

Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

CORSO DI LAUREA IN INFORMATICA



Studio del Back-end a microservizi e
implementazione del Front-End in Angular
nel progetto web SyncRec

Tesi di laurea triennale

Relatore

Prof. Claudio Palazzi

Laureando

Ludovico Brocca

ANNO ACCADEMICO 2018-2019

Indice

1	Introduzione	3
1.1	L'azienda: SyncLab s.r.l.	3
1.1.1	Servizi offerti	3
2	Organizzazione del progetto	5
2.1	Proposta di stage	5
2.2	Il progetto: SyncRec	6
2.2.1	Prime cinque settimane	6
2.2.2	Successive tre settimane	6
2.3	Obiettivi	6
2.3.1	Notazione	7
2.3.2	Obiettivi fissati	7
2.4	Pianificazione	7
3	Analisi dei requisiti	9
3.1	Tipologie di utenti	9
3.2	Casi d'uso	9
3.2.1	UC-1: Visualizzazione lista degli Applicant	9
3.2.2	UC-2: Modifica di un Applicant	10
3.2.3	UC-3: Eliminazione di un Applicant	20
3.2.4	UC-4: Visualizzazione dettaglio di un Applicant	21
3.2.5	UC-5: Aggiunta filtro sulla lista degli Applicant	26
3.2.6	UC-6: Ordinamento degli Applicant	26
3.2.7	UC-7: Aggiunta dei filtri	28
3.2.8	UC-8: Rimozione dei filtri selezionati	31
3.2.9	UC-9: Visualizzazione skillmatrix di un Applicant	31
3.2.10	UC-10: Modifica skill di un Applicant	32
3.3	Tracciamento dei requisiti	32
3.3.1	Requisiti funzionali	33
3.3.2	Requisiti di vincolo	36
4	Tecnologie e framework adottati	37
4.1	Linguaggi e framework	37
4.1.1	TypeScript	37
4.1.2	Angular	38
4.1.3	Npm	40
4.1.4	Bootstrap	40
4.1.5	HTML e CSS	40

4.1.6	Java	40
4.1.7	Spring	41
4.1.8	MongoDb	41
4.2	Ambiente di Sviluppo	42
4.2.1	Sistema operativo	42
4.2.2	IDE	42
4.2.3	Versionamento	43
4.2.4	Back-end	43
4.2.5	Analisi statica del codice	43
4.2.6	Altri strumenti utilizzati	43
5	Progettazione e codifica	45
5.1	L'applicativo SyncRec: Struttura e sviluppo del progetto	45
5.1.1	Homepage	46
5.1.2	Maschera della visualizzazione degli Applicant	47
5.1.3	Maschera delle operazioni CRUD su un Applicant	50
5.1.4	Maschera di visualizzazione di uno skillmatrix	52
A	Altre tecnologie	55
A.1	Oracle PL/SQL	55
A.1.1	Struttura di uno script Oracle PL/SQL	55
A.2	Architetture a microservizi	56
A.3	JavaServer Pages (JSP)	57
	Glossary	59

Elenco delle figure

1.1	Logo dell'azienda ospitante	3
2.1	Diagramma di Gantt per la pianificazione dello stage	8
3.1	Diagramma di Use Case da 1 a 4	10
3.2	Diagramma sottocasi UC-2	11
3.3	Diagramma Use Case sulla visualizzazione errori relativi al salvataggio dell' <i>applicant</i>	17
3.4	Diagramma sottocasi UC-4	22
3.5	Diagramma di Use Case da UC-5 a UC-10	26
3.6	Diagramma di Use Case delle operazioni ausiliare sulla lista degli <i>applicant</i>	27
4.1	Logo di Angular	38
4.2	Logo di Spring	41
5.1	Struttura dei <i>component</i> sviluppati per l'applicativo SyncRec	46
5.2	Integrazione tra Front-end e Back-end	47
5.3	Schermata della homepage	47
5.4	Schermata della visualizzazione lista degli <i>applicant</i>	48
5.5	Schermata della selezione dei filtri da applicare alla lista degli <i>applicant</i>	50
5.6	Maschera CRUD del singolo <i>applicant</i>	51
5.7	Maschera CRUD dello skillmatrix	53

Sommario

Il presente documento relaziona il risultato del lavoro prodotto a seguito del periodo di stage formativo svolto dal laureando Ludovico Brocca presso l'azienda SyncLab s.r.l., della durata di circa trecento ore.

Prima dell'inizio dell'attività di lavoro, sono stati concordati gli obiettivi con il tutor aziendale dr. Fabio Pallaro e con il relatore prof. Claudio Palazzi.

Tali obiettivi prevedevano un periodo iniziale di formazione su tecnologie in uso presso l'azienda e di particolare interesse personale, a cui poi è seguita l'implementazione di alcune maschere di Front-End per l'applicativo SyncRec, la cui finalità è registrare le persone richiedenti un periodo di formazione presso l'azienda.

Lo scopo è stato fornire un prodotto in grado di sostituire la versione precedente, ormai datata, implementando un back-end a microservizi e un front-end in Angular 8.

Il lavoro è stato svolto in collaborazione con altri studenti aventi il periodo di stage formativo presso l'azienda; ciò ha permesso l'adozione di metodologie AGILE/ Scrum e il raggiungimento di una maggiore comprensione della prospettiva lavorativa di un team.

La tesi è suddivisa in 6 capitoli:

- Nel primo viene presentata l'azienda e le metodologie di lavoro in uso presso di essa;
- Nel secondo viene presentato il progetto e l'organizzazione generale dello stage;
- Il terzo prosegue analizzando nel dettaglio gli obiettivi, i requisiti e i casi d'uso dell'applicativo SyncRec;
- Il quarto presenta una panoramica delle tecnologie concernenti il progetto di stage;
- Il quinto presenta il dettaglio della soluzione progettata e sviluppata.
- Il sesto presenta una relazione degli obiettivi raggiunti, insieme al loro grado di soddisfacimento e a una valutazione personale e soggettiva del lavoro svolto.

Convenzioni tipografiche

Durante la stesura del testo sono state adottate le seguenti convenzioni tipografiche:

- gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati sono definiti nel glossario, situato alla fine del presente documento e ogni occorrenza è evidenziata in blu, come l'esempio seguente: [Spring](#);

- i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere corsivo.

Capitolo 1

Introduzione

1.1 L'azienda: SyncLab s.r.l.



Figura 1.1: Logo dell'azienda ospitante

SyncLab nasce nel 2002 come Software house, trasformandosi poi nella figura di [System Integrator](#) in conseguenza del rapido ampliamento delle competenze e del dominio tecnologico offerto; oggi SyncLab vanta un organico di oltre 200 persone e 4 sedi sul territorio italiano (Napoli, Roma, Padova e Milano).

La grande attenzione alla gestione delle risorse umane ha fatto di Sync Lab un riferimento in positivo per quanti volessero avviare o far evolvere in chiave professionale la propria carriera. Il basso turn-over testimonia la voglia dei collaboratori di condividere il progetto comune, assumendo all'interno di esso ruoli e responsabilità che solo un processo evolutivo così intenso può offrire. I ricavi hanno avuto un incremento proporzionale alla crescita dell'azienda beneficiando dell'approccio adattivo e diversificato al mercato

1.1.1 Servizi offerti

Sync Lab Srl è un'azienda leader nella consulenza tecnologica, impegnata in un processo continuo di identificazione e messa in opera di soluzioni per i clienti finalizzate alla creazione di valore. Essa supporta le esigenze di innovazione di tutte le organizzazioni ed in ogni settore di mercato nell'ambito Information Technology, con servizi in ambito:

- *Business Consultancy*;
- *Project Financing*;
- *IT Consultancy*.

L'azienda ha come punti di forza la qualità dei servizi offerti (certificazioni [ISO 9001](#), [ISO 14001](#), [ISO 27001](#), [OHSAS 18001](#)) ed un'accurata gestione delle risorse umane. L'approfondita conoscenza di processi e tecnologie, maturata in esperienze altamente

significative e qualificanti le fornisce l'expertise e Know How necessari per gestire progetti di elevata complessità, dominando l'intero ciclo di vita: Studio di fattibilità, Progettazione, Implementazione, Governance e Post Delivery. L'offerta di consulenza specialistica trova le punte di eccellenza nella progettazione di architetture Software avanzate, siano esse per applicativi di dominio, per sistemi di supporto al business, per sistemi di integrazione o per sistemi di monitoraggio applicativo/territoriale. Il suo laboratorio RD è sempre al passo con i nuovi paradigmi tecnologici e di comunicazione (*Big Data Analysis*, *Cloud Computing*, *Internet delle Cose*, *Mobile* e Sicurezza IT) per supportare i propri clienti nella creazione ed integrazione di applicazioni, processi e dispositivi. Le attività in ambito Educational ed RD hanno permesso di acquisire una profonda conoscenza degli strumenti di finanza agevolata fruendone direttamente ed interagendo con enti di supporto ai progetti innovativi dei propri clienti. L'azienda, grazie alla rete di relazioni a livello nazionale ed internazionale, ha ottenuto importanti finanziamenti in progetti RD (FP7 e H2020) della Comunità Europea. Sync Lab si sta sempre più specializzando in vari settori d'impiego: dal settore bancario a quello assicurativo con una nicchia importante nell'ambito sanità, in cui vanta un prodotto d'eccellenza per la gestione delle cliniche private. L'azienda inoltre ha recentemente fondato una collegata Sync Security che si occupa espressamente del mondo della cyber security e sicurezza informatica in genere.

Capitolo 2

Organizzazione del progetto

Il seguente capitolo ha lo scopo di descrivere nel dettaglio il progetto di stage, gli obiettivi e il piano di lavoro concordati.

2.1 Proposta di stage

Lo stage ha avuto luogo nella sede di Padova dell'azienda SyncLab, esso ha avuto una durata di circa 300 ore, suddivise in 8 settimane, con data di inizio il 17 giugno 2019 e fine il 30 agosto 2019.

Lo scopo del progetto di stage consiste nello sviluppo di una [Web Application](#) avente architettura a [microservizi](#). L'applicazione ha la finalità di sostituire la compilazione di un foglio di calcolo contenente l'insieme delle competenze possedute da una persona insieme al livello raggiunto per ciascuna di esse. Tale foglio è richiesto dall'azienda prima dello svolgimento di un colloquio.

L'idea è quindi di sviluppare un portale in grado di raccogliere e organizzare i dati delle persone richiedenti un colloquio presso SyncLab.

L'approccio adottato prevede l'utilizzo di [Framework](#) moderni:

- **Front-end:** il [Framework](#) adottato è [Angular](#);
- **Back-end** la realizzazione del [Back-end](#) ha richiesto l'utilizzo di più tecnologie:
 - Per la realizzazione dell'architettura a [microservizi](#) viene utilizzato [Spring](#), un [Framework](#) scritto in [Java](#)
 - Come [DBMS](#) la scelta viene utilizzato [MongoDb](#), database *NoSql* orientato ai documenti e non relazionale;

Il carico di lavoro richiesto per la realizzazione del progetto è suddiviso fra diverse persone aventi lo stage presso SyncLab, dove il ruolo del sottoscritto comprendeva la realizzazione di diverse maschere facenti parte dell'interfaccia grafica.

2.2 Il progetto: SyncRec

2.2.1 Prime cinque settimane

Come prima fase del tirocinio è richiesto l'apprendimento di diverse tecnologie necessarie alla comprensione del progetto, indispensabili per il compimento delle attività di analisi e progettazione del software da sviluppare.

Tali tecnologie, concordate con il tutor aziendale e utili per la realizzazione del progetto sono:

- [JavaScript](#)
- [TypeScript](#)
- [Angular](#)

A scopo didattico, inoltre, sono state incluse le seguenti tecnologie:

- [Spring Core, Boot, Data e Data REST](#)
- [PL/SQL Oracle](#)
- [MongoDb](#)
- [Java Standard Edition](#)
- [Java Enterprise](#)
- [JavaServer Pages \(JSP\)](#)

Per quest'ultime tecnologie non è prevista la realizzazione di alcun prodotto e sono state inserite allo scopo di ampliare le conoscenze possedute dal sottoscritto.

Spring Core, Boot, Data, Data REST e MongoDb sono stati inseriti allo scopo ulteriore di comprendere appieno la logica sottostante l'implementazione del [Back-end](#), non prevista in alcun modo nel piano di lavoro (v. sezione [4.1.7](#) per approfondimenti).

2.2.2 Successive tre settimane

In questa fase è prevista la realizzazione di 3 maschere dell'applicazione, tali maschere sono:

- Maschera di login;
- Maschera di configurazione dell'applicativo;
- Maschere per le operazioni [CRUD](#) sull'entità persona.

Il committente del prodotto è l'azienda SyncLab stessa, dato che si tratta di un applicativo gestionale ad uso esclusivo interno, e la funzione viene svolta dal dr. Fabio Pallaro.

2.3 Obiettivi

La seguente sezione ha lo scopo di fornire un dettaglio maggiore sugli obiettivi e i requisiti concordati per lo svolgimento dello stage.

2.3.1 Notazione

Si farà riferimento ai requisiti secondo le seguenti notazioni:

- **O** per i requisiti obbligatori, vincolanti in quanto obiettivo primario richiesto dal committente;
- **D** per i requisiti desiderabili, non vincolanti o strettamente necessari, ma di riconoscibile valore aggiunto;
- **F** per i requisiti facoltativi, rappresentanti valore aggiunto non strettamente competitivo.

Le sigle precedentemente indicate saranno eseguite da una coppia sequenziale di numeri, identificativo del requisito.

2.3.2 Obiettivi fissati

- **Obbligatori**
 - *O01*: Acquisizione competenze sulle tecnologie concordate;
 - *O02*: Capacità di raggiungere gli obiettivi richiesti in autonomia seguendo il cronoprogramma;
 - *O03*: Portare a termine le modifiche richieste dal cliente con una percentuale di superamento degli *item* di collaudo pari al 50%.
- **Desiderabili**
 - *D01*: Portare un buon contributo durante le fasi di analisi e progettazione;
 - *D02*: Portare a termine le modifiche richieste dal cliente con una percentuale di superamento degli *item* di collaudo pari al 80%.
- **Facoltativi**
 - *F01*: Acquisizione competenze su [Framework Spring Security](#).

2.4 Pianificazione

La seguente sezione ha lo scopo di approfondire la pianificazione dello stage concordata con il tutor aziendale e con il Relatore.

In accordo con l'azienda, il piano è organizzato a cadenza settimanale, dove le prime 5 settimane sono dedicate all'apprendimento e studio delle tecnologie, mentre le ultime 3 sono dedicate all'implementazione del software richiesto. Il tempo allocato prevede uno *slack* nel caso vi siano imprevisti.

- **Prima settimana**
 - Presentazione strumenti di lavoro per la condivisione del materiale di studio e per la gestione dell'avanzamento del percorso (Slack, Trello ecc.);
 - Condivisione scaletta argomenti;
 - Veloce panoramica su metodologie Agile/Scrum;
 - Approfondimenti del linguaggio procedurale *Oracle PL/SQL* e *Mongo DB*.

- **Seconda settimana**

- *Java Enterprise Edition*: Ripasso Generale;
- Apprendimento di *Spring Core* presente in Java Enterprise.

- **Terza settimana**

- Apprendimento di *Spring Data Rest*;
- Apprendimento di *Spring Data*;

- **Quarta settimana**

- Front-end Web: Apprendimento di [JavaScript/TypeScript](#) in [Angular](#).

- **Quinta settimana**

- Front-end Web: Apprendimento di [JavaScript/TypeScript](#) in [Angular](#).

- **Sesta settimana**

- Introduzione all'applicativo SMS(legacy);
- Analisi dei requisiti richiesti dal cliente e degli impatti;
- Implementazione Maschera di Login con [Angular](#).

- **Settima settimana**

- Implementazione Maschera di Configurazione di Sistema;
- Implementazione Maschere [CRUD](#) dell'entità Persona.

- **Ottava Settimana**

- Conclusione dell'implementazione richiesta;
- Verifica dell'intervento - Collaudo Finale;
- Consegna software e messa in esercizio.

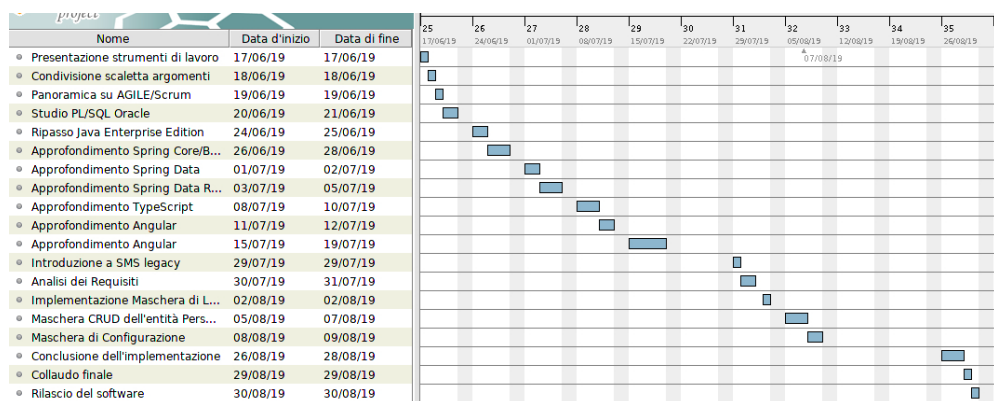


Figura 2.1: Diagramma di Gantt per la pianificazione dello stage

Sono presenti alcune settimane non aventi alcuna attività pianificata: ciò è dovuto a impegni pregressi pianificati dal sottoscritto e dalla chiusura dell'azienda per ferie.

Capitolo 3

Analisi dei requisiti

In questa sezione vengono illustrati i requisiti individuati per il progetto SyncRec.

3.1 Tipologie di utenti

Questa sezione ha lo scopo di dettagliare gli attori individuati; ciascun attore si differenzia dagli altri per i permessi di accesso a determinate parti del sistema:

- **Utente non riconosciuto** : utente che non ha eseguito il *login* presso l'applicazione, nel corso del lavoro svolto dal laureando non sono stati modellati casi d'uso per questo attore.
- **Utente riconosciuto** : utente che ha effettuato l'accesso alla piattaforma SyncRec.
- **applicant**: utente a cui è stato inviato un link specifico per la compilazione dello skillmatrix prima dello svolgimento di un colloquio lavorativo.

3.2 Casi d'uso

La sezione presenta i casi d'uso individuati per l'intero progetto SyncRec.

3.2.1 UC-1: Visualizzazione lista degli Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di visualizzazione lista degli *applicant*.
- **Scenario Principale:**
 1. l'utente visualizza il cognome dell'*applicant*;
 2. l'utente visualizza il nome dell'*applicant*;
 3. l'utente visualizza l'email dell'*applicant*;
- **PostCondizione:** l'utente si trova nella vista di visualizzazione lista degli *applicant*.

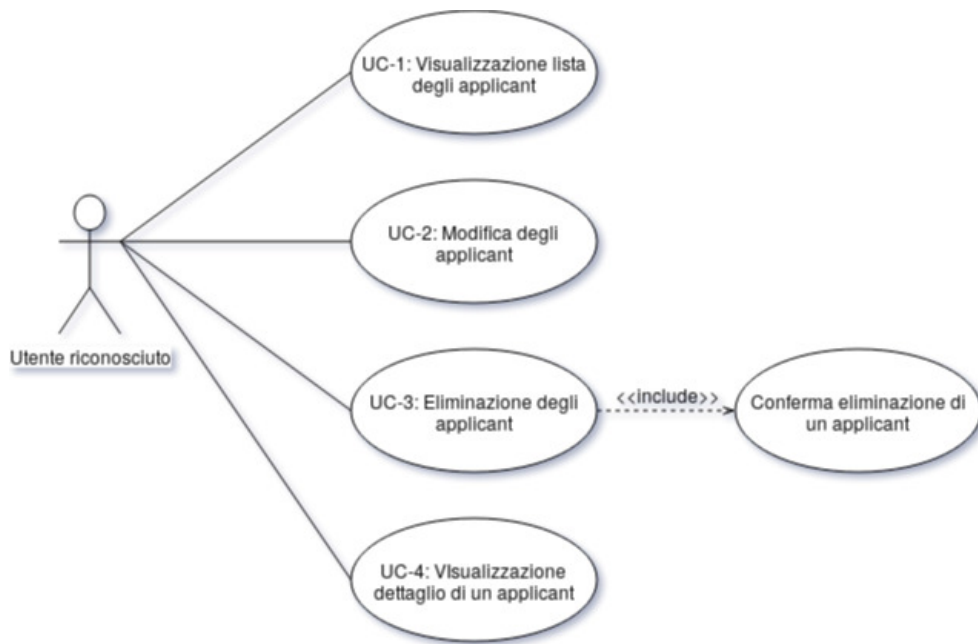


Figura 3.1: Diagramma di Use Case da 1 a 4

UC-1.1: Visualizzazione dettaglio di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di visualizzazione lista degli *applicant*.
- **Scenario Principale:**
 1. l'utente clicca sul pulsante relativo a un *applicant*;
 2. l'utente viene reindirizzato alla vista di dettaglio di un *applicant*;
 3. l'utente visualizza tramite la vista il dettaglio degli *applicant*.
- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;

3.2.2 UC-2: Modifica di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente clicca sul campo dati di un *applicant* che desidera modificare;
 2. l'utente visualizza il widget di modifica per il campo dati relativo;
 3. l'utente inserisce le modifiche desiderate.
- **PostCondizione:**

1. l'utente visualizza il campo dati modificato;
2. l'utente si trova nella vista di dettaglio di un *applicant*.

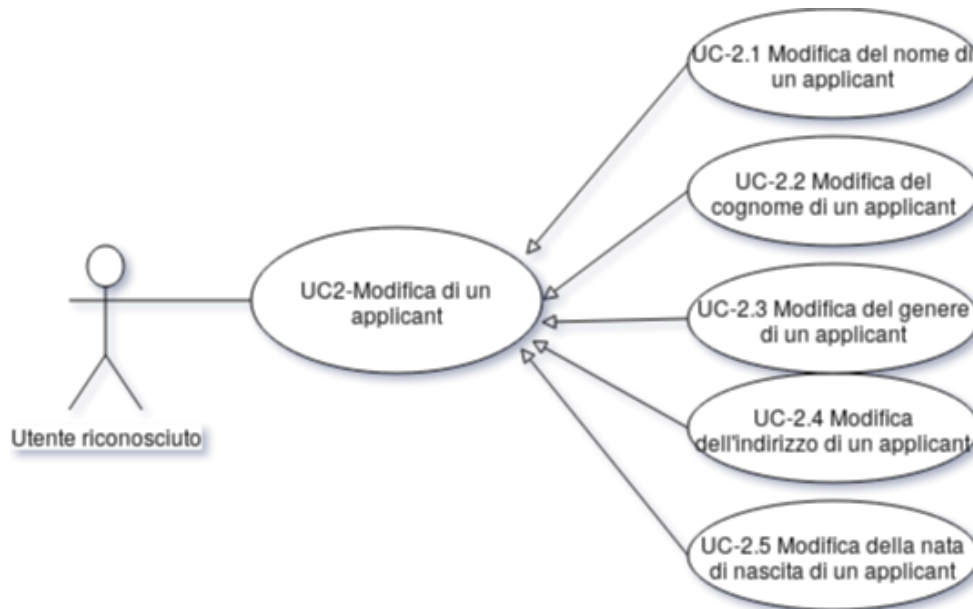


Figura 3.2: Diagramma sottocasi UC-2

UC-2.1: Modifica del nome di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente clicca sul box di testo relativo al nome;
 2. l'utente visualizza il widget di inserimento testo relativo al nome;
 3. l'utente inserisce le modifiche desiderate.
- **PostCondizione:**
 1. l'utente visualizza il campo dati modificato;
 2. l'utente si trova nella vista di dettaglio di un *applicant*.

UC-2.2: Modifica del cognome di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**

1. l'utente clicca sul box di testo relativo al cognome;
2. l'utente visualizza il widget di inserimento testo relativo al cognome;
3. l'utente inserisce le modifiche desiderate.

- **PostCondizione:**

1. l'utente visualizza il campo dati modificato;
2. l'utente si trova nella vista di dettaglio di un *applicant*.

UC-2.3: Modifica del genere di un Applicant

- **Attori:** Utente riconosciuto

- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;

- **Scenario Principale:**

1. l'utente clicca sul box di testo relativo al nome;
2. l'utente visualizza il menù a tendina relativo al genere;
3. l'utente seleziona il dato dal menù a tendina.

- **PostCondizione:**

1. l'utente visualizza il campo dati modificato;
2. l'utente si trova nella vista di dettaglio di un *applicant*.

UC-2.4: Modifica dell'indirizzo di un Applicant

- **Attori:** Utente riconosciuto

- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;

- **Scenario Principale:**

1. l'utente clicca sul box di testo relativo all'indirizzo;
2. l'utente visualizza il widget di inserimento testo relativo all'indirizzo;
3. l'utente inserisce le modifiche desiderate.

- **PostCondizione:**

1. l'utente visualizza il campo dati modificato;
2. l'utente si trova nella vista di dettaglio di un *applicant*.

UC-2.5: Modifica della data di nascita di un Applicant

- **Attori:** Utente riconosciuto

- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;

- **Scenario Principale:**

1. l'utente clicca sul box di testo relativo alla data di nascita;
2. l'utente visualizza il widget di inserimento data relativo alla data di nascita;

3. l'utente inserisce le modifiche desiderate.

- **PostCondizione:**

1. l'utente visualizza il campo dati modificato;
2. l'utente si trova nella vista di dettaglio di un *applicant*.

UC-2.6: Modifica della città di residenza di un Applicant

- **Attori:** Utente riconosciuto

- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;

- **Scenario Principale:**

1. l'utente clicca sul box di testo relativo alla città di residenza;
2. l'utente visualizza il widget di inserimento testo relativo alla città di residenza;
3. l'utente inserisce le modifiche desiderate.

- **PostCondizione:**

1. l'utente visualizza il campo dati modificato;
2. l'utente si trova nella vista di dettaglio di un *applicant*.

UC-2.7: Modifica dell'email di un Applicant

- **Attori:** Utente riconosciuto

- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;

- **Scenario Principale:**

1. l'utente clicca sul box di testo relativo all'email;
2. l'utente visualizza il widget di inserimento testo relativo all'email;
3. l'utente inserisce le modifiche desiderate.

- **PostCondizione:**

1. l'utente visualizza il campo dati modificato;
2. l'utente si trova nella vista di dettaglio di un *applicant*.

UC-2.8: Modifica dell'ambito lavorativo di un Applicant

- **Attori:** Utente riconosciuto

- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;

- **Scenario Principale:**

1. l'utente clicca sul box di testo relativo all'ambito lavorativo,
2. l'utente visualizza il menù a tendina relativo all'ambito lavorativo;
3. l'utente seleziona il dato desiderato dal menù a tendina.

- **PostCondizione:**

1. l'utente visualizza il campo dati modificato;
2. l'utente si trova nella vista di dettaglio di un *applicant*.

UC-2.9: Modifica dell'anzianità di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente clicca sul box di testo relativo all'anzianità;
 2. l'utente visualizza il menù a tendina relativo all'anzianità;
 3. l'utente seleziona il dato desiderato dal menù a tendina.
- **PostCondizione:**
 1. l'utente visualizza il campo dati modificato;
 2. l'utente si trova nella vista di dettaglio di un *applicant*.

UC-2.10: Modifica del titolo di studio di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente clicca sul box di testo relativo all'anzianità;
 2. l'utente visualizza il menù a tendina relativo al titolo di studio;
 3. l'utente seleziona il dato desiderato dal menù a tendina.
- **PostCondizione:**
 1. l'utente visualizza il campo dati modificato;
 2. l'utente si trova nella vista di dettaglio di un *applicant*.

UC-2.11: Modifica della disponibilità geografica di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente clicca sul box di testo relativo alla disponibilità geografica;
 2. l'utente visualizza form relativo alla disponibilità geografica;
 3. l'utente seleziona le opzioni desiderate.
- **PostCondizione:**
 1. l'utente visualizza il campo dati modificato;
 2. l'utente si trova nella vista di dettaglio di un *applicant*.

UC-2.12: Modifica del preavviso di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente clicca sul box di testo relativo al preavviso;
 2. l'utente visualizza il widget di inserimento testo relativo al preavviso;
 3. l'utente inserisce le modifiche desiderate.
- **PostCondizione:**
 1. l'utente visualizza il campo dati modificato;
 2. l'utente si trova nella vista di dettaglio di un *applicant*.

UC-2.13: Modifica del telefono di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente clicca sul box di testo relativo al telefono;
 2. l'utente visualizza il widget di inserimento testo relativo al telefono;
 3. l'utente inserisce le modifiche desiderate.
- **PostCondizione:**
 1. l'utente visualizza il campo dati modificato;
 2. l'utente si trova nella vista di dettaglio di un *applicant*.

UC-2.14: Download del curriculum di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente clicca sul pulsante relativo al curriculum di un *applicant*.
- **PostCondizione:**
 1. l'utente effettua il download del curriculum dell'*applicant*.

UC-2.15: Salvataggio delle modifiche di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente clicca sul pulsante di salvataggio delle modifiche
- **PostCondizione:**
 1. l'utente visualizza i campi dati modificati;
 2. le modifiche vengono confermate;
 3. l'utente si trova nella vista di dettaglio di un *applicant*;
 4. viene visualizzato un messaggio di successo.
- **Estensioni:**
 1. l'utente visualizza il messaggio di errore per un campo obbligatorio relativo al nome (UC-2.15.1);
 2. l'utente visualizza il messaggio di errore relativo all'errato inserimento del nome (UC-2.15.2);
 3. l'utente visualizza il messaggio di errore per un campo obbligatorio relativo al cognome (UC-2.15.3);
 4. l'utente visualizza il messaggio di errore relativo all'errato inserimento del cognome (UC-2.15.4);
 5. l'utente visualizza il messaggio di errore per un campo obbligatorio relativo alla data di nascita (UC-2.15.5);
 6. l'utente visualizza il messaggio di errore relativo all'errato inserimento della data di nascita (UC-2.15.6);
 7. l'utente visualizza il messaggio di errore per un campo obbligatorio relativo all'email (UC-2.15.1);
 8. l'utente visualizza il messaggio di errore relativo all'errato inserimento dell'email (UC-2.15.2).

UC-2.15.1: Visualizzazione errore nome obbligatorio

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant* e ha tentato di cancellare del tutto il campo del nome di un *applicant*.
- **Scenario Principale:**
 1. Viene visualizzato un errore che previene il salvataggio dei dati inconsistenti.
- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*.

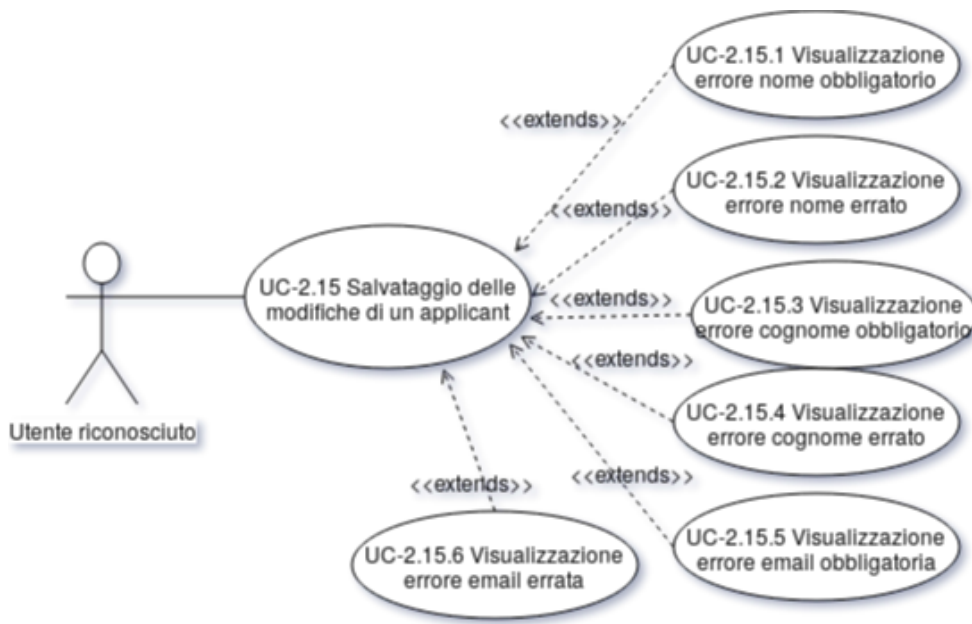


Figura 3.3: Diagramma Use Case sulla visualizzazione errori relativi al salvataggio dell'*applicant*

UC-2.15.2: Visualizzazione errore nome errato

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant* e ha tentato di inserire un nome con caratteri speciali.
- **Scenario Principale:**
 1. Viene visualizzato un errore che previene il salvataggio dei dati inconsistenti.
- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*.

UC-2.15.3: Visualizzazione errore cognome obbligatorio

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant* e ha tentato di cancellare del tutto il campo del cognome di un *applicant*.
- **Scenario Principale:**
 1. Viene visualizzato un errore che previene il salvataggio dei dati inconsistenti.
- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*.

UC-2.15.4: Visualizzazione errore cognome errato

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant* e ha tentato di inserire un cognome con caratteri speciali.
- **Scenario Principale:**
 1. Viene visualizzato un errore che previene il salvataggio dei dati inconsistenti.
- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*.

UC-2.15.5: Visualizzazione errore email obbligatoria

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant* e ha tentato di cancellare del tutto il campo della email di un *applicant*.
- **Scenario Principale:**
 1. Viene visualizzato un errore che previene il salvataggio dei dati inconsistenti.
- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*.

UC-2.15.6: Visualizzazione errore email errata

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant* e ha tentato di inserire una email che non rispetta il formato comunemente accettato.
- **Scenario Principale:**
 1. Viene visualizzato un errore che previene il salvataggio dei dati inconsistenti.
- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*.

UC-2.16: Annullamento delle modifiche di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente clicca sul pulsante di annullamento delle modifiche.
- **PostCondizione:**
 1. l'utente visualizza i campi dati nello stato antecedente alla modifica degli stessi;
 2. le modifiche vengono annullate;
 3. l'utente si trova nella vista di dettaglio di un *applicant*.

UC-2.17: Modifica delle note generiche di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente clicca sul box di testo relativo alle note generiche;
 2. l'utente visualizza il widget di inserimento testo relativo alle note generiche;
 3. l'utente inserisce le modifiche desiderate.
- **PostCondizione:**
 1. l'utente visualizza il campo dati modificato;
 2. l'utente si trova nella vista di dettaglio di un *applicant*.

UC-2.18: Modifica delle note relative a un colloquio di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente clicca sul box di testo relativo alle note del colloquio;
 2. l'utente visualizza il widget di inserimento testo relativo alle note del colloquio;
 3. l'utente inserisce le modifiche desiderate.
- **PostCondizione:**
 1. l'utente visualizza il campo dati modificato;
 2. l'utente si trova nella vista di dettaglio di un *applicant*.

UC-2.19: Modifica delle note riguardanti lo status lavorativo di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente clicca sul box di testo relativo alle note generiche;
 2. l'utente visualizza il widget di inserimento testo relativo alle note generiche;
 3. l'utente inserisce le modifiche desiderate.
- **PostCondizione:**
 1. l'utente visualizza il campo dati modificato;
 2. l'utente si trova nella vista di dettaglio di un *applicant*.

UC-2.20: Modifica delle note riguardanti le considerazioni aggiuntive relative a un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente clicca sul box di testo relativo alle note aggiuntive;
 2. l'utente visualizza il widget di inserimento testo relativo alle note aggiuntive;
 3. l'utente inserisce le modifiche desiderate.
- **PostCondizione:**
 1. l'utente visualizza il campo dati modificato;
 2. l'utente si trova nella vista di dettaglio di un *applicant*.

3.2.3 UC-3: Eliminazione di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di visualizzazione lista degli *applicant*.
- **Scenario Principale:**
 1. l'utente clicca il pulsante "Elimina" riferito a un certo *applicant*.
- **PostCondizione:** Viene visualizzato un messaggio di richiesta di conferma.

UC-3.1: Conferma eliminazione di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di visualizzazione lista degli *applicant* e ha richiesto l'eliminazione di un *applicant*.
- **Scenario Principale:**
 1. L'utente visualizza il messaggio di richiesta di conferma;
 2. L'utente preme il tasto "Ok".
- **PostCondizione:** si chiude il messaggio di conferma, avviene l'eliminazione dell'*applicant* e il *refresh* della pagina.
- **Estensioni:**
 1. l'utente preme il tasto "Annulla";
 2. il messaggio di conferma viene chiuso.

3.2.4 UC-4: Visualizzazione dettaglio di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente visualizza il nome dell'*applicant* (UC-4.1);
 2. l'utente visualizza il cognome dell'*applicant* (UC-4.2);
 3. l'utente visualizza il genere dell'*applicant* (UC-4.3);
 4. l'utente visualizza la data di nascita dell'*applicant* (UC-4.4);
 5. l'utente visualizza l'email dell'*applicant* (UC-4.5);
 6. l'utente visualizza il numero di telefono dell'*applicant* (UC-4.6);
 7. l'utente visualizza l'indirizzo dell'*applicant* (UC-4.7);
 8. l'utente visualizza la città residenza dell'*applicant* (UC-4.8);
 9. l'utente visualizza il preavviso richiesto dall'*applicant* per una chiamata (UC-4.9);
 10. l'utente visualizza il titolo di studio dell'*applicant* (UC-4.10);
 11. l'utente visualizza il livello di anzianità dell'*applicant* (UC-4.11);
 12. l'utente visualizza l'ambito di lavoro dell'*applicant* (UC-4.12);
 13. l'utente visualizza se lo status dell'*applicant* è scartato o meno (UC-4.13);
 14. l'utente visualizza le città per cui l'*applicant* ha fornito disponibilità per il trasferimento (UC-4.14);
 15. l'utente visualizza delle note generiche relative all'*applicant* (UC-4.15);
 16. l'utente visualizza delle note relative al colloquio dell'*applicant* (UC-4.16);
 17. l'utente visualizza delle note riguardanti lo status lavorativo dell'*applicant* (UC-4.17);
 18. l'utente visualizza altre considerazioni aggiuntive relative all'*applicant* (UC-4.18).
- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*.

UC-4.1: Visualizzazione nome di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente visualizza il nome dell'*applicant*;
- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*.

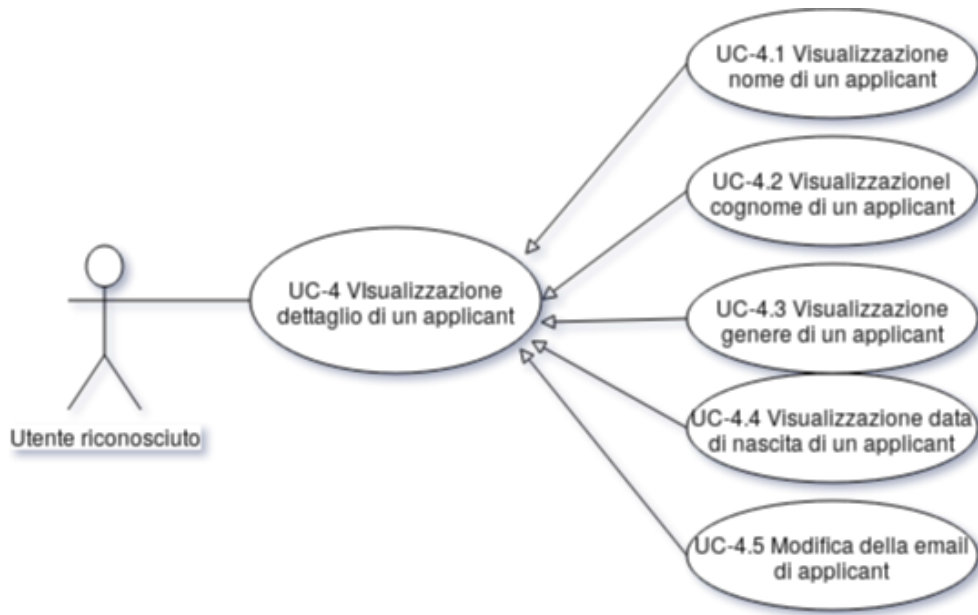


Figura 3.4: Diagramma sottocasi UC-4

UC-4.2: Visualizzazione cognome di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente visualizza il cognome dell'*applicant*;
- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*.

UC-4.3: Visualizzazione genere di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente visualizza il genere dell'*applicant*;
- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*.

UC-4.4: Visualizzazione data di nascita di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**

1. l'utente visualizza il nome dell'*applicant*;

- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*.

UC-4.5: Visualizzazione email di un Applicant

- **Attori:** Utente riconosciuto

- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;

- **Scenario Principale:**

1. l'utente visualizza il nome dell'*applicant*;

- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*.

UC-4.6: Visualizzazione numero di telefono di un Applicant

- **Attori:** Utente riconosciuto

- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;

- **Scenario Principale:**

1. l'utente visualizza il numero di telefono dell'*applicant*;

- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*.

UC-4.7: Visualizzazione l'indirizzo di un Applicant

- **Attori:** Utente riconosciuto

- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;

- **Scenario Principale:**

1. l'utente visualizza l'indirizzo dell'*applicant*;

- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*.

UC-4.8: Visualizzazione città di residenza di un Applicant

- **Attori:** Utente riconosciuto

- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;

- **Scenario Principale:**

1. l'utente visualizza la città di residenza dell'*applicant*;

- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*.

UC-4.9: Visualizzazione preavviso richiesto da un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente visualizza il preavviso richiesto dall'*applicant*;
- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*.

UC-4.10: Visualizzazione titolo di studio di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente visualizza il titolo di studio dell'*applicant*;
- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*.

UC-4.11: Visualizzazione livello di anzianità di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente visualizza il livello di anzianità dell'*applicant*;
- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*.

UC-4.12: Visualizzazione ambito lavorativo di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente visualizza l'ambito lavorativo dell'*applicant*;
- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*.

UC-4.13: Visualizzazione status (scartato o meno) di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente visualizza se lo status dell'*applicant* è scartato o meno;
- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*.

UC-4.14: Visualizzazione disponibilità geografica di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente visualizza le città per cui l'*applicant* ha fornito disponibilità per il trasferimento;
- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*.

UC-4.15: Visualizzazione note generiche di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente visualizza le note generiche relative all'*applicant*;
- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*.

UC-4.16: Visualizzazione note relative al colloquio di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente visualizza le note relative al colloquio dell'*applicant*;
- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*.

UC-4.17: Visualizzazione note sullo status lavorativo di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente visualizza note riguardanti lo status lavorativo dell'*applicant*;
- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*.

UC-4.18: Visualizzazione note aggiuntive relative a un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*;
- **Scenario Principale:**
 1. l'utente visualizza note riguardanti le considerazioni aggiuntive relative all'*applicant*;
- **PostCondizione:** l'utente si trova nella vista di dettaglio di un *applicant*.

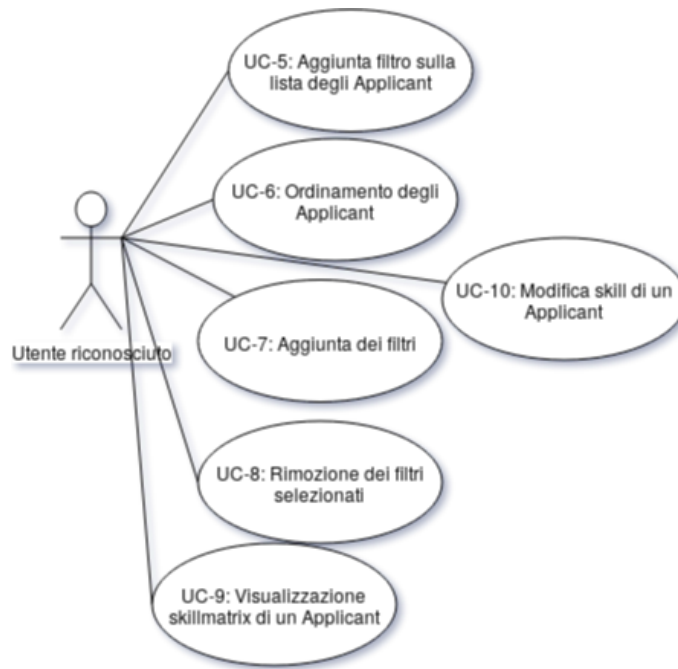


Figura 3.5: Diagramma di Use Case da UC-5 a UC-10

3.2.5 UC-5: Aggiunta filtro sulla lista degli Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di visualizzazione lista degli *applicant*.
- **Scenario Principale:**
 1. l'utente inserisce nel campo di testo un carattere o un insieme di caratteri.
- **PostCondizione:** viene visualizzata la lista degli *applicant* dove nome e cognome contengono i caratteri inseriti.

3.2.6 UC-6: Ordinamento degli Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di visualizzazione lista degli *applicant*.
- **Scenario Principale:**
 1. l'utente clicca sul *header* della lista riportante il campo desiderato;
 2. gli *applicant* vengono posti in ordine crescente secondo il campo desiderato.
- **PostCondizione:** l'utente si trova nella vista di visualizzazione lista degli *applicant*.

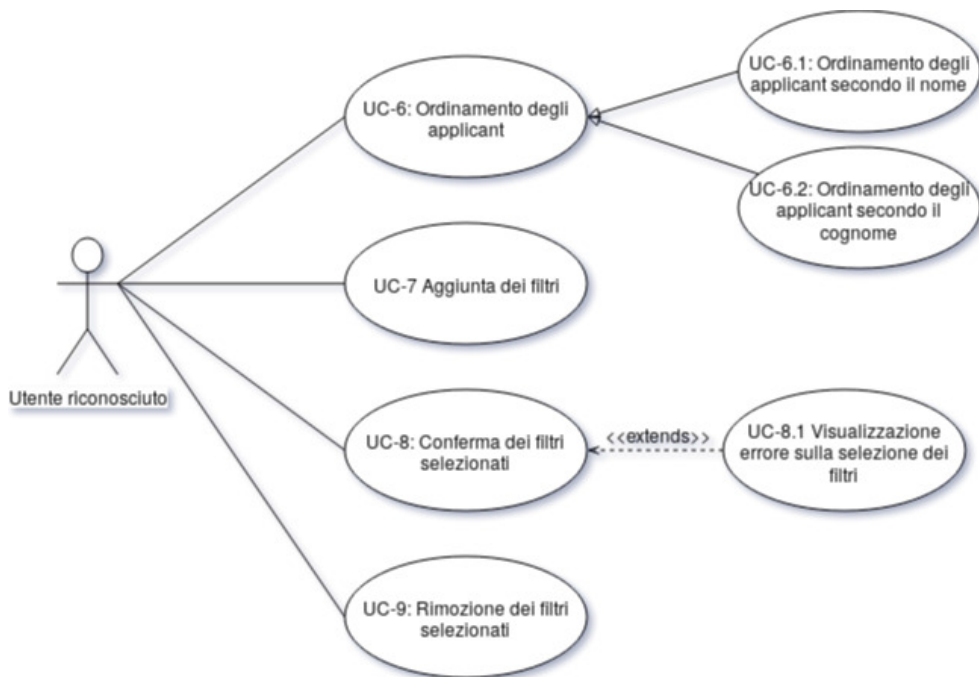


Figura 3.6: Diagramma di Use Case delle operazioni ausiliare sulla lista degli *applicant*

UC-6.1: Ordinamento degli Applicant secondo il nome

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di visualizzazione lista degli *applicant*.
- **Scenario Principale:**
 1. l'utente clicca sul *header* della lista riportante il campo "Nome";
 2. gli *applicant* vengono posti in ordine crescente secondo il nome.
- **PostCondizione:** l'utente si trova nella vista di visualizzazione lista degli *applicant*.

UC-6.2: Ordinamento degli Applicant secondo il cognome

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di visualizzazione lista degli *applicant*.
- **Scenario Principale:**
 1. l'utente clicca sul *header* della lista riportante il campo "Cognome";
 2. gli *applicant* vengono posti in ordine crescente secondo il cognome.
- **PostCondizione:** l'utente si trova nella vista di visualizzazione lista degli *applicant*.

3.2.7 UC-7: Aggiunta dei filtri

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista relativa ai filtri.
- **Scenario Principale:**
 1. l'utente aggiunge un filtro mediante l'inserimento di un input;
- **PostCondizione:** l'utente si trova nella vista relativa ai filtri e visualizza l'input immesso sullo schermo.

UC7.1: Aggiunta di un filtro per il nome

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista relativa ai filtri.
- **Scenario Principale:**
 1. l'utente inserisce una parola chiave in un campo di testo dedicato al nome;
- **PostCondizione:** l'utente si trova nella vista relativa ai filtri e visualizza l'input immesso sullo schermo.

UC7.2: Aggiunta di un filtro per il cognome

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista relativa ai filtri.
- **Scenario Principale:**
 1. l'utente inserisce una parola chiave in un campo di testo dedicato al cognome.
- **PostCondizione:** l'utente si trova nella vista relativa ai filtri e visualizza l'input immesso sullo schermo.

UC-7.3: Aggiunta di un filtro per il genere

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista relativa ai filtri.
- **Scenario Principale:**
 1. l'utente seleziona da un menù a tendina il genere desiderato.
- **PostCondizione:** l'utente si trova nella vista relativa ai filtri e visualizza l'opzione selezionata sullo schermo.

UC-7.4: Aggiunta di un filtro per il titolo di studio

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista relativa ai filtri.
- **Scenario Principale:**
 1. l'utente seleziona da un menù a tendina il titolo di studio desiderato.
- **PostCondizione:** l'utente si trova nella vista relativa ai filtri e visualizza l'opzione selezionata sullo schermo.

UC-7.5: Aggiunta di un filtro per l'ambito lavorativo

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista relativa ai filtri.
- **Scenario Principale:**
 1. l'utente seleziona da un menù a tendina l'ambito lavorativo desiderato.
- **PostCondizione:** l'utente si trova nella vista relativa ai filtri e visualizza l'opzione selezionata sullo schermo.

UC-7.6: Aggiunta di un filtro relativo ad una skill

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista relativa ai filtri.
- **Scenario Principale:**
 1. l'utente seleziona da un menù a tendina il nome della skill desiderato.
- **PostCondizione:** l'utente si trova nella vista relativa ai filtri e visualizza l'opzione selezionata sullo schermo.

UC-7.7: Aggiunta di un filtro relativo al livello di una skill

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista relativa ai filtri.
- **Scenario Principale:**
 1. l'utente seleziona da un menù a tendina un valore da 1 a 4 rappresentante il livello della skill desiderato.
- **PostCondizione:** l'utente si trova nella vista relativa ai filtri e visualizza l'opzione selezionata sullo schermo.

UC-7.8: Aggiunta di un filtro per la disponibilità geografica

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista relativa ai filtri.
- **Scenario Principale:**
 1. l'utente seleziona una o più città desiderate.
- **PostCondizione:** l'utente si trova nella vista relativa ai filtri e visualizza le opzioni selezionate sullo schermo.

UC-7.9: Aggiunta di un filtro per l'anzianità

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista relativa ai filtri.
- **Scenario Principale:**
 1. l'utente seleziona da un menù a tendina il livello di anzianità desiderato.
- **PostCondizione:** l'utente si trova nella vista relativa ai filtri e visualizza l'opzione selezionata sullo schermo.

UC-7.10: Conferma dei filtri selezionati

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista relativa ai filtri.
- **Scenario Principale:**
 1. l'utente clicca sul tasto "Applica filtri selezionati";
 2. l'utente viene reindirizzato alla vista di visualizzazione lista degli *applicant*.
- **PostCondizione:** l'utente si trova nella vista relativa alla visualizzazione degli *applicant* e visualizza li visualizza secondo i filtri selezionati;
- **Estensioni:** l'utente visualizza un messaggio di errore relativo all'inserimento dei filtri (UC-7.11.1);

UC-7.10.1: Visualizzazione errore sulla selezione dei filtri

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista relativa ai filtri e non ha selezionato congiuntamente una skill e un livello, ma solo uno dei due.
- **Scenario Principale:**
 1. viene visualizzato un errore a schermo che invita l'utente a completare la selezione dei filtri.
- **PostCondizione:** l'utente si trova nella vista relativa ai filtri.

3.2.8 UC-8: Rimozione dei filtri selezionati

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista relativa ai filtri
- **Scenario Principale:**
 1. l'utente clicca sul tasto "Annulla";
 2. vengono rimossi tutti i filtri precedentemente selezionati.
- **PostCondizione:** l'utente si trova nella vista relativa ai filtri.

3.2.9 UC-9: Visualizzazione skillmatrix di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un *applicant*
- **Scenario Principale:**
 1. l'utente clicca sul pulsante relativo alla visualizzazione skillmatrix di un *applicant*
- **PostCondizione:** l'utente si trova nella vista relativa alla visualizzazione skillmatrix di un *applicant*.

UC-9.1: Visualizzazione nome della skill

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista relativa alla visualizzazione skillmatrix di un *applicant*.
- **Scenario Principale:**
 1. l'utente visualizza il nome della skill posseduta dall'*applicant*
- **PostCondizione:** l'utente si trova nella vista relativa alla visualizzazione skillmatrix di un *applicant*.

UC-9.2: Visualizzazione livello della skill

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista relativa alla visualizzazione skillmatrix di un *applicant*.
- **Scenario Principale:**
 1. l'utente visualizza il livello della skill posseduta dall'*applicant*
- **PostCondizione:** l'utente si trova nella vista relativa alla visualizzazione skillmatrix di un *applicant*.

3.2.10 UC-10: Modifica skill di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista relativa alla visualizzazione skillmatrix di un *applicant*.
- **Scenario Principale:**
 1. l'utente seleziona una skill da modificare;
 2. l'utente seleziona il livello desiderato da un menù a tendina.
- **PostCondizione:** l'utente si trova nella vista relativa alla visualizzazione skillmatrix di un *applicant* e viene visualizzato un pulsante di conferma.

UC-10.1: Conferma della modifica skill di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista relativa alla visualizzazione skillmatrix di un *applicant* e ha effettuato una modifica su uno o più livelli di una skill.
- **Scenario Principale:**
 1. l'utente clicca sul pulsante di conferma.
- **PostCondizione:** l'utente si trova nella vista relativa alla visualizzazione skillmatrix di un *applicant* e le modifiche precedentemente apportate risultano salvate.

UC-10.2: Annullamento della modifica skill di un Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista relativa alla visualizzazione skillmatrix di un *applicant* e ha effettuato una modifica su uno o più livelli di una skill.
- **Scenario Principale:**
 1. l'utente clicca sul pulsante di annullamento.
- **PostCondizione:** l'utente si trova nella vista relativa alla visualizzazione skillmatrix di un *applicant* e le modifiche precedentemente apportate vengono eliminate, ripristinando lo stato originale dello skillmatrix.

3.3 Tracciamento dei requisiti

In corrispondenza dell'analisi degli use case effettuata con il supporto del tutor aziendale, vengono identificati i requisiti relativi al progetto di stage, essi sono definiti secondo la seguente notazione:

R(Importanza)(Tipologia)(Codice)

Dove per ciascun parametro possono essere assegnate le seguenti opzioni:

- **Importanza**
 - O: indica un requisito obbligatorio
 - D: indica un requisito desiderabile
 - P: indica un requisito facoltativo (opzionale)
- **Tipologia**
 - F : indica un requisito funzionale. Rappresenta le funzionalità che il prodotto deve fornire.
 - Q : indica un requisito qualitativo. Rappresenta gli elementi per aumentare la qualità del prodotto finale
 - P : indica un requisito prestazionale. Rappresenta un valore misurabile delle performance raggiunte dal prodotto terminato.
 - V : indica un requisito di vincolo. Rappresenta le limitazioni che il prodotto deve rispettare.
- **Codice** : un codice progressivo utile per identificare univocamente i requisiti individuati.

3.3.1 Requisiti funzionali

Codice	Descrizione	Use Case
ROF01	L'utente riconosciuto deve poter visualizzare la lista degli <i>applicant</i> registrati presso SyncRec	UC-1
ROF02	L'utente deve poter accedere alla vista di dettaglio di un <i>applicant</i> a partire dalla lista del totale delle persone richiedenti registrate	UC-1.1
ROF03	L'utente deve poter modificare i dati registrati di un <i>applicant</i>	UC-2
ROF04	L'utente deve poter modificare il nome di un <i>applicant</i>	UC-2.1
ROF05	L'utente deve poter modificare il nome di un <i>applicant</i>	UC-2.2
ROF06	L'utente deve poter modificare il genere di un <i>applicant</i>	UC-2.3
ROF07	L'utente deve poter modificare l'indirizzo di un <i>applicant</i>	UC-2.4
ROF08	L'utente deve poter modificare la data di nascita di un <i>applicant</i>	UC-2.5
ROF09	L'utente deve poter modificare la città di residenza di un <i>applicant</i>	UC 2.6
ROF10	L'utente deve poter modificare l'email di un <i>applicant</i>	UC-2.7
ROF11	L'utente deve poter modificare l'ambito lavorativo di un <i>applicant</i>	UC-2.8
ROF12	L'utente deve poter modificare il titolo di studio di un <i>applicant</i>	UC-2.10
ROF13	L'utente deve poter modificare la disponibilità geografica di un <i>applicant</i>	UC-2.11
ROF14	L'utente deve poter modificare il tempo di preavviso di un <i>applicant</i>	UC-2.12

Codice	Descrizione	Use Case
ROF15	L'utente deve poter modificare il telefono di un <i>applicant</i>	UC-2.13
ROF16	L'utente deve poter scaricare il curriculum di un <i>applicant</i>	UC-2.14
ROF17	L'utente deve poter confermare le modifiche apportate a un <i>applicant</i>	UC-2.15
ROF18	L'utente deve poter annullare le modifiche apportate a un <i>applicant</i>	UC-2.16
ROF19	L'utente deve poter visualizzare un messaggio di errore se il nome non è presente	UC-2.15.1
ROF20	L'utente deve poter visualizzare un messaggio di errore se il nome presenta dei caratteri non alfabetici	2.15.2
ROF21	L'utente deve poter visualizzare un messaggio di errore se il cognome non è presente	UC-2.15.3
ROF22	L'utente deve poter visualizzare un messaggio di errore se il cognome presenta dei caratteri non alfabetici	UC-2.15.4
ROF23	L'utente deve poter visualizzare un messaggio di errore se l'email non è presente	UC-2.15.5
ROF24	L'utente deve poter visualizzare un messaggio di errore se l'email presenta una struttura non corretta	UC-2.15.6
ROF24	L'utente deve poter modificare le note generiche di un <i>applicant</i>	UC-2.17
ROF25	L'utente deve poter modificare le note relative al colloquio di un <i>applicant</i>	UC-2.18
ROF26	L'utente deve poter le note riguardanti lo status lavorativo di un <i>applicant</i>	UC-2.19
ROF27	L'utente deve poter mopdificare le note riguardanti le considerazioni aggiuntive relative a un <i>applicant</i>	UC-2.20
ROF28	L'utente deve poter eliminare definitivamente un <i>applicant</i>	UC-3
ROF29	L'utente deve poter confermare l'eliminazione di un <i>applicant</i>	UC-3.1
ROF30	L'utente deve poter visualizzare il dettaglio di un <i>applicant</i>	UC-4
ROF31	L'utente deve poter visualizzare il nome di un <i>applicant</i>	UC-4.1
ROF32	L'utente deve poter visualizzare il cognome di un <i>applicant</i>	UC-4.2
ROF33	L'utente deve poter visualizzare il genere di un <i>applicant</i>	UC-4.3
ROF34	L'utente deve poter visualizzare la data di nascita di un <i>applicant</i>	UC-4.4
ROF35	L'utente deve poter visualizzare l'email di un <i>applicant</i>	UC-4.5
ROF36	L'utente deve poter visualizzare il numero di telefono di un <i>applicant</i>	UC-4.6
ROF37	L'utente deve poter visualizzare l'indirizzo di un <i>applicant</i>	UC-4.7
ROF38	L'utente deve poter visualizzare la città di residenza di un <i>applicant</i>	UC-4.8
ROF39	L'utente deve poter visualizzare il preavviso richiesto da un <i>applicant</i>	UC-4.9
ROF40	L'utente deve poter visualizzare il titolo di studio di un <i>applicant</i>	UC-4.10
ROF41	L'utente deve poter visualizzare il livello di anzianità di un <i>applicant</i>	UC-4.11

Codice	Descrizione	Use Case
ROF42	L'utente deve poter visualizzare l'ambito di lavoro di un <i>applicant</i>	UC-4.12
ROF43	L'utente deve poter visualizzare se l' <i>applicant</i> è scartato o meno	UC-4.13
ROF44	L'utente deve poter visualizzare le note generiche relative a un <i>applicant</i>	UC-4.15
ROF45	L'utente deve poter visualizzare le note relative al colloquio dell' <i>applicant</i>	UC-4.16
ROF46	L'utente deve poter visualizzare le note relative allo status lavorativo dell' <i>applicant</i>	UC-4.17
ROF47	L'utente deve poter visualizzare altre considerazioni aggiuntive relative all' <i>applicant</i>	UC-4.18
ROF48	L'utente deve poter filtrare la lista degli <i>applicant</i> secondo il nome e cognome	UC-5
ROF49	L'utente deve poter ordinare la lista degli <i>applicant</i> secondo determinati criteri	UC-6
ROF50	L'utente deve poter ordinare la lista degli <i>applicant</i> secondo il nome	UC-6.1
ROF51	L'utente deve poter ordinare la lista degli <i>applicant</i> secondo il cognome	UC-6.2
ROF52	L'utente deve poter aggiungere filtri alla lista degli <i>applicant</i> secondo determinati criteri	UC-7
ROF53	L'utente deve aggiungere un filtro per il nome sulla lista degli <i>applicant</i>	UC-7.1
ROF54	L'utente deve aggiungere un filtro per il cognome sulla lista degli <i>applicant</i>	UC-7.2
ROF55	L'utente deve aggiungere un filtro per il genere sulla lista degli <i>applicant</i>	UC-7.3
ROF56	L'utente deve aggiungere un filtro per il titolo di studio sulla lista degli <i>applicant</i>	UC-7.4
ROF57	L'utente deve aggiungere un filtro per l'ambito lavorativo sulla lista degli <i>applicant</i>	UC-7.5
ROF58	L'utente deve aggiungere un filtro per il nome di una skill sulla lista degli <i>applicant</i>	UC-7.6
ROF59	L'utente deve aggiungere un filtro per il livello di una skill sulla lista degli <i>applicant</i>	UC-7.7
ROF60	L'utente deve aggiungere un filtro per la disponibilità geografica di un <i>applicant</i>	UC-7.8
ROF61	L'utente deve aggiungere un filtro per l'anzianità di un <i>applicant</i>	UC-7.9
ROF62	L'utente deve poter confermare l'aggiunta dei filtri selezionati	UC-7.10
ROF63	L'utente deve poter annullare l'aggiunta dei filtri selezionati	UC-7.11
ROF64	L'utente deve poter visualizzare un eventuale errore commesso nell'aggiunta dei filtri	7.10.1
ROF65	L'utente deve poter rimuovere i filtri precedentemente selezionati	UC-8

Codice	Descrizione	Use Case
ROF66	L'utente deve poter visualizzare lo skillmatrix di un <i>applicant</i>	UC-9
ROF67	L'utente deve poter visualizzare il nome di una skill posseduta da un <i>applicant</i>	UC-9.1
ROF68	L'utente deve poter visualizzare il livello di una skill posseduta da un <i>applicant</i>	UC-9.2
ROF69	L'utente deve poter modificare il livello di una skill posseduta da un <i>applicant</i>	UC-10
ROF70	L'utente deve poter confermare le modifiche apportate allo skillmatrix di un <i>applicant</i>	UC-10.1
ROF71	L'utente deve poter annullare le modifiche apportate allo skillmatrix di un <i>applicant</i>	UC-10.2

3.3.2 Requisiti di vincolo

Codice	Descrizione
ROV01	Utilizzo del Framework Angular per lo sviluppo dell'applicazione
ROV02	Utilizzo di MongoDB per la gestione dei dati
ROV03	Utilizzo di Spring per il Back-end dell'applicazione
ROV04	Utilizzo di Visual Studio Code come IDE

Capitolo 4

Tecnologie e framework adottati

Il seguente capitolo ha lo scopo di illustrare nel dettaglio le tecnologie e gli strumenti utilizzati nel corso del progetto di stage.

4.1 Linguaggi e framework

La seguente sezione approfondisce le tecnologie adottate(intese come *framework* e linguaggi di programmazione) durante il corso del progetto.

4.1.1 TypeScript

TypeScript è un linguaggio di programmazione open-source sviluppato e mantenuto da Microsoft, esso si configura come un livello di astrazione aggiuntivo a [JavaScript](#), introducendo i classici paradigmi della programmazione ad oggetti, insieme alle funzionalità introdotte da ECMAScript 3 in poi .

Una volta compilato il codice TypeScript, esso produce in *output* codice JavaScript nativo, in questo modo è garantita la compatibilità con tutti browser, [Node.js](#), o *engine* JavaScript che supportino ECMAScript 3; in un progetto [Angular](#), tale processo avviene in automatico tramite esecuzione dei comandi di [Angular CLI](#).

L'utilizzo di tale linguaggio si è reso necessario una volta scelto e adottato *Angular* come *framework* di riferimento per lo sviluppo delle maschere di Front-End.

4.1.2 Angular



Figura 4.1: Logo di Angular

Angular è un framework sviluppato da Google utilizzato per lo sviluppo di applicazioni web responsive e compatibili con la gran parte dei dispositivi odierni, esso nasce come evoluzione di *AngularJS*, abbandonando l'utilizzo di [JavaScript](#) in favore di TypeScript (v. sezione [4.1.1](#)), ciò rende le due versioni non compatibili. I principali vantaggi offerti da Angular sono:

- **Cross-platform:** Angular permette lo sviluppo di *web application*, *mobile Web Application*, applicazioni *desktop* e applicazioni *mobile*;
- **Velocità e performance:** la struttura del *framework* permette di sviluppare applicazioni veloci e dinamiche, l'elaborazione, infatti, viene eseguita quasi interamente su lato client, una volta scaricata l'applicazione dal [Web Server](#); il peso delle applicazioni, inoltre, è stato sensibilmente ridotto;
- **Compatibilità:** rispetto al suo predecessore, Angular semplifica l'interazione e la compatibilità con librerie esterne [JavaScript](#), come [RxJs](#) o [immutable.js](#).

In un progetto Angular, l'elemento cardine risulta essere il *component*, introdotto dalla seconda versione, esso rappresenta l'elemento base della struttura gerarchica, organizzata in moduli, che permette un'efficiente suddivisione del carico di lavoro tra più programmatori, i quali non hanno necessità di conoscere interamente la logica dell'applicazione per apportare un contributo allo sviluppo.

Le successive sezioni hanno lo scopo di dettagliare accuratamente i principali elementi che fanno parte di un'applicazione sviluppata in Angular.

Module

I moduli rappresentano contenitori di blocchi di codice coesi afferenti a una certa funzionalità da implementare, a un *workflow* da eseguire o a un insieme strettamente correlato di capacità sviluppate. Essi possono contenere:

- **Services:** classi addette alla gestione di transizioni con l'esterno o alla manipolazione dei dati (per maggiori dettagli v. sez. [4.1.2](#));
- **Routing Modules:** moduli particolari che permettono di mappare richieste [URL](#) alla visualizzazione di *component* (v. sez. [4.1.2](#));
- **Components:** elemento base di una [Web Application](#) sviluppata in *Angular* (v. sezione [4.1.2](#));

- Altri *file* contenenti codice il cui *scope* è definito dal modulo stesso.

Ogni applicazione sviluppata in *Angular* contiene almeno un modulo, definito *root module* e contenuto nel file `app.module.ts`, il quale deve illustrare i moduli e i *component* dichiarati.

La *Best practice* suggerita dalla documentazione afferma che è opportuno dichiarare per ogni *component* un suo modulo di appartenenza, salvo rari casi dove ciò risulterebbe ridondante (si pensi alla dichiarazione di un *component* contenente una logica basilare e un *template HTML* di dimensioni ridotte), in questo modo, il *root module* conterrà solamente le dichiarazioni dei moduli figli, e andrà aggiornato solo nel caso in cui si renda necessaria l'aggiunta di un nuovo *component*.

Component

I *component* costituiscono elementi di una [Web Application](#) sviluppata in *Angular*, i quali assumono controllo della visualizzazione dei dati e la gestione degli eventi in determinate circostanze definite dal programmatore.

Ogni *component* è costituito da 3 parti:

- un *file* `.ts` che definisce la logica sottostante per l'elaborazione dei dati;
- un *file* `.html` che costituisce il *template* per la visualizzazione su schermo del *component*;
- un *file* `.css` che definisce posizione, dimensione e colore degli elementi definiti nel *template*.

Questo tipo di struttura permette una grande possibilità di riutilizzo del codice: ogni *component*, una volta definito, può essere visualizzato secondo quanto dichiarato nel suo modulo di appartenenza. In questo modo, nel caso in cui fosse necessario riutilizzare un *component* precedentemente definito, sarà sufficiente importare il suo modulo di riferimento ove desiderato.

Service

Angular ha predisposto una struttura facente uso della [Dependency Injection](#), la quale permette di iniettare dipendenze nei *component* senza pregiudicare la manutenibilità del codice; applicando questo approccio è possibile dividere chiaramente la visualizzazione dei dati sullo schermo dal suo relativo recupero.

L'elemento che si occupa del *fetch* dei dati in una applicazione *Angular* è il *service*: la documentazione suggerisce di rendere un *service* **Injectable**, in questo modo è sufficiente che il *component* dichiari la dipendenza nel suo costruttore per poterlo utilizzare. Un'approccio di questo tipo ha l'ulteriore vantaggio di rendere molto più pulita la gestione degli errori nelle transazioni, che è interamente adibita ai *service* facenti le chiamate.

Routing

Il *Routing* costituisce il meccanismo implementato da *Angular* per la navigazione all'interno di una [Web Application](#).

Normalmente un utente per navigare all'interno di un sito tramite *browser* effettua una tra queste tre azioni:

- Digita un [URL](#) nella barra degli indirizzi e naviga direttamente alla pagina desiderata;
- Clicca un *link* all'interno della pagina e il browser effettua la navigazione;
- Usa i pulsanti di *back* e *forward* per visualizzare pagine precedentemente visitate.

Il *Router* di *Angular*, utilizzando questo modello, è in grado di mappare le richieste [URL](#) a delle *view* generate dal client, con l'ulteriore possibilità di inserimento di parametri opzionali per la visualizzazione di determinati dati.

Nel caso del progetto *SyncRec*, il *Routing* tramite parametri opzionali si è rivelato indispensabile per la visualizzazione del dettaglio di una persona.

Funzionamento e implementazione Il meccanismo di implementazione del *Routing* è molto simile a quello dei moduli: ogni *component* deve essere corredato da un *Routing Module* che definisca il *path* e il corrispettivo *component* collegato, successivamente, ogni *Routing Module* deve essere importato e dichiarato nel corrispettivo modulo di appartenenza, in modo che il *root module* possa essere a conoscenza della mappatura completa degli [URL](#) con i *component* associati.

4.1.3 Npm

Npm (Node Package Manager) si configura come il gestore di pacchetti predefinito per l'ambiente *runtime* di [JavaScript NodeJs](#); il suo utilizzo si è rivelato necessario per l'inclusione di alcune dipendenze necessarie per il completamento del progetto ([Angular CLI](#) primo fra tutti).

I pacchetti sono memorizzati in un database remoto chiamato *npm registry* e l'aggiunta, rimozione e aggiornamento degli stessi avviene tramite il *client* a riga di comando messo a disposizione, chiamato anch'esso *npm*.

4.1.4 Bootstrap

Bootstrap è una raccolta di strumenti [Open-source](#) per lo sviluppo di siti e applicazioni web. Essa contiene moduli basati su *HTML* e *CSS* per la creazione dei componenti facenti parte di un'interfaccia (moduli, pulsanti e navigazione), insieme all'integrazione con alcune componenti opzionali di [JavaScript](#).

Si tratta di un *toolkit* compatibile con tutti i browser moderni e facente ampio uso del *responsive web design*, è stato integrato nel progetto allo scopo di applicare rapidamente uno stile comune agli elementi facenti parte dell'applicazione.

4.1.5 HTML e CSS

L'utilizzo di *HTML* e *CSS* si è reso necessario una volta scelto *Angular* come [Framework](#) di riferimento; essi sono linguaggi consolidati che permettono la definizione della struttura delle pagine web (nel caso di *Angular* essi permettono di definire la struttura dei singoli *component*).

4.1.6 Java

Java è il linguaggio di programmazione utilizzato da [Spring](#), il [Framework](#) di riferimento per lo sviluppo del [Back-end](#) dell'applicativo *SyncRec*.

4.1.7 Spring



Figura 4.2: Logo di Spring

Spring è un **Framework** scritto in Java utilizzato per lo sviluppo di applicazioni *enterprise*; esso è suddiviso in moduli che rispondono a vari tipi di esigenze, in modo da poter includere nella propria applicazione solo le funzionalità desiderate.

Spring è fortemente basato sul principio **Inversion of Control**, che delega la maggior parte del controllo sul flusso di esecuzione al *framework* stesso, in modo da ridurre il più possibile l'introduzione di errori.

Spring Core

Detto anche *core container*, esso rappresenta la parte principale di Spring, e tutto il framework è costruito sopra di esso. Insieme al modulo Beans è responsabile della funzionalità di IoC (precedentemente descritta) e di **Dependency Injection**, introdotta tramite *getters*, *setters*, *factory methods* e costruttori. Ciò permette di concretizzare i due principi, delegando al *framework* il compito di iniettare le dipendenze del *container*.

Spring Boot

Costituisce il modulo necessario per l'esecuzione di applicazioni *Spring*, spesso esso non necessita di particolari configurazioni, e risulta molto facile includere altre librerie di terze parti e moduli *Spring* aggiuntivi.

Spring Data

Costituisce il modulo di riferimento per la lettura e scrittura dei dati in applicazioni *Spring*, vi sono varie diramazioni a seconda della tecnologia utilizzata (ad esempio JDBC, MongoDB, Apache); uno dei moduli più importanti approfonditi durante il periodo di stage risulta essere *Spring Data Rest*, che permette la creazione di un'applicazione a microservizi molto facilmente, tramite le tipiche *annotations* di *Spring* (definite tramite l'operatore).

4.1.8 MongoDB

MongoDB è un **DBMS** non relazionale orientato ai documenti scritti in *JSON*.

Si tratta di un software estremamente diffuso e fortemente scalabile, cosa che ha permesso il suo utilizzo anche in ambienti enterprise con milioni di transazioni giornaliere. Come affermato in precedenza, esso è stato utilizzato come DBMS di riferimento all'interno del progetto di stage *SyncRec*, con risultati estremamente positivi.

Per ulteriori informazioni sulle tecnologie apprese nel corso del periodo di stage v. sezione [A](#)

4.2 Ambiente di Sviluppo

La sezione seguente illustra gli strumenti facenti parte dell'ambiente di sviluppo e utilizzati nel corso del progetto.

4.2.1 Sistema operativo

I sistemi operativi adottati per il completamento del progetto sono due:

- Ubuntu KDE 2019
- Ubuntu 18.04.2 LTS

4.2.2 IDE

L'adozione di un corretto ambiente di sviluppo integrato (*Integrated development environment*, spesso abbreviato con **IDE**), è di primaria importanza per il corretto sviluppo di un *software*, la scelta è spesso soggettiva e determinata dai gusti personali del programmatore. In questo progetto, viene adottato *Visual Studio Code*, una **IDE Open-source** sviluppata da Microsoft.

Si tratta di un *software* molto leggero e adattabile alle proprie esigenze tramite l'aggiunta di estensioni; queste due qualità rendono *VSCode* il software ideale per lo sviluppo di un applicativo in **Angular**. Alcuni dei principali vantaggi di *VSCode* sono

- Piena integrazione con **Git**;
- Ampia gamma di estensioni;
- Introduzione di **IntelliSense**, software per l'autocompletamento del codice
- Personalizzabile.

Secondo un sondaggio effettuato da *StackOverflow* nel 2019, *VSCode* risulta l'ambiente di sviluppo integrato più utilizzato dai programmatori, con il 50.7% di scelta su 87.317 partecipanti.

Estensioni

- **Angular 1Snippets 8:** Permette l'aggiunta di **Snippets** per **Angular** e **HTML**;
- **angular2-inline:** Per l'autocompletamento e l'evidenziazione dei *template inline* che possono essere utilizzati nei *component*;
- **Debugger for Chrome:** Questa estensione, sviluppata anch'essa da Microsoft, permette di utilizzare un **Debugger** per **JavaScript/TypeScript**;
- **Material Icon Theme:** Permette di visualizzare immediatamente l'estensione di un *file* tramite l'utilizzo di un'icona a fianco del nome;
- **npm:** Permette l'esecuzione rapida di script tramite **npm**;
- **TSLint:** Fornisce supporto per l'utilizzo di *TSLint* all'interno di *Visual Studio Code* (v. sezione 4.2.5) ;

4.2.3 Versionamento

Lo strumento di versionamento utilizzato è [Git](#) e il codice viene memorizzato in una [Repository](#) remota presso *GitLab*. La repository è privata, e l'accesso è stato consentito dal tutor aziendale dr Fabio Pallaro.

La necessità di lavorare in gruppo, inoltre, richiede un processo di [Continuous Integration](#) rapido ed efficiente: a tal scopo, viene integrata l'estensione *git-flow* al normale strumento di versionamento. Essa permette la creazione immediata di [Branch](#) di *feature* separate, il successivo [Merge](#) è altresì possibile mediante l'utilizzo di un singolo comando.

La necessità di venire incontro a casistiche particolari che possono incorrere nella gestione della *repository* ha richiesto la scelta di una [IDE](#) che possa svolgere tale compito. Il *software* adottato per questo progetto è *GitKraken*, un applicativo [Open-source](#) che fornisce supporto alla gestione di repository [Git](#), insieme all'integrazione dell'estensione *git-flow*.

4.2.4 Back-end

I microservizi sono installati in una macchina di proprietà di SyncLab, e sono accessibili tramite un [URL](#) dalle persone autorizzate.

L'integrazione, quindi, è avvenuta presso l'azienda, in quanto non è possibile accedere al [Back-end](#) da dispositivi esterni.

4.2.5 Analisi statica del codice

Viene utilizzato TSLint come strumento di analisi statica del codice, allo scopo di aumentare la leggibilità, manutenibilità e prevenzione di errori funzionali all'interno del codice, la sua configurazione avviene tramite un *file* `tslint.json` e gli errori sono immediatamente visibili al momento della scrittura.

4.2.6 Altri strumenti utilizzati

Per effettuare richieste al [Back-end](#), viene utilizzato *Postman*, che permette di testare il corretto funzionamento di un microservizio tramite le classiche chiamate *GET*, *PUT*, *POST*, *DELETE*; in questo modo si può verificare prima della scrittura del codice che il comportamento del [Back-end](#) sia conforme rispetto ai risultati attesi.

Capitolo 5

Progettazione e codifica

La seguente sezione ha lo scopo di illustrare la soluzione ideata dal laureando per il soddisfacimento dei requisiti concordati con il tutor aziendale descritti nel capitolo 3. In ciascuna sezione viene presentata una maschera dell'applicativo sviluppata, con il dovuto corredo di immagini e [Snippets](#) di codice.

5.1 L'applicativo SyncRec: Struttura e sviluppo del progetto

Come già accennato in precedenza, [Angular](#) suggerisce di applicare una forte struttura modulare alle proprie [Web Application](#), tale approccio è stato nella gran parte perseguito, con alcune eccezioni per determinati elementi condivisi fra più *component*; un'applicazione di tale struttura, inoltre, permette lo sviluppo delle proprie componenti senza dover necessariamente conoscere l'intera architettura nel dettaglio.

La figura 5.1 illustra la struttura dei *component* sviluppati per l'applicativo SyncRec, a scopo di sintesi sono state omesse alcune parti proprie del [Framework](#) di riferimento (come i moduli e gli *asset*) o sviluppate da altri studenti nel corso del progetto.

I moduli *applicant-list*, *applicant-detail* e *viewskillmatrix* rappresentano i componenti grafici dell'applicazione, il modulo *shared* rappresenta l'insieme di *component* condivisi nel *namespace* globale e il modulo *core* contiene i *service*; quest'ultimi hanno il duplice scopo di recuperare i dati dal [Back-end](#) e di fornire alcuni metodi di utilità necessari a svolgere determinate operazioni (come l'ordinamento o l'emissione di eventi verso *components* padri).

La figura 5.2 illustra come il modulo *service* si occupi di dialogare con il [Back-end](#) scritto in [Spring](#).

In ambito di progettazione, dialogando con gli altri laureandi che hanno preso parte alla realizzazione di SyncRec, si sono definite alcune linee guida da perseguire nel corso della codifica.

Tali linee guida sono:

- Utilizzare il più possibile le direttive [Angular](#) (come [ng-Model](#)) all'interno del *HTML* dei *component*.
- Definire per ciascun *component* il relativo modulo di riferimento, tale prassi è fortemente consigliata dalla documentazione di [Angular](#).

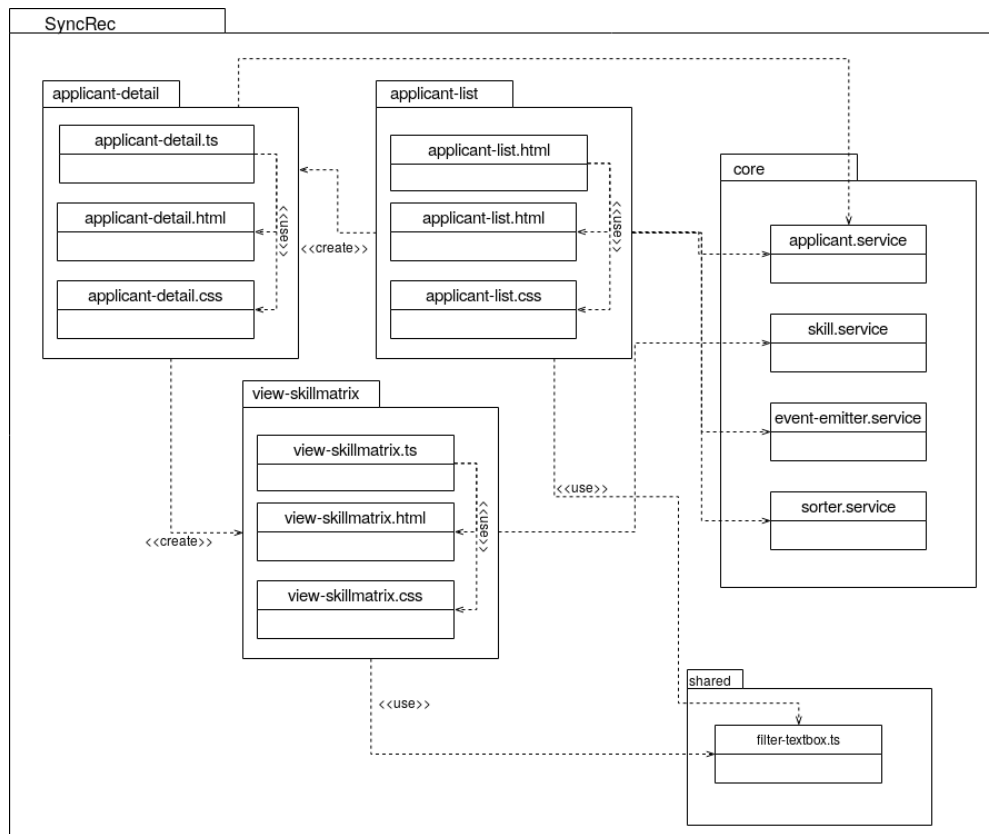


Figura 5.1: Struttura dei *component* sviluppati per l'applicativo SyncRec

- Per quanto riguarda il *CSS*, favorire l'utilizzo della versione 3.0 piuttosto che della libreria esterna *bootstrap*.

5.1.1 Homepage

La homepage dell'applicativo (v. figura 5.3) si presenta come un semplice menù con 3 voci.

- **Aggiungi persone** reindirizza verso il *component* adibito all'aggiunta degli *applicant*;
- **Visualizza persone** reindirizza verso il *component* adibito alla visualizzazione della lista degli *applicant*, sviluppato dal laureando nel corso del progetto di stage;
- **Ricerca Persone** reindirizza verso il *component* adibito alla ricerca delle persone;

In aggiunta, è possibile effettuare il *logout* con un pulsante posto in alto, vicino al logo.

5.1. L'APPLICATIVO SYNCREC: STRUTTURA E SVILUPPO DEL PROGETTO47

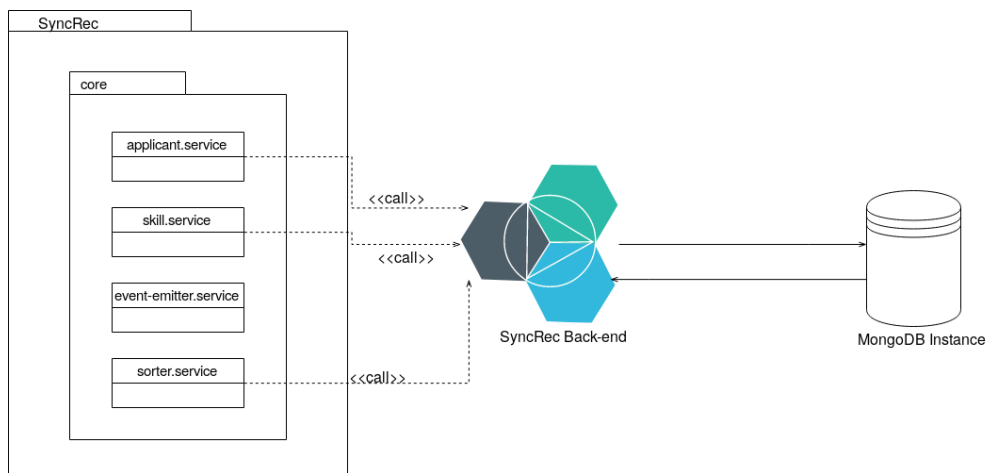


Figura 5.2: Integrazione tra Front-end e Back-end



Figura 5.3: Schermata della homepage

5.1.2 Maschera della visualizzazione degli Applicant

La figura 5.4, mostra il *component* sviluppato per la visualizzazione degli *applicant*, per ciascun *applicant* viene visualizzato il Cognome, il Nome e l'email, insieme a un tasto per l'eliminazione e uno per la visualizzazione del dettaglio della persona, il quale reindirizza verso la maschera descritta nella sezione 5.1.3.

Per realizzare questa *task*, è stato necessario reperire le informazioni tramite un *service* definito appositamente, il quale effettua una semplice *GET* al relativo microservizio sviluppato nel *Back-end* (v. codice 5.1). Successivamente, i dati vengono memorizzati in un apposito *array* e visualizzati tramite una direttiva *Angular* chiamata *ng-for* (v. listing 5.2).

Listing 5.1: Funzione del service che effettua la chiamata GET

SyncRec

Lista delle persone richiedenti

[Torna alla Homepage](#)

Filter:

Applica filtri aggiuntivi

Cognome	Nome	Email	Elimina	Visualizza
Marchiori	Matteo	matt.marchiori97@gmail.com	Elimina	
Motricala	Victor	f.pallaro@synclab.it	Elimina	
Rossi	Giorgio	g.rossi@gmail.com	Elimina	
Brocca	Ludovico	lbocca29@gmail.com	Elimina	

Numero di persone richiedenti: 4

Figura 5.4: Schermata della visualizzazione lista degli *applicant*

```
getAllApplicants(): Observable<Applicant[]> {
  return this.http.get<Applicant[]>(this.baseUrl).pipe(
    catchError(this.handleError)
  );
}
```

Listing 5.2: Visualizzazione degli *applicant* nel codice HTML

```
<tr class="hover-tr" *ngFor="let appl of filteredApplicants;" >
  <td>
    {{ appl.surname }}
  </td>
  <td>
    {{ appl.name }}
  </td>
  <td>
    {{ appl.email }}
  </td>
  <td>
    <button class="btn" id="btn-delete-applicant-list"
      (click)="deleteApplicant(appl.id)">Elimina</button>
  </td>
  <td>
    <span class="immagineEsterna" (click)="routerRedirect(
      appl.id)"><i class="fa fa-search fa-2x"></i></span>
  </td>
</tr>
```

Il pulsante posto in alto a destra visualizza il *component* adibito all'applicazione di filtri aggiuntivi sul totale degli *applicant*, come si può vedere in figura 5.5. Posti sopra la tabella, inoltre, è possibile visualizzare un campo di testo, che permette di filtrare i risultati secondo nome, cognome o email. Tale filtro dinamico è stato ottenuto tramite l'azione congiunta di un *service* (v *listing. 5.3*) e di un *component* costituito da un singolo campo di testo; il *component* padre in questo modo può

applicare il filtro sugli *applicant* ogni qualvolta il contenuto del campo di testo venga cambiato (v *listing 5.4*), in modo simile a quanto accade con l'[Observer Pattern](#).

Listing 5.3: Event-Emitter service

```
export class EventEmitterService {

    invokeFirstComponentFunction = new EventEmitter();
    // variabile per la sottoscrizione di un component per la
    // reazione ad eventi lanciati
    subsVar: Subscription;

    constructor() { }

    onComponentAction() {
        this.invokeFirstComponentFunction.emit();
    }
}
```

Listing 5.4: Funzione che filtra i dati degli applicant

```
filter(data: string) {
    if (data) {
        this.filteredApplicants = this._applicants.filter
            ((applicant: Applicant) => {
                return applicant.name.toLowerCase().
                    indexOf(data.toLowerCase()) > -1 ||
                    applicant.surname.toLowerCase().indexOf(
                        data.toLowerCase()) > -1 ||
                    applicant.email.toLowerCase().indexOf(
                        data.toLowerCase()) > -1;
            });
    } else {
        this.filteredApplicants = this._applicants;
    }
}
```

La visualizzazione del *component* relativo all'aggiunta di filtri aggiuntivi viene gestita tramite una semplice direttiva [Angular](#) chiamata *Ng-if*, la quale si lega a una variabile booleana e determina la visualizzazione o meno del *component* a seconda del valore assegnato. La selezione dei valori, invece, viene gestita tramite i *form* [Angular](#), i quali legano i valori contenuti nei vari *widget* a delle strutture che gestiscono autonomamente valori di default, validazione, *submit* dei risultati e quant'altro, rendendo la struttura molto solida, sicura e poco soggetta a errori. Nel *listing 5.5* è possibile vedere l'inizializzazione del *FormBuilder*, la sopracitata struttura in grado di gestire i dati di un *form*, notare che viene inserito un pattern per la validazione ove necessario.

Listing 5.5: Inizializzazione di un FormBuilder

```

this.filterGroup = this.formBuilder.group({
  sesso: ['',],
  name: ['', Validators.pattern('^[a-zA-Z]*$')],
  surname: ['', Validators.pattern('^[a-zA-Z]*$')],
  qualification: ['',],
  seniority: ['',],
  ambito: ['',],
  geodisp: this.formBuilder.array([]),
  skill: ['',],
  skillLevel: ['',],
  scartato: [false],
});

```

SyncRec

Lista delle persone richiedenti

Torna alla Homepage

♂

Genere

Seleziona il genere

👤

Nome

Inserisci nome

👤

Cognome

Inserisci cognome

☰

Skill Posseduta

Scegli la competenza

1
2
3

Livello Skill

Scegli il livello di skill acquisito

⚙️

Titolo di studio

Seleziona titolo di studio

👜

Ambito lavorativo

Seleziona ambito lavorativo

🏠

Anzianità

Seleziona anzianità

🌐

Disponibilità geografica

Napoli

Milano

Roma

Padova

Verona

Bologna

Applica filtri

Annulla

Figura 5.5: Schermata della selezione dei filtri da applicare alla lista degli *applicant*















5.1.3 Maschera delle operazioni CRUD su un Applicant

La gestione delle operazioni **CRUD** di un *applicant* avviene tramite la maschera visibile nella figura 5.6; l'approccio adottato è molto simile a quanto avviene con la gestione della maschera per la selezione dei filtri aggiuntivi descritta nella sezione 5.1.2. Viene

5.1. L'APPLICATIVO SYNCREC: STRUTTURA E SVILUPPO DEL PROGETTO51

SyncRec

Ludovico Brocca
[Torna alla Homepage](#)

 Nome	Ludovico	 Cognome	Brocca
 Genere	Maschio	 Indirizzo	via monteverdi 27
 Data di nascita	1996-11-04	 Città	Abano Terme
 Email	lbocca29@gmail.com	 Ambito	Sviluppatore
 Anzianità	Junior	 Titolo di studio	Diploma
 Disponibilità geografica	Napoli Padova Verona	 Preavviso	3 mesi
 Telefono	366698855	 Curriculum	Download
Note			
Status Lavorativo			
Colloqui			
Considerazioni			

[←](#)[Elimina](#)[Visualizza SkillMatrix](#)

Figura 5.6: Maschera **CRUD** del singolo *applicant*

utilizzato un *FormBuilder* per la gestione dei singoli valori di un *applicant*, una volta modificati, un controllo su un parametro dei *FormControl*, chiamato **touched**, permette la visualizzazione dei tasti di conferma o annullamento delle modifiche, in caso di conferma viene effettuata una chiamata *PUT* tramite l'apposito servizio (descritta nel *listing 5.6*), in caso contrario vengono ripristinati i valori antecedenti alle modifiche apportate dall'utente.

Listing 5.6: chiamata PUT al microservizio del Back-end di SyncRec

```

putApplicant(applicant: Applicant): Observable<any> {
    return this.http.patch(this.baseUrl + '/' + applicant.id,
        applicant).pipe(
            catchError((err: HttpErrorResponse) => {
                if (err.error instanceof Error) {
                    // A client-side or network error
                    occurred.
                    const details = {detail: err.
                        error, status: err.status};
                    return throwError(details);
                } else {
                    // The backend returned an
                    unsuccessful response code.
                    const details = {detail: err.
                        error, status: err.status};
                    return throwError(details);
                }
            })
        );
}

```

Con i pulsanti posti in basso è possibile tornare indietro, eliminare un *applicant* (tramite una chiamata DELETE al microservizio di [Back-end](#)), o visualizzare l'insieme di competenze possedute tramite l'apposita maschera descritta nella sezione [5.1.4](#).

5.1.4 Maschera di visualizzazione di uno skillmatrix

Uno *skillmatrix* rappresenta l'insieme delle competenze possedute da un *applicant* insieme al relativo livello, che può variare da 0 a 5; una competenza per essere considerata posseduta deve avere un livello ≥ 5 .

È importante evidenziare come all'interno di SyncLab le competenze prese in considerazione siano molto numerose e tra le più disparate: nel contesto del progetto esse sono suddivise in categorie e il loro numero complessivo supera 130; ciò ha richiesto la formulazione di una soluzione che rendesse immediatamente visibile il distacco fra *skill* possedute o meno e che al contempo permettesse di modificare i valori in modo rapido. Come si può vedere nella figura [5.7](#), l'approccio adottato prevede la predisposizione di 2 tabelle, una immediatamente visibile contenente l'insieme di *skill* possedute per ciascuna categoria e una visibile tramite la pressione di un tasto, contenente il totale delle *skill* considerate.

La visualizzazione delle skill è legata alla funzione [5.7](#), che restituisce un booleano a una direttiva *Ng-If* all'interno del codice *HTML*; la quale determina la visualizzazione o meno del codice afferente al *tag* legato alla direttiva, come si può vedere nel *listing* [5.8](#).

5.1. L'APPLICATIVO SYNCREC: STRUTTURA E SVILUPPO DEL PROGETTO53

SyncRec

Competenze Possedute

[Torna al dettaglio della persona](#)

Filter:

Processi e metodologie di Ing. del Software	Livello	Modifica
AGILE	3	<button>Modifica</button>

Tool di sviluppo	Livello	Modifica
Eclipse	1	<button>Modifica</button>

↑

Lista totale delle competenze

Filter:

Nome	Livello	Modifica
Objective-C	0	<button>Modifica</button>
Eclipse	1	<button>Modifica</button>
Mercury	0	<button>Modifica</button>
SQL	0	<button>Modifica</button>

Figura 5.7: Maschera CRUD dello skillmatrix

Listing 5.7: Funzione che determina la visualizzazione di una categoria avente *skills* possedute

```
hasThisCategorySkills(cat: string): boolean {
  let result = false;
  const category: Category = this.catalog.filter ((c:
    Category) => c.name === cat)[0];
  category.skills.forEach((categoryskill) => {
    this.skills.filter((sk: Skill) => {
      if (categoryskill.name === sk.name && sk.
        level > 0) {
        result = true;
      }
    });
  });
  return result;
}
```

Per facilitare la navigazione fra le due tabelle, viene aggiunto il *component* per il filtraggio dei dati, adottando lo stesso approccio della maschera di visualizzazione lista degli *applicant* (v. sez. 5.1.2). La modifica avviene tramite un menù a tendina visibile tramite la pressione di un tasto

"Modifica", la tabella di *skill* possedute si aggiorna in conseguenza alla conferma del nuovo dato selezionato (aggiungendo o togliendo righe a seconda del livello della *skill*). Allo scopo di rendere i dati consistenti fra le due tabelle (in modo da evitare che i livelli risultino discordanti), viene utilizzato un singolo *FormBuilder* e una singola struttura dati che tenga traccia dei valori per il *submit* delle modifiche.

Listing 5.8: HTML e direttiva Ng-If

```
<ng-container *ngIf="hasThisCategorySkills(catalog[0].name)">
<!-- direttiva che lega al FormBuilder del component -->
<form [formGroup]="skillGroup">
<!-- visualizzazione HTML di una tabella -->
</form>
</ng-container>
```

Appendice A

Altre tecnologie

Lo scopo di questa sezione è fornire una panoramica delle tecnologie studiate durante il periodo di stage e previste nel piano di lavoro e che non hanno trovato un'applicazione concreta nel contesto del progetto SyncRec.

Tali tecnologie sono state incluse con il duplice scopo di ampliare il bagaglio di conoscenze del laureando e comprendere meglio l'architettura del progetto sul quale andavano fatti gli interventi concordati.

A.1 Oracle PL/SQL

Oracle PL/SQL è un linguaggio sviluppato da Oracle negli anni '80, si tratta di un linguaggio procedurale in grado di eseguire *query SQL* insieme ad alcune strutture molto simili a quanto si può trovare in altri linguaggi procedurali (come [Python](#) o [ADA](#)). PL/SQL fornisce le seguenti strutture:

- Gestione degli errori;
- Tipizzazione dei dati;
- Le classiche strutture della programmazione (cicli, blocchi condizionali, funzioni etc.);
- Procedure;
- PSP (PL/SQL Server Pages);
- Piena integrazione con SQL (direttive DDL e DML sono disponibili in modo statico e dinamico)

A.1.1 Struttura di uno script Oracle PL/SQL

Uno *script Oracle PL/SQL* è sempre suddiviso in al più tre parti:

- **DECLARE:** si tratta di una sezione opzionale che contiene la dichiarazione di variabili, cursori e quant'altro;
- **BEGIN/END:** sezione obbligatoria che esegue le operazioni PL/SQL richieste;
- **EXCEPTION:** sezione facoltativa che si occupa della gestione degli errori all'interno dello *script*.

All'interno del blocco *DECLARE* è possibile definire inoltre due tipi di strutture: *procedure* e *funzioni*, dove l'unica differenza è che le seconde possono fornire valori in output tramite la keyword *RETURN*.

Ogni script viene poi eseguito tramite la keyword *Execute*.

Cursori

Per processare dichiarazioni SQL Oracle fornisce un'area di memoria conosciuta come *context area*, i *cursori* costituiscono dei puntatori a tale area di memoria.

Il set di tuple risultanti da un'operazione SQL e puntato da un cursore è definito come l' *Active Set*.

Per ogni operazione eseguita, Oracle istanzia un cursore implicito, ed è inoltre possibile definire dei cursori *custom* che processano le tuple una alla volta, applicando determinate trasformazioni definite dal programmatore. I cursori vengono definiti nei blocchi PL/SQL e restringono la *context area* tramite delle classiche *SELECT* in SQL. I cursori, in ultima analisi, sono uno strumento molto potente per manipolare i risultati delle operazioni SQL, tuttavia oggi risultano piuttosto datati se si considerano le potenzialità dei moderni [Framework](#) e la loro capacità di processare i dati contenuti nei database.

A.2 Architetture a microservizi

Un'architettura basata su microservizi viene applicata nella realizzazione di un'applicazione, essa è costituita da componenti indipendenti che eseguono ciascun processo applicativo come un **servizio**.

Tali servizi comunicano attraverso un'interfaccia ben definita che utilizza API leggere. Ciò risulta vantaggioso poiché ciascun componente viene eseguito in modo indipendente, e ciascun servizio può essere aggiornato, distribuito e ridimensionato per rispondere alla richiesta di funzioni specifiche di un'applicazione.

Ciascun servizio è progettato per una serie di capacità e si concentra sulla risoluzione di un problema specifico. Se, nel tempo, gli sviluppatori aggiungono del codice aggiuntivo a un servizio rendendolo più complesso, il servizio può essere scomposto in servizi più piccoli.

Tra le principali qualità di un'architettura a microservizi troviamo:

- **Agilità:**I microservizi promuovono le organizzazioni di team indipendenti di dimensioni ridotte che diventano proprietari del servizio che gestiscono. Ciò comporta tempi di sviluppo minori in contesti ridotti e ben delineati.
- **Scalabilità e flessibilità**I microservizi consentono di scalare ciascun servizio in modo indipendente per rispondere alla richiesta delle funzionalità che l'applicazione supporta. Ciò permette ai team di adattare in modo corretto l'infrastruttura rispetto alle necessità, con la possibilità di monitorare accuratamente i servizi nel caso in cui essi rilevino un aumento del flusso di transazioni.
- **Semplicità di distribuzione**I microservizi supportano l'integrazione continua e la distribuzione continua; l'apporto di modifiche errate non inficia sul corretto funzionamento generale dell'applicazione e gli errori influiscono di meno sul costo delle operazioni.
- **Libertà tecnologica** L'utilizzo di un'architettura a microservizi non è legata a un particolare *stack* tecnologico: i team di sviluppo hanno piena libertà nella

scelta e adozione delle tecnologie da utilizzare nel corso dello sviluppo di un progetto.

- **Codice riutilizzabile** La suddivisione in moduli ben delineati ha il grande vantaggio di rendere il codice riutilizzabile; è possibile quindi utilizzare moduli e servizi pienamente funzionanti per lo sviluppo di componenti aggiuntivi.
- **Resilienza** A differenza di quanto accade in un'architettura monolitica, un errore non è in grado in alcun modo di pregiudicare il corretto funzionamento di un'applicazione; ciò è dovuto alla natura indipendente dei servizi per come sono concepiti.

A.3 JavaServer Pages (JSP)

Glossario

ADA Sviluppato verso la fine degli anni settanta, è un linguaggio *general purpose* che si presta all'utilizzo in qualsiasi dominio applicativo. [55](#)

Angular CLI Sigla per *Angular Command Line Interface*, si tratta di un applicativo a riga di comando compreso nella *suite Angular* in grado di facilitare lo sviluppo di applicazioni *web*. [37](#), [40](#)

Back-end Componente aggregata che permette l'effettivo funzionamento delle interazioni tra l'utente e l'interfaccia grafica (o **Front-end**). [5](#), [6](#), [36](#), [40](#), [43](#), [52](#)

Big Data Analysis Processo di raccolta e analisi di grandi volumi di dati (*big data*) allo scopo di ricavarne informazioni utili.. [4](#)

Branch Puntatore a un determinato stato di avanzamento del *software* all'interno di una *repository*, tipicamente vengono utilizzati per suddividere logicamente gli incrementi necessari durante il ciclo di vita di un *software*. [43](#)

Cloud Computing Indica un paradigma di erogazione di servizi offerti *on demand* da un fornitore ad un cliente finale attraverso la rete Internet. [4](#)

Continuos Integration Pratica che si applica in contesti in cui lo sviluppo del *software* avviene attraverso un sistema di versionamento. Consiste nell'allineamento frequente (ovvero "molte volte al giorno") dagli ambienti di lavoro degli sviluppatori verso l'ambiente condiviso. [43](#)

CRUD Sigla che indica le classiche operazioni di manipolazioni di dati (*Create, Read Update e Delete*). [v](#), [6](#), [8](#), [50](#), [51](#)

DBMS Sigla per *Database Management System* sistema *software* progettato per consentire la creazione, la manipolazione e l'interrogazione efficiente di database. [5](#), [41](#)

Debugger É un programma/software specificatamente progettato per l'analisi e l'eliminazione dei bug, ovvero errori di programmazione interni al codice di altri programmi. [42](#)

Dependency Injection Design pattern della Programmazione orientata agli oggetti il cui scopo è quello di semplificare lo sviluppo e migliorare la testabilità di software di grandi dimensioni. [39](#), [41](#)

Framework Un'architettura logica di supporto su cui un software può essere progettato e realizzato. [5](#), [7](#), [36](#), [40](#), [41](#), [45](#), [56](#)

Git È un *software* di controllo versione distribuito utilizzabile da interfaccia a riga di comando. [42](#), [43](#)

IDE *Software* che, in fase di programmazione, supporta i programmatori nello sviluppo del codice sorgente di un programma. [36](#), [42](#), [43](#)

immutable.js description. [38](#)

Internet delle Cose Si riferisce all'estensione della rete internet verso oggetti e luoghi concreti. [4](#)

Inversion of Control Pattern che prevede che le componenti di libreria esterne posseggano il controllo del flusso di esecuzione del programma, e che possano cedere tale controllo a componenti applicativi a loro discrezione. [41](#)

ISO Sigla per *International Organization for Standardization*, si tratta della più importante organizzazione mondiale per la definizione di norme tecniche. [3](#)

Java description. [5](#)

JavaScript Linguaggio di scripting orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione *web* lato *client*. [37](#), [38](#), [40](#), [42](#)

Merge Eliminazione di un *branch* e conseguente allineamento degli incrementi sviluppati in un secondo *branch*. [43](#)

ng-Model L'insieme di direttive *Angular* che lega vari *tag HTML* a variabili utili nella gestione di interazioni con l'utente durante l'utilizzo dell'applicazione. [45](#)

Node.js *Runtime* di JavaScript Open-source multiplatforma orientato agli eventi per l'esecuzione di codice JavaScript, costruita sul motore JavaScript V8 di *Google Chrome*. [37](#)

NodeJs description. [40](#)

npm Sigla per *Node Package Manager*, è un gestore di pacchetti per il linguaggio di programmazione JavaScript. [42](#)

Observer Pattern Pattern che sostanzialmente permette a componenti applicativi di "osservare" l'esecuzione di un altro componente e agire in conseguenza dello scatenarsi di un evento, è la base del *reactive programming*. [49](#)

Open-source Un software open source è reso tale per mezzo di una licenza attraverso cui i detentori dei diritti ne favoriscono la modifica, lo studio, l'utilizzo e la redistribuzione. Caratteristica principale dunque delle licenze open source è la pubblicazione del codice sorgente. [40](#), [42](#)

Python Python è un linguaggio di programmazione ad alto livello, orientato agli oggetti, adatto, tra gli altri usi, a sviluppare applicazioni distribuite, scripting, computazione numerica e *system testing*. [55](#)

Repository Un ambiente di sistema informatico condiviso, utilizzato per mantenere allineato l'avanzamento dell'applicativo nel corso del suo ciclo di vita. [43](#)

RxJs Libreria per il *reactive programming* fortemente basata sull'*Observer Pattern*. [38](#)

Snippets Frammento o esempio di codice sorgente. [42](#)

System Integrator Persona o azienda specializzata nell'integrazione fra sottosistemi, assicurando il loro corretto funzionamento. [3](#), [61](#)

URL Sigla per *Uniform Resource Locator*: si tratta di una sequenza di caratteri che identifica univocamente l'indirizzo di una risorsa su una rete di computer. [40](#)

Web Application Indica un'applicazione *web-based* accessibile tramite *browser*. [5](#), [39](#)

Web Server Applicazione software che, in esecuzione su un *server*, è in grado di gestire le richieste di trasferimento di pagine *web* di un *client*, tipicamente un *web browser*. [38](#)