | not null |
| priority | enum ('HIGH', 'LOW', 'MEDIUM') | | not null |
| startDate | date | | not null |
| dueDate | date | | not null |
| completionDate | date | | null |
| status | enum ('CANCELLED', 'COMPLETED', 'IN_PROGRESS', 'NOT_STARTED') | | not null |
| project_manager_id | bigint | foreign key references user (id) | not null |



UNIVERSITÀ DEGLI STUDI
DI SALERNO

| | | | |
|----------------|--------|----------------------------------|------|
| team_id | bigint | foreign key references team (id) | null |
|----------------|--------|----------------------------------|------|

7.1.3.6 *Comment*

| Nome entità | | | |
|---------------------|---------------|-------------------------------------|----------------|
| Descrizione | | | |
| Nome Campo | Tipo | Vincolo di chiave | Altri vincoli |
| id | bigint | primary key | auto_increment |
| content | varchar (500) | | not null |
| creationDate | date | | not null |
| modified | bit | | not null |
| modifiedDate | date | | null |
| deleted | bit | | not null |
| parent_id | bigint | foreign key references comment (id) | null |
| user_id | bigint | foreign key references user (id) | not null |
| project_id | bigint | foreign key references project (id) | not null |



7.2 Appendice B

7.2.1 Appendice B.1: Descrizione approfondita dei package

7.2.1.1 com.progresso.backend.authsubsystem

| AuthService | | | |
|------------------------------|----------------------|---|--|
| Metodo | Tipo di ritorno | Parametri | Descrizione |
| generateUsername | String | String <i>firstName</i> , String <i>lastName</i> , Role <i>role</i> | Genera un username univoco per un nuovo utente basato sul nome, cognome e ruolo. Il formato dell'username è: <primaLetteraNome> . <cognome> . <inizialiRuolo> . <numeroProgressivo> @progresso.com. Il numero progressivo viene determinato contando quanti utenti con lo stesso ruolo sono già presenti nel database. |
| incrementTokenVersion | Integer | Integer <i>version</i> | Incrementa il valore della versione del token. Se il valore attuale ha raggiunto Integer.MAX_VALUE, viene resettato a 0 per evitare overflow. Altrimenti, viene semplicemente incrementato di 1. |
| registerUser | UserResponseDto | UserRegistrationDto <i>userRegistrationDto</i> | Registra un nuovo utente verificando l'unicità dell'e-mail, generando username e password, e salvando l'utente. |
| authenticateUser | UserLoginResponseDto | UserLoginDto <i>loginDto</i> | Autentica l'utente verificando le credenziali e, in caso di successo, genera un token JWT. |
| logout | UserResponseDto | | Effettua il logout dell'utente aggiornando il tokenVersion per invalidare i token attivi. |
| deactivateUser | UserResponseDto | Long <i>userId</i> | Disattiva l'utente, aggiornando il tokenVersion e gestendo correttamente le assegnazioni dei task in corso. |
| activateUser | UserResponseDto | Long <i>userId</i> | Attiva un utente precedentemente disattivato, salvando lo stato aggiornato. |

| AuthController /api/auth | | | | |
|-----------------------------|-----------|-------------|---|---------------------------|
| Metodo | Endpoint | Http Method | Parametri | Sicurezza / PreAuthorize |
| login | /login | POST | RequestBody: UserLoginDto <i>userLoginDto</i> (validato) | |
| register | /register | POST | RequestBody: UserRegistrationDto | hasAuthority ('ADMIN') |



| | | | | |
|-----------------------|--------------------------|------|--|---------------------------|
| | | | <i>userRegistrationDto</i> (validato) | |
| logout | /logout | POST | | |
| deactivateUser | /{userId} /deactivate | PUT | PathVariable: Long <i>userId</i> | hasAuthority ('ADMIN') |
| activateUser | /{userId} /activate | PUT | PathVariable: Long <i>userId</i> | hasAuthority ('ADMIN') |

7.2.1.2 com.progresso.backend.commentmanagement

| CommentRepository | | |
|--------------------------|-----------------|---|
| Metodo | Tipo di ritorno | Parametri |
| findByIdProjectId | Page<Comment> | Long <i>projectId</i> , Pageable <i>pageable</i> |

| CommentService | | | |
|-----------------------------------|------------------|---|---|
| Metodo | Tipo di ritorno | Parametri | Descrizione |
| convertToDto | CommentDto | Comment <i>comment</i> | Converte un'entità <i>Comment</i> in un DTO (<i>CommentDto</i>), trasferendo tutti i campi rilevanti, inclusi quelli relativi al commento padre, se presente. |
| isUserInProject | Boolean | User <i>user</i> , Project <i>project</i> | Verifica se l'utente è associato al progetto, controllando se l'utente è presente nel team assegnato al progetto o se è il project manager. |
| isManagerOrMemberOfProject | Boolean | Long <i>projectId</i> , String <i>username</i> | Controlla se l'utente, identificato da <i>username</i> , è il project manager o un membro del team associato al progetto identificato da <i>projectId</i> . |
| isCommentOwner | Boolean | Long <i>commentId</i> , String <i>username</i> | Verifica se l'utente (identificato da <i>username</i>) è il proprietario del commento identificato da <i>commentId</i> . |
| findByIdProjectId | Page<CommentDto> | Long <i>projectId</i> , Pageable <i>pageable</i> | Recupera in modalità paginata tutti i commenti associati al progetto identificato da <i>projectId</i> , convertendoli in <i>CommentDto</i> . |
| createComment | CommentDto | CommentDto <i>commentDto</i> | Crea un nuovo commento per un progetto: valida l'esistenza e lo stato attivo dell'utente e del progetto, gestisce eventuali risposte (commento padre) e salva il commento, restituendo il relativo DTO. |
| updateComment | CommentDto | Long <i>commentId</i> , String <i>newContent</i> | Aggiorna il contenuto di un commento esistente, impostando il flag di modifica e aggiornando la data di modifica, a condizione che il commento |



| | | | |
|----------------------|------------|-----------------------|---|
| | | | non sia già eliminato e il progetto sia attivo. |
| deleteComment | CommentDto | Long <i>commentId</i> | "Elimina" un commento segnalandolo come cancellato (impostando un messaggio standard e il flag di eliminazione), se il progetto è attivo e il commento non è già stato eliminato. |

| CommentController <i>/api/comments</i> | | | | |
|---|-------------------------------|-------------|---|---|
| Metodo | Endpoint | Http Method | Parametri | Sicurezza / PreAuthorize |
| findByIdProject | /project/{projectId}/comments | GET | PathVariable: Long <i>projectId</i> Pageable <i>pageable</i> | hasAuthority('ADMIN') or @commentService.isManagerOrMemberOfProject(#projectId, authentication.name) |
| createComment | | POST | RequestBody: CommentDto <i>commentDto</i> (validato) | hasAuthority('ADMIN') or @commentService.isManagerOrMemberOfProject(#commentDto.projectId, authentication.name) |
| updateComment | /{id} | PUT | PathVariable: Long <i>id</i> RequestParam: String <i>newContent</i> (non vuoto, massimo 500 caratteri) | @commentService.isCommentOwner(#id, authentication.name) |
| deleteComment | /{id} | DELETE | PathVariable: Long <i>id</i> | hasAuthority('ADMIN') or @commentService.isCommentOwner(#id, authentication.name) |

7.2.1.3 com.progresso.backend.projectmanagement

| ProjectRepository | | |
|---|-----------------|---|
| Metodo | Tipo di ritorno | Parametri |
| findAllWithFilters | Page<Project> | Status <i>status</i> , Priority <i>priority</i> , String <i>name</i> , Pageable <i>pageable</i> |
| findByProjectManagerUsernameAndFilters | | String <i>managerUsername</i> , Status <i>status</i> , Priority <i>priority</i> , String <i>name</i> , Pageable <i>pageable</i> |
| findActiveProjectsByTeamMemberUsername | Page<Project> | String <i>teamMemberUsername</i> , Pageable <i>pageable</i> |



| | | |
|--|---------|---|
| countByProjectManagerAndStatusNotIn | long | User projectManager, List<Status> excludedStatus |
| countByTeamAndStatusNotIn | long | Team team, List<Status> excludedStatus |
| existsByNameIgnoreCase | boolean | String name |

| ProjectService | | | |
|---|------------------|---|---|
| Metodo | Tipo di ritorno | Parametri | Descrizione |
| convertToDto | ProjectDto | Project <i>project</i> | Converte un'entità Project in un <i>ProjectDto</i> , trasferendo tutte le proprietà rilevanti (inclusi campi calcolati come la percentuale di completamento) e i dati correlati a project manager, team, task e commenti. |
| updateProjectPriority | Priority | Project <i>project</i> | Calcola e aggiorna la priorità di un progetto in base alla data corrente, alla data di inizio e alla data di scadenza. Se il progetto è completato o cancellato, mantiene la priorità esistente. |
| getProjectsDto | Page<ProjectDto> | Page<Project> <i>projectsPage</i> | Aggiorna la priorità dei progetti presenti nella pagina e li converte in DTO. |
| isTeamMemberOfProject | boolean | Long <i>projectId</i> , String <i>username</i> | Verifica se l'utente identificato dall' <i>username</i> è membro del team associato al progetto specificato. |
| isManagerOfProject | boolean | Long <i>projectId</i> , String <i>username</i> | Controlla se l'utente, identificato dall' <i>username</i> , è il project manager del progetto indicato. |
| getProjectCompletionPercentage | long | Long <i>projectId</i> | Calcola la percentuale di completamento di un progetto in base al numero di task completati rispetto al totale dei task (escludendo quelli cancellati). |
| findAllProjectsWithFilters | Page<ProjectDto> | String <i>status</i> , String <i>priority</i> , String <i>name</i> , Pageable <i>pageable</i> | Recupera in modalità paginata i progetti che rispettano i filtri opzionali su <i>status</i> , <i>priority</i> e <i>name</i> , convertendoli in DTO. |
| findProjectsByProjectManagerUsernameAndFilters | Page<ProjectDto> | String <i>managerUsername</i> , String <i>status</i> , String <i>priority</i> , String <i>name</i> , Pageable <i>pageable</i> | Recupera in modalità paginata i progetti gestiti da un project manager specifico, applicando filtri su <i>status</i> , <i>priority</i> e <i>name</i> , convertendoli in DTO. |
| findActiveProjectsByTeamMemberUsername | Page<ProjectDto> | String <i>teamMemberUsername</i> , Pageable <i>pageable</i> | Recupera in modalità paginata i progetti attivi (con status IN_PROGRESS o NOT_STARTED) in cui è coinvolto un team member specificato. |
| findProjectById | ProjectDto | Long <i>id</i> | Recupera un progetto dato il suo ID, aggiornando la |



| | | | |
|------------------------------|------------|---|--|
| | | | priorità e salvando eventuali modifiche prima di convertirlo in DTO, se non trovato. |
| createProject | ProjectDto | ProjectDto <i>projectDto</i> | Crea un nuovo progetto dopo aver validato le date (start e due), verificato la validità del project manager (ruolo, stato attivo, numero massimo di progetti attivi) e generato un nome univoco per il progetto. |
| updateProject | ProjectDto | Long <i>projectId</i> , ProjectDto <i>projectDto</i> | Aggiorna un progetto esistente dopo aver eseguito varie validazioni (date, priorità, task, team, commenti) e controllato lo stato del progetto. |
| updateProjectManager | ProjectDto | Long <i>projectId</i> , Long <i>projectManagerId</i> | Aggiorna il project manager di un progetto, verificando che il nuovo manager sia attivo, abbia il ruolo corretto e non superi il limite di progetti attivi. |
| assignTeamToProject | ProjectDto | Long <i>projectId</i> , Long <i>teamId</i> | Assegna un team ad un progetto, dopo aver verificato che il progetto non sia già assegnato, che il team sia attivo, che il progetto non sia completato/cancellato e che il team non sia già impegnato in un altro progetto attivo. |
| reassignTeamToProject | ProjectDto | Long <i>projectId</i> , Long <i>teamId</i> | Riassegna un team ad un progetto, aggiornando le assegnazioni dei task per il team corrente, rimuovendo il progetto dal team attuale e associandolo al nuovo team, dopo aver effettuato le necessarie validazioni. |
| completeProject | ProjectDto | Long <i>projectId</i> | Completa un progetto verificando che non vi siano task in corso; se tutti i task sono completati, aggiorna lo stato a COMPLETED, imposta la data di completamento e regola la priorità a LOW. |
| removeProject | ProjectDto | Long <i>projectId</i> | "Rimuove" un progetto impostandolo come CANCELLED, rimuovendo le associazioni dei task collegati al progetto e aggiornando il relativo stato. |



| ProjectController <i>/api/projects</i> | | | | |
|--|--|-------------|--|---|
| Metodo | Endpoint | Http Method | Parametri | Sicurezza / PreAuthorize |
| getProjectCompletionPercentage | <code>/{{projectId}}/completion</code> | GET | PathVariable: Long <i>projectId</i> | has Authority (ADMIN) or (hasAuthority (PROJECTMANAGER) and @projectService .isManagerOfProject (#projectId, authentication.name)) or (hasAuthority (TEAMMEMBER) and @projectService .isTeamMemberOfProject (#projectId, authentication.name)) |
| getAllProjectsByFilters | | GET | RequestParam (required=false): String <i>status</i> , String <i>priority</i> , String <i>name</i> , Pageable <i>pageable</i> | hasAuthority (ADMIN) |
| getProjectsByManagerAndFilters | <code>/manager/{{managerUsername}}</code> | GET | PathVariable: String <i>managerUsername</i> , RequestParam (required=false): String <i>status</i> , String <i>priority</i> , String <i>name</i> Pageable <i>pageable</i> | hasAuthority (ADMIN) or (hasAuthority (PROJECTMANAGER) and #managerUsername == authentication.name)) |
| getActiveProjectsByTeamMemberUsername | <code>/active/teamMember/{{teamMemberUsername}}</code> | GET | PathVariable: String <i>teamMemberUsername</i> Pageable <i>pageable</i> | hasAuthority (ADMIN) or (hasAuthority (TEAMMEMBER) and #teamMemberUsername == authentication.name)) |
| getProjectById | <code>/{{id}}</code> | GET | PathVariable: Long <i>id</i> | hasAuthority (ADMIN) or (hasAuthority (PROJECTMANAGER) and @projectService .isManagerOfProject (#id, authentication.name)) or |



| | | | | |
|------------------------------|--|------|---|---|
| | | | | (hasAuthority('TEAMMEMBER') and @projectService.isTeamMemberOfProject(#id, authentication.name)) |
| createProject | | POST | RequestBody: ProjectDto <i>projectDto</i> (validato) | hasAuthority('ADMIN') |
| updateProject | /{projectId} | PUT | PathVariable: Long <i>projectId</i> RequestBody: ProjectDto <i>projectDto</i> (validato) | hasAuthority('ADMIN') or (hasAuthority('PROJECTMANAGER') and @projectService.isManagerOfProject(#projectId, authentication.name)) |
| removeProject | /{projectId} /remove | PUT | PathVariable: Long <i>projectId</i> | hasAuthority('ADMIN') or (hasAuthority('PROJECTMANAGER') and @projectService.isManagerOfProject(#projectId, authentication.name)) |
| updateProjectManager | /{projectId} /update-manager /{projectManagerId} | PUT | PathVariable: Long <i>projectId</i> , Long <i>projectManagerId</i> | hasAuthority('ADMIN') |
| assignTeamToProject | /{projectId} /assign-team /{teamId} | PUT | PathVariable: Long <i>projectId</i> , Long <i>teamId</i> | hasAuthority('ADMIN') or (hasAuthority('PROJECTMANAGER') and @projectService.isManagerOfProject(#projectId, authentication.name)) |
| reassignTeamToProject | /{projectId} /reassign-team /{teamId} | PUT | PathVariable: Long <i>projectId</i> , Long <i>teamId</i> | hasAuthority('ADMIN') or (hasAuthority('PROJECTMANAGER') and @projectService.isManagerOfProject(#projectId, authentication.name)) |
| completeProject | /{projectId} /complete | PUT | PathVariable: Long <i>projectId</i> | hasAuthority('ADMIN') or (hasAuthority('PROJECTMANAGER')) |



| | | | | |
|--|--|--|--|--|
| | | | | ('PROJECTMANAGER') and @projectService .isManagerOfProject (#projectId, authentication.name)) |
|--|--|--|--|--|

7.2.1.4 com.progresso.backend.taskmanagement

| TaskRepository | | | |
|---|-----------------|---|--|
| Metodo | Tipo di ritorno | Parametri | |
| findByProjectIdAndStatus AndPriority | Page<Task> | Long <i>projectId</i> , Status <i>status</i> , Priority <i>priority</i> , Pageable <i>pageable</i> | |
| existsByProjectIdAndName | boolean | Long <i>projectId</i> , String <i>name</i> | |

| TaskService | | | |
|---|-----------------|---|--|
| Metodo | Tipo di ritorno | Parametri | Descrizione |
| convertToDto | TaskDto | Task <i>task</i> | Converte un'entità Task in un DTO (<i>TaskDto</i>), trasferendo le proprietà rilevanti (come nome, descrizione, date, priorità, status, e dati relativi a progetto e utente assegnato). |
| getProjectIdByTaskId | Long | Long <i>taskId</i> | Recupera l'ID del progetto associato al task identificato da <i>taskId</i> . |
| findByProjectIdAndStatus AndPriority | Page<TaskDto> | Long <i>projectId</i> , String <i>status</i> , String <i>priority</i> , Pageable <i>pageable</i> | Recupera in modalità paginata i task associati al progetto identificato da <i>projectId</i> , applicando filtri opzionali su <i>status</i> e <i>priority</i> . |
| createAndAssignTask | TaskDto | TaskDto <i>taskDto</i> , Long <i>userId</i> | Crea un nuovo task, assegnandolo a un utente specificato. Il metodo esegue varie validazioni su date, stato del progetto e del team, e il ruolo dell'utente prima di salvare il task e aggiornare le assegnazioni. |
| updateTask | TaskDto | Long <i>taskId</i> , TaskDto <i>taskDto</i> | Aggiorna un task esistente. Effettua controlli per impedire modifiche a campi non ammessi (ad es. completion date, status, utente assegnato) e aggiorna il nome se necessario, garantendo la coerenza con le regole di business. |
| reassignTaskToTeamMember | TaskDto | Long <i>taskId</i> , Long <i>userId</i> | Riassegna un task a un nuovo team member. Il metodo verifica che il task non sia completato o cancellato, che il nuovo utente sia attivo, faccia parte del team e abbia il ruolo corretto, aggiornando |



| | | | |
|------------------------------|---------|---|--|
| | | | di conseguenza le assegnazioni. |
| completeTask | TaskDto | Long <i>taskId</i> | Segna un task come completato, aggiornando lo status a COMPLETED e impostando la data di completamento. Prima di completare il task, vengono verificate le condizioni relative alla data e allo stato del task e del progetto. |
| removeTaskFromProject | TaskDto | Long <i>projectId</i> , Long <i>taskId</i> | Cancella un task associato a un progetto, impostando lo status a CANCELLED e aggiornando la priorità a LOW. Il metodo esegue controlli sullo stato del progetto e del task per garantire che l'operazione sia consentita. |

| TaskController <i>/api/tasks</i> | | | | |
|--------------------------------------|--|-------------|---|--|
| Metodo | Endpoint | Http Method | Parametri | Sicurezza / PreAuthorize |
| getTasksByProjectIdAndFilters | /project/{projectId} | GET | PathVariable: Long <i>projectId</i> RequestParam (required=false): String <i>status</i> , String <i>priority</i> Pageable <i>pageable</i> | has Authority ('ADMIN') or (hasAuthority ('PROJECTMANAGER') and @projectService .isManagerOfProject (#projectId, authentication.name)) or (hasAuthority ('TEAMMEMBER') and @projectService .isTeamMemberOfProject (#projectId, authentication.name)) |
| createAndAssignTask | | POST | RequestBody: TaskDto <i>taskDto</i> (validato) RequestParam: Long <i>userId</i> | hasAuthority ('ADMIN') or (hasAuthority ('PROJECTMANAGER') and @projectService .isManagerOfProject (#taskDto.projectId, authentication.name)) |
| reassignTaskToUser | { <i>taskId</i> } /reassign /{ <i>userId</i> } | POST | PathVariable: Long <i>taskId</i> , Long <i>userId</i> | hasAuthority ('ADMIN') or (hasAuthority ('PROJECTMANAGER') and |



| | | | | |
|-----------------------------------|--|--------|---|--|
| | | | | @projectService .isManagerOfProject (@taskService .getProjectIdByTaskI d (#taskId), authentication.name)) |
| updateTask | /{taskId} | PUT | PathVariable: Long <i>taskId</i> RequestBody: TaskDto <i>taskDto</i> (validato) | hasAuthority ('ADMIN') or (hasAuthority ('PROJECTMANAG ER') and @projectService .isManagerOfProject (#taskDto.projectId, authentication.name)) |
| completeTask | /{taskId} /complete | PATCH | PathVariable: Long <i>taskId</i> | hasAuthority ('ADMIN') or (hasAuthority ('PROJECTMANAG ER') and @projectService .isManagerOfProject (@taskService .getProjectIdByTaskI d (#taskId), authentication.name)) |
| removeTaskFromPr oject | /project /{projectId}/task /{taskId} | DELETE | PathVariable: Long <i>projectId</i> , Long <i>taskId</i> | hasAuthority ('ADMIN') or (hasAuthority ('PROJECTMANAG ER') and @projectService .isManagerOfProject (#projectId, authentication.name)) |

7.2.1.5 com.progresso.backend.teammanagement

| TeamRepository | | |
|---------------------------------------|-----------------|--|
| Metodo | Tipo di ritorno | Parametri |
| findAllTeamsWithFilters | Page<Team> | Boolean <i>active</i> , String <i>searchTerm</i> , Pageable <i>pageable</i> |
| existsByNameIgnoreCaseAndIdNot | Boolean | String <i>name</i> , Long <i>id</i> |
| existsByNameIgnoreCase | Boolean | String <i>name</i> |
| findTeamsWithoutActiveProjects | Page<Team> | List<Status> <i>activeStatuses</i> , String <i>searchTerm</i> , Pageable <i>pageable</i> |



| TeamService | | | |
|---------------------------------------|-----------------|--|--|
| Metodo | Tipo di ritorno | Parametri | Descrizione |
| convertToDto | TeamDto | Team <i>team</i> | Converte un'entità Team in un DTO (<i>TeamDto</i>), trasferendo i campi essenziali (ID, nome, stato, elenco degli ID dei membri e dei progetti associati). |
| isProjectManagerOfTeamProjects | boolean | Long <i>teamId</i> , String <i>username</i> | Verifica se l'utente, identificato da <i>username</i> , è project manager di uno o più progetti attivi associati al team con ID <i>teamId</i> . |
| isTeamMemberOfTeam | boolean | Long <i>teamId</i> , String <i>username</i> | Controlla se l'utente, identificato da <i>username</i> , è membro del team specificato dal <i>teamId</i> . |
| getTeamById | TeamDto | Long <i>id</i> | Recupera il team con l'ID specificato, convertendolo in TeamDto. |
| getAllTeamsWithFilters | Page<TeamDto> | Pageable <i>pageable</i> , String <i>searchTerm</i> | Recupera in modalità paginata i team attivi che non sono associati a progetti attivi, filtrando opzionalmente per nome. |
| getTeamsWithoutActiveProjects | Page<TeamDto> | Pageable <i>pageable</i> , String <i>searchTerm</i> | Recupera in modalità paginata i team attivi che non sono associati a progetti attivi, filtrando opzionalmente per nome. |
| createTeam | TeamDto | String <i>teamName</i> | Crea un nuovo team verificando che il nome sia valido e univoco (aggiungendo un suffisso se necessario), imposta lo stato attivo e restituisce il DTO del team creato. |
| updateTeam | TeamDto | Long <i>id</i> , String <i>newName</i> | Aggiorna il nome del team con l'ID specificato, dopo aver verificato che il nuovo nome sia valido e non sia già in uso da altri team; salva le modifiche e restituisce il DTO aggiornato. |
| addMembersToTeam | TeamDto | Long <i>teamId</i> , List<Long> <i>userIds</i> | Aggiunge gli utenti, identificati dagli ID in <i>userIds</i> , al team specificato; verifica che gli utenti esistano, siano attivi, abbiano il ruolo corretto e non siano già membri di un altro team attivo; aggiorna le associazioni bidirezionali e restituisce il DTO aggiornato del team. |
| removeMembersFromTeam | TeamDto | Long <i>teamId</i> , List<Long> <i>userIds</i> | Rimuove dal team gli utenti indicati in <i>userIds</i> dopo aver verificato che siano membri attivi; aggiorna le assegnazioni dei task per questi utenti e restituisce il DTO del team aggiornato. |



| | | | |
|-------------------|---------|--------------------|---|
| deleteTeam | TeamDto | Long <i>teamId</i> | Disattiva il team con l'ID specificato, verificando che il team non abbia progetti attivi; salva lo stato aggiornato e restituisce il DTO del team disattivato. |
|-------------------|---------|--------------------|---|

| TeamController <i>/api/teams</i> | | | | |
|-------------------------------------|---------------------------------|-------------|--|---|
| Metodo | Endpoint | Http Method | Parametri | Sicurezza / PreAuthorize |
| getAllTeams | | GET | RequestParam (required = false): Boolean active, String searchTerm Pageable pageable | hasAuthority ('ADMIN') |
| getTeamById | <i>{teamId}</i> | GET | PathVariable: Long <i>teamId</i> | |
| getAvailableTeams | /availableTeams | GET | RequestParam (required = false): String <i>searchTerm</i> Pageable pageable | hasAuthority ('ADMIN') or hasAuthority ('PROJECTMANAGER') |
| createTeam | | POST | RequestParam: String <i>teamName</i> (non vuoto, massimo 100 caratteri) | hasAuthority ('ADMIN') or hasAuthority ('PROJECTMANAGER') |
| updateTeam | <i>{teamId}</i> | PUT | PathVariable: Long <i>teamId</i> RequestParam: String <i>newName</i> (non vuoto, massimo 100 caratteri) PathVariable: Long <i>teamId</i> RequestBody: List<Long> <i>userIds</i> | hasAuthority ('ADMIN') or (hasAuthority ('PROJECTMANAGER') and @teamService.isProjectManagerOf TeamProjects (#teamId, authentication.name)) |
| addMembersToTeam | <i>{teamId}</i> /members | POST | PathVariable: Long <i>teamId</i> RequestBody: List<Long> <i>userIds</i> | hasAuthority ('ADMIN') or (hasAuthority ('PROJECTMANAGER') and @teamService.isProjectManagerOf TeamProjects (#teamId, authentication.name)) |
| removeMembersFromTeam | <i>{teamId}</i> /remove-members | POST | PathVariable: Long <i>teamId</i> RequestBody: List<Long> <i>userId</i> | hasAuthority ('ADMIN') or (hasAuthority ('PROJECTMANAGER') and @teamService |



| | | | | |
|-------------------|-------------|--------|-------------------------------------|---|
| | | | | .isProjectManagerOf TeamProjects (#teamId, authentication.name)) |
| deleteTeam | /{{teamId}} | DELETE | PathVariable: Long <i>teamId</i> | hasAuthority ('ADMIN') |

7.2.1.6 com.progresso.backend.usermanagement

| UserRepository | | | |
|-------------------------------------|-----------------|---|--|
| Metodo | Tipo di ritorno | Parametri | |
| findByUsername | Optional<User> | String <i>username</i> | |
| findByEmail | Optional<User> | String <i>email</i> | |
| countByRole | int | Role <i>role</i> | |
| findAllWithFilters | Page<User> | Role <i>role</i> , Boolean <i>active</i> , String <i>searchTerm</i> , Pageable <i>pageable</i> | |
| findAvailableProjectManagers | Page<User> | String <i>searchTerm</i> , Pageable <i>pageable</i> | |
| findAvailableTeamMembers | Page<User> | String <i>searchTerm</i> , Pageable <i>pageable</i> | |
| findUsersByTeamId | Page<User> | Long <i>teamId</i> , String <i>searchTerm</i> , Pageable <i>pageable</i> | |

| UserService | | | |
|------------------------------------|-----------------------|---|--|
| Metodo | Tipo di ritorno | Parametri | Descrizione |
| convertToDtoCommon | UserResponseDto | User <i>user</i> | Converte un oggetto User in <i>UserResponseDto</i> con informazioni comuni. |
| convertToDtoToken | UserLoginResponseDto | User <i>user</i> , String <i>token</i> | Converte un oggetto User in <i>UserLoginResponseDto</i> includendo un token. |
| convertToDto | UserResponseDto | User <i>user</i> | Converte un User in <i>UserResponseDto</i> utilizzando <i>convertToDtoCommon</i> . |
| getUserById | UserResponseDto | Long <i>id</i> | Recupera un utente per ID e lo converte in <i>UserResponseDto</i> . |
| getAllUsersWithFilters | Page<UserResponseDto> | Pageable <i>pageable</i> , String <i>searchTerm</i> , String <i>role</i> , Boolean <i>active</i> | Recupera utenti filtrati per ruolo, stato e termine di ricerca. |
| getAvailableProjectManagers | Page<UserResponseDto> | Pageable <i>pageable</i> , String <i>searchTerm</i> | Recupera i Project Manager disponibili con filtro di ricerca. |
| getAvailableTeamMembers | Page<UserResponseDto> | Pageable <i>pageable</i> , String <i>searchTerm</i> | Recupera i team member disponibili con filtro di ricerca. |
| getUsersByTeamId | Page<UserResponseDto> | Long <i>teamId</i> , Pageable <i>pageable</i> , String <i>searchTerm</i> | Recupera gli utenti appartenenti a un team specifico. |

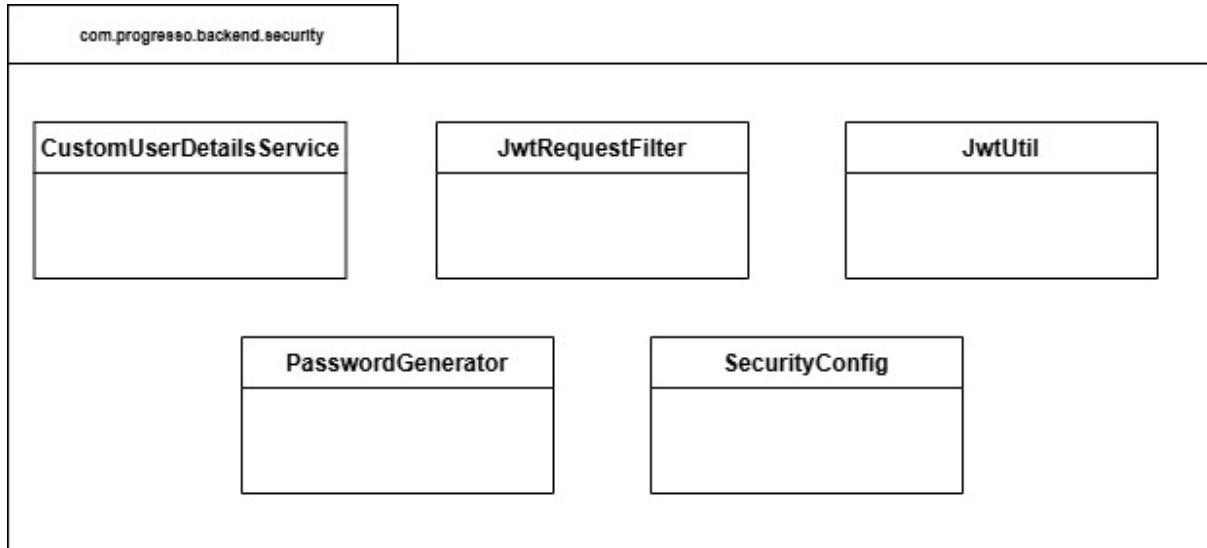


UNIVERSITÀ DEGLI STUDI
DI SALERNO

| UserController <i>/api/users</i> | | | | |
|-------------------------------------|---|-------------|--|--|
| Metodo | Endpoint | Http Method | Parametri | Sicurezza / PreAuthorize |
| getUserById | <code>/{{userId}}</code> | GET | PathVariable: Long <i>userId</i> | |
| getAllUsers | | GET | Pageable <i>pageable</i> RequestParam (required = false): String <i>searchTerm</i> , String <i>role</i> , Boolean <i>active</i> | hasAuthority ('ADMIN') |
| getAvailablePms | <code>/available-project-managers</code> | GET | Pageable <i>pageable</i> RequestParam (required = false): String <i>searchTerm</i> | hasAuthority ('ADMIN') |
| getAvailableTeamMembers | <code>/available-team-members</code> | GET | RequestParam (required = false): String <i>searchTerm</i> Pageable <i>pageable</i> | hasAuthority ('ADMIN') or hasAuthority ('PROJECTMANAGER') |
| getUsersByTeamId | <code>/teams/{{teamId}}/team-members</code> | GET | PathVariable: Long <i>teamId</i> RequestParam (required = false): String <i>searchTerm</i> Pageable <i>pageable</i> | hasAuthority ('ADMIN') or hasAuthority ('PROJECTMANAGER') or (hasAuthority ('TEAMMEMBER') and @teamService .isTeamMemberOfTeam (#teamId, authentication.name)) |



7.2.1.7 com.progresso.backend.security



| CustomUserDetailsService | | | |
|---------------------------|-----------------|------------------------|--|
| Metodo | Tipo di ritorno | Parametri | Descrizione |
| loadUserByUsername | UserDetails | String <i>username</i> | Carica un utente tramite il suo nome utente. |

| JwtRequestFilter | | | |
|-------------------------|-----------------|--|---|
| Metodo | Tipo di ritorno | Parametri | Descrizione |
| doFilterInternal | void | HttpServletRequest <i>request</i> , HttpServletResponse <i>response</i> , FilterChain <i>chain</i> | Filtrà le richieste HTTP, estrae e convalida il token JWT; quindi, imposta l'autenticazione nel contesto di sicurezza se il token è valido. |

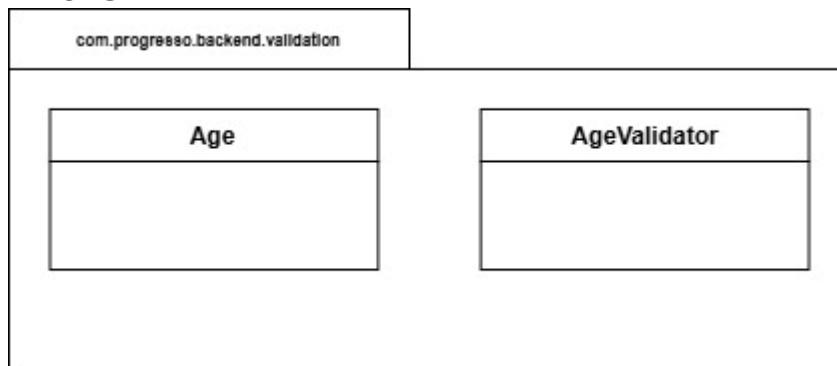
| JwtUtil | | | |
|--------------------------|-----------------|---|--|
| Metodo | Tipo di ritorno | Parametri | Descrizione |
| extractUsername | String | String <i>token</i> | Estrae il nome utente dal token JWT. |
| extractExpiration | Date | String <i>token</i> | Estrae la data di scadenza del token JWT. |
| extractClaim | <T> | String <i>token</i> , Function<Claims, T> <i>claimsResolver</i> | Estrae un claim specifico dal token JWT usando il resolver passato come parametro. |
| extractAllClaims | Claims | String <i>token</i> | Estrae tutti i claims dal token JWT. |
| isTokenExpired | Boolean | String <i>token</i> | Verifica se il token JWT è scaduto. |
| generateToken | String | User <i>user</i> | Genera un token JWT a partire dalle informazioni dell'utente. |
| createToken | String | Map<String, Object> <i>claims</i> , String <i>subject</i> | Crea un token JWT a partire dai claims e dal soggetto passato come parametri. |
| validateToken | Boolean | String <i>token</i> , String <i>username</i> | Valida il token JWT verificando che il nome utente e la versione del token corrispondano e che il token non sia scaduto. |



| PasswordGenerator | | | |
|-------------------------------------|-----------------|---------------------|---|
| Metodo | Tipo di ritorno | Parametri | Descrizione |
| <code>generateSecurePassword</code> | String | | Genera una password sicura con almeno 12 caratteri, includendo lettere maiuscole, minuscole, numeri e caratteri speciali. |
| <code>shuffleString</code> | String | String <i>input</i> | Mescola i caratteri di una stringa per ottenere una disposizione casuale. |

| SecurityConfig | | | |
|----------------------------------|---------------------|--------------------------|--|
| Metodo | Tipo di ritorno | Parametri | Descrizione |
| <code>securityFilterChain</code> | SecurityFilterChain | HttpSecurity <i>http</i> | Configura la sicurezza HTTP, permette l'accesso a <code>/api/auth/login</code> con il metodo POST e autentica tutte le altre richieste. Aggiunge il filtro JWT. |
| <code>passwordEncoder</code> | PasswordEncoder | | Crea e restituisce un'istanza di BCryptPasswordEncoder per la codifica delle password. |
| <code>corsFilter</code> | CorsFilter | | Configura il CORS per permettere le richieste da <code>http://localhost:4200</code> , con metodi e intestazioni specifici. |

7.2.1.8 com.progresso.backend.validation



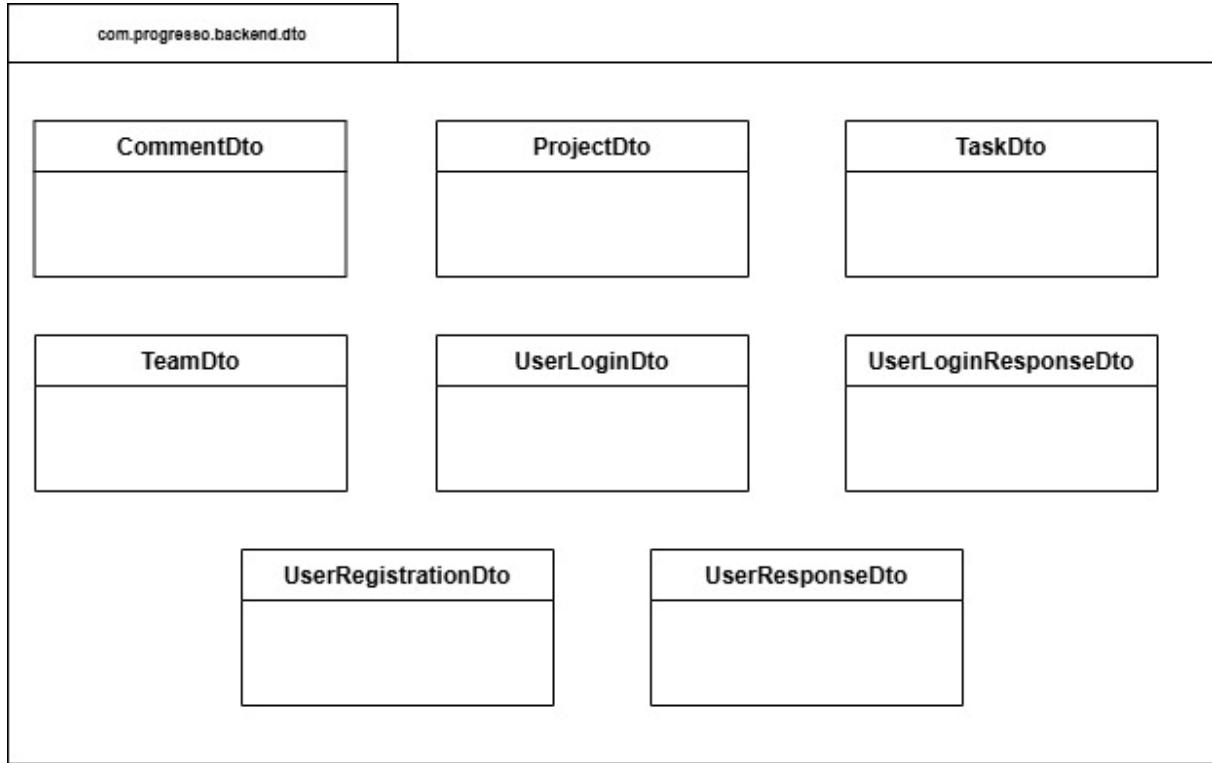
| Age (Annotation personalizzata) | | | |
|------------------------------------|----------------------------|---------------------------|--|
| Attributo | Tipo | Valore di default | Descrizione |
| <code>message</code> | String | "Age must be 17 or older" | Il messaggio d'errore restituito se il vincolo di età non è soddisfatto. |
| <code>groups</code> | Class<?>[] | | Consente di specificare i gruppi di validazione a cui applicare questo vincolo, per gestire scenari di validazione differenti. |
| <code>payload</code> | Class<? extends Payload>[] | | Permette di associare informazioni aggiuntive al risultato della validazione, |



| | | | |
|--|--|--|--|
| | | | come il livello di severità o altre informazioni personalizzate. |
|--|--|--|--|

| AgeValidator | | | |
|--------------|-----------------|--|--|
| Metodo | Tipo di ritorno | Parametri | Descrizione |
| initialize | void | Age constraintAnnotation | Metodo di inizializzazione dell'annotazione. |
| isValid | boolean | LocalDate birthDate, ConstraintValidatorContext context | Calcola l'età a partire dalla data di nascita e restituisce <i>true</i> se l'età è maggiore o uguale a 17, altrimenti <i>false</i> . |

7.2.1.9 com.progresso.backend.dto



| CommentDto | | | |
|----------------------|---------------|---|---|
| Campo | Tipo | Annotazioni/Vincoli | Descrizione |
| id | Long | | Identificativo univoco del commento. |
| content | String | @NotBlank @Size(max = 500) @Pattern (regexp = "^\S.*") | Testo del commento. |
| creationDate | LocalDateTime | | Data e ora di creazione del commento. |
| userId | Long | @NotNull | Identificativo dell'utente autore del commento. |
| userFirstName | String | | Nome dell'utente autore. |
| userLastName | String | | Cognome dell'utente autore. |
| userUsername | String | | Username dell'utente autore. |



UNIVERSITÀ DEGLI STUDI
DI SALERNO

| | | | |
|----------------------------|---------------|----------|---|
| projectId | Long | @NotNull | Identificativo del progetto a cui il commento è associato. |
| modified | Boolean | | Indica se il commento è stato modificato. |
| modifiedDate | LocalDateTime | | Data e ora dell'ultima modifica del commento. |
| deleted | Boolean | | Indica se il commento è stato cancellato (logicamente). |
| parentId | Long | | Identificativo del commento padre, se esiste (per le risposte). |
| parentContent | String | | Testo del commento padre, se presente. |
| parentCreationDate | LocalDateTime | | Data e ora di creazione del commento padre. |
| parentUserId | Long | | Identificativo dell'utente autore del commento padre. |
| parentUserFirstName | String | | Nome dell'utente autore del commento padre. |
| parentUserLastName | String | | Cognome dell'utente autore del commento padre. |
| parentUsername | String | | Username dell'utente autore del commento padre. |
| parentModified | Boolean | | Indica se il commento padre è stato modificato. |
| parentModifiedDate | LocalDateTime | | Data e ora dell'ultima modifica del commento padre. |
| parentDeleted | Boolean | | Indica se il commento padre è stato cancellato. |

| ProjectDto | | | |
|--------------------------------|-----------|--|---|
| Campo | Tipo | Annotazioni/Vincoli | Descrizione |
| id | Long | | Identificativo univoco del progetto. |
| name | String | @NotBlank, @Size(max = 100), @Pattern (regexp = "^\\$.*") | Nome del progetto. |
| description | String | @NotBlank, @Size(max = 255), @Pattern (regexp = "^\\$.*") | Descrizione del progetto. |
| priority | String | | Priorità del progetto. |
| startDate | LocalDate | @NotNull | Data di inizio del progetto. |
| dueDate | LocalDate | @NotNull | Data di scadenza del progetto. |
| completionDate | LocalDate | | Data di completamento del progetto. |
| completionPercentage | Long | | Percentuale di completamento del progetto, calcolata in base al completamento dei task. |
| status | String | | Stato del progetto. |
| projectManagerId | Long | @NotNull | Identificativo del project manager assegnato al progetto. |
| projectManagerFirstName | String | | Nome del project manager. |
| projectManagerLastName | String | | Cognome del project manager. |



UNIVERSITÀ DEGLI STUDI
DI SALERNO

| | | | |
|-------------------------------|------------|--|---|
| projectManagerUsername | String | | Username del project manager. |
| taskIds | List<Long> | | Elenco degli ID dei task associati al progetto. |
| teamId | Long | | Identificativo del team assegnato al progetto. |
| teamName | String | | Nome del team assegnato al progetto. |
| commentIds | List<Long> | | Elenco degli ID dei commenti associati al progetto. |

| TaskDto | | | |
|-----------------------------|-----------|---|---|
| Campo | Tipo | Annotazioni/Vincoli | Descrizione |
| id | Long | | Identificativo univoco del task. |
| name | String | @NotBlank, @Size(max = 100), @Pattern (regexp = "^\S.*") | Nome del task. |
| description | String | @NotBlank, @Size(max = 255), @Pattern (regexp = "^\S.*") | Descrizione del task. |
| priority | String | @NotNull | Priorità del task. |
| startDate | LocalDate | @NotNull | Data di inizio del task. |
| dueDate | LocalDate | @NotNull | Data di scadenza del task. |
| completionDate | LocalDate | | Data di completamento del task. |
| status | String | | Stato corrente del task. |
| projectId | Long | @NotNull | Identificativo del progetto a cui il task appartiene. |
| assignedUserId | Long | | Identificativo dell'utente a cui il task è assegnato. |
| assignedUserUsername | String | | Username dell'utente assegnato al task. |

| TeamDto | | | |
|----------------------|------------|---|--|
| Campo | Tipo | Annotazioni/Vincoli | Descrizione |
| id | Long | | Identificativo univoco del team. |
| name | String | @NotBlank @Size(max = 100) @Pattern (regexp = "^\S.*") | Nome del team. |
| active | Boolean | | Indica se il team è attivo. |
| teamMemberIds | List<Long> | | Elenco degli ID degli utenti che fanno parte del team. |
| projectIds | List<Long> | | Elenco degli ID dei progetti associati al team. |



UNIVERSITÀ DEGLI STUDI
DI SALERNO

UserLoginDto

| Campo | Tipo | Annotazioni/Vincoli | Descrizione |
|-----------------|--------|--|---|
| username | String | @NotBlank @Pattern (regexp = "^[a-zA-Z]\\.[a-zA-Z_] +\\. (am pm tm) [0-9] + @progresso\\.com\$") | Username dell'utente, deve seguire un formato rigoroso che rappresenta una combinazione del nome, cognome, ruolo e un numero identificativo e, per concludere, il dominio fisso @progresso.com. |
| password | String | @NotBlank @Size(min = 8) | Password dell'utente. |

UserLoginResponseDto

| Campo | Tipo | Annotazioni/Vincoli | Descrizione |
|------------------------|-----------------|---------------------|---|
| token | String | | Token JWT restituito al momento del login per l'autenticazione successiva dell'utente. |
| userResponseDto | UserResponseDto | | Oggetto contenente le informazioni di base dell'utente, come ID, nome, cognome e altri dettagli necessari per il front-end. |

UserRegistrationDto

| Campo | Tipo | Annotazioni/Vincoli | Descrizione |
|----------------------------|-----------|--|---|
| firstName | String | @NotBlank, @Size(max = 50), @Pattern (regexp = "^[\\S[a-zA-ZÀ-Ñ\\s]+\$") | Il nome dell'utente. |
| lastName | String | @NotBlank, @Size(max = 50), @Pattern (regexp = "^[\\S[a-zA-ZÀ-Ñ\\s]+\$") | Il cognome dell'utente. |
| birthDate | LocalDate | @Past, @Age | La data di nascita dell'utente. |
| phoneNumber | String | @NotBlank | Il numero di telefono dell'utente. |
| streetAddress | String | @NotBlank, @Size(max = 100), @Pattern (regexp = "^[\\S[a-zA-Z0-9\\s,-]+\$") | L'indirizzo di residenza dell'utente. |
| city | String | @NotBlank, @Size(max = 50), @Pattern (regexp = "^[\\S[a-zA-ZÀ-Ñ\\s]+\$") | La città di residenza dell'utente. |
| stateProvinceRegion | String | @NotBlank, @Size(max = 50), @Pattern (regexp = "^[\\S[a-zA-ZÀ-Ñ\\s]+\$") | Lo stato, provincia o regione di residenza dell'utente. |
| country | String | @NotBlank, @Size(max = 50), @Pattern | Il paese di residenza dell'utente. |



UNIVERSITÀ DEGLI STUDI
DI SALERNO

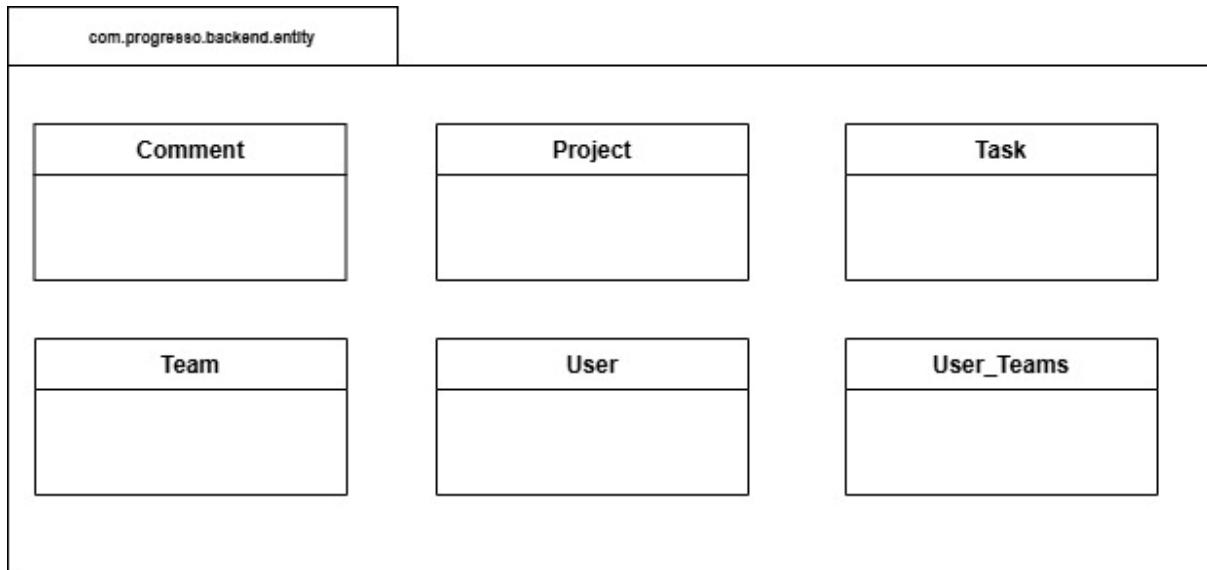
| | | | |
|----------------|--------|--|--|
| | | (regexp = "^\S[a-zA-ZÀ-Ñ\s]+\$") | |
| zipCode | String | @NotBlank, @Pattern (regexp = "^[0-9]{5}(?:-[0-9]{4})?\$") | Il codice di avviamento postale dell'utente. |
| email | String | @NotBlank, @Email, @Pattern (regexp = "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.(com org net edu gov)\$") | L'e-mail dell'utente. |
| role | String | @NotBlank | Il ruolo dell'utente. |

| UserResponseDto | | | |
|--------------------------|------------|---------------------|---|
| Campo | Tipo | Annotazioni/Vincoli | Descrizione |
| id | Long | | L'ID univoco dell'utente. |
| firstName | String | | Il nome dell'utente. |
| lastName | String | | Il cognome dell'utente. |
| username | String | | L'username dell'utente. |
| role | String | | Il ruolo dell'utente. |
| assignedTaskIds | List<Long> | | La lista degli ID dei task assegnati all'utente. |
| managedProjectIds | List<Long> | | La lista degli ID dei progetti gestiti dall'utente. |
| teamIds | List<Long> | | La lista degli ID dei team a cui l'utente appartiene. |
| commentIds | List<Long> | | La lista degli ID dei commenti scritti dall'utente. |
| active | Boolean | | Indica se l'utente è attivo o meno. |



UNIVERSITÀ DEGLI STUDI
DI SALERNO

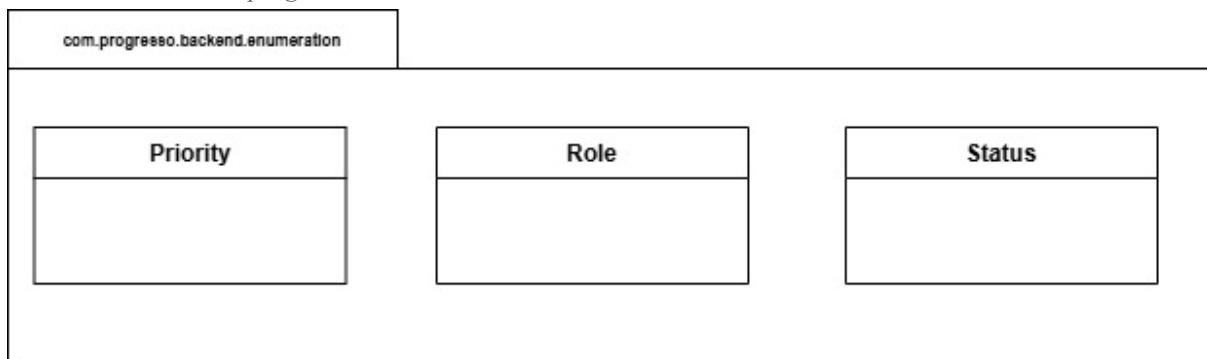
7.2.1.10 com.progresso.backend.entity



Per visualizzare i dettagli delle entità del sistema, è possibile consultare lo schema presente al paragrafo:

[3.3.1 Entity Class Diagram.](#)

7.2.1.11 com.progresso.backend.enumeration



Per visualizzare i dettagli delle enumeration del sistema, è possibile consultare lo schema presente al paragrafo:

[3.3.1 Entity Class Diagram.](#)



7.2.1.12 com.progresso.backend.exception



La classe ***GlobalExceptionHandler*** sfrutta il meccanismo di gestione centralizzata delle eccezioni offerto da *Spring Boot* attraverso l'annotazione `@RestControllerAdvice`. In pratica, quando un'eccezione viene lanciata all'interno di un controller (o da un service propagata al controller), *Spring Boot* la intercetta e la passa al ***GlobalExceptionHandler***. Ecco come funziona in dettaglio:

- L'annotazione `@RestControllerAdvice` consente di creare una classe globale che gestisce le eccezioni per i controller annotati con `@RestController`. Quando un controller non gestisce direttamente un'eccezione, questa viene inviata a questa classe per un'elaborazione centralizzata.
- All'interno del ***GlobalExceptionHandler***, ogni metodo annotato con `@ExceptionHandler` è configurato per gestire uno o più tipi specifici di eccezioni. Quando un'eccezione del tipo indicato viene lanciata, il metodo corrispondente viene invocato automaticamente da *Spring Boot*, bypassando la normale propagazione dell'errore.
- Ogni metodo che gestisce le eccezioni è stato annotato con `@ResponseStatus`, che specifica il codice *HTTP* da restituire (ad esempio, 404 per una risorsa non trovata o



400 per una richiesta non valida). In questo modo, il **GlobalExceptionHandler** costruisce una risposta *HTTP* che comunica al client il tipo di errore verificatosi.

Il flusso della gestione delle eccezioni, dunque, è il seguente:

- **Service:** Se un metodo nel service layer lancia un'eccezione, quest'ultima viene propagata al controller.
- **Controller:** Il controller non gestisce direttamente l'eccezione, per cui essa viene intercettata dal meccanismo di *Spring Boot*.
- **GlobalExceptionHandler:** Grazie a `@RestControllerAdvice` e ai metodi annotati con `@ExceptionHandler`, l'eccezione viene catturata e il corrispondente metodo gestore produce una risposta *HTTP* adeguata, contenente lo *status code* definito con `@ResponseStatus` e il messaggio d'errore fornito dall'eccezione.

| Nome eccezione | Tipo | Http Status | Descrizione |
|------------------------------------|----------------|-------------------|--|
| ActiveProjectsException | Personalizzata | 400 (BAD REQUEST) | Sollevata quando un Admin cerca di disattivare un Project manager che gestisce progetti attivi. |
| CommentNotFoundException | Personalizzata | 404 (NOT FOUND) | Sollevata quando non viene trovato il commento richiesto. |
| EmailAlreadyExistsException | Personalizzata | 400 (BAD REQUEST) | Sollevata quando un Admin tenta di registrare un utente con un un'e-mail già presente nel sistema. |
| InvalidPasswordException | Personalizzata | 400 (BAD REQUEST) | Sollevata quando in fase di login un utente inserisce una password errata. |
| InvalidRoleException | Personalizzata | 403 (FORBIDDEN) | Sollevata quando si tenta di eseguire un'operazione riservata a un determinato ruolo su un utente che non possiede quel ruolo. Ad esempio, se si tenta di assegnare un progetto a un utente che non è un Project manager oppure di assegnare un task a un utente che non è un Team member. |
| NoDataFoundException | Personalizzata | 404 (NOT FOUND) | Sollevata quando un metodo che interroga il database si aspetta di ricevere una pagina di risultati, ma invece riceve una pagina vuota. |
| ProjectNotFoundException | Personalizzata | 404 (NOT FOUND) | Sollevata quando non viene trovato il progetto richiesto. |
| TaskNotFoundException | Personalizzata | 404 (NOT FOUND) | Sollevata quando non viene trovato il task richiesto. |
| TeamNotFoundException | Personalizzata | 404 (NOT FOUND) | Sollevata quando non viene trovato il team richiesto. |
| UserNotActiveException | Personalizzata | 403 (FORBIDDEN) | Sollevata quando viene richiesta un'operazione su un utente che non è attivo. |
| UserNotFoundException | Personalizzata | 404 (NOT FOUND) | Sollevata quando non viene trovato l'utente richiesto. |



UNIVERSITÀ DEGLI STUDI
DI SALERNO

| | | | |
|--|----------|-----------------------------|---|
| AccessDeniedException | Standard | 403 (FORBIDDEN) | Sollevata quando un utente tenta di accedere a una risorsa per la quale non dispone dei permessi necessari. |
| DateTimeParseException | Standard | 400 (BAD REQUEST) | Sollevata quando il formato di una data non è valido. |
| MethodArgumentNotValidException | Standard | 400 (BAD REQUEST) | Sollevata quando la validazione dei parametri di un metodo fallisce (ad es., tramite annotazioni di validazione). |
| ConstraintViolationException | Standard | 400 (BAD REQUEST) | Sollevata quando vengono violate le constraint di validazione sui parametri o sugli attributi degli oggetti. |
| IllegalStateException | Standard | 400 (BAD REQUEST) | Sollevata quando lo stato dell'applicazione non consente l'esecuzione dell'operazione richiesta. |
| IllegalArgumentException | Standard | 400 (BAD REQUEST) | Sollevata quando un argomento passato a un metodo non è valido. |
| HttpMessageNotReadableException | Standard | 400 (BAD REQUEST) | Sollevata quando il corpo della richiesta HTTP non è leggibile (es. formato JSON errato). |
| Exception | Standard | 500 (INTERNAL SERVER ERROR) | Sollevata per errori generici imprevisti, restituendo un messaggio d'errore generico al client. |



7.3 Appendice C

7.3.1 Appendice C.1: Documentazione dei test case unitari

7.3.1.1 AuthServiceTest – registerUser

| AuthService | | | |
|--------------|-----------------|---|---|
| Metodo | Tipo di ritorno | Parametri | Descrizione |
| registerUser | UserResponseDto | UserRegistrationDto <i>userRegistrationDto</i> | Registra un nuovo utente verificando l'unicità dell'e-mail, generando username e password, e salvando l'utente. |

| UserRegistrationDto | | | |
|---------------------|--------|---|-----------------------|
| Campo | Tipo | Annotazioni/Vincoli | Descrizione |
| email | String | @NotBlank, @Email, @Pattern (regexp = " ^[a-zA-Z0-9._%+-] +@[a-zA-Z0-9.] +\.(com org net edu gov)\$") | L'e-mail dell'utente. |

| ID Test case | Condizione/Scenario | Output atteso | Note |
|--------------|---|---|--|
| TC_1.1 | L'e-mail fornita dall' <i>UserRegistrationDto</i> esiste già nel sistema. | Solleva un'eccezione <i>EmailAlreadyExistsException</i> . | Verifica il controllo di unicità sull'e-mail. |
| TC_1.2 | Registrazione con dati validi ed e-mail univoca. | Registrazione dell'utente avvenuta con successo. | Verifica che l'utente salvato corrisponda ai dati forniti dal DTO. |



7.3.1.2 CommentServiceTest – createComment

| CommentService | | | |
|----------------------|-----------------|---------------------------------|---|
| Metodo | Tipo di ritorno | Parametri | Descrizione |
| createComment | CommentDto | CommentDto <i>commentDto</i> | Crea un nuovo commento per un progetto: valida l'esistenza e lo stato attivo dell'utente e del progetto, gestisce eventuali risposte (commento padre) e salva il commento, restituendo il relativo DTO. |

| CommentDto | | | |
|------------------|------|---------------------|---|
| Campo | Tipo | Annotazioni/Vincoli | Descrizione |
| userId | Long | @NotNull | Identificativo dell'utente autore del commento. |
| projectId | Long | @NotNull | Identificativo del progetto a cui il commento è associato. |
| parentId | Long | | Identificativo del commento padre, se esiste (per le risposte). |

| ID Test case | Condizione/Scenario | Output atteso | Note |
|---------------|---|--|--|
| TC_2.1 | L' <i>userId</i> fornito dal <i>commentDto</i> non corrisponde a nessun utente. | Viene sollevata un'eccezione <i>UserNotFoundException</i> . | Verifica il controllo sull'esistenza dell'utente. |
| TC_2.2 | L' <i>userId</i> fornito dal <i>commentDto</i> corrisponde ad un utente che non è attivo. | Viene sollevata un'eccezione <i>UserNotActiveException</i> . | Verifica che l'utente debba essere attivo. |
| TC_2.3 | L' <i>userId</i> fornito dal <i>commentDto</i> corrisponde ad un utente attivo che non ha ruolo <i>ADMIN</i> e non sta lavorando al progetto. | Viene sollevata un'eccezione <i>UserNotFoundException</i> . | Verifica che il metodo controlli correttamente l'appartenenza dell'utente al progetto prima di creare il commento. |



| | | | |
|----------------|--|--|---|
| TC_2.4 | Il <i>projectId</i> fornito dal <i>commentDto</i> non corrisponde a nessun progetto. | Viene sollevata un'eccezione <i>ProjectNotFoundException</i> . | Verifica il controllo sull'esistenza del progetto. |
| TC_2.5 | Il <i>projectId</i> fornito dal <i>commentDto</i> corrisponde ad un progetto completato. | Viene sollevata un'eccezione <i>IllegalArgumentException</i> . | Verifica che non si possano aggiungere commenti a progetti completati. |
| TC_2.6 | Il <i>projectId</i> fornito dal <i>commentDto</i> corrisponde ad un progetto cancellato. | Viene sollevata un'eccezione <i>IllegalArgumentException</i> . | Verifica che non si possano aggiungere commenti a progetti cancellati. |
| TC_2.7 | Il <i>parentId</i> fornito dal <i>commentDto</i> non corrisponde a nessun commento. | Viene sollevata un'eccezione <i>CommentNotFoundException</i> . | Verifica che il metodo gestisca correttamente il caso in cui il commento padre non venga trovato. |
| TC_2.8 | Il <i>parentId</i> fornito dal <i>commentDto</i> corrisponde ad un commento marcato come eliminato. | Viene sollevata un'eccezione <i>IllegalArgumentException</i> . | Verifica che non si possa rispondere a un commento eliminato. |
| TC_2.9 | L' <i>userId</i> e il <i>projectId</i> forniti dal <i>commentDto</i> corrispondono rispettivamente ad un utente attivo che ricopre il ruolo di <i>ADMIN</i> . | Creazione del commento avvenuta con successo. | Verifica che il commento salvato corrisponda ai dati forniti dal <i>DTO</i> . |
| TC_2.10 | L' <i>userId</i> e il <i>projectId</i> forniti dal <i>commentDto</i> identificano, rispettivamente, un utente attivo senza il ruolo di <i>ADMIN</i> e un progetto attivo in cui l'utente è effettivamente coinvolto. | Creazione del commento avvenuta con successo. | Verifica che il commento salvato corrisponda ai dati forniti dal <i>DTO</i> . |
| TC_2.11 | L' <i>userId</i> fornito dal <i>commentDto</i> identifica un utente attivo che non riveste il ruolo di | Creazione del commento risposta avvenuta con successo. | Verifica che il commento salvato corrisponda ai |



| | | | |
|--|--|--|-------------------------------|
| | <i>ADMIN</i> , ma risulta coinvolto nel progetto attivo associato al <i>projectId</i> . Inoltre, il <i>parentId</i> specificato nel <i>commentDto</i> individua un commento esistente che non è stato eliminato. | | dati forniti dal <i>DTO</i> . |
|--|--|--|-------------------------------|

7.3.1.3 ProjectServiceTest – *createProject*

| ProjectService | | | |
|----------------------|-----------------|---------------------------------|--|
| Metodo | Tipo di ritorno | Parametri | Descrizione |
| createProject | ProjectDto | ProjectDto <i>projectDto</i> | Crea un nuovo progetto dopo aver validato le date (start e due), verificato la validità del project manager (ruolo, stato attivo, numero massimo di progetti attivi) e generato un nome univoco per il progetto. |

| ProjectDto | | | |
|-------------------------|-----------|--|---|
| Campo | Tipo | Annotazioni/Vincoli | Descrizione |
| name | String | @NotBlank, @Size(max = 100), @Pattern (regexp = "^\\$.*") | Nome del progetto. |
| startDate | LocalDate | @NotNull | Data di inizio del progetto. |
| dueDate | LocalDate | @NotNull | Data di scadenza del progetto. |
| projectManagerId | Long | @NotNull | Identificativo del project manager assegnato al progetto. |

| ID Test case | Condizione/Scenario | Output atteso | Note |
|---------------|--|--|---|
| TC_3.1 | La <i>startDate</i> fornita dal <i>projectDto</i> corrisponde ad una data antecedente alla data odierna. | Solleva un'eccezione <i>IllegalArgumentException</i> . | Verifica che la data di inizio non sia antecedente alla data odierna. |



| | | | |
|---------------|---|---|---|
| TC_3.2 | La <i>startDate</i> e la <i>dueDate</i> fornite dal <i>projectDto</i> sono impostate in modo tale che la data di inizio (<i>startDate</i>) sia successiva alla data di scadenza (<i>dueDate</i>). | Solleva un'eccezione <i>IllegalArgumentException</i> . | Verifica la correttezza dell'ordine delle date. |
| TC_3.3 | Il <i>name</i> fornito dal <i>projectDto</i> corrisponde a quello di un progetto esistente. | Assegna automaticamente un nome univoco aggiungendo un suffisso. | Verifica la gestione dei nomi duplicati. |
| TC_3.4 | Il <i>name</i> fornito dal <i>projectDto</i> corrisponde a quello di un progetto esistente e, dopo aver aggiunto il suffisso, supera i cento caratteri. | Il sistema tronca il nome per rispettare il limite di cento caratteri, aggiungendo correttamente il suffisso. | Verifica che la logica per la generazione di nomi univoci tronca il nome quando necessario. |
| TC_3.5 | Il <i>projectManagerId</i> fornito dal <i>projectDto</i> non corrisponde a nessun utente. | Solleva un'eccezione <i>UserNotFoundException</i> . | Verifica il controllo sull'esistenza del Project Manager |
| TC_3.6 | Il <i>projectManagerId</i> fornito dal <i>projectDto</i> corrisponde ad un utente che non ricopre il ruolo di <i>PROJECTMANAGER</i> . | Solleva un'eccezione <i>InvalidRoleException</i> . | Verifica che solo un Project Manager possa essere assegnato al progetto. |
| TC_3.7 | Il <i>projectManagerId</i> corrisponde ad un utente con ruolo <i>PROJECTMANAGER</i> che non è attivo. | Solleva un'eccezione <i>UserNotActiveException</i> . | Verifica che il Project Manager sia attivo. |
| TC_3.8 | Il <i>projectManagerId</i> corrisponde ad un utente con il ruolo <i>PROJECTMANAGER</i> ma quest'ultimo ha raggiunto il limite di progetti attivi. | Solleva un'eccezione <i>IllegalArgumentException</i> . | Verifica che il Project manager non superi il limite di progetti attivi. |
| TC_3.9 | Input completamente valido. | Creazione del progetto avvenuta con successo. | Verifica che il progetto salvato corrisponda ai dati forniti dal <i>DTO</i> . |



7.3.1.4 TaskServiceTest – createAndAssignTask

| TaskService | | | |
|---------------------------------|-----------------|--|--|
| Metodo | Tipo di ritorno | Parametri | Descrizione |
| createAndAssignTa sk | TaskDto | TaskDto <i>taskDto</i> , Long <i>userId</i> | Crea un nuovo task, assegnandolo a un utente specificato. Il metodo esegue varie validazioni su date, stato del progetto e del team, e il ruolo dell'utente prima di salvare il task e aggiornare le assegnazioni. |

| TaskDto | | | |
|------------------|-----------|--|---|
| Campo | Tipo | Annotazioni/Vincoli | Descrizione |
| name | String | @NotBlank, @Size(max = 100), @Pattern (regexp = "^\\$.*") | Nome del task. |
| startDate | LocalDate | @NotNull | Data di inizio del task. |
| dueDate | LocalDate | @NotNull | Data di scadenza del task. |
| projectId | Long | @NotNull | Identificativo del progetto a cui il task appartiene. |

| ID Test case | Condizione/Scenario | Output atteso | Note |
|---------------|--|--|---|
| TC_4.1 | Non è stato fornito il parametro <i>userId</i> . | Solleva un'eccezione <i>IllegalArgumentException</i> . | Verifica che il parametro <i>userId</i> sia obbligatorio per la funzione. |
| TC_4.2 | La <i>startDate</i> fornita dal <i>taskDto</i> è antecedente alla data odierna. | Solleva un'eccezione <i>IllegalArgumentException</i> . | Verifica che la data di inizio non sia antecedente alla data odierna. |
| TC_4.3 | La <i>startDate</i> e la <i>dueDate</i> fornite dal <i>taskDto</i> sono impostate in modo che la data di inizio preceda la scadenza. | Solleva un'eccezione <i>IllegalArgumentException</i> . | Verifica la correttezza dell'ordine delle date. |



| | | | |
|----------------|---|--|--|
| TC_4.4 | La <i>startDate</i> fornita dal <i>taskDto</i> è antecedente alla data di inizio del progetto associato al <i>projectId</i> . | Solleva un'eccezione <i>IllegalArgumentException</i> . | Verifica che il task non inizi prima del progetto. |
| TC_4.5 | La <i>dueDate</i> fornita dal <i>taskDto</i> è successiva alla data di scadenza del progetto associato al <i>projectId</i> . | Solleva un'eccezione <i>IllegalArgumentException</i> . | Verifica che il task non finisca oltre la scadenza del progetto. |
| TC_4.6 | Il <i>projectId</i> fornito dal <i>taskDto</i> non corrisponde a nessun progetto esistente. | Solleva un'eccezione <i>ProjectNotFoundException</i> . | Verifica l'esistenza del progetto. |
| TC_4.7 | Il progetto associato al <i>projectId</i> fornito dal <i>taskDto</i> è marcato come completato. | Solleva un'eccezione <i>IllegalStateException</i> . | Verifica che non sia possibile creare un task per un progetto completato. |
| TC_4.8 | Il progetto associato al <i>projectId</i> fornito dal <i>taskDto</i> è marcato come cancellato. | Solleva un'eccezione <i>IllegalStateException</i> . | Verifica che non sia possibile creare un task per un progetto cancellato. |
| TC_4.9 | Il progetto associato al <i>projectId</i> fornito dal <i>taskDto</i> non è stato assegnato a un team. | Solleva un'eccezione <i>IllegalArgumentException</i> . | Verifica che il progetto abbia un team assegnato. |
| TC_4.10 | Il team assegnato al progetto associato al <i>projectId</i> fornito dal <i>taskDto</i> non è attivo. | Solleva un'eccezione <i>IllegalStateException</i> . | Verifica che il team assegnato al progetto sia attivo. |
| TC_4.11 | L' <i>userId</i> fornito dal <i>taskDto</i> non corrisponde a un utente esistente. | Solleva un'eccezione <i>UserNotFoundException</i> . | Verifica che l'utente esista nel sistema. |
| TC_4.12 | L' <i>userId</i> fornito dal <i>taskDto</i> corrisponde a un utente non attivo. | Solleva un'eccezione <i>UserNotActiveException</i> . | Verifica che l'utente sia attivo. |
| TC_4.13 | L' <i>userId</i> fornito dal <i>taskDto</i> corrisponde a un utente che non possiede il ruolo di <i>TEAMMEMBER</i> . | Solleva un'eccezione <i>InvalidRoleException</i> . | Verifica che l'utente abbia un ruolo appropriato per essere assegnato al task. |
| TC_4.14 | L' <i>userId</i> fornito nel <i>taskDto</i> corrisponde a un utente con ruolo adeguato che però non | Solleva un'eccezione <i>IllegalArgumentException</i> . | Verifica che l'utente sia effettivamente |



| | | | |
|----------------|---|--|--|
| | sta lavorando sul progetto. | | assegnato al progetto. |
| TC_4.15 | Il <i>name</i> fornito dal <i>taskDto</i> è già stato assegnato a un altro task all'interno del progetto associato. | Assegna automaticamente un nome univoco aggiungendo un suffisso. | Verifica la gestione dei nomi duplicati all'interno di un progetto. |
| TC_4.16 | Il <i>name</i> fornito nel <i>taskDto</i> , dopo l'aggiunta di un suffisso per unicità, supera i cento caratteri. | Il sistema tronca il nome e aggiunge correttamente il suffisso. | Verifica che i nomi dei task rispettino il limite di cento caratteri con suffisso incluso. |
| TC_4.17 | Input completamente valido. | Creazione e assegnazione del task avvenute con successo. | Verifica che il task salvato corrisponda ai dati forniti dal <i>DTO</i> . |

7.3.1.5 TeamServiceTest – *createTeam*

| TeamService | | | |
|-------------------|-----------------|------------------------|--|
| Metodo | Tipo di ritorno | Parametri | Descrizione |
| createTeam | TeamDto | String <i>teamName</i> | Crea un nuovo team verificando che il nome sia valido e univoco (aggiungendo un suffisso se necessario), imposta lo stato attivo e restituisce il DTO del team creato. |

| ID Test case | Condizione/Scenario | Output atteso | Note |
|---------------|--|--|---|
| TC_5.1 | Il parametro <i>teamName</i> non è stato fornito. | Solleva un'eccezione <i>IllegalArgumentException</i> . | Verifica che il parametro <i>teamName</i> sia obbligatorio. |
| TC_5.2 | Il parametro <i>teamName</i> è stato fornito ma è vuoto. | Solleva un'eccezione <i>IllegalArgumentException</i> . | Verifica che il parametro <i>teamName</i> non sia vuoto. |
| TC_5.3 | Il parametro <i>teamName</i> corrisponde a un nome già assegnato a un team presente nel sistema. | Assegna automaticamente un nome univoco aggiungendo un suffisso. | Verifica la gestione dei nomi duplicati. |
| TC_5.4 | Il parametro <i>teamName</i> , dopo l'aggiunta del suffisso per garantire unicità, | Il sistema tronca il nome e aggiunge correttamente il suffisso. | Verifica che i nomi dei team rispettino il limite di cento |



UNIVERSITÀ DEGLI STUDI
DI SALERNO

| | supera il limite di cento caratteri. | | caratteri con suffisso incluso. |
|--------|--------------------------------------|---|---|
| TC_5.5 | Input completamente valido. | Creazione del team avvenuta con successo. | Verifica che il team salvato abbia il nome specificato (o modificato secondo le regole di unicità). |

7.3.2 Appendice C.2: Documentazione dei test case dell'interfaccia utente

7.3.2.1 *Registrazione utente*

| birthDate | | | |
|--------------|-----------|--|------------------------|
| ID Test case | Validator | Scenario | Output atteso |
| TC_1.3.1 | required | Viene selezionato solo l'anno. | Birthdate is required. |
| TC_1.3.2 | required | Vengono selezionati anno e mese, ma non il giorno. | Birthdate is required. |



phonePrefix

pattern

(/^|+|d*\$/)

| ID Test case | Validator | Scenario | Output atteso |
|--------------|-----------|---|---|
| TC_1.4.1 | required | Campo lasciato vuoto. | Phone prefix is required. |
| TC_1.4.2 | minLength | Inseriti meno di due caratteri. | Invalid phone prefix. |
| TC_1.4.3 | maxLength | Inseriti più di sei caratteri (incluso il "+"). | Invalid phone prefix. |
| TC_1.4.4 | pattern | Il prefisso non rispetta il formato "+XXX". | The phone prefix must start with an '+' and must contain only digits. |

phoneNumber

pattern

(^|d{3} |d{3} |d{4,}\$)

| ID Test case | Validator | Scenario | Output atteso |
|--------------|-----------|---|---------------------------|
| TC_1.5.1 | required | Campo lasciato vuoto. | Phone number is required. |
| TC_1.5.2 | pattern | Inserite meno di dieci cifre. | Invalid phone number. |
| TC_1.5.3 | pattern | Inserite nove cifre ed un carattere non numerico. | Invalid phone number. |
| TC_1.5.4 | maxLength | Inserite più di quindici cifre. | Invalid phone number. |

role

| ID Test case | Validator | Scenario | Output atteso |
|--------------|-----------|---------------------------------|-------------------|
| TC_1.6.1 | required | Non viene selezionato il ruolo. | Role is required. |

streetAddress

pattern

(^|[a-zA-Z0-9|\s,.-]+\$)

| ID Test case | Validator | Scenario | Output atteso |
|--------------|-----------|----------------------------------|------------------------------|
| TC_1.7.1 | required | Campo lasciato vuoto. | Street address is required. |
| TC_1.7.2 | minLength | Inseriti meno di due caratteri. | Street address is too short. |
| TC_1.7.3 | maxLength | Inseriti più di cento caratteri. | Street address is too long. |



UNIVERSITÀ DEGLI STUDI
DI SALERNO

| | | | |
|-----------------|---------|------------------------------|--------------------------------|
| TC_1.7.4 | pattern | Inseriti caratteri speciali. | Invalid street address format. |
|-----------------|---------|------------------------------|--------------------------------|



| zipCode <i>pattern</i> ('^[0-9]{5}(:-[0-9]{4})?\$\$') | | | |
|--|------------------|--|--------------------------|
| ID Test case | Validator | Scenario | Output atteso |
| TC_1.11.1 | required | Campo lasciato vuoto. | Zip Code is required. |
| TC_1.11.2 | pattern | Inserite meno di cinque cifre. | Invalid Zip Code format. |
| TC_1.11.3 | pattern | Inserite quattro cifre ed un carattere speciale. | Invalid Zip Code format. |

| email <i>pattern</i> ('^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.(com org net edu gov))\$') | | | |
|---|------------------|--|-----------------------|
| ID Test case | Validator | Scenario | Output atteso |
| TC_1.12.1 | required | Campo lasciato vuoto. | Email is required. |
| TC_1.12.2 | pattern | Inserita e-mail non valida (ad esempio a.smith05.gmail.com). | Invalid email format. |

| ID Test case | Validator | Scenario | Output atteso |
|---------------------|--|--|--|
| TC_1.13.1 | Tutti i <i>validator</i> vengono rispettati. | Il form viene compilato correttamente e inviato, ma l'e-mail risulta già registrata nel sistema. | Un toast contenente il messaggio: "This email already exists." |
| TC_1.13.2 | Tutti i <i>validator</i> vengono rispettati. | Il form viene compilato correttamente, inviato e la registrazione va a buon fine. | Un toast contenente il messaggio: "Registration successful!" |



7.3.2.2 Creazione di un commento

| content | | | |
|--------------|--|--|-----------------------|
| ID Test case | Validator | Scenario | Output atteso |
| TC_2.1 | minLength | Inseriti meno di due caratteri. | Comment is too short. |
| TC_2.2 | maxLength | Inseriti più di cinquecento caratteri. | Comment is too long. |
| TC_2.3 | Tutti i validator vengono rispettati. | Il form viene compilato correttamente, inviato e la pubblicazione del commento va a buon fine. | |
| TC_2.4 | minLength (commento di risposta) | Inseriti meno di due caratteri. | Reply is too short. |
| TC_2.5 | maxLength (commento di risposta) | Inseriti più di cinquecento caratteri. | Reply is too long. |
| TC_2.6 | Tutti i validator vengono rispettati (commento di risposta). | Il form viene compilato correttamente, inviato e la pubblicazione del commento di risposta va a buon fine. | |

7.3.2.3 Creazione di un progetto

| name | | | |
|--------------|-----------|----------------------------------|----------------------------|
| ID Test case | Validator | Scenario | Output atteso |
| TC_3.1.1 | required | Campo lasciato vuoto. | Project name is required. |
| TC_3.1.2 | minLength | Inseriti meno di due caratteri. | Project name is too short. |
| TC_3.1.3 | maxLength | Inseriti più di cento caratteri. | Project name is too long. |

| description | | | |
|--------------|-----------|---------------------------------|---------------------------|
| ID Test case | Validator | Scenario | Output atteso |
| TC_3.2.1 | required | Campo lasciato vuoto. | Description is required. |
| TC_3.2.2 | minLength | Inseriti meno di due caratteri. | Description is too short. |



| | | | |
|----------|-----------|---|--------------------------|
| TC_3.2.3 | maxLength | Inseriti più di duecento cinquantacinque caratteri. | Description is too long. |
|----------|-----------|---|--------------------------|

| startDate | | | |
|--------------|--------------------------------|--|-------------------------------------|
| ID Test case | Validator | Scenario | Output atteso |
| TC_3.3.1 | required | Viene selezionato solo l'anno. | Start date is required. |
| TC_3.3.2 | required | Vengono selezionati anno e mese, ma non il giorno. | Start date is required. |
| TC_3.3.3 | todayDateBeforeStar tValidator | Viene selezionata una data precedente alla data odierna. | Start date must not be in the past. |

| dueDate | | | |
|--------------|--------------------------|--|--|
| ID Test case | Validator | Scenario | Output atteso |
| TC_3.4.1 | required | Viene selezionato solo l'anno. | Due date is required. |
| TC_3.4.2 | required | Vengono selezionati anno e mese, ma non il giorno. | Due date is required. |
| TC_3.4.3 | startBeforeDueValid ator | Viene selezionata una data precedente alla data di inizio. | Due date must be after the start date. |

| ID Test case | Validator | Scenario | Output atteso |
|--------------|---|--|--|
| TC_3.5.1 | Tutti i validator del form del primo step vengono rispettati. | Il form del primo step viene completato correttamente e l'utente accede al secondo step. Tuttavia, tentando di avanzare al terzo step, l'operazione viene bloccata poiché non è stata effettuata la selezione del project manager. | Il pulsante "Next" del secondo step rimane disabilitato. |
| TC_3.5.2 | Tutti i validator del form del primo step vengono rispettati. | Il form del primo step viene completato | Il pulsante "Next" del secondo step viene abilitato. |



| | | | |
|-----------------|---|---|---|
| | | correttamente, l'utente accede al secondo step e seleziona il project manager. | |
| TC_3.6.1 | Tutti i validator del form del primo step vengono rispettati. | Il form del primo step viene completato correttamente, l'utente accede al secondo step e seleziona il project manager. Successivamente l'utente passa al terzo step ed utilizza il comando “Assign Team Later”. | Un toast contenente il messaggio “Project created successfully!”. |

7.3.2.4 *Creazione e assegnazione di un task*

| name | | | |
|-----------------|-----------|----------------------------------|-------------------------|
| ID Test case | Validator | Scenario | Output atteso |
| TC_4.1.1 | required | Campo lasciato vuoto. | Task name is required. |
| TC_4.1.2 | minLength | Inseriti meno di due caratteri. | Task name is too short. |
| TC_4.1.3 | maxLength | Inseriti più di cento caratteri. | Task name is too long. |

| description | | | |
|-----------------|-----------|---|---------------------------|
| ID Test case | Validator | Scenario | Output atteso |
| TC_4.2.1 | required | Campo lasciato vuoto. | Description is required. |
| TC_4.2.2 | minLength | Inseriti meno di due caratteri. | Description is too short. |
| TC_4.2.3 | maxLength | Inseriti più di duecento cinquantacinque caratteri. | Description is too long. |

| priority | | | |
|-----------------|-----------|-----------------------|-----------------------|
| ID Test case | Validator | Scenario | Output atteso |
| TC_4.3.1 | required | Campo lasciato vuoto. | Priority is required. |



| startDate | | | |
|--------------|---|--|--|
| ID Test case | Validator | Scenario | Output atteso |
| TC_4.4.1 | required | Viene selezionato solo l'anno. | Start date is required. |
| TC_4.4.2 | required | Vengono selezionati anno e mese, ma non il giorno. | Start date is required. |
| TC_4.4.3 | todayDateBeforeStartValidator | Viene selezionata una data precedente alla data odierna. | Start date must not be in the past. |
| TC_4.4.4 | startDateAfterProjectStartDateValidator | Viene selezionata una data precedente alla data di inizio del progetto in cui si vuole creare il task. | Start date must be after the project start date. |
| TC_4.4.5 | startDateBeforeProjectDueDateValidator | Viene selezionata una data successiva alla data di scadenza del progetto in cui si vuole creare il task. | Start date must be before the project due date. |

| dueDate | | | |
|--------------|--------------------------------------|--|---|
| ID Test case | Validator | Scenario | Output atteso |
| TC_4.5.1 | required | Viene selezionato solo l'anno. | Due date is required. |
| TC_4.5.2 | required | Vengono selezionati anno e mese, ma non il giorno. | Due date is required. |
| TC_4.5.3 | startBeforeDueValidator | Viene selezionata una data precedente alla data odierna. | Due date must be after the start date. |
| TC_4.5.4 | dueDateBeforeProjectDueDateValidator | Viene selezionata una data successiva alla data di scadenza del progetto in cui si vuole creare il task. | Due date must be before the project due date. |

| ID Test case | Validator | Scenario | Output atteso |
|--------------|---|---|--|
| TC_4.6.1 | Tutti i validator del form del primo step vengono rispettati. | Il form del primo step viene completato correttamente e l'utente accede al secondo step. Tuttavia, il processo di creazione non può | Il pulsante "Complete" del secondo step rimane disabilitato. |



| | | | |
|-----------------|---|---|---|
| | | essere completato poiché non è stato selezionato l'utente a cui assegnare il task. | |
| TC_4.6.2 | Tutti i validator del form del primo step vengono rispettati. | Il form del primo step viene completato correttamente e l'utente accede al secondo step. Successivamente seleziona l'utente a cui desidera assegnare il task. | Il pulsante “Complete” viene abilitato. |
| TC_4.7.1 | Tutti i validator del form del primo step vengono rispettati. | Il form del primo step viene completato correttamente e l'utente accede al secondo step. Successivamente seleziona l'utente a cui desidera assegnare il task ed utilizza il comando “Complete”. | Un toast contenente il messaggio “Task created successfully!” |



7.3.2.5 *Creazione e assegnazione di un team ad un progetto*

I seguenti test case sono stati condotti all'interno della pagina di un progetto, per il quale, durante la fase di creazione, è stata scelta l'opzione di assegnare il team successivamente tramite il comando "Assign Team Later".

| name | | | |
|--------------|-----------|----------------------------------|-------------------------|
| ID Test case | Validator | Scenario | Output atteso |
| TC_5.1.1 | required | Campo lasciato vuoto. | Team name is required. |
| TC_5.1.2 | minLength | Inseriti meno di due caratteri. | Team name is too short. |
| TC_5.1.3 | maxLength | Inseriti più di cento caratteri. | Team name is too long. |

| ID Test case | Validator | Scenario | Output atteso |
|--------------|---|---|---|
| TC_5.2.1 | Tutti i validator del form del primo step vengono rispettati. | Il form del primo step viene completato correttamente e l'utente accede al secondo step. Tuttavia, il processo di creazione non può essere completato poiché non è stato selezionato alcun utente da aggiungere al team. | Il pulsante “Complete” rimane disabilitato. |
| TC_5.2.2 | Tutti i validator del form del primo step vengono rispettati. | Il form del primo step viene completato correttamente e l'utente accede al secondo step. Successivamente seleziona gli utenti che desidera aggiungere al team. | Il pulsante “Complete” viene abilitato. |
| TC_5.3.1 | Tutti i validator del form del primo step vengono rispettati. | Il form del primo step viene completato correttamente e l'utente accede al secondo step. Successivamente seleziona gli utenti che desidera | Un toast contenente il messaggio “Team assigned successfully!”. |



UNIVERSITÀ DEGLI STUDI
DI SALERNO

| | | | |
|--|--|--|--|
| | | aggiungere al team ed utilizza il comando “Complete”. | |
|--|--|--|--|