System Design Document

MeerKat

System Design Document

Versione 0.1.7

Coordinatore del progetto:

Nome	Matricola

Revision History

Data	Versione	Descrizione	Autore
20/11/2024	0.1	Scrittura indice.	Stefano Nicolò , Francesco Giuseppe Trotta, Giuseppe Ballacchino, Gabriel Tabasco
21/11/2024	0.1.1	Persistent data management, edit design goals	Stefano Nicolò Zito
24/11/2024	0.1.2	Global Software Control	Francesco Giuseppe Trotta
24/11/2024	0.1.3	Boundary conditions	Gabriel Tabasco
25/11/2024	0.1.4	Hardware/software mapping	Francesco Giuseppe Trotta
25/11/2024	0.1.5	Subsystem services + Overview 3.1	Giuseppe Ballacchino

Data	Versione	Descrizione	Autore
25/11/2024	0.1.6	Glossario + Revision	Stefano Nicolò , Francesco Giuseppe Trotta, Giuseppe Ballacchino, Gabriel Tabasco
11/02/2025	0.1.7	Revisione Finale Pre-Delivery	Stefano Nicolò Zito, Francesco Giuseppe Trotta

Indice

1. Introduzione

1.1 Purpose of the System

Lo scopo del sistema MeerKat è creare un unico luogo in cui le aziende possano organizzare progetti e gruppi di lavoro, consentendo di gestire e creare task, comunicare con gli altri membri del progetto in modo organizzato e ricevere dati in tempo reale sul progresso del team e sull'andamento del progetto.

1.2 Design Goals

In questa sezione sono presentati gli obiettivi di design e spiegati eventuali trade-off effettuati.

1.2.1 List of design goals

1. **DG_1 tempo di risposta:** Il sistema deve rispondere alle richieste degli utenti in tempi brevi, non superiori ai 10 secondi.
2. **DG_2 Persistenza:** Il Sistema deve garantire la persistenza dei dati in modo organizzato e a lungo termine, anche se questo significa un uso di memoria notevole.
3. **DG_3 Privacy dei dati:** Il sistema deve garantire la privacy dei dati degli utenti, permettendo l'accesso solo a utenti autorizzati e impedendo le diffusione dei dati a terze parti.
4. **DG_4 Robustezza:** Il sistema deve essere in grado di gestire eventuali input errati senza causare malfunzionamenti.
5. **DG_5 Readability:** Il sistema deve essere sviluppato garantendo la leggibilità del codice, per facilitare modifiche future.
6. **DG_6 Tracciabilità dei requisiti:** I requisiti del sistema devono essere tracciabili facilmente attraverso la documentazione.
7. **DG_7 Usability:** Le funzionalità del sistema devono essere presentate in un interfaccia utente di facile comprensione.
8. **DG_8 Transparency:** Il sistema deve essere un sistema distribuito che integri trasparenza di locazione e accesso per i dati che tratta.

1.2.2 Trade-off

- **Funzionalità vs Usabilità:** Il sistema non ha un elevato numero di funzionalità, il focus durante la fase di design fu quello di rendere ciascuna di queste funzionalità quanto più intuitiva e facile da usare possibile, piuttosto che espandere lo scope del sistema.

1.3 Definitions, acronyms, and abbreviations

- **UI: User Interface** - L'interfaccia attraverso cui gli utenti interagiscono con il sistema.
- **API RESTful** - Application Programming Interface - Un insieme di regole e protocolli che definiscono come diverse applicazioni software possono comunicare tra loro.
- **ORM -** Object-relational mapping - Tecnica di programmazione che favorisce l'integrazione di sistemi software aderenti al paradigma della programmazione orientata agli oggetti con sistemi RDBMS.
- **RBAC -** Role-Based Access Control - Modello per autorizzare l'accesso degli utenti finali a sistemi, applicazioni e dati in base al ruolo predefinito dell'utente
- **Hashing -** Processo unidirezionale che trasforma i dati in un valore di hash di lunghezza fissa utilizzando una funzione hash.
- **Event-driven -** Il sistema è guidato dal flusso di eventi all'interno dei componenti interni

1.4 References

1. Problem Statement
2. RAD

1.5 Overview

2. Current system architecture

Nel dominio dei sistemi di gestione progetti, le piattaforme come Trello e Asana adottano architetture basate su microservizi per garantire scalabilità e flessibilità.

Tuttavia, entrambi i sistemi presentano limiti nell'integrazione con soluzioni per esigenze aziendali specifiche. Questo progetto mira a integrare le funzionalità di monitoraggio, offrendo un sistema altamente configurabile.

3. Proposed software architecture

3.1 Overview

La nuova architettura software per **MeerKat** è progettata per integrare funzionalità di gestione dei progetti, comunicazione e monitoraggio delle performance in un'unica piattaforma scalabile e sicura. Si basa su un'architettura **modulare e orientata ai servizi**, che garantisce flessibilità, manutenibilità e un'efficiente distribuzione delle risorse.

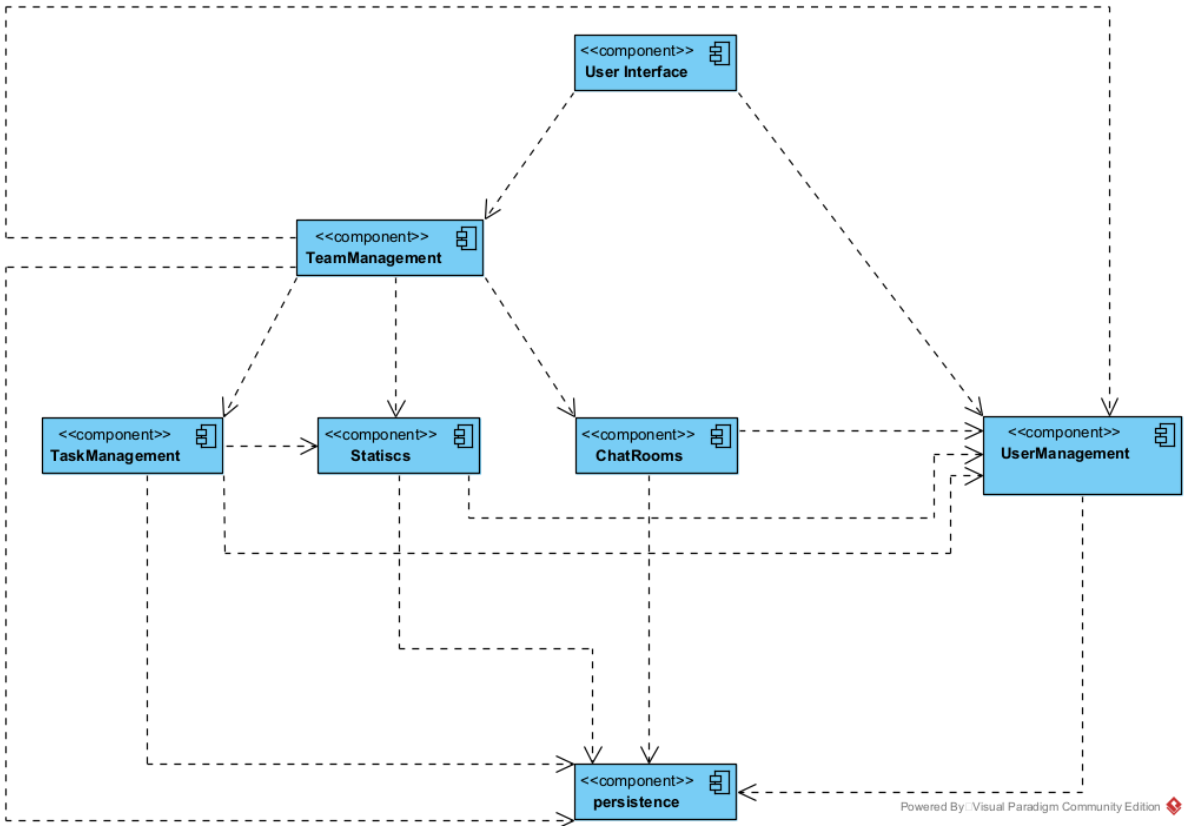
L'architettura è composta da più sottosistemi indipendenti che interagiscono tramite API definite. Questi sottosistemi sono organizzati per separare le responsabilità, ridurre le dipendenze.

Caratteristiche chiave:

- **Modularità**: Ogni sottosistema è progettato per gestire un aspetto specifico del sistema (e.g., gestione dei task, autenticazione).
- **Scalabilità**: Capacità di supportare un numero crescente di utenti e task senza compromettere le prestazioni.
- **Sicurezza**: Uso di meccanismi avanzati per la protezione dei dati, inclusi crittografia e controllo degli accessi.

3.2 Subsystem Decomposition

3.2.1 Subsystem diagram



3.2.2 Subsystem explanation

- **TeamManagement:** gestisce le funzioni relativi alla creazione, eliminazione e modifica dei team di lavoro
- **UserManagement:** gestisce gli account utente, occupandosi di gestire login e registrazione degli utenti.
- **TaskMangement:** gestisce le funzioni riguardanti la creazione, gestione e eliminazione di tasks e tasklists.
- **ChatRooms:** gestisce le chat individuali e di gruppo tra utenti membri dello stesso progetto e fornendo notifiche.
- **Statistics:** gestisce la raccolta e elaborazione dei dati statistici riguardanti i tempi di lavoro e lo stato di completamento del progetto.
- **Persistence:** salva i dati prodotti dagli altri sottosistemi al interno del database, e permette agli altri sottosistemi di accedervi quando richiesto.

3.3 Hardware/software mapping

Componenti software

1. **Interfaccia utente (UI)**

   Gestisce l'interazione con l'utente, presente su client (browser o app mobile).

2. **Gestione attività (Task Management)**

Gestisce la logica delle attività, riutilizzando moduli comuni come database connector o autenticazione.

3. **Gestione delle chat (Chat Rooms)**

   Sistema di messaggistica che riutilizza il framework di comunicazione WebSocket per comunicazioni in tempo reale.

4. **Statistiche (Statistics)**

   Utilizza algoritmi riusabili per analisi dei dati.

5. **Database**

   Archivia i dati delle attività, utenti e chat. Riutilizza schemi di tabelle standard per modularità.

Componenti hardware

1. **Client (PC)**

   Dispositivi personali degli utenti, contenenti solo il frontend per l'interazione.

2. **Backend Server**

   Include nodi multipli per scalabilità e fault tolerance. Ogni nodo backend riutilizza componenti software standard.

3. **Database Server**

   Uno o più nodi che supportano il clustering e il partizionamento per la gestione dei dati in ambienti multi-nodo.

4. **Rete**

   Si occupa della comunicazione tra nodi (backend, database, client), implementando protocolli sicuri come HTTPS.

Problematiche introdotte dai molteplici nodi

- **Sincronizzazione tra i diversi nodi del backend**
  - Distribuzione non uniforme delle richieste
  - Bilanciamento del carico
- **Condivisione delle sessioni**
  - Condivisione dei dati di sessione tra i diversi nodi del backend
- **Scalabilità del database**
  - Sincronizzazione delle query sul database
  - Problemi di gestione della latenza
  - Database con grandi quantità di dati

3.3.6 Interazione tra i Sottosistemi

I sottosistemi comunicano tramite **API RESTful** e sono orchestrati da un **Application Server**, che funge da mediatore per le richieste degli utenti e la gestione delle risorse.

3.4 Persistent data management

3.4.1 Sistema di gestione dei dati persistenti scelto

Come spiegato dal design goal DG_2, il sistema ha bisogno di gestire dati persistenti, per gestire questi dati persistenti abbiamo scelto di usare un **Database Relazionale Distribuito** generato attraverso **ORM.**

3.4.2 Punti a favore

- Il Database è distribuito, cosa fondamentale in quanto più utenti del sistema dovranno avere accesso alle informazioni da macchine diverse, richiedendo anche trasparenza di locazione dei relativi dati;
- Permette maggiore organizzazione dei dati e maggior facilità nel realizzare il controllo sugli accessi, incontrando DG_2 e DG_3;
- Fornisce dei meccanismi di backup che semplificano il mantenimento dei dati gestiti in caso di malfunzionamenti;
- Le funzionalità del sistema prevedono che l'accesso in scrittura della maggior parte dei dati sia possibile solo da parte del project manager del relativo team, facendo in modo che non ci sia bisogno di gestire eccessivamente le condizioni di lock date dalla concorrenza.

3.4.3 Punti a sfavore

Il tempo di risposta di alcune operazioni di un Database potrebbe accrescere all'aumentare al volume di dati contenuti da questo. Probabilmente andando contro il DG_1.

3.5 Access control and security

Il sistema MeerKat implementa un modello di accesso basato su ruoli (**Role-Based Access Control, RBAC**). I principali ruoli definiti sono:

- **Manager**: Ha accesso completo per creare, modificare e gestire progetti, team e task. Può monitorare le performance dei membri e generare report.
- **Membro del Team**: Può visualizzare e aggiornare i task assegnati, avviare il tracciamento del tempo, e comunicare tramite il sistema di messaggistica.

Matrice d'Accesso

Ruolo	Visualizza Task	Edit Task	Crea Progetto	Gestione Utenti	Accesso alle Statistiche
Manager	V	V	V	V	V
Membro del team	V	V (Solo Assegnati)	X	X	V (Limitato)

3.5.1 Autenticazione

Il sistema utilizza un meccanismo di autenticazione per garantire che solo gli utenti autorizzati possano accedere al sistema stesso. Le funzionalità del sistema includono:

- **Registrazione**: Gli utenti forniscono nome, cognome, e-mail e password.
- Politiche di sicurezza delle password, che includono requisiti minimi come lunghezza, utilizzo di caratteri speciali e rotazione periodica.
- **Crittografia Password**: Le password sono memorizzate utilizzando un algoritmo di hashing sicuro.

3.5.2 Protezione dei dati

- **Protezione dell'Accesso ai Progetti**: I progetti e i relativi dati sono accessibili solo ai membri autorizzati.

3.6 Global software control

Il sistema utilizza un modello di controllo **event-driven** per gestire le comunicazioni e le interazioni tra i vari sottosistemi. Le richieste sono inviate dall'interfaccia utente e da eventi interni al sistema, garantendo un flusso coerente e sincronizzato tra i diversi componenti.

Questo tipo di approccio assicura che le operazioni siano gestite in modo efficiente e che i dati siano coerenti, anche in presenza di operazioni concorrenti

3.6.1 Requests Initiation

- **User Request**

  Le richieste degli utenti vengono originate attraverso l'interfaccia utente e inoltrate ai sottosistemi appropriati attraverso chiamate sincrone o asincrone. Ad esempio:

  - Un utente che aggiorna un'attività nel sottosistema **TaskManagement** genera una richiesta che viene reindirizzata ai sottosistemi **TaskStorage** e **Statistics**
  - L'invio di un messaggio in una chat avvia una richiesta al sottosistema **ChatRooms**, che si sincronizza con **ChatStorage** per la persistenza del messaggio.

- **Internal Events**

  Gli eventi interni, come aggiornamenti automatici di statistiche o notifiche di sistema, sono generati dai sottosistemi come risposta a un cambiamento nello stato globale.

  Questi eventi vengono propagati tramite meccanismi di messaggistica.

3.6.2 Subsystem Synchronization

I sottosistemi si sincronizzano utilizzando un **gestore di eventi** e **API** per lo scambio di dati. Ogni sottosistema è responsabile di:

- **Ascoltare** gli eventi rilevanti generati dagli altri sottosistemi.

- **Inviare** notifiche in caso di aggiornamenti significativi (ad esempio, un nuovo messaggio di chat o un'attività completata).
- **Sincronizzare** i dati con il database centrale in modo asincrono, evitando blocchi del sistema principale.

3.7 Boundary conditions

Per garantire l'affidabilità della piattaforma, è essenziale definire delle boundary conditions che coprono le fasi di avvio, inizializzazione e terminazione del sistema. L'obiettivo è preservare l'integrità dei dati degli utenti e garantirne la disponibilità in caso di interruzioni.

Fase di Inizializzazione:

- Attivazione del thread di Logging per il monitoraggio diagnostico
- Caricamento della configurazione precedente o predefinita
- Verifica dell'esistenza delle connessioni critiche (database, servizi esterni)
- Attivazione degli endpoint per la comunicazione con i client

Fase di Terminazione:

- Chiusura sicura delle connessioni critiche
- Rilascio controllato delle risorse allocate (thread, file aperti)

Gestione delle Interruzioni Improvvise:

- Transazioni del database con meccanismo di commit/rollback per garantire l'integrità dei dati
- Thread di Logging per identificare e potenzialmente risolvere le cause dell'interruzione
- Meccanismo di ritrasmissione lato client (3 tentativi con intervallo di 10-15 secondi prima della segnalazione di errore)

  Questi accorgimenti assicurano l'affidabilità del sistema e ottimizzano l'esperienza utente in caso di imprevisti.

4. Subsystem services

4.1 UserManagement Services

- **User Registration Service**: Permette la creazione di nuovi account utente con verifica via e-mail.
- **Authentication Service**: Gestisce il login, logout e l’autenticazione a due fattori (2FA).
- **Authorization Service**: Verifica i permessi degli utenti in base ai ruoli e alle risorse richieste.

4.2 Project Management Services

- **Project Creation and Management Service**: Consente ai manager di creare, modificare e rimuovere progetti.
- **Progress Reporting Service**: Genera report sullo stato di avanzamento dei progetti e sulle prestazioni del team.
- **Team Management Service**: Permette l'aggiunta e la rimozione di membri ai progetti e ai team.

4.3 Task Management Services

- **Task Management Service**: Gestisce la creazione, modifica, rimozione dei task e l'aggiornamento del loro stato.
- **Tasklist Management Service**: Gestisce la creazione, modifica e rimozione di liste di task.
- **Task assignement Service**: Gestisce l'assegnazione di task alle liste corrispondente e gli spostamenti di task da una lista al altra.

4.4 Communication Services

- **Group Messaging Service**: Consente la creazione di chat di gruppo con discussioni tematiche (e.g., thread legati a progetti specifici).
- **Direct Messaging Service**: Supporta conversazioni individuali tra utenti.
- **Notification Service**: Invia notifiche in tempo reale per messaggi ricevuti, nuovi task assegnati o aggiornamenti di progetto.

4.5 Persistence Services

- **User Data Management Service**: Archivia informazioni relative agli utenti, incluse credenziali e ruoli.
- **Project Data Management Service**: Gestisce dati relativi a progetti, team e task.
- **Message Data Management Service**: Conserva i messaggi inviati nelle chat di gruppo o individuali.
- **Backup and Recovery Service**: Esegue backup periodici dei dati e fornisce strumenti di ripristino in caso di incidenti.
- **Data Encryption Service**: Applica algoritmi di crittografia per proteggere i dati sensibili.

4.6. Statistcs Services

- **Project Analytics Service**: Analizza lo stato del progetto e fornisce insight sulle scadenze e sui task completati.
- **User Performance Service**: Monitora e valuta le performance individuali basate sui task completati e sul tempo speso.
- **Timer Service**: Gestisce l'avvio, la pausa e la registrazione del tempo lavorativo.
- **Time Aggregation Service**: Calcola il tempo totale speso su task, progetti e per utente.

- **Performance Analysis Service**: Genera grafici e statistiche basati sui dati raccolti dal tracciamento del tempo.

Glossario

Termine/Acronimo	Definizione
API RESTful	Architettura di comunicazione tra sistemi che utilizza HTTP per inviare e ricevere dati (JSON/XML).
WebSocket	Protocollo full-duplex per la comunicazione in tempo reale tra client e server.
RBAC (Role-Based Access Control)	Modello di controllo degli accessi basato sui ruoli degli utenti.
Backup and Recovery	Processi per creare copie di sicurezza dei dati e ripristinarli in caso di perdita o corruzione.
Distributed Database	Sistema di gestione dati distribuito su più nodi per scalabilità, affidabilità e trasparenza.
DG (Design Goal)	Obiettivi di progettazione per guidare lo sviluppo del sistema.
2FA (Two-Factor Authentication)	Metodo di autenticazione con due livelli di verifica.
CRUD	Operazioni di base sui dati: Create, Read, Update, Delete.
HTTPS	Versione sicura di HTTP con crittografia SSL/TLS per proteggere le comunicazioni.
Team Management	Modulo per creare, gestire e organizzare team di lavoro.
Task Management	Funzionalità per creare, assegnare e monitorare attività nei progetti.
Statistics	Servizio per analizzare e visualizzare performance individuali e di progetto.
Boundary Conditions	Condizioni che definiscono avvio, inizializzazione e terminazione del sistema per affidabilità.
Event-Driven Model	Architettura software in cui le operazioni sono attivate da eventi, come input o cambiamenti di stato.
Persistent Data Management	Sistema per garantire che i dati siano disponibili anche dopo la terminazione del sistema.