# Spendly: Documentazione Tecnica

Amin Borqal, Loris Iacoban, Diego Rossi

January 17, 2025

## Contents



## List of Figures



## List of Tables

# Iterazione 0

Questa sezione analizza i requisiti iniziali del progetto Spendly.

## 0.1 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.
- **Requisiti Non Funzionali**:
  - L'applicazione deve essere intuitiva e facile da usare.
  - Il sistema deve garantire la sicurezza dei dati personali degli utenti.
  - Le transazioni devono essere sincronizzate in tempo reale tra tutti i dispositivi degli utenti.

## 0.2 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.

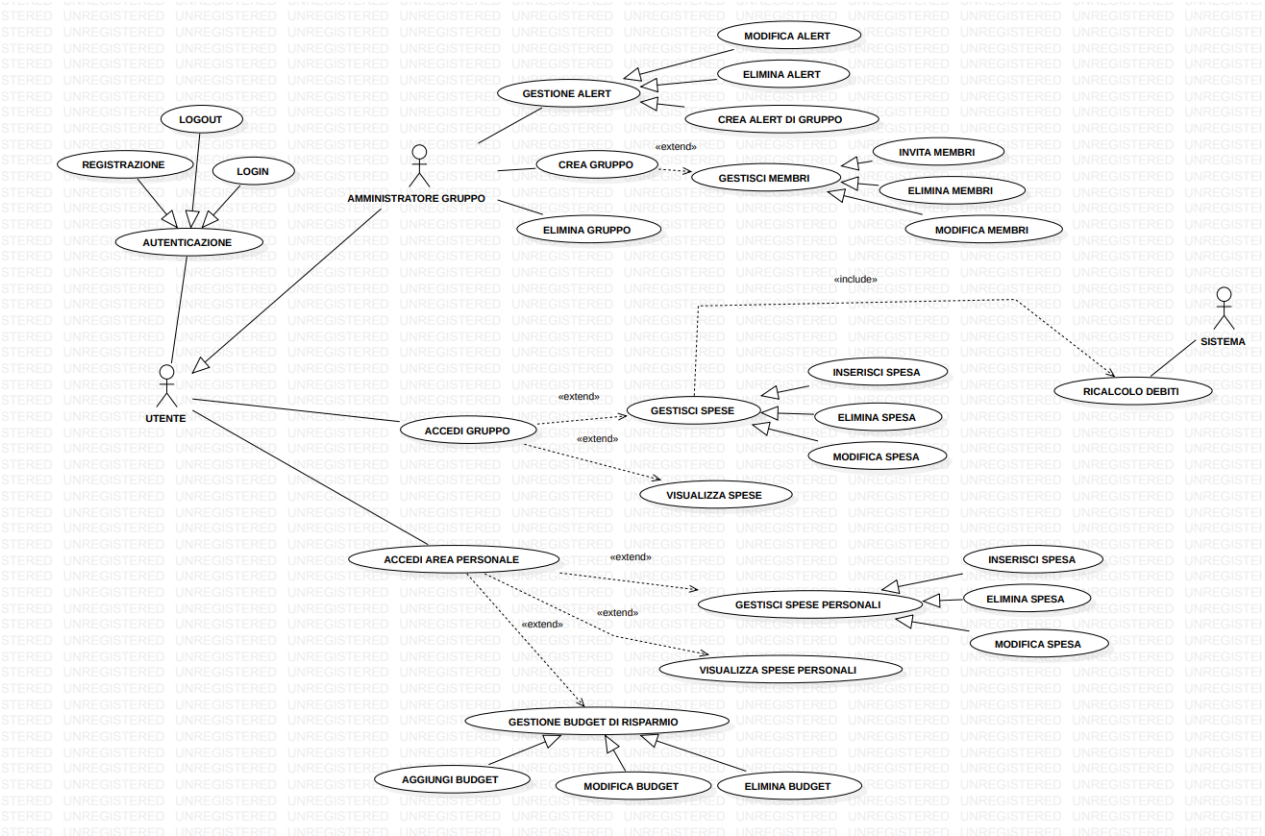


Figure 1: Diagramma casi d'uso

- **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.

## 0.3 Architettura

L'architettura del sistema è basata su un'architettura a microservizi con i seguenti componenti principali:

- **Client/Frontend**: Implementato in React.js, il frontend fornisce un'interfaccia utente intuitiva per la gestione delle spese.
- **API Gateway**: Un punto di ingresso unico per tutte le richieste del client, responsabile dell'instradamento verso i microservizi corretti.
- **Backend (Microservizi)**: I microservizi sono sviluppati in Spring Boot e includono:
  - **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.
- **Database**: Utilizzo di PostgreSQL per la gestione dei dati relazionali.

## 0.4 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 |

Table 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 |

Table 2: Coda media priorità

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

Table 3: Coda bassa priorità

## 0.5 Topologia del Sistema

La topologia del sistema è rappresentata nel seguente schema:

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

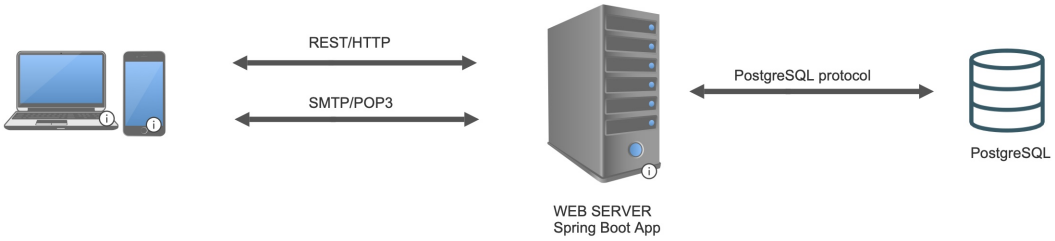


Figure 2: Diagramma dell'architettura della webapp

si interfaccia con un database **PostgreSQL** tramite il **protocollo PostgreSQL**, garantendo la persistenza dei dati.

### 0.5.1 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**.

### 0.5.2 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.

## Tool Chain

Lo sviluppo del progetto è supportato dai seguenti tool:

- **Eclipse**: IDE per lo sviluppo software utilizzato per la scrittura di tutto il codice **(Java)** del sistema. Alcuni degli altri tool utilizzati sono integrati con Eclipse.
- **Spring Boot**: framework Java per lo sviluppo di applicazioni web basate sui microservizi.
- **MongoDB Atlas**: cloud database non relazionale (i dati sono salvati come documenti JSON). Il cluster utilizzato si appoggia ad un server **AWS**.
- **JAutoDoc**: plugin di Eclipse per la generazione dei **Javadoc** che permettono di generare la documentazione del codice Java a partire dai commenti del codice.
- **JUnit 4**: framework per i test di unità in Java.
- **EclEmma**: plugin di Eclipse per la verifica della copertura del codice.
- **JGraphT**: libreria Java per la modellazione di grafi.
- **STAN4J**: software per l'analisi statica di progetti Java.
- **GitHub**: piattaforma per il versionamento basata su Git. È stato utilizzato anche GitHub Desktop che permette di interagire dal proprio PC con GitHub utilizzando una semplice GUI.
- **StarUML**: software per la creazione di modelli UML.

- **diagrams.net**: software online per la creazione di modelli in notazione libera.

# Iterazione 1

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

# Conclusioni

Il progetto Spendly rappresenta un'applicazione concreta delle conoscenze tecniche e pratiche, offrendo un esempio di sviluppo software iterativo e modulare.