Spendly: Documentazione Tecnica

Amin Borqal, Loris Iacoban, Diego Rossi

3 febbraio 2025

Indice

Elenco delle figure

Elenco delle tabelle

1 Iterazione 0

Questa sezione analizza i requisiti iniziali del progetto Spendly.

1.1 Introduzione

Spendly è una web app progettata per semplificare la gestione delle spese personali e di gruppo, offrendo funzionalità che permettono a singoli utenti, famiglie o gruppi di organizzarsi in modo efficiente.

Con Spendly, è possibile:

- Creare gruppi di utenti, dove ciascun membro può registrare le spese effettuate per conto degli altri. L'applicazione calcolerà automaticamente il numero minimo di transazioni necessarie per saldare i debiti all'interno del gruppo.
- Impostare avvisi personalizzati per monitorare il livello di spesa del gruppo, fornendo notifiche utili per evitare superamenti del budget condiviso.

A livello individuale, ogni utente può:

- Visualizzare, registrare e gestire le proprie spese in modo autonomo.
- Creare e monitorare un budget di risparmio per pianificare meglio le proprie finanze personali.

Grazie a queste funzionalità, Spendly rende la gestione economica più semplice, trasparente ed efficace, sia per singoli utenti che per gruppi.

Perché scegliere Spendly?

Spendly è strutturata attorno a una serie di funzionalità che coprono ogni aspetto della gestione delle spese. Queste funzionalità sono state sviluppate con l'obiettivo di offrire la massima flessibilità, sia per gli utenti individuali che per i gruppi di condivisione spese. Spendly è più di una semplice applicazione di gestione delle spese. È una soluzione completa che integra funzionalità avanzate, come il calcolo automatico dei debiti e la gestione centralizzata delle spese. Questo la rende ideale per ogni tipo di utente, dai gruppi di amici che vogliono semplificare la divisione delle spese, alle famiglie che desiderano tenere sotto controllo i propri budget. Con una piattaforma sicura, flessibile e facile da usare, Spendly trasforma il modo in cui le persone gestiscono i loro soldi, riducendo lo stress finanziario e migliorando la trasparenza nelle relazioni economiche. Spendly è lo strumento perfetto per chi desidera una gestione semplice, efficace e organizzata delle spese. Con funzionalità pensate per rispondere alle esigenze di utenti individuali e gruppi, questa web-app rappresenta una soluzione innovativa e accessibile per migliorare la vita quotidiana di chiunque voglia un controllo totale sulle proprie finanze.

1.2 Requisiti

Il sistema deve soddisfare i seguenti requisiti funzionali e non funzionali:

- **Requisiti Funzionali**:
  - Gli utenti devono poter registrarsi e autenticarsi al sistema.
  - Gli utenti devono poter aggiungere, modificare e cancellare le proprie spese.
  - Gli utenti devono poter creare gruppi e gestire le spese condivise.
  - Il sistema deve calcolare automaticamente il bilancio dei debiti tra i membri del gruppo.
  - Il sistema deve inviare notifiche o alert agli utenti quando il loro budget per una determinata categoria di spesa viene superato.
  - Gli utenti devono poter impostare budget personalizzati per categorie specifiche (es. alimentari, trasporti, svago, ecc.).
- **Requisiti Non Funzionali**:
  - L'applicazione deve essere intuitiva e facile da usare.
  - Il sistema deve garantire la sicurezza dei dati personali degli utenti mediante crittografia e autenticazione sicura.
  - Le transazioni e le modifiche devono essere sincronizzate in tempo reale tra tutti i dispositivi degli utenti.
  - L'applicazione deve essere accessibile da diversi dispositivi (smartphone, tablet, desktop).
  - Il sistema deve garantire elevate prestazioni, assicurando una risposta rapida alle richieste degli utenti.

1.3 Casi d'Uso

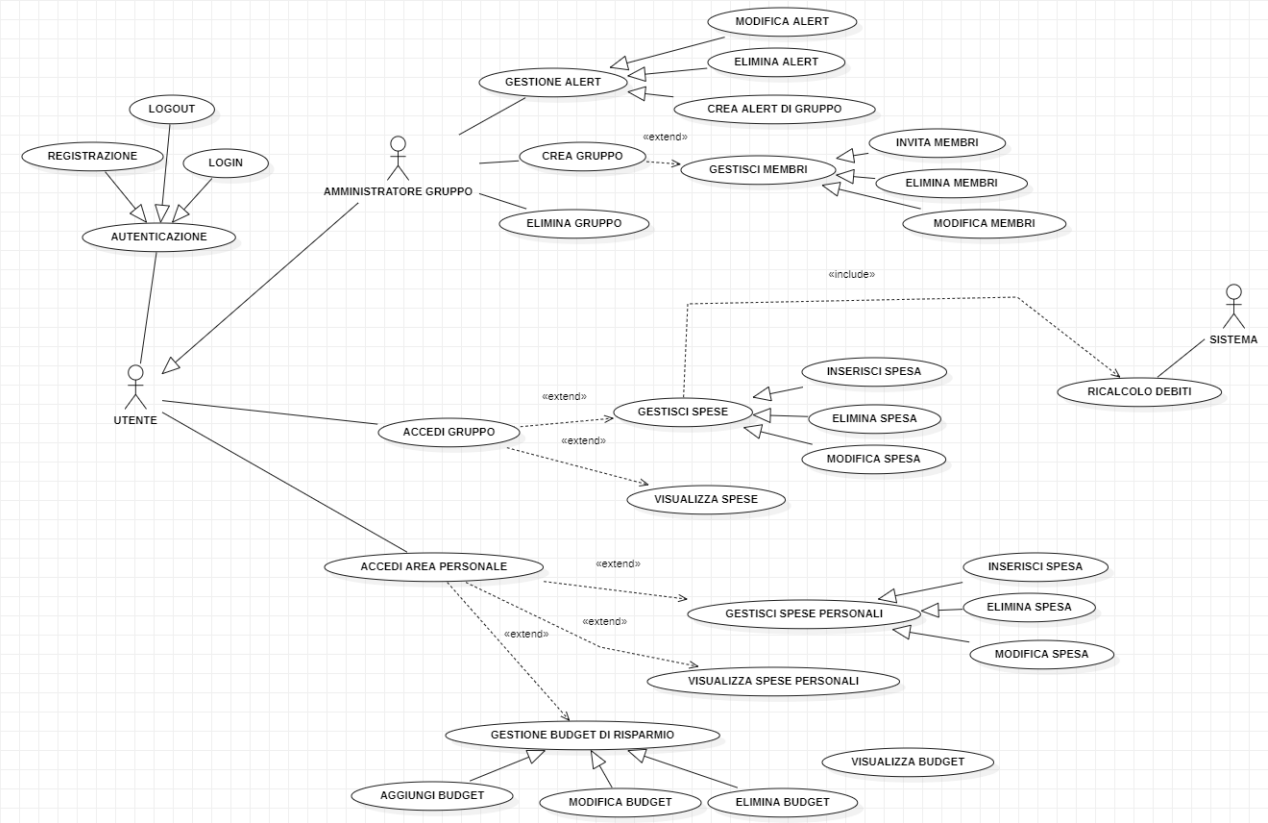


Figura 1: Diagramma casi d'uso

Di seguito sono riportati alcuni casi d'uso principali:

- **UC1 - Login**: Come utente, voglio poter accedere al mio account per gestire le mie spese.
- **UC2 - Registrazione**: Come nuovo utente, voglio potermi registrare al sistema per iniziare a usare la web-app.
- **UC3 - Logout**: Voglio poter terminare la mia sessione.
- **UC4 - Crea alert di gruppo**: Come utente amministratore di un gruppo spesa voglio poter inserire un alert, dove un alert è un avviso che ci permette di avvisare se si sta raggiungendo una soglia limite di spesa.
- **UC5 - Modifica alert**: Voglio poter modificare i valori dell'alert.
- **UC6 - Elimina alert**: Voglio poter eliminare l'alert.
- **UC7 - Crea gruppo**: Voglio poter creare un gruppo di condivisione spese.
- **UC8 - Invita memebri**: Voglio come amministratore invitare membri nel gruppo di spese.
- **UC9 - Elimina membri**: Voglio come amministratore poter eliminare membri del gruppo di spese.

- **UC10 - Modifica membri**: Voglio come amministratore modificare i membri nel gruppo di spese.
- **UC11 - Elimina gruppo**: Voglio poter eliminare un gruppo di condivisione spese.
- **UC12 - Accedi gruppo**: Voglio poter accedere ad un gruppo di condivisione spese.
- **UC13 - Inserisci spesa**: Voglio poter inserire una spesa di gruppo.
- **UC14 - Elimina spesa**: Voglio poter eliminare una spesa di gruppo.
- **UC15 - Modifica spesa**: Voglio poter modificare una spesa di gruppo.
- **UC16 - Visualizza spese**: Voglio poter visualizzare le spesa di gruppo.
- **UC17 - Ricalcolo debiti**: Voglio poter calcolare i debiti.
- **UC18 - Inserisci spesa personale**: Voglio poter inserire una spesa personali.
- **UC19 - Elimina spesa personale**: Voglio poter eliminare una spesa personali.
- **UC20 - Modifica spesa personale**: Voglio poter modificare una spesa personali.
- **UC21 - Visualizza spesa personale**: Voglio poter visualizzare le spesa personali.
- **UC22 - Inserisci budget**: Voglio poter inserire un budget di risparmio.
- **UC23 - Elimina budget**: Voglio poter eliminare un budget di risparmio.
- **UC24 - Modifica budget**: Voglio poter modificare un budget di risparmio.
- **UC25 - Visualizza budget**: Voglio poter visualizzare i budget di risparmio.

1.4 Architettura

L'architettura del sistema è basata su un'architettura MVC (Model View Controller) con le seguenti componenti principali:

- **Frontend**: Implementato in HTML e CSS, il frontend fornisce un'interfaccia utente intuitiva per la gestione delle spese.
- **Database**: Utilizza PostgreSQL come sistema di gestione dei dati.
- **Backend**: Implementato in Spring Boot e include:
  - **Servizio Utente**: Gestisce la registrazione, l'autenticazione e il profilo utente.
  - **Servizio Gestione Spese**: Gestisce le operazioni CRUD sulle spese.
  - **Servizio Gruppi**: Gestisce la creazione e la modifica dei gruppi.
- **Flusso delle richieste**:
  - L'utente interagisce con il frontend (browser) tramite pagine HTML.
  - Le richieste vengono inviate al backend (controller Spring Boot) attraverso HTTP.
  - Il backend elabora la richiesta, utilizza i servizi per applicare la logica di business e accede al database.

– La risposta (dati o una nuova pagina HTML) viene inviata al frontend e presentata all'utente.

1.5 Priorità casi d'uso

I casi d'uso possono essere suddivisi in tre categorie, a seconda della loro priorità nel processo di sviluppo:

- **Coda ad alta priorità:** Contiene i requisiti essenziali per il corretto funzionamento dell'applicativo. Questi includono la creazione dei profili utente, la creazione di nuovi gruppi spese, con le funzionalità annesse(invita,elimina membri ecc ecc) ed infine la gestione spesa con l'algoritmo di calcolo debiti.
- **Coda a media priorità:** Questa coda include funzionalità di supporto, principalmente orientate alla gestione spese personali.
- **Coda a bassa priorità:** Contiene le funzionalità meno rilevanti ossia la tematica del budget , per le quali non è prevista una loro implementazione immediata. Tuttavia, potrebbero essere implementate in futuro, a seconda dell'andamento del progetto.

ID	Titolo
UC1	Login
UC2	Registrazione
UC3	Logout
UC8	Invita memebri
UC9	Elimina membri
UC10	Modica memebri
UC11	Elimina gruppo
UC12	Accedi gruppo
UC13	Inserisci spesa
UC14	Elimina spesa
UC15	Modifica spese
UC16	Visualizza spese
UC17	Ricalcolo debiti

Tabella 1: Coda alta priorità

ID	Titolo
UC4	Crea alert di gruppo
UC5	Modica alert
UC6	Elimina alert
UC18	Inserisci spesa personale
UC19	Elimina spesa personale
UC20	Modica spesa personale
UC21	Visualizza spesa personale

Tabella 2: Coda media priorità

ID	Titolo
UC22	Inserisci budget
UC23	Elimina budget
UC24	Modica budget
UC25	Visualizza budget

Tabella 3: Coda bassa priorità

1.6 Topologia del Sistema

La topologia del sistema è rappresentata nel seguente schema:

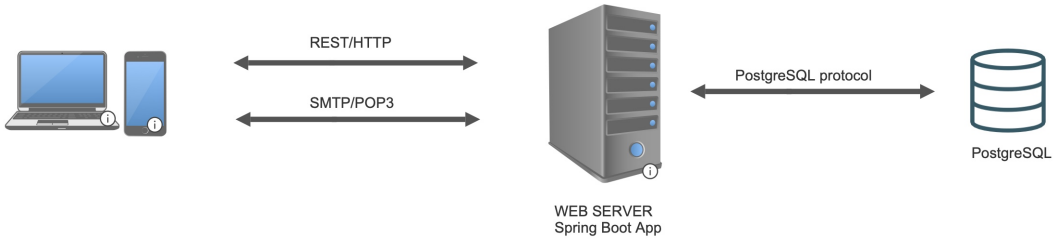


Figura 2: Diagramma dell'architettura della webapp

Il sistema è basato su un'architettura **Client-Server** in cui i client interagiscono con un **web server** sviluppato con il framework **Spring Boot**. I client possono essere:

- **Applicazione Web**: gli utenti accedono alla webapp tramite un browser, comunicando con il server attraverso **API REST/HTTP**.
- **Applicazione Mobile**: gli utenti possono interagire con il sistema tramite un'app mobile che utilizza **REST/HTTP** per le richieste e **SMTP/POP3** per le notifiche via email.

Il web server, sviluppato con **Spring Boot**, gestisce le richieste provenienti dai client e si interfaccia con un database **PostgreSQL** tramite il **protocollo PostgreSQL**, garantendo la persistenza dei dati.

1.6.1 Gestione delle comunicazioni

Il sistema sfrutta il protocollo **REST/HTTP** per lo scambio di dati in formato **JSON**, garantendo interoperabilità con diverse piattaforme. Inoltre, per le notifiche agli utenti, viene utilizzato il protocollo **SMTP/POP3**.

1.6.2 Architettura a Microservizi

L'architettura del sistema segue il paradigma dei **microservizi**, in cui ogni funzionalità del web server è implementata come un servizio indipendente. Questo approccio consente maggiore scalabilità, modularità e manutenibilità del sistema.

1.7 Strumenti Utilizzati

Lo sviluppo del progetto è supportato da una serie di strumenti software che facilitano la progettazione, lo sviluppo, il testing e la gestione del codice. Di seguito, una panoramica dettagliata dei tool impiegati:

- **Visual Studio Code**: IDE (Integrated Development Environment) scelto per lo sviluppo del progetto. VS Code offre un ambiente di sviluppo leggero ma potente, con un ampio supporto per il linguaggio Java e numerose estensioni utili. Tra i plugin utilizzati troviamo:
  - **Spring Boot Extension Pack**: per il supporto avanzato nello sviluppo di applicazioni Spring Boot.
  - **Java Extension Pack**: per il supporto completo al linguaggio Java, con funzionalità di autocompletamento, refactoring e debugging.
- **Spring Boot**: Framework Java basato su Spring, progettato per lo sviluppo di applicazioni web scalabili e strutturate secondo l'architettura a microservizi. Grazie alla sua configurazione automatica e al supporto integrato per REST API, Spring Boot semplifica la gestione del backend e garantisce un'elevata manutenibilità del codice.
- **PostgreSQL**: Sistema di gestione di database relazionale (RDBMS) scelto per la sua affidabilità, scalabilità e aderenza agli standard SQL. PostgreSQL supporta transazioni ACID (Atomicità, Coerenza, Isolamento, Durabilità) ed è ottimizzato per operazioni complesse e interrogazioni avanzate. Il database è stato configurato per garantire prestazioni elevate e sicurezza dei dati.
- **Postman**: Strumento essenziale per il testing delle API REST sviluppate con Spring Boot. Consente di effettuare richieste HTTP, validare le risposte del server e automatizzare test, facilitando così il debug e l'integrazione tra i diversi componenti del sistema.
- **Grok AI**: Tecnologia avanzata per la generazione automatica di immagini basata su intelligenza artificiale. Grok AI viene utilizzato per creare rappresentazioni visive intuitive e schematiche di concetti complessi, supportando la documentazione e la comunicazione grafica del progetto.
- **JAutoDoc**: Plugin per la generazione automatica della documentazione Javadoc a partire dai commenti presenti nel codice sorgente. Questo strumento semplifica la creazione di documentazione chiara e dettagliata per ogni componente del sistema.
- **JUnit 4**: Framework per il testing unitario in Java, impiegato per validare il corretto funzionamento delle classi e dei metodi sviluppati. L'uso di test automatizzati consente di rilevare tempestivamente eventuali errori e di garantire la robustezza del codice.
- **EclEmma**: Plugin per la misurazione della copertura del codice nei test unitari. Fornisce statistiche dettagliate che permettono di individuare porzioni di codice non coperte dai test, migliorando così la qualità complessiva del software.

- **JGraphT**: Libreria Java dedicata alla modellazione e alla gestione di strutture dati basate su grafi. Utilizzata per la rappresentazione e la manipolazione di relazioni complesse all'interno del sistema.
- **STAN4J**: Software per l'analisi statica del codice Java, impiegato per individuare problemi di progettazione, dipendenze indesiderate e violazioni dei principi di modularità.
- **GitHub**: Piattaforma per il versionamento del codice basata su Git, utilizzata per il controllo delle versioni e la collaborazione tra gli sviluppatori. Grazie a GitHub, è possibile tracciare le modifiche al codice, gestire le revisioni e garantire un workflow ordinato ed efficiente.
- **GitHub Desktop**: Applicazione con interfaccia grafica che semplifica l'interazione con il repository GitHub direttamente dal PC. Permette di eseguire operazioni di commit, push e pull senza dover utilizzare la riga di comando, facilitando la gestione del codice per gli sviluppatori.
- **StarUML**: Software utilizzato per la modellazione di diagrammi UML (Unified Modeling Language), fondamentale nella fase di progettazione dell'architettura del sistema. StarUML consente di rappresentare visivamente classi, casi d'uso e flussi operativi.
- **diagrams.net**: Applicazione web per la creazione di diagrammi e schemi con notazione libera. Utilizzata per rappresentare graficamente flussi di dati, processi e architetture software, facilitando la comprensione e la condivisione delle informazioni progettuali.
- **WhatsApp**: Applicazione di messaggistica utilizzata come strumento di comunicazione interna tra i membri del team. Attraverso WhatsApp, è possibile coordinare le attività di sviluppo, discutere problemi tecnici e organizzare riunioni in tempo reale, garantendo un flusso comunicativo rapido ed efficiente.

2 Iterazione 1

In questa iterazione, ci concentriamo sul design e sull'implementazione iniziale del sistema.

2.1 Casi d'Uso Implementati

In questa iterazione sono stati sviluppati i seguenti casi d'uso ritenuti prioritari per lo sviluppo di **Spendly**.

2.1.1 UC1: Login

Attori: Utente, Sistema.

Descrizione: L'utente può autenticarsi nel sistema per accedere al proprio account.

Flusso degli eventi:

1. L'utente accede alla pagina di login.
2. Inserisce email e password.
3. Il sistema verifica le credenziali e autentica l'utente.
4. L'utente viene reindirizzato alla dashboard.

2.1.2 UC2: Registrazione

Attori: Utente, Sistema.

Descrizione: Un nuovo utente può registrarsi creando un account.

Flusso degli eventi:

1. L'utente accede alla pagina di registrazione.
2. Inserisce nome, email e password.
3. Il sistema verifica che l'email non sia già registrata.
4. Se la verifica è superata, il sistema crea l'account e lo memorizza.
5. L'utente viene reindirizzato alla dashboard.

2.1.3 UC3: Logout

Attori: Utente, Sistema.

Descrizione: L'utente può terminare la sessione ed effettuare il logout.

Flusso degli eventi:

1. L'utente clicca su "Logout".
2. Il sistema invalida la sessione e mostra la schermata di login.

2.1.4 UC7: Creazione Gruppo

Attori: Utente amministratore, Sistema.

Descrizione: L'utente può creare un nuovo gruppo di spese per la condivisione con altri membri.

Flusso degli eventi:

1. L'utente clicca su "Crea Gruppo".
2. Inserisce il nome del gruppo e una descrizione opzionale.
3. Il sistema crea il gruppo e assegna l'utente come amministratore.
4. L'utente viene reindirizzato alla pagina del gruppo.

2.1.5 UC8: Invita Membri

Attori: Utente amministratore, Sistema.

Descrizione: L'amministratore di un gruppo può invitare altri utenti a unirsi al gruppo di spese.

Flusso degli eventi:

1. L'amministratore accede alla pagina del gruppo.
2. Clicca su "Invita Membri" e inserisce l'email degli utenti da invitare.
3. Il sistema invia un'email con l'invito e memorizza la richiesta.
4. Gli utenti ricevono l'invito e possono accettarlo per entrare nel gruppo.

2.1.6 UC9: Elimina Membri

Attori: Utente amministratore, Sistema.

Descrizione: L'amministratore di un gruppo può rimuovere un membro dal gruppo.

Flusso degli eventi:

1. L'amministratore accede alla lista dei membri del gruppo.
2. Seleziona il membro da rimuovere e clicca su "Elimina".
3. Il sistema rimuove il membro e aggiorna la lista.

2.1.7 UC10: Modifica Membri

Attori: Utente amministratore, Sistema.

Descrizione: L'amministratore di un gruppo può modificare i dettagli dei membri (ad esempio assegnare nuovi ruoli).

Flusso degli eventi:

1. L'amministratore accede alla lista dei membri.
2. Seleziona un membro e modifica i dettagli (es. ruolo nel gruppo).

3. Il sistema aggiorna i dati e notifica il cambiamento.

2.1.8 UC11: Eliminazione Gruppo

Attori: Utente amministratore, Sistema.

Descrizione: L'amministratore di un gruppo può eliminare definitivamente un gruppo di spese.

Flusso degli eventi:

1. L'amministratore accede alle impostazioni del gruppo.
2. Clicca su "Elimina Gruppo".
3. Il sistema chiede conferma prima di procedere.
4. Se confermato, il gruppo e tutte le sue spese vengono eliminate.

2.1.9 UC12: Accesso a un Gruppo

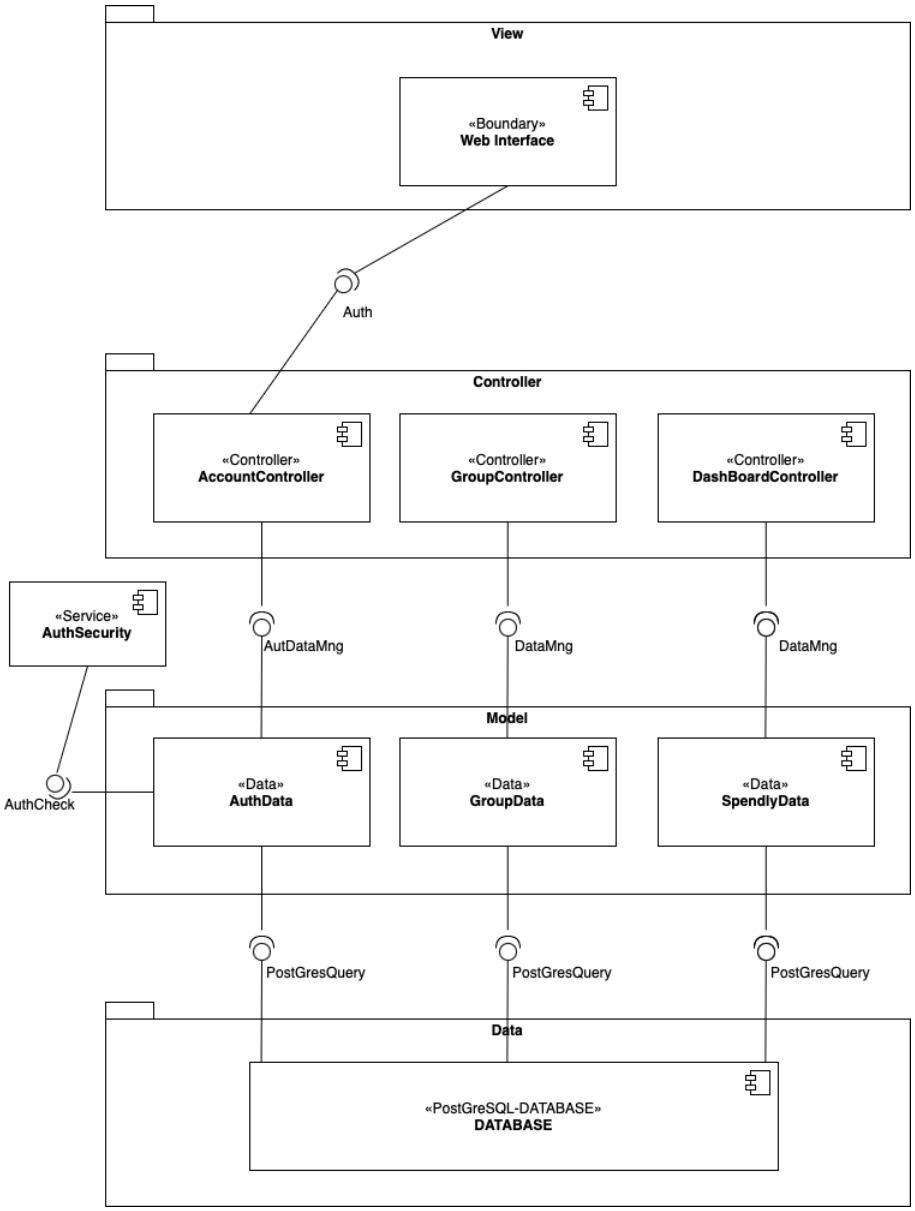
Attori: Utente, Sistema.

Descrizione: Un utente può accedere a un gruppo di spese a cui è stato invitato.

Flusso degli eventi:

1. L'utente riceve un invito via email o notifica nell'app.
2. Clicca sul link di invito e accede alla web-app.
3. Il sistema verifica la validità dell'invito e aggiunge l'utente al gruppo.
4. L'utente viene reindirizzato alla pagina del gruppo.

2.2 UML Component Diagram



Partendo dai casi d'uso selezionati per questa iterazione e procedendo con l'utilizzo delle euristiche di design, è stato possibile progettare l'architettura software del sistema **Spendly**. I componenti sono organizzati secondo il pattern **MVC (Model-View-Controller)**, con la suddivisione in:

- **Boundary** - Interfaccia utente, responsabile dell'interazione con l'utente finale.
- **Controller** - Gestione logica di business.
- **Model** - Gestione dei dati e accesso al database.
- **Service** - Servizi di sicurezza e autenticazione.
- **Database** - PostgreSQL.

2.3 UML Class Diagram per tipi di dato