Progetto: **"Outsourcing Hub"**

System Design Document

Ingegneria del Software e Interazione Uomo Macchina

Antonio Onorato Mat:0124002413

Mattia Danisi Mat:0124002281

Fabio Migliaccio Mat:0124002170

Anno accademico 2022-2023

Indice

1 Introduzione

1.1 Scopo del sistema

La nostra applicazione nasce per gestire a 360 gradi il processo di outsourcing, dal singolo developer all'azienda che richiede il servizio
La nostra applicazione offre tutte le funzionalità necessarie per l'outsourcing:

- Creazione di progetti a cui le aziende di outsourcing potranno candidarsi;
- Ricerca di clienti a cui fornire i propri servizi;
- Ricerca di developers;
- Gestione delle proprie informazioni personali e del proprio portfolio;

1.2 Obiettivi di progettazione

- **Usabilità:** UI facile e intuitiva.
- **Familiarità:** UI pensata seguendo i canoni della progettazione moderna.
- **Response time:** La richiesta viene soddisfatta in modo rapido ed efficiente.
- **Estensibilità:** Sviluppo che segue i princi base del software design.
- **Piattaforma di destinazione:** Web. Implementazione tramite Spring Boot.
- **Portabilità:** La natura web fa si che l'app possà essere utilizzata da tutti i dispositivi dotati di un'interfaccia web.
- **Gestione dei Dati Persistenti:** Per la memorizzazione dei dati sono state proposte due tecnologie differenti: Cassandra, un database NoSql per quanto riguarda la parte relative ai contenuti user-generated, e MySQL per gestire i dati transazionali come l'user authentication.

1.3 Definzioni, acronimi e abbreviazioni

All'interno della documentazione sono stati utilizzati degli acronomi e quindi per agevolare la comprensione del lettore nella tabella riportata qui sotto definiamo il loro significato.

Termini	Definizioni
Client / Client Company	Utente che richiede un servizio di Outsourcing
Outsourcing Company	Compagnia che offre il servizio di Outsourcing
Developer	Utente che offre il servizio all'azienda di outsourcing in cambio di un compenso economico

1.4 Riferimenti

Per una migliore comprensione del lavoro presentato in questo documento, si consiglia di consultare il modello di analisi dei requisiti in allegato.

1.5 Panoramica

L'analisi del progetto ha portato alla seguente suddivisione in classi.

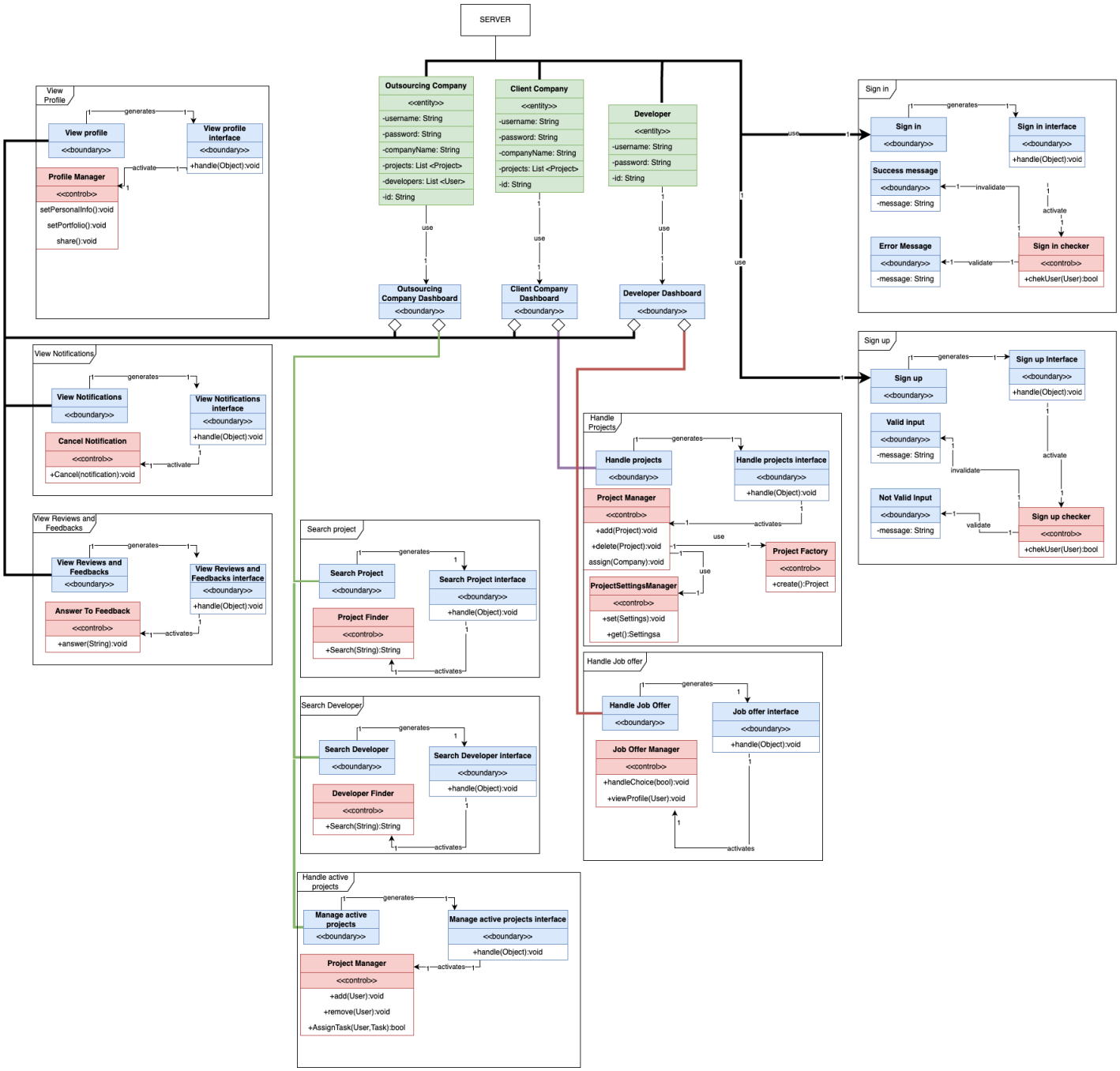


Figure 1: visione completa

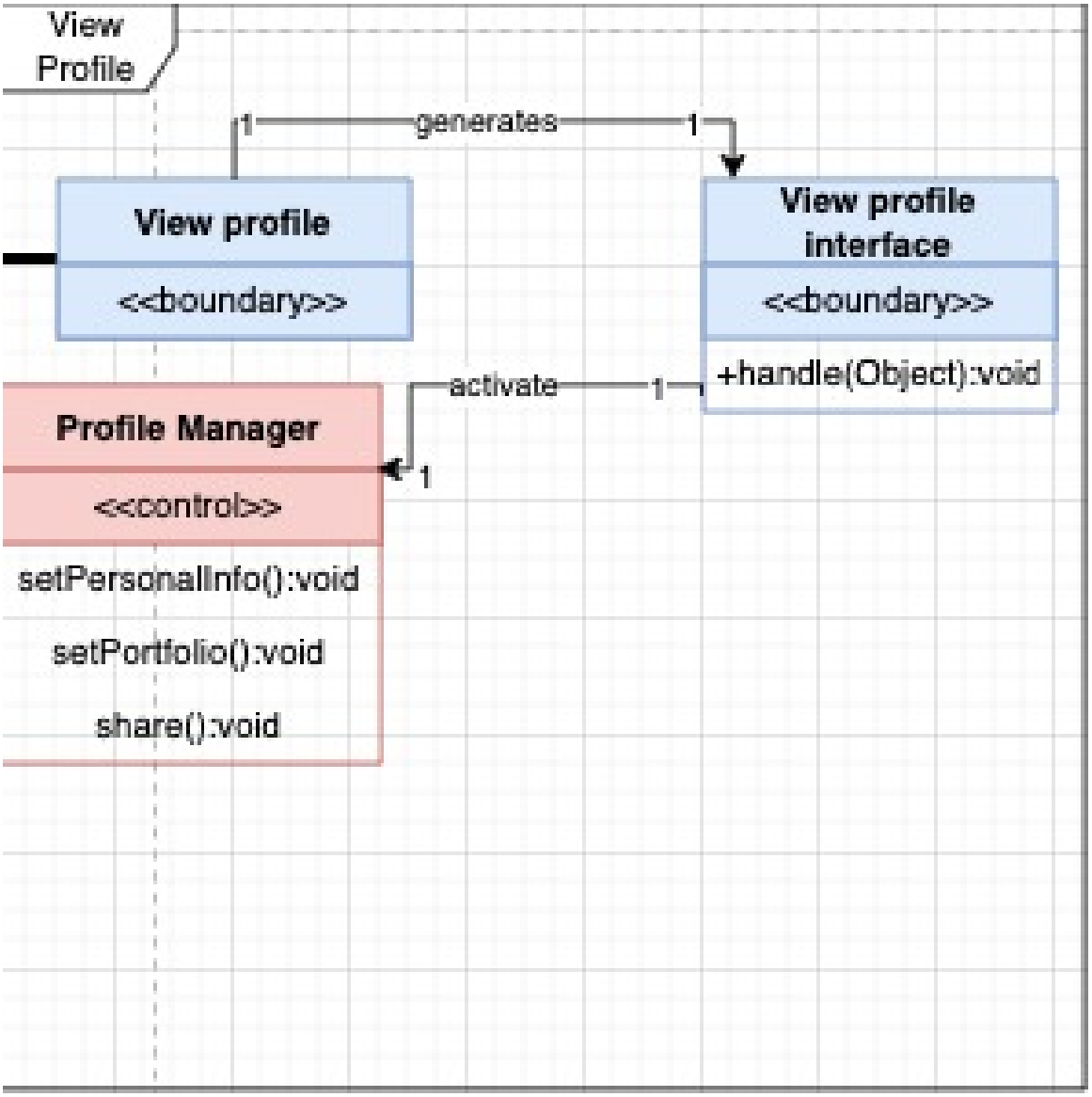


Figure 2: Classe utilizzata da ogni attore per visualizzare il proprio profilo e/o apportare modfiche alle info personali, al portfolio o condividere il profilo

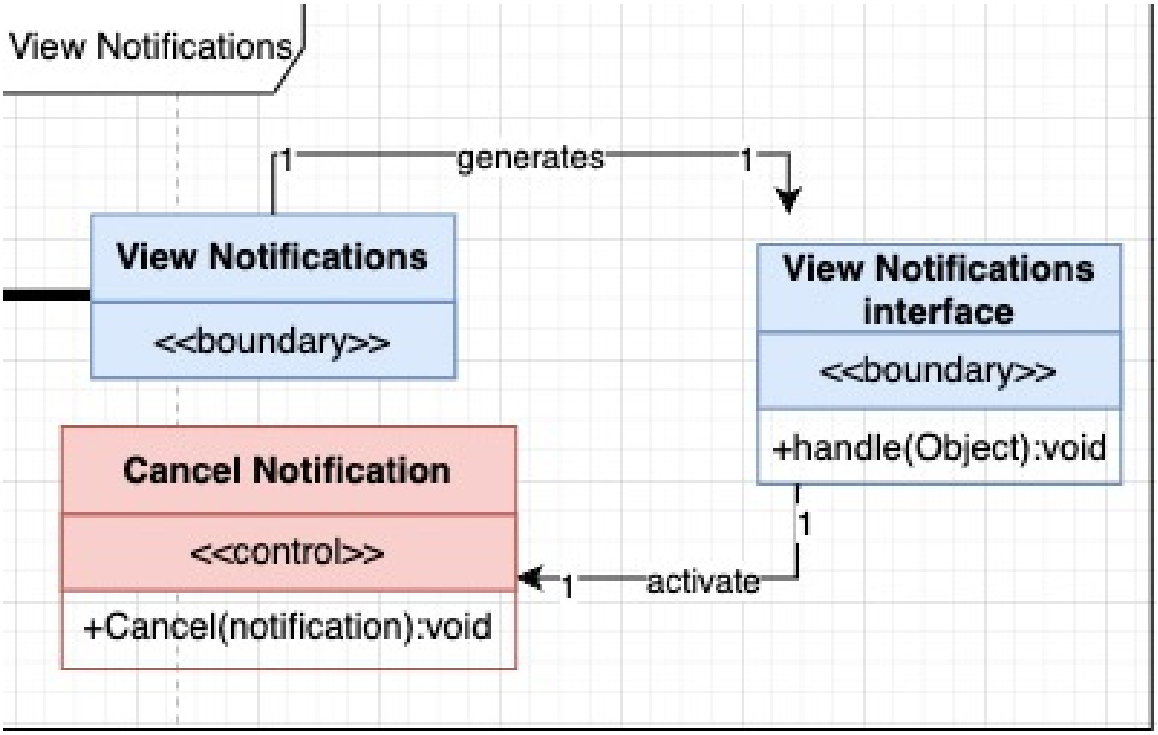


Figure 3: Classe utilizzata da ogni attore per visualizzare le notifiche ed eventualmente cancellarle

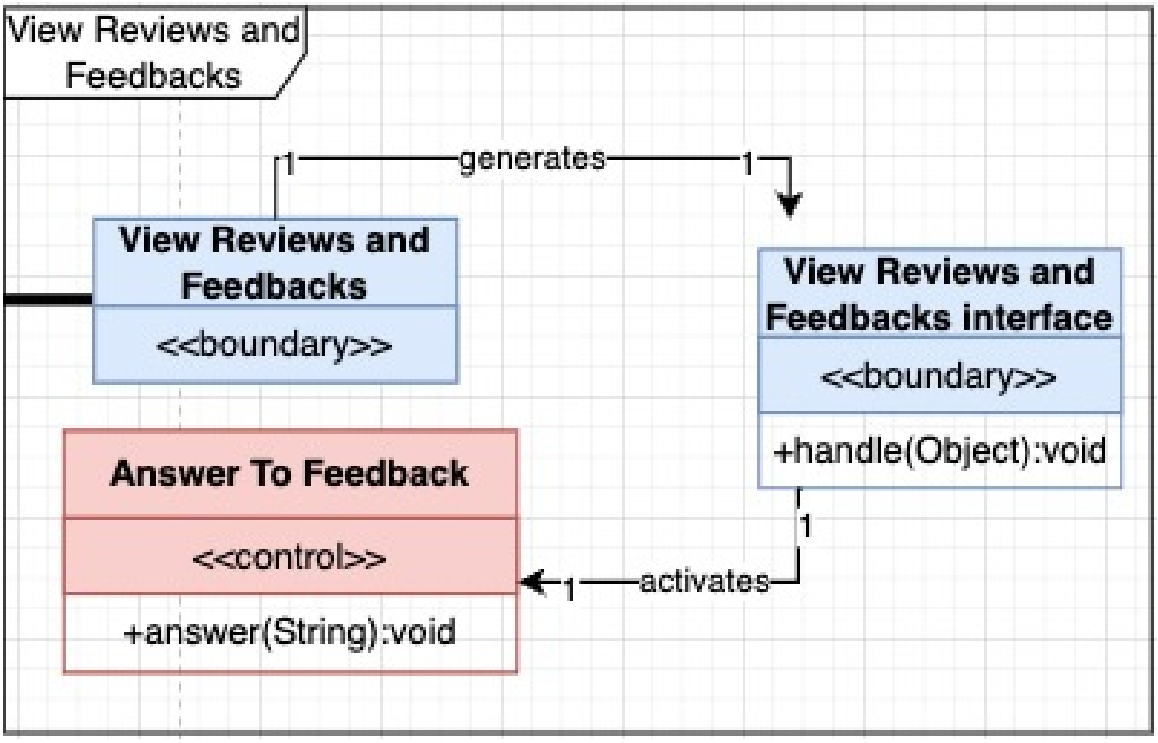


Figure 4: Classe utilizzata da ogni attore per revisionare i feedback ricevuti dagli utenti. Contiene una funzione *Answer()* grazie al quale è possibile rispondere a un commento

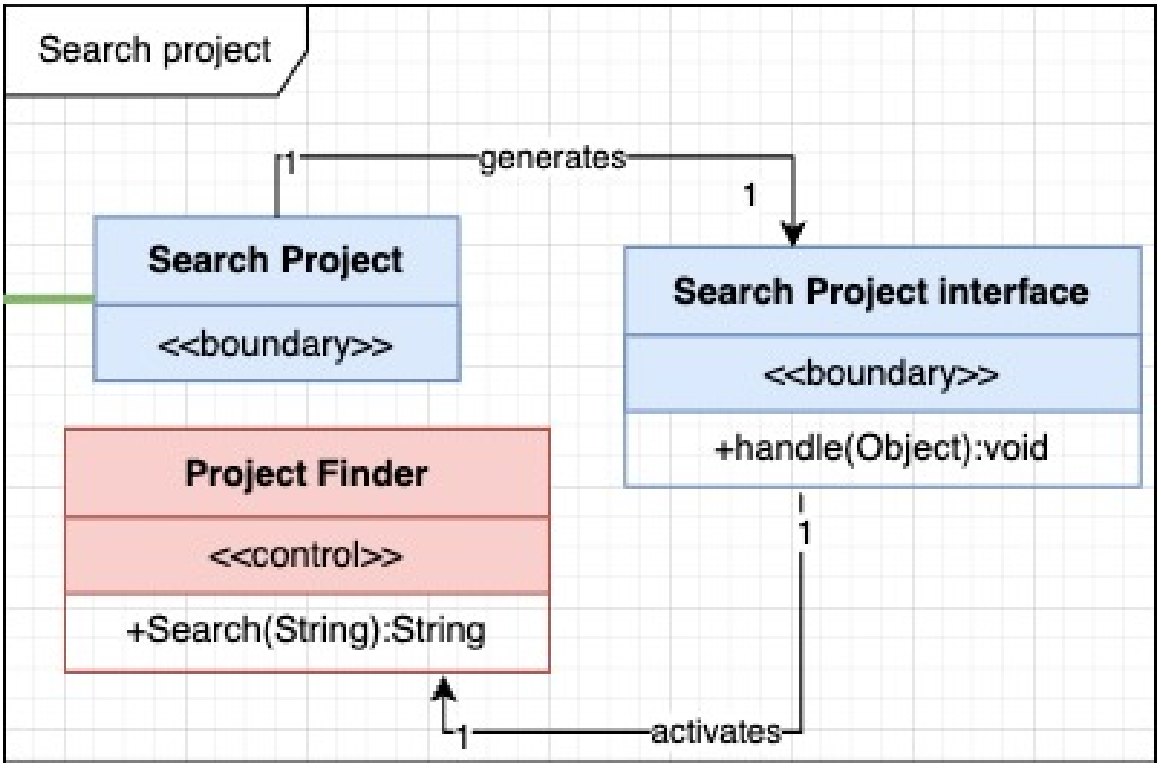


Figure 5: Classe utilizzata da outsourcing company per cercare un progetto all'interno del network. La classe utilizza Project Finder, il cui metodo *Search()* prende in input una stringa e restituisce una stringa Json

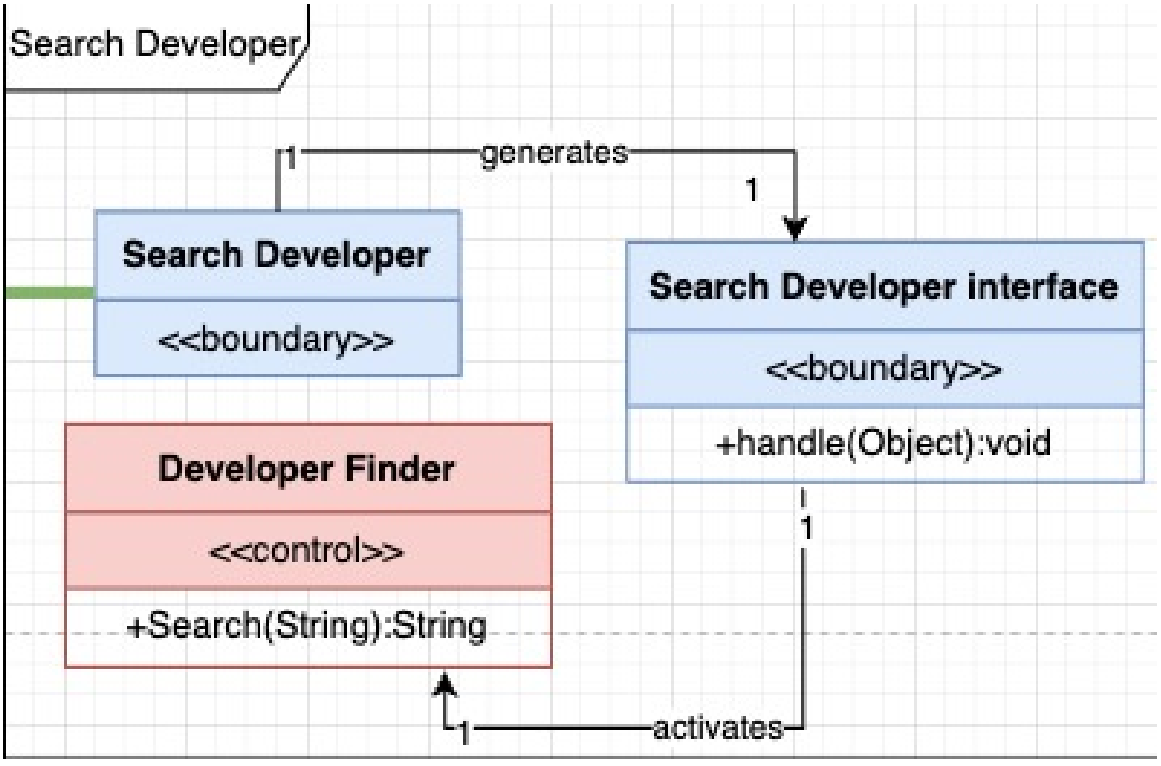


Figure 6: Search Developer è una classe utilizzata da Cliente/Outsourcing Company per cercare un developer. Anche questa classe utilizza il metodo *Search()* e funziona in modo analogo a quanto descritto nella classe *Search Project*

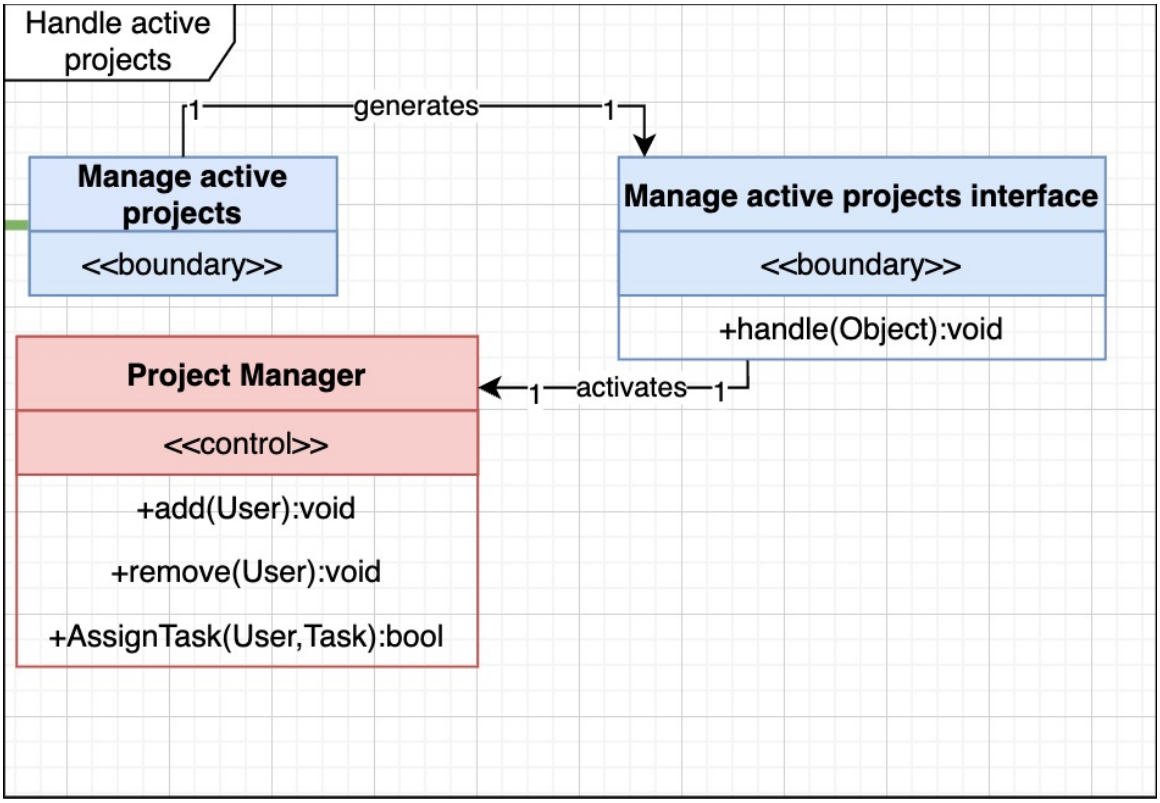


Figure 7: Classe utilizzata da Outsourcing Company per gestire gli sviluppatori esterni all'azienda reclutati tramite la piattaforma. Tramite la classe è possibile aggiungere o rimuovere developers dal progetto o assegnare una task specifica a uno di essi

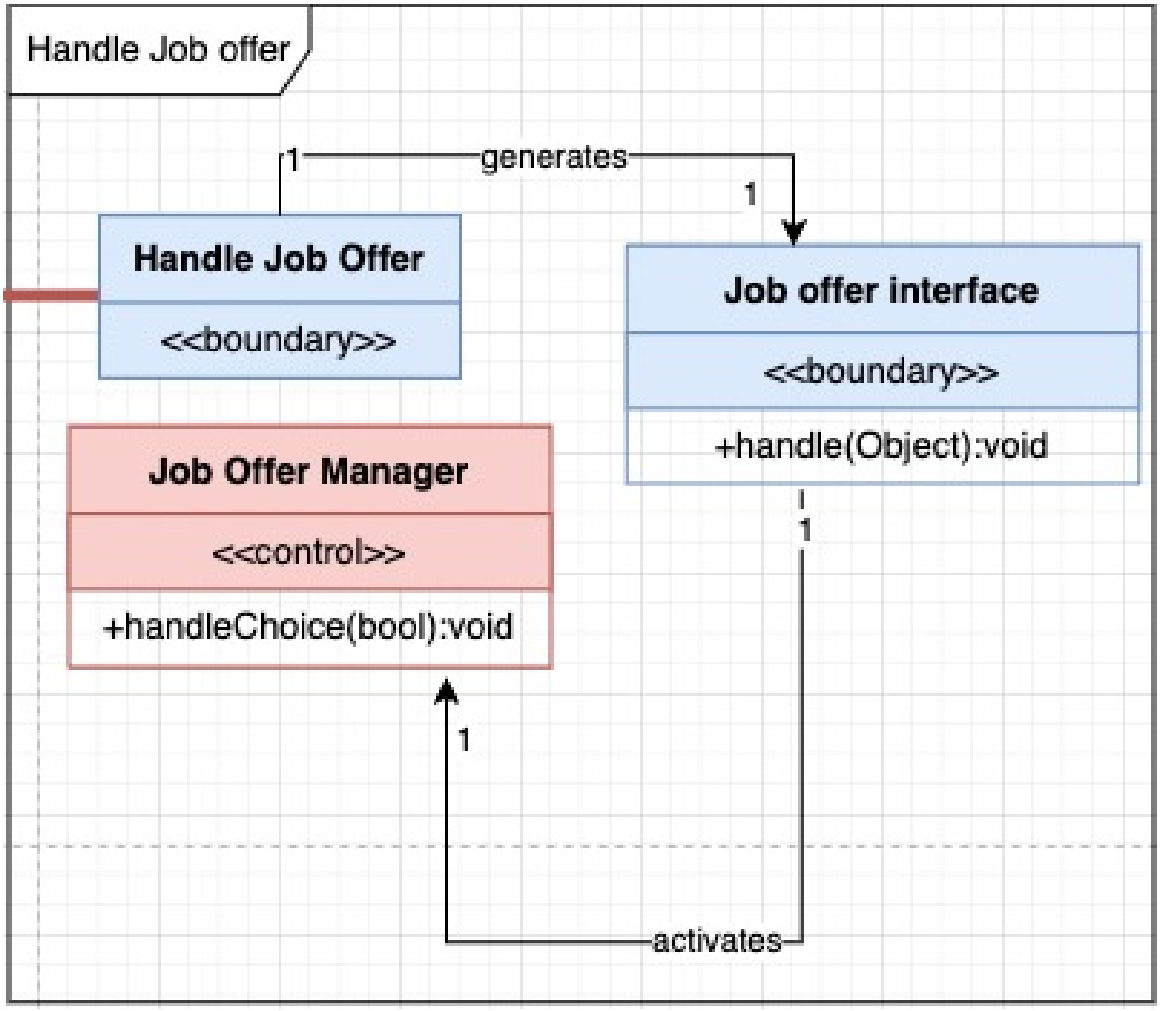


Figure 8: Classe utilizzata da Developer per gestire le offerte di lavoro in arrivo. Tramite la classe l'utente può accettare o rifiutare l'offerta

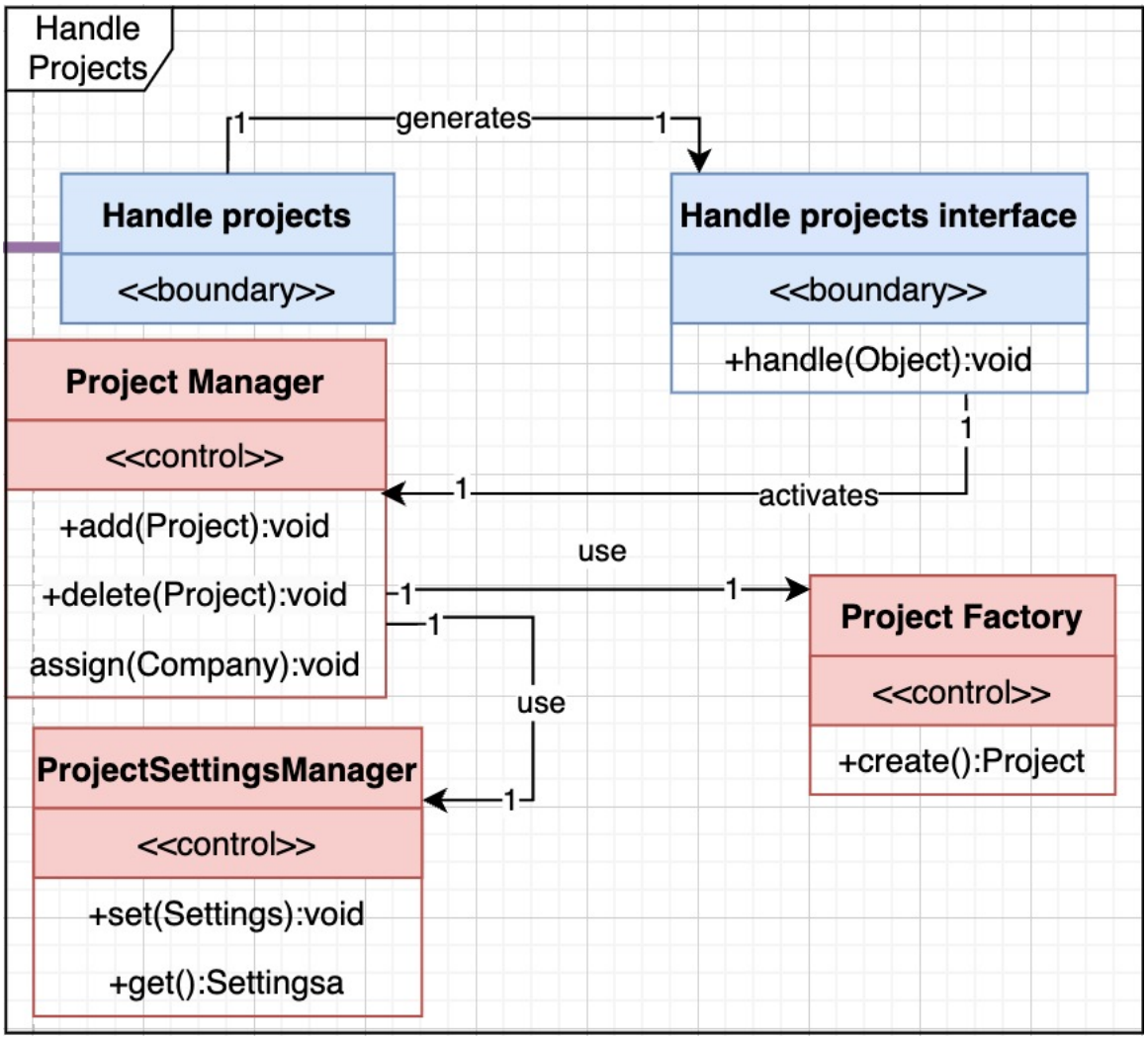


Figure 9: Classe utilizzata dall'azienda cliente per gestire i progetti attivi. Può eliminare progetti già esistenti, aggiungerne di nuovi, assegnare una company ad un progetto o gestire le impostazioni di progetto (visibilità, budget, scadenza . . .)

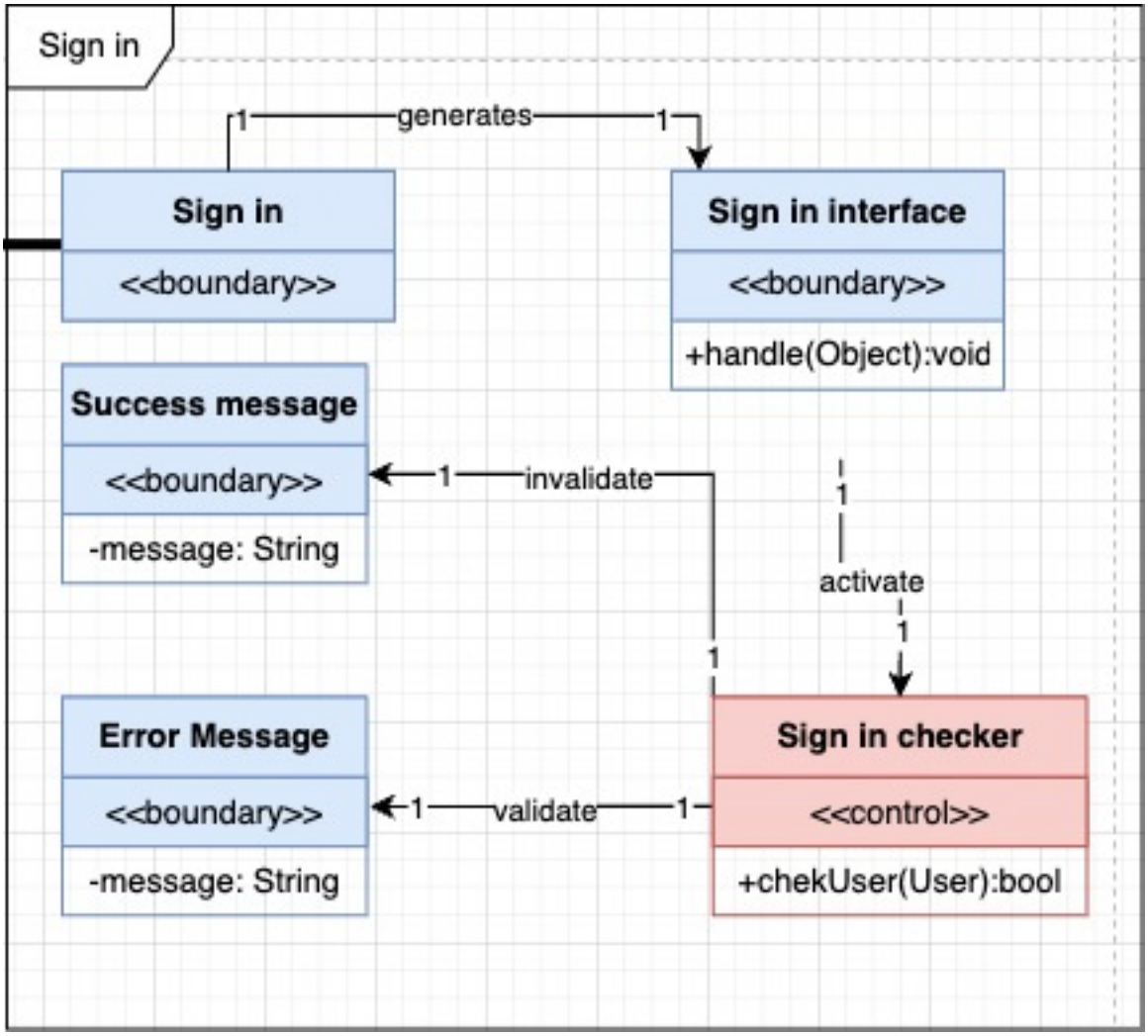


Figure 10: Classe utilizzata da tutti gli utenti per l'autenticazione sulla piattaforma

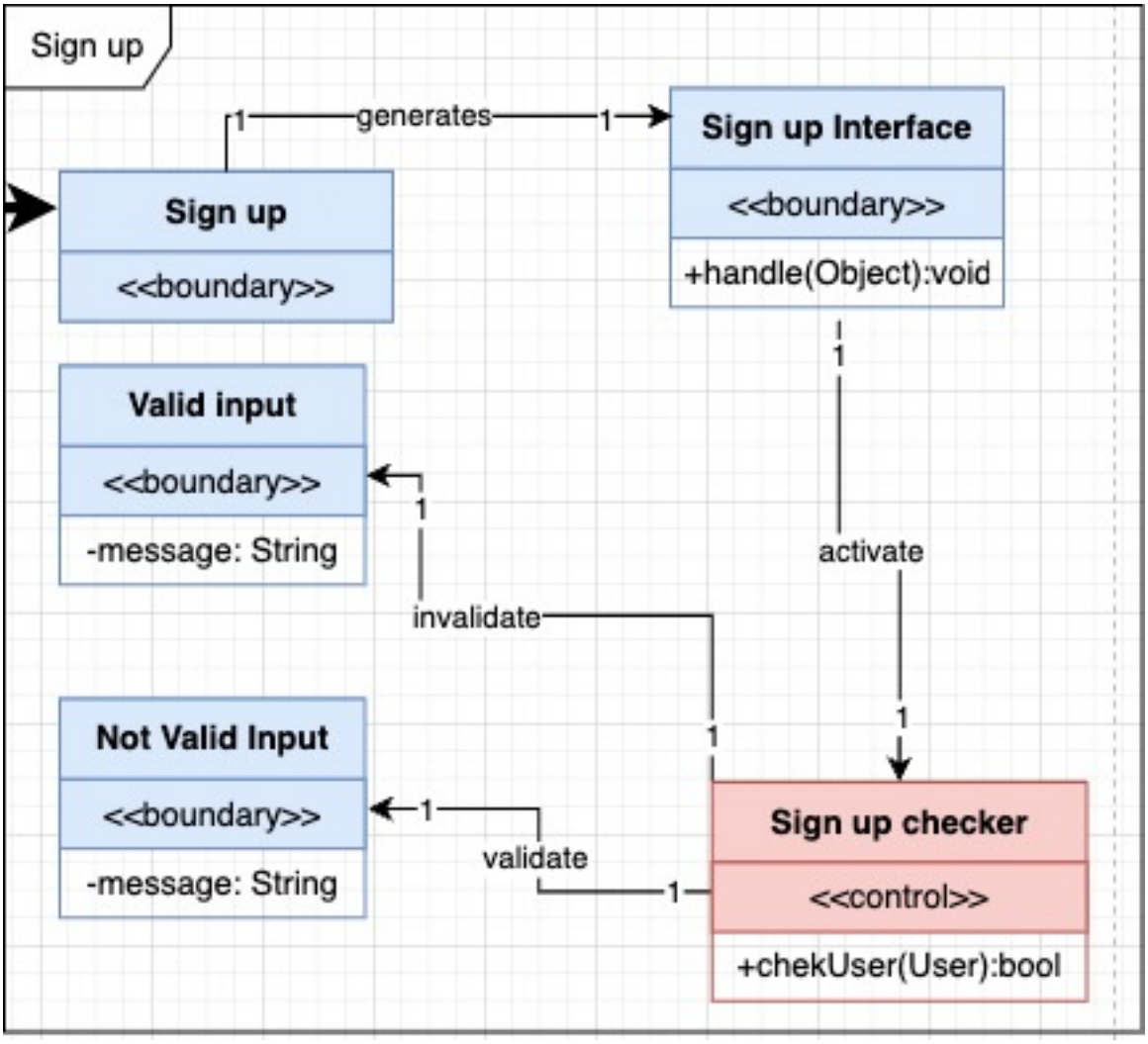
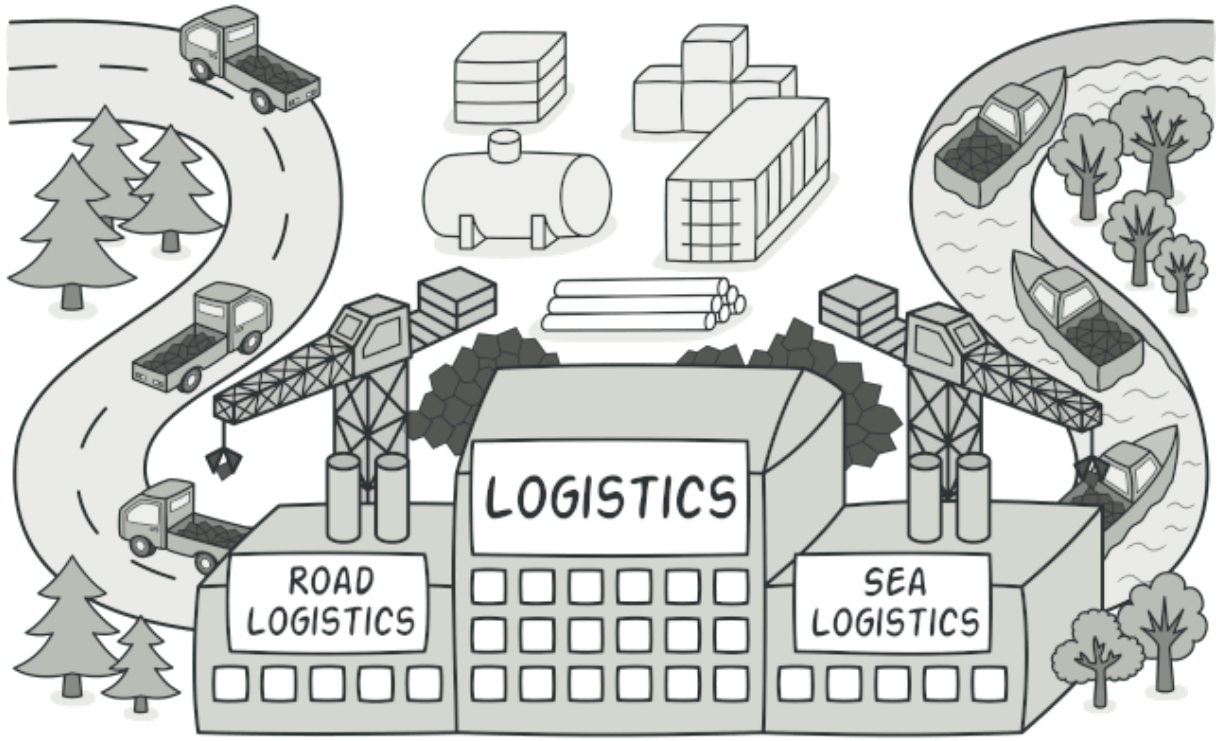
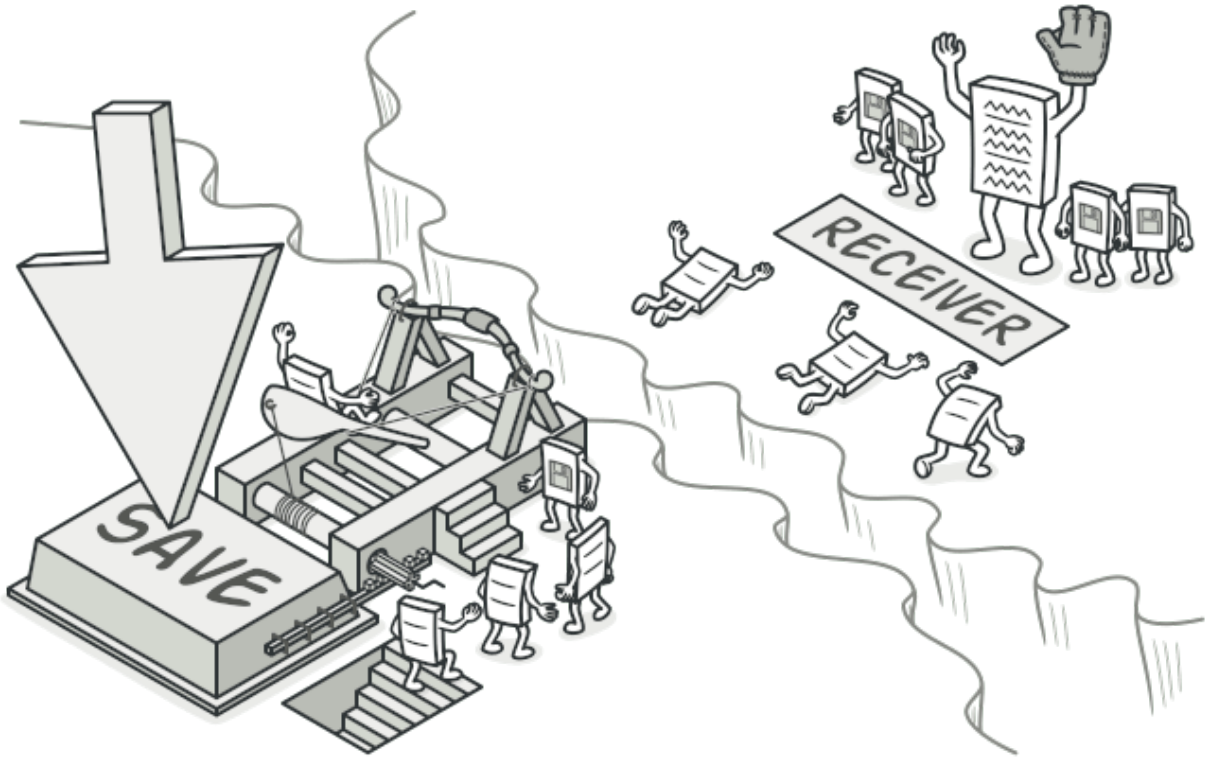


Figure 11: Classe utilizzata da tutti gli attori per la creazione di un nuovo utente

1.6 Pattern Utilizzati



Il pattern Factory Method è stato introdotto per permettere la selezione a run time durante la creazione utente. Questo pattern è necessario dato che abbiamo 3 tipologie di account che l'utente può creare.



Il pattern Command è stato introdotto per facilitare la creazione della user interface, separando la UI dalla business logic delle varie operazioni. In questo modo possiamo creare elementi UI indipendenti dall'azione che

compiono

Il pattern strategy è stato introdotto per permettere un'astrazione del tool di ricerca. Come spiegato in precedenza l'utente può utilizzare il tool di ricerca per cercare progetti a cui candidarsi, ma anche developers che combaciano con un determinato profilo professionale richiesto dall'azienda. Grazie al pattern strategy è possibile astrarre la classe relativa al tool di ricerca e decidere a run time l'algoritmo da utilizzare

2 Sistema corrente

I sistemi analizzati già presenti sul mercato sono progettati in modo da massimizzare il numero di utenti della piattaforma; Il target ha quindi un range molto alto e la competizione è al massimo. Questo tipo di sistema però presenta alcune gravi problematiche:

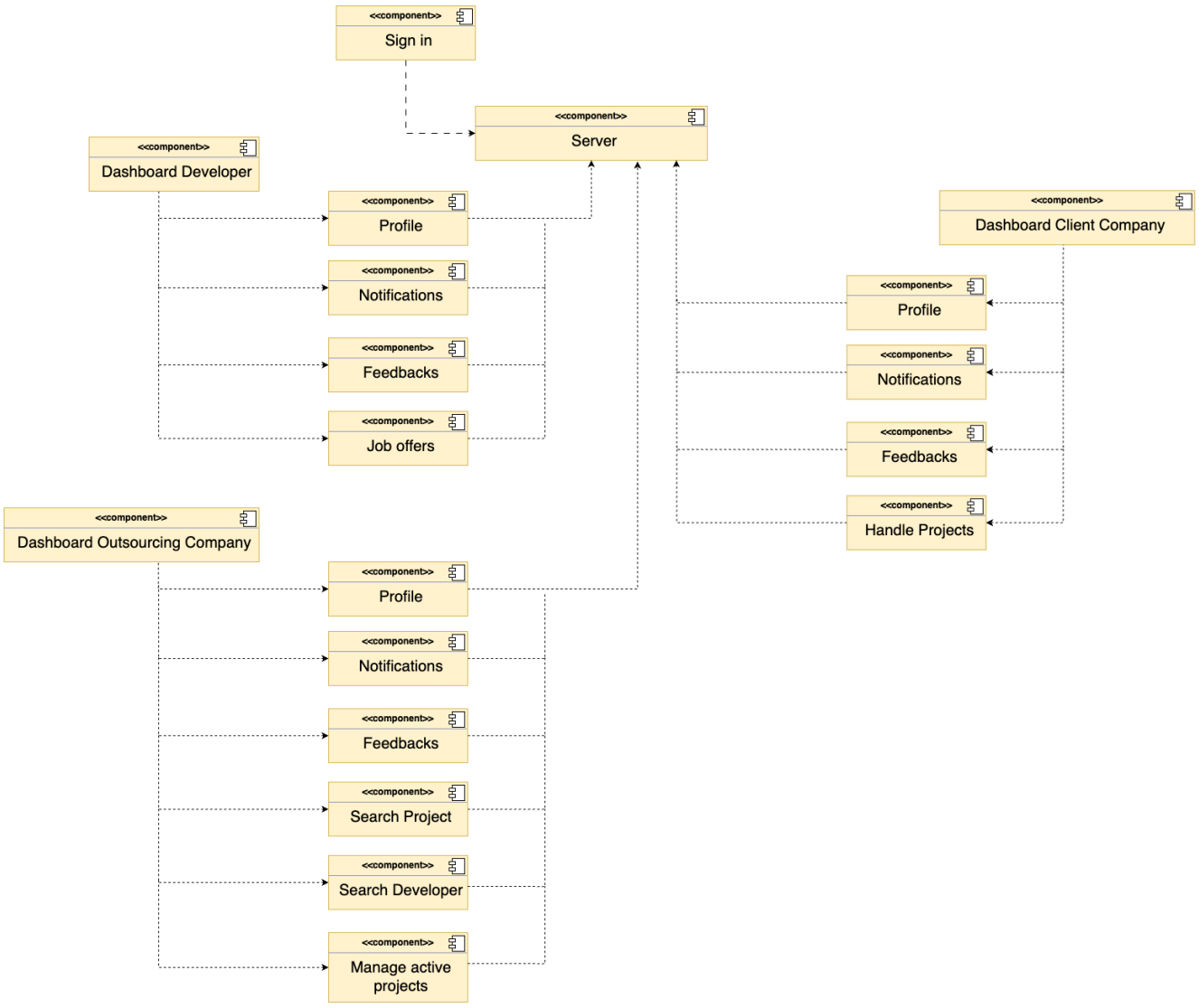
- **Troppa generalità:** In sistemi come questo il mercato finisce per saturarsi in tempi davvero brevi, rendendo impossibile crearsi un network di clienti.
- **Poca comunicazione:** La comunicazione molto spesso avviene esclusivamente alla commissionamento del progetto. È impossibile quindi per l'utente apportare modifiche al prodotto mentre esso è in fase di sviluppo

In seguito a tale analisi, abbiamo ritenuto opportuno, proporre un nuovo sistema che, seppur mantendendo i punti di forza dei competitor, risolve quelle che sono le carenze degli altri sistemi, progettando una soluzione developer-oriented.

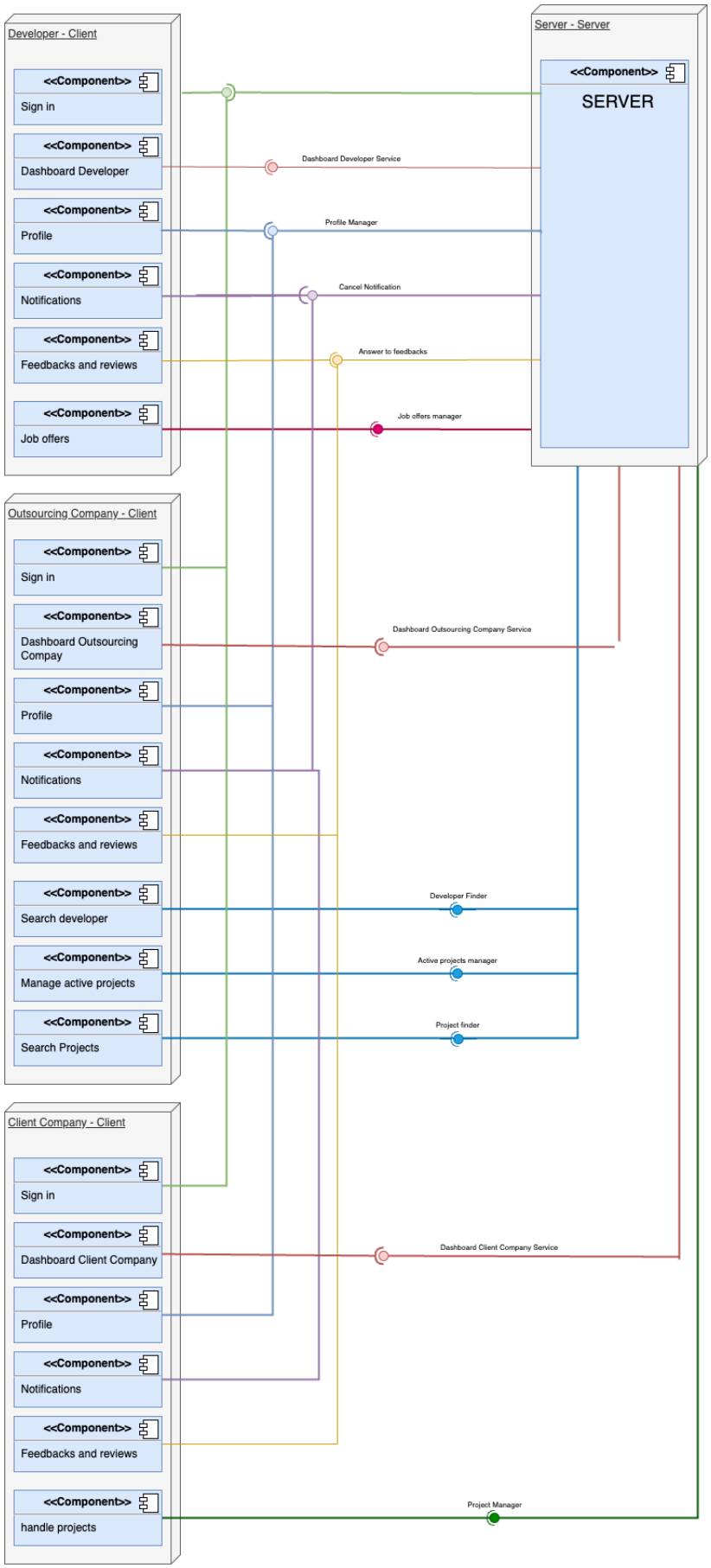
3 Sistema Proposto

3.1 Panoramica

Per la progettazione del sistema, l'architettura più adeguata è quella client/server. Nel sistema sono presenti tre tipi di client : Il developer, l'azienda cliente e l'azienda di outsourcing che funge da intermediario.



3.2 Hardware/Software mapping



3.3 Gestione dei dati persistenti

Il nostro sistema utilizza un database NoSQL per la memorizzazione dei dati generati dall'utente. Grazie alla sua flessibilità e scalabilità, i dati possono essere facilmente recuperati e mostrati all'utente attraverso il sistema. Per quanto riguarda la gestione dei dati transazionali come l'user authentication è stato scelto un database SQL.

3.4 Controllo accessi e sicurezza

Poiché il nostro sistema prevede tre tipi di utenti, abbiamo creato una dashboard che avesse una parte comune a tutti gli utenti, ma che allo stesso tempo aggiungesse funzionalità specifiche per tipologia di utente.

- **Area Personale Utente:** Area comune a tutti i client.
- **Area Specifica Utente:** Ogni utente può trovare aree specifiche del suo ruolo.