Progetto ingegneria del software "Lavoratori stagionali"

Enrico Bragastini
Davide Bianchini
Andrea Mafficini

13 febbraio 2023

Indice

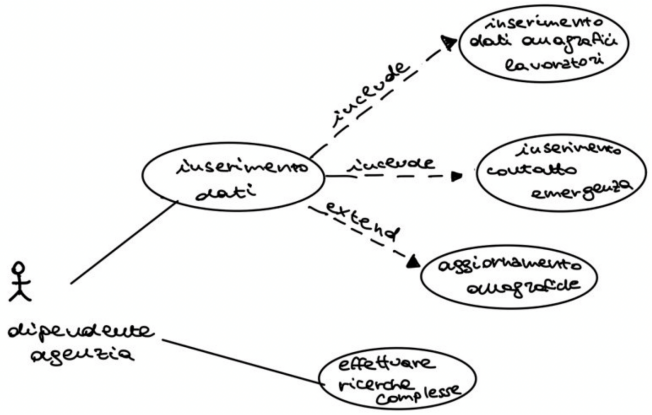
Specifiche dei casi d'uso

0.1 Note generali

Si vuole progettare un sistema informatico di una agenzia che fornisce servizi di supporto alla ricerca di lavoro stagionale. I lavoratori interessati possono iscriversi al servizio, rivolgendosi agli sportelli dell'agenzia. Il sistema deve permettere la gestione delle anagrafiche e la ricerca di lavoratori stagionali, nei settori dell'agricoltura e del turismo. I responsabili del servizio, dipendenti dell'agenzia, inseriscono i dati dei lavoratori. Per ogni lavoratore vengono memorizzati i dati anagrafici (nome, cognome, luogo e data di nascita, nazionalità), indirizzo, recapito telefonico personale (se presente), email, le eventuali specializzazioni/esperienze precedenti (bagnino, barman, istruttore di nuoto, viticultore, floricultore), lingue parlate, il tipo di patente di guida e se automunito. Sono inoltre memorizzati i periodi e le zone (comuni), per i quali il lavoratore è disponibile. Di ogni lavoratore si memorizzano anche le informazioni di almeno una persona da avvisare in caso di urgenza: nome, cognome, telefono, indirizzo email. I dipendenti dell'agenzia devono autenticarsi per poter accedere al sistema e inserire i dati dei lavoratori. Il sistema permette ai dipendenti dell'agenzia di aggiornare le anagrafiche con tutti i lavori che i lavoratori stagionali hanno svolto negli ultimi 5 anni. Per ogni lavoro svolto vanno registrati: periodo, nome dell'azienda, mansioni svolte, luogo di lavoro, retribuzione lorda giornaliera. Per i dipendenti dell'agenzia si memorizzano i dati anagrafici, l'indirizzo email, il telefono e le credenziali di accesso (login e password). Una volta registrate le informazioni sui lavoratori, il personale dell'agenzia può effettuare ricerche rispetto a possibili profili richiesti. In particolare, il sistema deve permettere ai dipendenti di effettuare ricerche per lavoratore, per lingue parlate, periodo di disponibilità, mansioni indicate, luogo di residenza, disponibilità di auto/patente di guida. Il sistema deve permettere di effettuare ricerche complesse, attraverso la specifica di differenti condizioni di ricerca (sia in AND che in OR).

0.2 Casi d'uso

Per i dipendenti dell'agenzia vengono memorizzate delle informazioni tra le quali delle credenziali di accesso per accedere al sistema. Una volta fatto l'accesso il dipendente può inserire i dati dei lavoratori stagionali, modificarne le anagrafiche ed effettuare ricerche in base a dei profili (filtrare la ricerca).



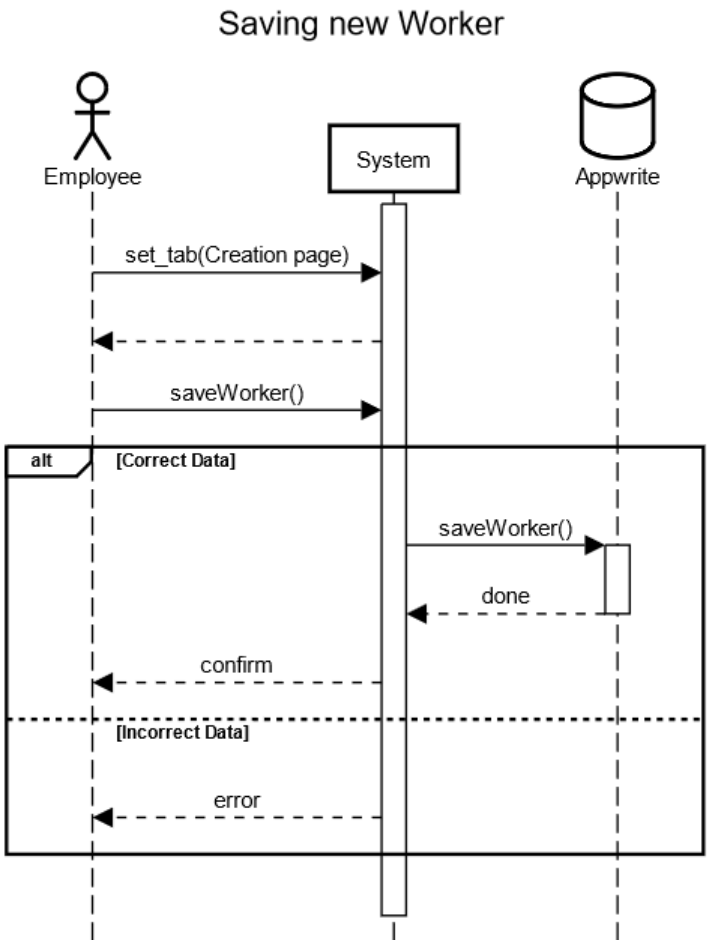
0.2.1 Inserimento nuovo lavoratore

I dipendenti dell'agenzia per poter effettuare l'inserimento e la gestione delle anagrafiche dei lavoratori è necessario che siano autenticati al sistema.

Attori: Dipendente dell'agenzia
Precondizioni: Il dipendente deve essere autenticato.
Passi:

1. Il dipendente accede al sistema
2. Il dipendente è introdotto all'interfaccia di base
3. Il dipendente compila il form con i dati anagrafici e il contatto di emergenza o modifica un form di un lavoratore esistente
4. Il dipendente inserisce il nuovo lavoratore o salva le modifiche del lavoratore modificato

Postcondizioni: Il lavoratore viene aggiunto dal database

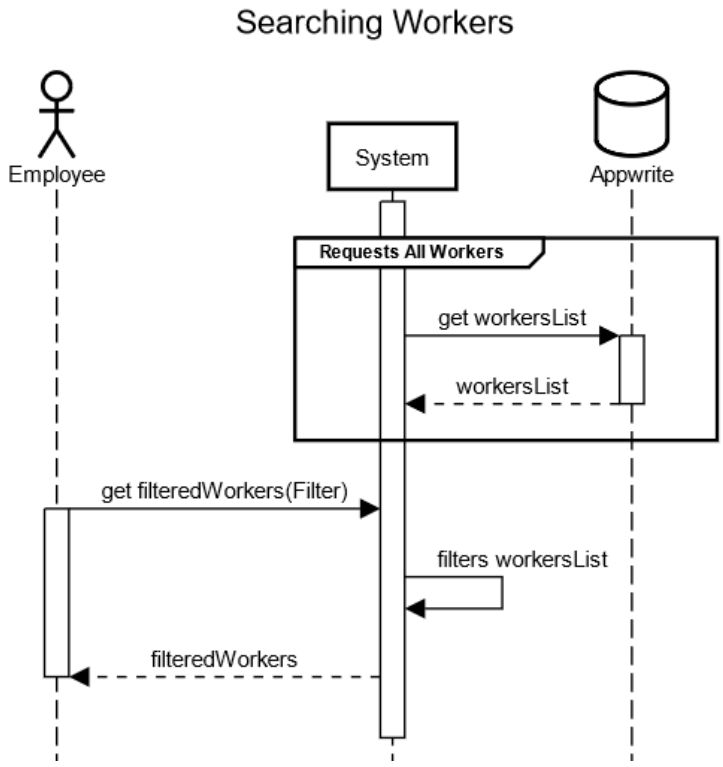


0.2.2 Effettuare ricerche

I dipendenti dell'agenzia per poter effettuare delle ricerche complesse è necessario che siano autenticati al sistema.
La ricerca può avvenire mediante una barra di testo tramite l'inserimento di nome e cognome. É inoltre possibile filtrare ulteriormente i risultati in base a: lingue parlate, mansioni effettuate, periodo e comuni di disponibilità, patenti di guida e disponibilità di auto propria. Tutti questi filtri possono inoltre essere applicati in modalità "AND" oppure "OR".

Attori: Dipendente dell'agenzia
Precondizioni: Il dipendente deve essere autenticato.
Passi:

1. Il dipendente accede al sistema
2. Il dipendente è introdotto all'interfaccia di base
3. Il dipendente visualizza la lista dei lavoratori
4. Il dipendente filtra la lista dei lavoratori



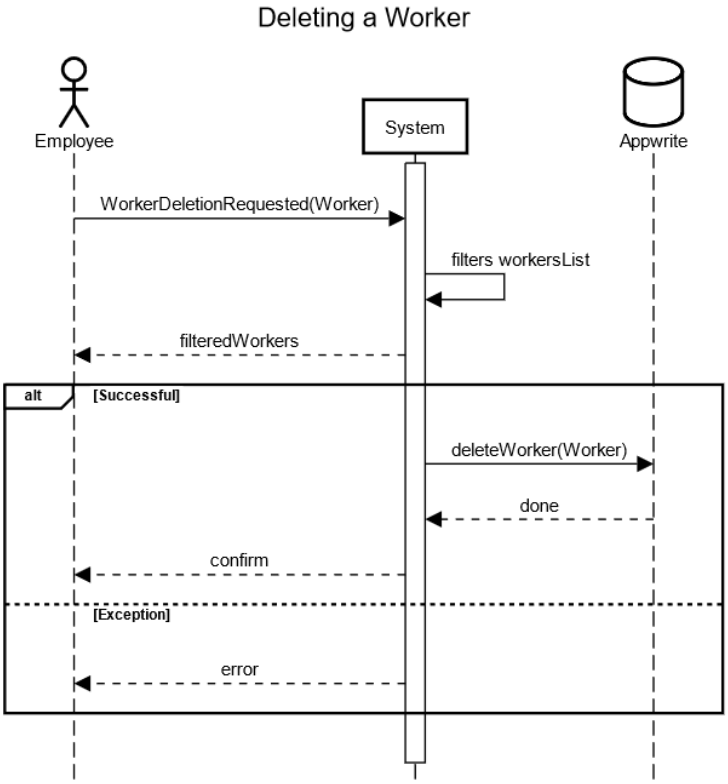
0.2.3 Cancellazione lavoratore

I dipendenti dell'agenzia per poter cancellare un lavoratore è necessario che siano autenticati al sistema.
La cancellazione avviene tramite un pulsante dedicato per ogni lavoratore.

Attori: Dipendente dell'agenzia
Precondizioni: Il dipendente deve essere autenticato.
Passi:

1. Il dipendente accede al sistema
2. Il dipendente è introdotto all'interfaccia di base
3. Il dipendente visualizza la lista dei lavoratori
4. Il dipendente trova il lavoratore da eliminare
5. Il dipendente elimina il lavoratore tramite apposito pulsante

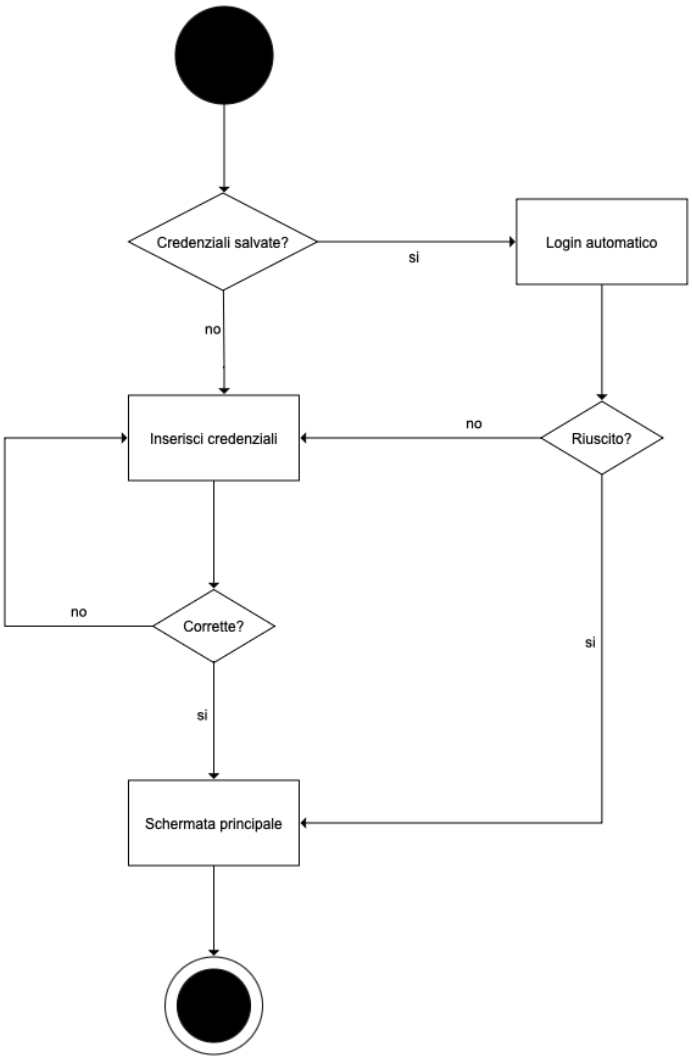
Postcondizioni: Il lavoratore viene eliminato dal database



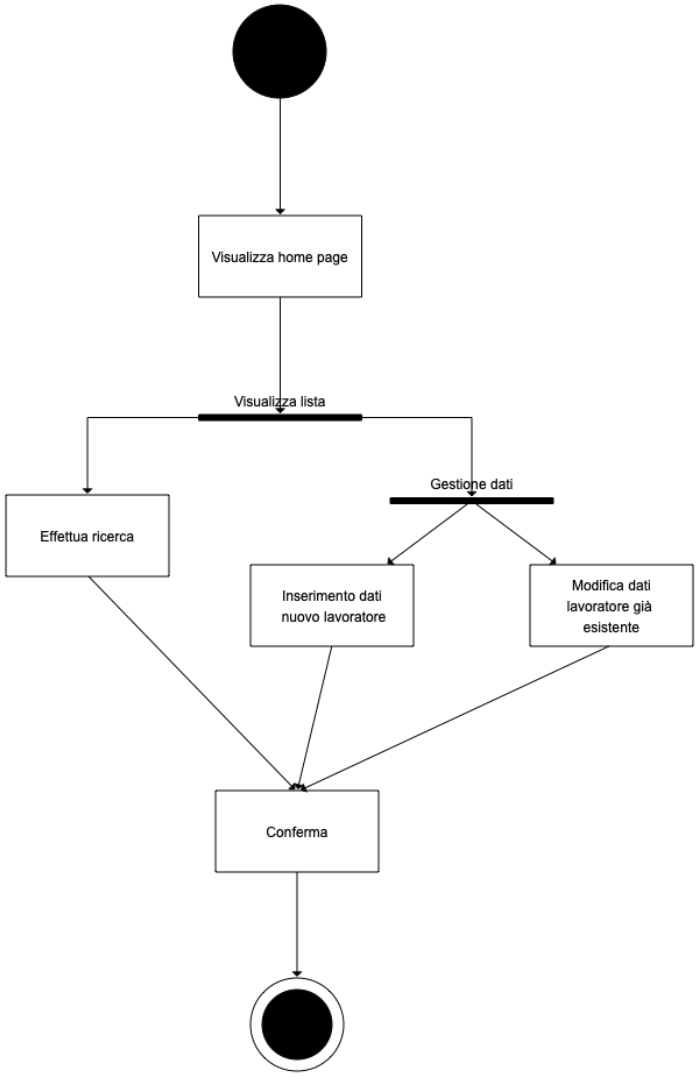
0.3 Diagrammi di attività

0.3.1 Autenticazione

I dipendenti dell'agenzia per poter poter utilizzare il software è necessario che siano autenticati al sistema.



0.3.2 Gestione delle anagrafiche



Nota: i seguenti diagrammi catturano una singola attività di un utente rispetto al sistema. Non è stata rappresentata nel diagramma la possibilità di ripetere più volte la stessa operazione, in sequenza, senza chiudere il software. Questo per semplici ragioni di chiarezza e leggerezza; si considerano quindi le singole attività d'interazione

0.4 Diagrammi delle classi

0.5 Note sul processo di sviluppo

Il processo di sviluppo è stato basato su un approccio Agile e Incrementale, con l'obiettivo di mantenere una solida base di lavoro. L'implementazione delle funzionalità non ha seguito un piano rigido, ma è stato scelto, ad ogni incremento, cosa fosse prioritario integrare ai fini di ridurre le tempistiche di sviluppo. Dopo ogni modifica significativa, è stata eseguita una rapida attività di test. La fase di analisi dei requisiti è stata completata prima dello sviluppo, che ha comportato la creazione di Use-Cases, diagrammi di attività e progettazione architetturale.

0.6 Design Pattern BLoC

Il design pattern scelto è *BLoC*, un design pattern creato da Google con lo scopo di separare la Business Logic dal Presentation Layer e permettere allo sviluppatore di riutilizzare il codice in modo efficiente. Per semplificare il tutto, viene messa a disposizione una libreria per Dart chiamata *flutter_bloc* che aiuta lo sviluppatore nell'implementazione del design pattern.

Nell'architettura Bloc, la Business Logic è centralizzata all'interno della classe blocco, mentre la visualizzazione è gestita da widget che ascoltano gli stati del blocco e reagiscono di conseguenza. Questo semplifica la separazione della logica di business dall'interfaccia utente e consente quindi di testare la logica di business in modo più efficiente.

Questo pattern è particolarmente utile per le app di medie e grandi dimensioni, dove la gestione della Business Logic risulta complessa.

BLoc Design Pattern si basa sul concetto di Programmazione Reattiva. La programmazione reattiva è un paradigma di programmazione che si concentra sulla gestione degli eventi e delle modifiche ai dati in tempo reale. Invece di eseguire il codice in modo sequenziale, la programmazione reattiva utilizza flussi di dati e eventi che possono essere trasmessi tra i diversi componenti dell'applicazione in modo asincrono.

Il concetto di base è quello dello Stream. Uno stream è una sequenza di dati che possono essere trasmessi e/o ricevuti in modo asincrono, anche da più componenti dell'applicazione.

Questo comporta che:

- quando succede qualcosa in qualsiasi componente (un evento, una variabile modificata ...) viene inviata una notifica a uno Stream
- chi si trova in ascolto su quello Stream, verrà avvisato e potrà quindi intraprendere le azioni appropriate, qualunque sia la sua posizione all'interno dell'applicazione

Ogni componente (*widget*) della User Interface ad ogni evento che lo riguarda dovrà semplicemente inviare una notifica dell'evento su uno Stream dedicato. Non dovrà preoccuparsi di chi userà quel dato, come o quando.

0.7 Architettura

La libreria BLoC consente di separare l'applicazione in tre livelli:

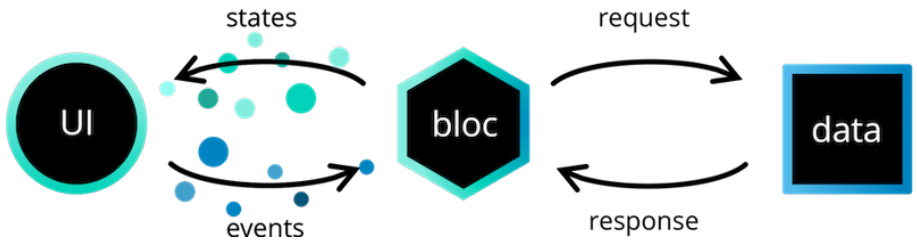
- **Presentation Layer** (interfaccia utente)
  La responsabilità del Presentation Layer è quella di generare l'interfaccia utente tenendo conto dello stato di uno o più Bloc. Inoltre deve gestire l'input dell'utente e i vari eventi del ciclo di vita dell'applicazione.

- **Business Logic Layer** (bloc)
  La responsabilità del Business Logic Layer è rispondere agli eventi ricevuti dal Presentation con la generazione di nuovi stati. Questo strato può appoggiarsi a uno o più Repository per recuperare i dati necessari alla creazione del nuovo stato.

- **Data Layer**
  La responsabilità del Data Layer è quella di recuperare/manipolare i dati da una o più fonti.

  · **Repository**
    È un wrapper attorno a uno o più Data Provider. Si occupa di fornire dei dati ben formattati e facilmente utilizzabili al Bloc

  · **Data Provider**
    Recupera i "dati grezzi" direttamente dalla sorgente (il database).



0.8 Backend

La realizzazione del Backend è stata affidata a Appwrite, un backend-as-a-service open source e self-hosted che semplifica lo sviluppo di app con una suite di SDK e API. Appwrite fornisce vari servizi tra cui autenticazione e database.

0.8.1 Autenticazione

Appwrite offre delle API dedicate per gestire l'autenticazione dell'app. Tramite la SDK è possibile accedere a delle funzioni per registrare e autenticare gli utenti e gestirne la sessione.

0.8.2 Database

Appwrite offre un database basato su documenti di facile utilizzo. Dietro le quinte si tratta di un database NoSQL basato su MariaDB. Il database si compone di:

- **Collection:** un gruppo di documenti. Ogni collezione ha degli attributi che definiscono la struttura del documento e le sue autorizzazioni per la lettura e la scrittura.
- **Document:** un oggetto JSON strutturato di chiavi e valori, appartenente a una Collection.

Le **Collection**, i **Document**, gli **Attributi** e i **Permessi** possono essere facilmente gestiti dalla console di Appwrite.
La SDK di Appwrite mette a disposizione dei metodi per caricare, aggiornare, eliminare i **Documents** delle rispettive **Collections**.