Università degli Studi di Salerno
Corso di Ingegneria del Software

WeCook
System Design Document
Versione 1.0

Data: 24/11/2024

Progetto: WeCook	Versione: 1.0
Documento: System Design Document	Data: 24/11/2024

Coordinatore del progetto:

Nome	Matricola

Partecipanti:

Nome	Matricola
Simone Francesco Albino	0512116140
Valentina Ferrentino	0512116842
Giovanni Semioli	0512117412

Scritto da:	Simone Francesco Albino - Valentina Ferrentino - Giovanni Semioli

Revision History

Data	Versione	Descrizione	Autore
24/11/2024	1.0	Prima stesura	Team

Indice

1. Introduzione

WeCook nasce con l'intento di unire persone provenienti da ogni angolo del mondo attraverso una passione universale: la cucina. La piattaforma si propone come un punto di incontro tra tradizioni culinarie, sapori autentici e storie personali, abbattendo le barriere culturali e geografiche. Il suo obiettivo è offrire uno spazio inclusivo dove chiunque possa condividere le proprie radici, reinterpretare la propria idea di cucina e celebrare la diversità gastronomica globale, valorizzando ogni ingrediente e ogni storia che lo accompagna.

1.1 Obiettivo del sistema

L'obiettivo di WeCook è diventare un social network che migliori la condivisione delle ricette e favorisca la connessione tra gli utenti, fornendo un'esperienza intuitiva e accessibile. La piattaforma è progettata per permettere a chiunque, anche a chi ha poca esperienza, di cimentarsi nella preparazione di piatti elaborati grazie a istruzioni semplici e dettagliate.

1.2 Design Goals

<u>Usabilità</u>

1. **Chiarezza delle notifiche**: Il sistema deve comunicare in modo immediato e comprensibile lo stato delle operazioni (successo o insuccesso), utilizzando notifiche visibili, temporanee e non invasive;

2. **Compatibilità multi-dispositivo**: L'interfaccia deve essere completamente responsive, adattandosi a schermi di qualsiasi dimensione per garantire una navigazione fluida su dispositivi mobili, tablet e desktop.

<u>Affidabilità</u>

3. **Protezione dei dati utente**: I dati personali e le informazioni degli utenti devono essere gestiti con rigorosi protocolli di sicurezza per garantire la protezione da accessi non autorizzati;

4. **Gestione robusta degli input**: Il sistema deve essere in grado di riconoscere e gestire input non validi o errati senza compromettere la stabilità della piattaforma.

Prestazioni

5. **Rapidità nella pubblicazione dei contenuti:** La pubblicazione di post (inclusi testo, immagini e ricette) deve avvenire entro un massimo di 10 secondi;

6. **Ottimizzazione delle immagini**: Le immagini caricate dagli utenti devono rispettare un limite massimo di 4 MB, con processi di compressione e verifica automatica per preservare la qualità e la velocità di caricamento;

7. **Velocità di caricamento delle pagine**: Tutte le pagine del sistema devono avere tempi di risposta inferiori ai 10 secondi, ottimizzando il rendering e l'accesso ai dati.

Supportabilità

8. **Tracciabilità delle operazioni**: Il sistema deve integrare un meccanismo di logging avanzato, che permetta agli amministratori di tracciare cronologicamente le operazioni effettuate dagli utenti e diagnosticarne eventuali problemi.

1.3 Riferimenti

Come ispirazione per la realizzazione di WeCook sono stati presi come riferimento diverse piattaforme, analizzandone il design e le principali funzioni social, come il feed delle notizie, la gestione dei profili utente e le interazioni tra utenti (Instagram, Facebook, ecc).
Il progetto si basa, inoltre, su metodologie descritte nei testi di elencati di seguito:

- Bruegge, Bernd, e Dutoit, Allen H., Object-Oriented Software Engineering Using UML, Patterns, and Java™ (Third Edition), Pearson;
- Problem Statement;
- Requirements Analysis Document.

1.4 Overview

Il documento è strutturato per fornire una visione completa dell'architettura del sistema, dalla descrizione generale fino ai dettagli tecnici più approfonditi. Ogni sezione si concentra su aspetti specifici del design, consentendo una comprensione modulare e progressiva del progetto. Nello specifico:

- Il primo paragrafo funge da introduzione e illustra gli obiettivi principali della piattaforma, il contesto in cui si inserisce, le sue finalità e le fonti di ispirazione. Fornisce inoltre una panoramica dei design goals, definizioni rilevanti e riferimenti utilizzati;
- Il secondo paragrafo descrive lo stato attuale delle soluzioni simili disponibili sul mercato, evidenziando i limiti e le opportunità che hanno guidato la progettazione di *WeCook*;
- Il terzo paragrafo presenta il modello architetturale progettato per *WeCook*, inclusa la decomposizione in sottosistemi, il mapping hardware/software, i dettagli sulla gestione dei dati persistenti, la sicurezza, i controlli di accesso e le condizioni di confine.
- Il quarto paragrafo approfondisce i servizi specifici offerti dai principali sottosistemi di *WeCook*, delineandone le responsabilità e le funzionalità principali.

Questa organizzazione modulare rende il documento un riferimento chiaro e completo sia per il team di sviluppo che per gli stakeholder, supportando l'implementazione e la manutenzione del sistema con una visione strutturata e coerente.

2. Architettura Software Corrente

Al momento, non esiste un sistema specifico dedicato alla condivisione di ricette che adotti un approccio tipico dei social network. Tuttavia, esistono diverse soluzioni che, pur non essendo direttamente comparabili, forniscono utili spunti per la progettazione del nuovo sistema. La maggior parte delle attuali piattaforme per la condivisione di ricette, come *Giallo Zafferano, Il Cucchiaio d'Argento* e *Agrodolce*, segue un modello di architettura centralizzata, progettato per la pubblicazione di contenuti statici (blog). Tali sistemi si focalizzano sull'erogazione di articoli e ricette in modo unidirezionale, con un'interazione utente limitata a commenti o valutazioni. D'altra parte, le piattaforme social generaliste come *Instagram*, *Facebook* e *X* utilizzano architetture orientate alla comunicazione bidirezionale e alla personalizzazione dei contenuti. Queste architetture sfruttano sistemi distribuiti per la gestione del feed e funzionalità interattive come post, like, commenti e messaggistica diretta.

Il design del nuovo sistema di WeCook parte dalle seguenti osservazioni e assunzioni:

- **Familiarità degli utenti con i social network**: concetti come "*Post*", "*Like*", "*Follower*", "*Feed*", sono entrati nella quotidianità, riducendo la curva di apprendimento per l'utilizzo di piattaforme basate su queste dinamiche;
- **Limiti delle piattaforme esistenti**: i blog non favoriscono la creazione di comunità attive e interattive, mentre i social generalisti mancano di un focus tematico specifico, diluendo l'esperienza culinaria.
- **Esigenza di una piattaforma tematica**: un sistema dedicato alla cucina e alla condivisione di ricette può rispondere meglio ai bisogni specifici degli utenti, offrendo funzionalità pensate per valorizzare l'aspetto culinario e culturale.

Il nuovo sistema affronterà le seguenti problematiche riscontrate nelle architetture esistenti:

- **Mancanza di strutturazione dei contenuti**: nei social generalisti, le ricette sono difficili da cercare e organizzare, mentre nei blog manca l'interattività e lo scambio diretto tra utenti;
- **Accessibilità per principianti**: molte piattaforme esistenti non sono progettate per supportare utenti con poca esperienza culinaria, mancando di tutorial dettagliati o di modalità interattive per imparare;

Con queste considerazioni come base, WeCook adotta un'architettura software che combina le dinamiche interattive dei social network con la specializzazione e l'organizzazione tipica dei sistemi tematici.

3. Architettura Software Proposta

3.1 Overview

Questa sezione offre una panoramica generale dell'architettura software del sistema, evidenziando la suddivisione funzionale tra i diversi sottosistemi. Ogni componente è progettato per gestire specifiche funzionalità, contribuendo in modo integrato al funzionamento complessivo della piattaforma.

3.2 Decomposizione del sottosistema

Come prima cosa identifichiamo i sottosistemi basandoci sui requisiti funzionali di WeCook e dall'analisi dei modelli. L'obiettivo è quello di dividere il sistema in componenti autonomi che possono essere gestiti in modo indipendente.

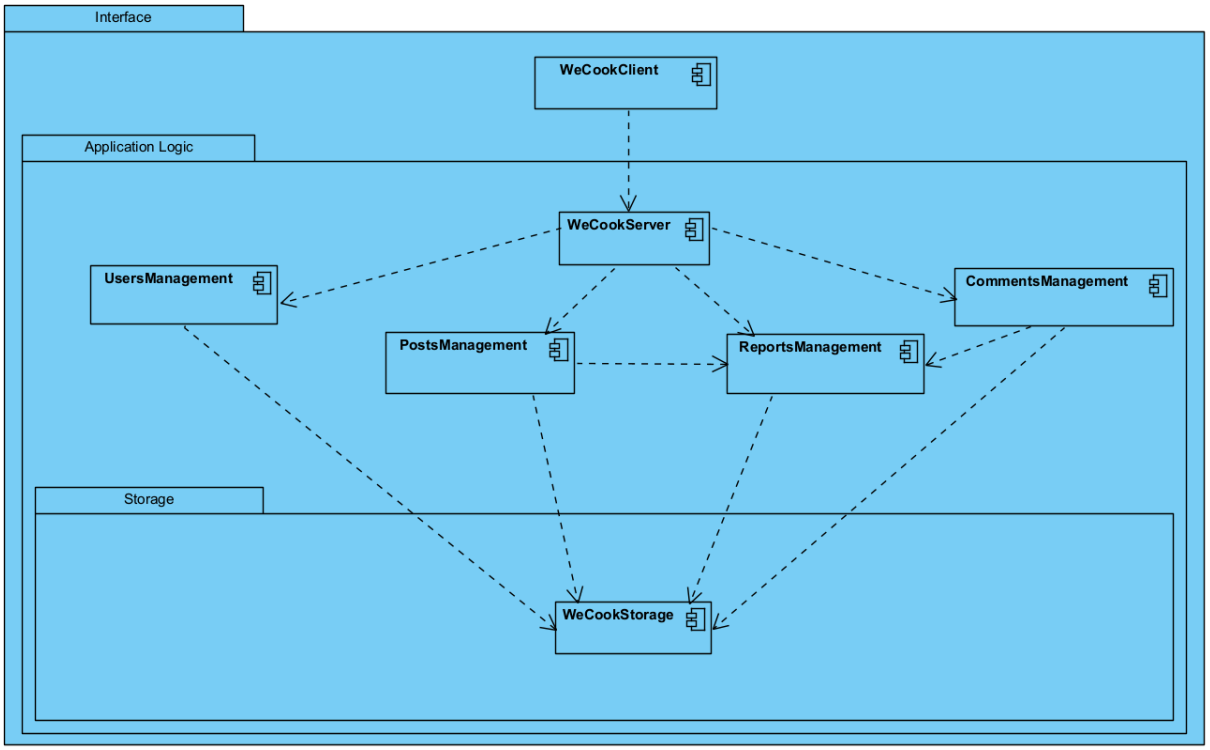
Come prima operazione distinguiamo due sottosistemi principali che fanno parte di WeCook:

- Una parte di organizzazione della visualizzazione dei post, che è responsabile del coordinamento degli utenti per la pubblicazione di un post che poi sarà visto da tutti gli utenti;
- Una parte di interazione tra utenti, responsabile del coordinamento degli utenti per le interazioni con gli altri utenti.

Per la parte relativa alla visualizzazione del post, selezioniamo uno stile architetturale three-tier, composto da:

- *WeCookClient*: Fornisce il front-end, l'interfaccia con cui gli utenti interagiscono. È il punto di accesso per svolgere tutte le attività principali del sistema: creare un post, lasciare un commento, navigare tra i contenuti e segnalare eventuali problemi;
- *WeCookServer*: Si tratta del centro della logica applicativa. Funziona come intermediario tra il client e i sottosistemi che gestiscono le funzionalità specifiche. Si occupa di garantire che ogni richiesta venga gestita correttamente e al suo interno, troviamo sottosistemi specializzati:
    - *UsersManagement*, che gestisce la registrazione, il login e tutte le operazioni legate ai profili utente;
    - *PostsManagement*, responsabile della creazione, modifica e gestione dei post condivisi dagli utenti;
    - *CommentsManagement*, che regola la pubblicazione e l'interazione tra gli utenti e i post attraverso i commenti;
    - *ReportsManagement*, che consente la creazione e gestione delle segnalazioni di post e commenti.
- *WeCookStorage*: Il sottosistema che si occupa di memorizzare in modo permanente i dati della piattaforma. Tutte le informazioni, dagli utenti ai post, dai commenti alle segnalazioni, sono archiviate in modo sicuro e facilmente accessibile, garantendo un'efficace gestione della persistenza.

Dal punto di vista delle interazioni, il WeCookClient comunica costantemente con il WeCookServer per tutte le operazioni applicative, mentre il WeCookServer si affida al WeCookStorage per la gestione e l'archiviazione dei dati.



Come prima cosa identifichiamo i sottosistemi prendendo come riferimento i requisiti funzionali di WeCook e dall'analisi dei modelli. L'obiettivo è quello di dividere il sistema in componenti autonomi che possono essere gestiti in modo indipendente.

3.3 Hardware/Software mapping

WeCook è un sistema distribuito progettato per consentire l'accesso alla piattaforma da qualsiasi località, garantendo un'esperienza utente continua e senza vincoli geografici. L'architettura del sistema si basa su due principali categorie di nodi:

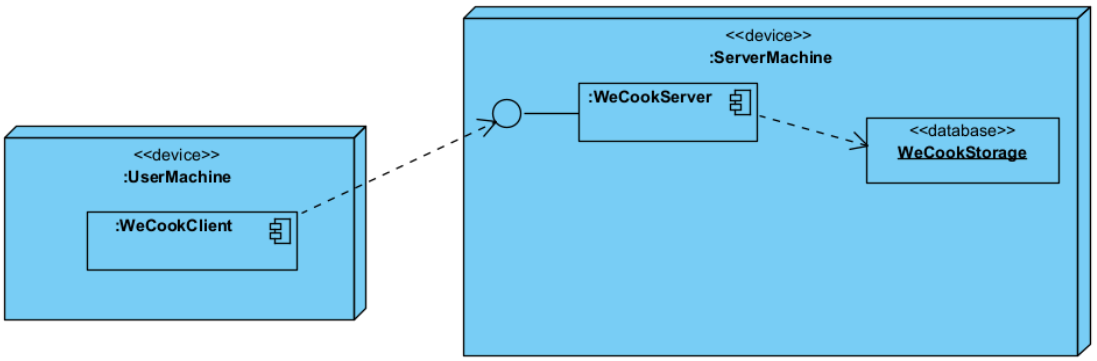
1. **UserMachine**: Rappresenta i dispositivi utilizzati dagli utenti per interagire con la piattaforma. Questi nodi si occupano dell'interfaccia utente, consentendo agli utenti di effettuare operazioni quali la visualizzazione, l'inserimento di contenuti e la ricezione di notifiche. Il software client comunica con i server di WeCook tramite richieste HTTPS, garantendo la sicurezza delle trasmissioni dei dati. Le UserMachine gestiscono operazioni come:
    a. Navigazione delle ricette e contenuti.
    b. Pubblicazione di ricette, commenti.
    c. Notifiche.
2. **ServerMachine**: Svolge un ruolo critico nell'infrastruttura, ospitando la logica applicativa, gestendo la persistenza dei dati tramite il database e processando le richieste provenienti dalle UserMachine. Questo livello garantisce il funzionamento centralizzato e scalabile del sistema, supportando le interazioni degli utenti con la piattaforma in modo efficiente.

La distinzione tra questi nodi riflette un modello distribuito che combina la facilità di accesso per gli utenti con una robusta gestione server-side, assicurando prestazioni elevate e una scalabilità adatta alle esigenze di una piattaforma globale.

Le **UserMachine** e le **ServerMachine** comunicano attraverso una connessione crittografata HTTPS, garantendo la sicurezza delle informazioni personali e delle operazioni. Ogni richiesta inviata dagli utenti viene processata dai server che, in base alla natura della richiesta, interagiscono con il database per fornire una risposta rapida e accurata.

3.3.1 Deployment Diagram

Il Deployment Diagram descrive come i componenti software di WeCook vengono distribuiti fisicamente e logicamente sui vari nodi hardware, fornendo una panoramica chiara dell'infrastruttura tecnica del sistema. L'obiettivo principale è garantire una piattaforma scalabile, sicura e ad alte prestazioni, capace di supportare un'esperienza utente fluida e accessibile da qualsiasi dispositivo. Il Deployment Diagram riflette questa configurazione, rappresentando graficamente i principali nodi (client e server) e le loro interconnessioni.



Il backend rappresenta il nucleo centrale dell'intera architettura, essendo responsabile della gestione della logica applicativa e delle interazioni con i dati. Per lo sviluppo di questa componente è stato adottato Jakarta EE, un framework affidabile, ideale per creare applicazioni robuste e scalabili. La definizione degli endpoint RESTful con Jakarta consente di strutturare le API in modo chiaro ed efficiente, favorendo un'interazione fluida tra il backend e il frontend.

Inoltre, per la gestione della persistenza dei dati, è stato scelto Hibernate, una libreria ORM che semplifica notevolmente l'interazione con il database. Grazie a Hibernate, è possibile mappare le entità Java direttamente alle tabelle del database, riducendo la complessità del codice e offrendo funzionalità avanzate come il caching e la gestione automatica delle transazioni. Questo approccio rende il backend non solo altamente performante, ma anche facile da estendere per supportare nuove funzionalità in futuro.

La componente frontend è stata sviluppata utilizzando Angular, una scelta strategica per garantire modularità e riutilizzabilità del codice. Angular permette di adottare un approccio basato sui componenti, dove ogni parte dell'interfaccia è progettata per essere indipendente, autonoma e riutilizzabile. Questo approccio semplifica l'implementazione di nuove funzionalità e rende il sistema più facile da mantenere e aggiornare nel tempo.

Per quanto riguarda la gestione dei dati il sistema si basa su MySQL, che si integra perfettamente con Hibernate, ottimizzando l'esecuzione delle query e garantendo tempi di risposta rapidi anche in presenza di grandi volumi di dati.

Nel complesso, il sistema è stato progettato per offrire un equilibrio tra modularità, efficienza e scalabilità. La combinazione di tecnologie come Jakarta EE, Hibernate, Angular e MySQL rappresenta una soluzione moderna e affidabile, capace di adattarsi alle esigenze di contesti diversi. Grazie alla suddivisione in componenti ben definiti, il sistema risulta facilmente manutenibile e predisposto per eventuali evoluzioni future, sia a livello funzionale che tecnologico.

3.4 Gestione dei dati persistenti

Il sistema memorizza e gestisce le seguenti entità:

- Utente
- Post
- Commento
- Like
- Segnalazione
- Ricetta
- Ingrediente
- Ingrediente ricetta
- Passaggio

Per la gestione dei dati persistenti viene utilizzato un database relazionale per ottimizzare le prestazioni e garantire un accesso efficiente ai dati, è stata scelta l'integrazione con *MySQL*, un database relazionale affidabile che supporta query concorrenti, vincoli di integrità referenziale.

Grazie all'utilizzo di *Hibernate*, una libreria ORM, che consente il mapping diretto tra le entità Java e le tabelle del database, l'interazione con MySQL risulta semplificata permettendo di astrarre i dettagli specifici del database, garantendo un backend performante e facilmente estensibile.

Nel caso di dati binari come immagini associate alle entità, il sistema non memorizza direttamente i file all'interno del database come BLOB, poiché questa soluzione potrebbe introdurre overhead significativi e ridurre le prestazioni. In alternativa, le immagini vengono salvate in una directory dedicata gestita dal server. Il database conserva invece il riferimento al file attraverso il path completo, consentendo un accesso rapido e riducendo l'impatto sulle risorse del sistema.

Nel complesso, il design adottato garantisce un equilibrio ottimale tra prestazioni e manutenibilità, mantenendo al tempo stesso flessibilità per rispondere alle esigenze future del sistema.

3.5 Controlli di accesso e sicurezza

WeCook è un sistema multiutente progettato per gestire una varietà di attori, ciascuno dotato di permessi e responsabilità distinti in base alle azioni che è autorizzato a compiere. Ogni utente è associato a un ruolo specifico che determina l'accesso alle diverse funzionalità del sistema e le operazioni che può eseguire. Questa configurazione consente di garantire una chiara separazione dei compiti e un controllo rigoroso sugli accessi, migliorando la sicurezza e la gestione operativa. Di seguito viene mostrata una tabella che riassume le operazioni consentite sulle entità da parte di ciascun attore del sistema:

	Ospite	**Standard**	**Moderatore**
Registrazione	X		
Autenticazione		X	X
Pubblicazione post		X	
Pubblicazione commento		X	
Lasciare un like ad un post		X	
Segnalazione post		X	
Segnalazione commento		X	
Ricerca ricetta		X	
Ricerca utenti		X	
Eliminazione post			X
Eliminazione commento			X
Visualizzazione post		X	X
Visualizzazione commento		X	X
Visualizzazione profilo utente		X	X
Salvataggio post		X	

Legenda: X = compie l'operazione;

3.6 Controllo del Software

Analizzando il dynamic model sviluppato durante la fase di analisi dei requisiti per WeCook, è stato deciso di adottare un controllo distribuito, con il coordinamento principale affidato a un controller centrale rappresentato dal *WeCookServer*. Questo approccio garantisce una gestione chiara delle interazioni tra i componenti del sistema e facilita la modularità del software.

Il sistema WeCook, progettato per gestire in modo efficiente le richieste degli utenti attraverso un'applicazione web, segue un flusso di controllo event-driven. Quando l'utente interagisce con l'interfaccia Web, le richieste vengono gestite da un Web Server, che funge da punto di accesso.

Il Web Server inoltra le richieste ad uno dei moduli di controllo che gestiscono logica applicativa (*UserManagement*, *PostsManagement*, *CommentsManagement* e *ReportsManagement*), coordina gli oggetti boundary e le entità per gestire lo stato del sistema.

Gli oggetti boundary sono progettati per gestire esclusivamente dati temporanei relativi alla richiesta in corso.

Ogni sessione utente disporrà di un proprio oggetto di controllo dedicato, evitando conflitti derivanti da richieste simultanee che coinvolgono lo stesso oggetto.

Gli oggetti entità forniranno accesso al loro stato interno esclusivamente tramite metodi dedicati. L'accesso concorrente alle entità sarà gestito tramite meccanismi di sincronizzazione forniti dal framework di persistenza utilizzato (Hibernate).

Poiché WeCook è pensato per operare in un contesto concorrente, si occuperà di gestire automaticamente la concorrenza e le transazioni. Questo approccio consente di ridurre la complessità del codice applicativo e garantisce che le operazioni di modifica dello stato del sistema secondo un approccio ACID.

L'architettura software di WeCook adotta un modello di controllo robusto, progettato per garantire efficienza e affidabilità in un contesto multiutente. La gestione della concorrenza e l'isolamento delle sessioni migliorano l'esperienza utente, riducendo il rischio di errori dovuti a conflitti di accesso.

3.7 Condizioni di confine

Le condizioni di confine comprendono i processi di inizializzazione, terminazione e gestione dei fallimenti del sistema, fondamentali per garantire un funzionamento affidabile e continuo della piattaforma.

L'inizializzazione del sistema avviene automaticamente all'avvio del *WeCookServer*. Durante questa fase, il sistema:

- Carica la configurazione necessaria dai file di setup;
- Verifica la disponibilità delle risorse critiche, come connessioni al database e accesso ai percorsi di archiviazione per i file multimediali;
- Registra i servizi RESTful e rende operativi gli endpoint, garantendo che il sistema sia pronto a rispondere alle richieste degli utenti.

In fase di terminazione il *WeCookServer* si occupa di:

- Scollega in modo sicuro le connessioni aperte al database;
- Rilascia eventuali risorse allocate, come thread o file temporanei;
- Salva eventuali log di terminazione necessari per analisi successive.

Per affrontare eventuali fallimenti, il sistema implementa meccanismi di gestione degli errori e di recupero:

- Durante tutta l'esecuzione il sistema crea dei file log dettagliati contenenti informazioni relativamente ad azioni effettuate, da quale entità del sistema, l'orario, l'esito di tale azione e eventuali errori;
- Tutti i processi di comunicazione con il database che prevedono una modifica dei dati sono gestiti in modo transazionale per garantire la consistenza dei dati e che tutte le operazioni di scrittura vadano a buon fine dall'inizio alla fine, in caso contrario viene garantita l'integrità dei dati annullando la transazione.

Grazie a questa gestione strutturata delle condizioni di confine, il sistema garantisce stabilità, affidabilità e un'esperienza utente ottimale anche in presenza di eventi imprevisti.

4. Servizi dei sottosistemi

Consideriamo i sottosistema: *UserManagement*, *PostsManagement*, *CommentsManagement* e *ReportsManagement.*

UsersManagement:

- Definiamo il *AuthenticationService* per la verifica delle credenziali di accesso e per la registrazione dell'utente;
- Definiamo il *RoleSerivice* per la selezione del ruolo.

PostsManagement:

- Definiamo il *PostService* per creare, eliminare e ottenere informazioni sui post della piattaforma;
- Definiamo il *PostState* per la verifica dei post attivi.

CommentsManagement:

- Definiamo il *CommentService* per creare, eliminare e ottenere informazioni sui commenti della piattaforma;
- Definiamo il *CommentState* per la verifica dei commenti attivi.

ReportsManagement:

- Definiamo il *ReportService* per creare, eliminare e ottenere informazioni sulle segnalazioni della piattaforma;
- Definiamo il *ReportState* per la verifica delle segnalazioni attive.