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

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 4 capitoli:

- Nel primo viene presentata l'azienda e le metodologie di lavoro in uso presso di essa;
- Nel secondo viene presentato il progetto e le tecnologie apprese e/o utilizzate per il suo completamento;
- Il terzo prosegue analizzando nel dettaglio gli obiettivi, i requisiti e i casi d'uso dell'applicativo SyncRec;
- Il quarto presenta una relazione degli obiettivi raggiunti, insieme al loro grado di soddisfacimento, insieme 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 Serivzi 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 comu- nicazione (*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'utlizzo 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 indispensabili 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.

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 oblbigatori, 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 unna 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.

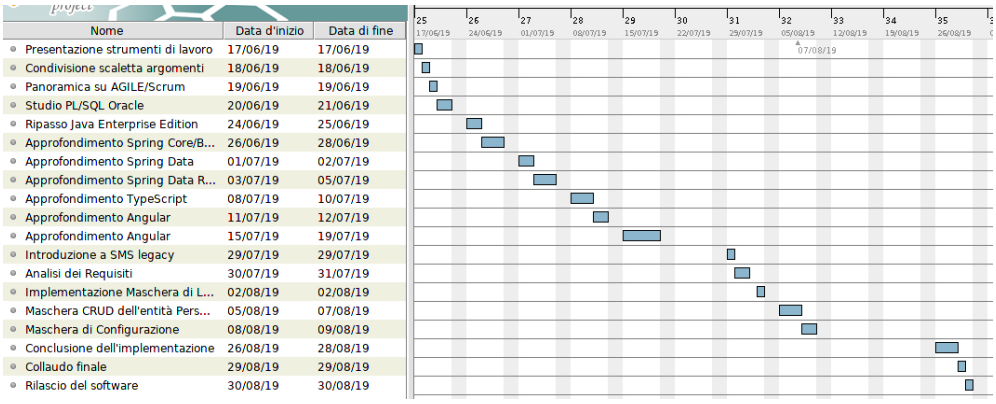

Figura 2.1: Diagramma di Gantt per la pianficazione dello stage

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;
- **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 vista degli Applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** L'utente si trova nella homepage dell'applicazione.
- **Scenario Principale:**
  1. L'utente clicca sul pulsante "Visualizza lista delle persone"
- **PostCondizione:** l'utente accede alla vista di visualizzazione lista degli applicant.

3.2.2 UC-2: 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.

3.2.3 UC-3 Aggiunta filtro su nome/cognome su 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

3.2.4 UC-4 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.

3.2.5 UC-4.1 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.

3.2.6 UC-5 visualizzazione vista dei filtri

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di visualizzazione lista degli applicant.
- **Scenario Principale:**
  1. l'utente clicca sul tasto "Applica filtri aggiuntivi"
- **PostCondizione:** l'utente visualizza la vista relativa ai filtri

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'utentesi trova nella vista relativa ai filtri e visualizza l'input immesso sullo schermo.

3.2.8 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.

3.2.9 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.

3.2.10 UC7.3- 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.

3.2.11 UC7.4- 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.

3.2.12 UC7.5- 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.

3.2.13 UC7.6- 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.

3.2.14 UC7.7- 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.

3.2.15 UC7.8- 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.

3.2.16 UC7.9- 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.

3.2.17 UC7.10- 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.

3.2.18 UC-8 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-8.1);

3.2.19 UC-8.1 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.

3.2.20 UC-9 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 flitri precedentemente selezionati
- **PostCondizione:** l'utente si trova nella vista relativa ai filtri.

UC-10 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 tabella 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-11 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 tabella 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.

3.2.21 UC-12 Visualizzazione vista di 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 riportante una lente di ingrandimento relativa 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;

3.2.22 UC-13 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;
  2. l'utente visualizza il cognome dell'applicant;
  3. l'utente visualizza il genere dell'applicant;
  4. l'utente visualizza la data di nascita dell'applicant;
  5. l'utente visualizza l'email dell'applicant;
  6. l'utente visualizza il numero di telefono dell'applicant;
  7. l'utente visualizza l'indirizzo dell'applicant;
  8. l'utente visualizza la città residenza dell'applicant;
  9. l'utente visualizza il preavviso richiesto dall'applicant per una chiamata;
  10. l'utente visualizza il titolo di studio dell'applicant;
  11. l'utente visualizza il livello di anzianità dell'applicant;

12. l'utente visualizza l'ambito di lavoro dell'applicant;
13. l'utente visualizza se lo status dell'applicant è scartato o meno;
14. l'utente visualizza le citta per cui l'applicant ha fornito disponibilità per il trasferimento;
15. l'utente visualizza delle note generiche relative all'applicant;
16. l'utente visualizza delle note relative al colloquio dell'applicant;
17. l'utente visualizza delle note riguardanti lo status lavorativo dell'applicant;
18. l'utente visualizza altre considerazioni aggiuntive relative all'applicant;

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

3.2.23 UC-14 Modifica di un dato di un applicant

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

  1. l'utente clicca su un pulsante posto a fianco di un campo dati di un applicant;
  2. l'utente visualizza il widget di modifica per il campo dati relativo;
  3. l'utente inserisce le modifiche desiderate;
  4. l'utente clicca sul relativo pulsante precedentemente premuto;

- **PostCondizione:**

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

3.2.24 UC-15 Salvataggio delle modifiche per un applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un applicant e visualizza il pulsante di salvataggio dopo aver modificato un campo.
- **Scenario Principale:**

  1. l'utente clicca sul pulsante di salvataggio delle modifiche

- **PostCondizione:** le modifiche vengono salvate e viene visualizzato un messaggio di successo.
- **Estensioni:** viene visualizzato un messaggio di errore causato da un inserimento errato nei campi modificati (UC 15.1);

3.2.25 UC-15.1 Visualizzazione messaggio di errore inserimento campo di un applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un applicant e ha inserito un campo di un applicant non rispettando i parametri imposti
- **Scenario Principale:**
  1. viene visualizzato un messaggio di errore in corrispondenza del campo dati inserito erroneamente
- **PostCondizione:** l'utente si trova nella vista di dettaglio di un applicant

3.2.26 UC-16 Annullamento delle modifiche precendentemente inserite

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un applicant e ha precedentemente confermato delle modifiche di alcuni dati relativi all'applicant
- **Scenario Principale:**
  1. l'utente clicca sul pulsante di annullamento delle modifiche
- **PostCondizione:** l'utente visualizza la vista di dettaglio dell'applicant nello stato antecedente alla modifica.

3.2.27 UC- 17 Ritorno alla lista degli applicant

- **Attori:** Utente riconosciuto
- **Precondizione:** l'utente si trova nella vista di dettaglio di un applicant
- **Scenario Principale:**
  1. l'utente clicca sul pulsante *back*
- **PostCondizione:** l'utente si trova nella vista di visualizzazione lista degli applicant.

3.2.28 UC-18 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.
- **Estensioni:** l'utente si trova nella vista relativa all'inserimento skillmatrix di un applicant.

Capitolo 4

Tecnologie e progettazione

IL seguente capitolo ha lo scopo di illustrare nel dettaglio le tecnologie e gli strumenti utilizzati nel corso del progetto di stage, insieme alla progettazione delle maschere richieste per il raggiungimento degli obiettivi.

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 tramtite 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 Agular

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 *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 immutbale.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 seguenti 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*, utlizzando 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 utlizzo 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.

Per ulteriori informazioni sulle tecnologie apprese nel corso del periodo di stage v. sezione A

4.1.5 HTML e CSS

L'utilizzo di HTML e CSS si è reso necessario una volta scelto *Agular* 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 sudddiviso 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 *Annotations*, *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

Spring Data

Spring Data Rest

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 ilcompletamento 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 giusti 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 sodaggio effettuato da *StckOverflow* 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 Snippets 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 4.2.6) ;

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 continuos 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 Testing

Viene adottato *Karma* come *Test runner* all'interno del progetto, come suggerito dalla documentazione di *Angular* I test vengono codificati in *file* con estensione `.spec.ts` (creati automaticamente da Angular CLI al momento dell'inserimento di un component) ed eseguiti tramite il comando `npm test`; la configurazione di *Karma*, invece, avviene tramite un *file* `karma.conf.js`.

4.2.5 Back-end

I microservizi sono installati in una macchina di proprietà di SyncLab, e sono accedibili 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.

4.2.6 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.

4.2.7 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 verficare prima della scritttura del codice che il comportamento del Back-end sia conforme rispetto ai risultati attesi.

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 instanzia un 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.

Glossario

ADA description. 16

Angular description. 5

Angular CLI description. 13

angular CLI description. 15

Back-end description. 12

branch description. 12

continuos integration description. 12

CRUD description. 6

DBMS description. 5

debugger description. 11

dependency injection description. 14

ERP TODO. 3, 17

framework description. 5, 7

Git description. 12

ICT TODO. 3, 17

IDE description. 11

immutbale.js description. 13

Java description. 5

JavaScript description. 13

merge description. 12

microservizi description. 5

MongoDb description. 5

Node.js description. 13

NodeJs description. 15

npm description. 11

open-source description. 15

Python description. 16

repository description. 12

RxJs description. 13

snippets description. 11

Spring Framework Java fortemente basato sul principio di Inversion of Control, fornisce uno stack di librerie molto versatile per lo sviluppo di applicazioni Java. 1, 5, 18

System Integrator TODO. 3

TypeScript description. 13

URL description. 15

web application description. 5, 15

web server description. 13