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).

1.1.1 Principali aree applicative

SyncLab offre offre i propri servizi operando su diverse aree applicative di primaria importanza, ponendo l'accento sulla predisposizione di un ambiente di lavoro collaborativo ed efficace, allo scopo di fornire prodotti di qualità che rispecchino gli standard adottati.
L'azienda, inoltre, offre ai propri clienti la possibilità di usufruire dell'*expertise* accumulata nel corso del suo operato nel campo della ICT (Information and Comunications Technology), fornendo consulenza informatica *ad hoc*, sviluppo di sistemi distribuiti *enterprise* supportati dagli standard ISO più diffusi (ISO 9001, ISO 14001, ISO 27001) e fornitura di servizi comprendenti la *Business Consultancy*, *It Consultancy* e *Project Financing*. Uno degli obiettivi di primaria importanza per SyncLab è fornire ai propri clienti la possibilità di adottare il miglior software ERP (Enterprise Resource Planning) possibile, valutando accuratamente caso per caso le necessità e i bisogni, ponendosi come partner di supporto per lo sviluppo e la crescita dei clienti.

1.1.2 Prodotti e soluzioni

- **E-Health:** uno dei principali prodotti sviluppati da SyncLab è SynClinic, un sitema informativo sanitario che offre la possibilità di gestire tutti i processi

clinici e amministrativi di ospedali, cliniche e case di cura.

- **Privacy e Sicurezza:** nel corso del suo operato nell'ambito della privacy e sicurezza, l'azienda ha sviluppato diversi applicativi: i principali sono DPS 4.0 (per la gestione delle informazioni personali in seguito al GDPR per la protezione dei dati) e StreamLog, al fine di monitorare gli accessi degli utenti ai propri sistemi informativi.
- **Big Data e Mobile development:** uno dei prodotti di punta di SyncLab è StreamCrusher, un applicativo in grado di raccogliere e unificare i dati generati da un'azienda, individuando rapidamente criticità e vulnerabilità, aiutando le aziende nel processo di *decision making*.

1.1.3 Ricerca e Svilupppo

Laboratorio

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 Descrizione dello 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.

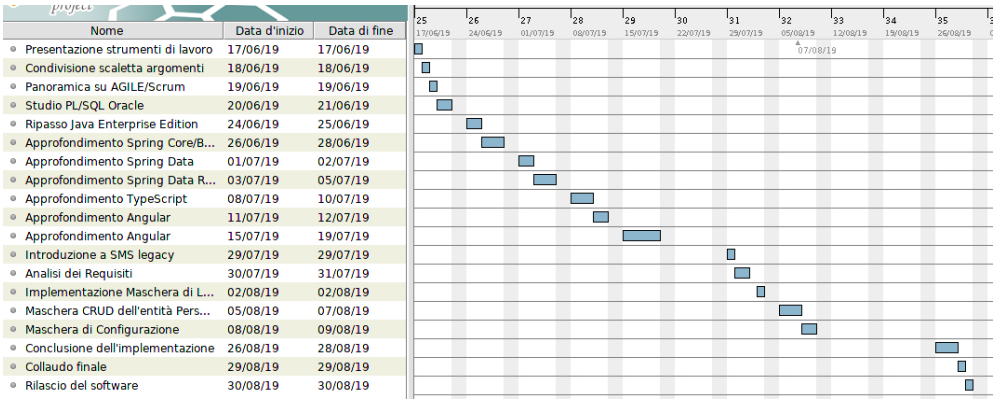

Figura 2.1: Diagramma di Gantt per la pianficazione dello stage

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

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 Ambiente di Sviluppo

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

4.1.1 Sistema operativo

I sistemi operativi adottati per ilcompletamento del progetto sono due:

- Ubuntu KDE 2019
- Ubuntu 18.04.2 LTS

4.1.2 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.

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 4.1.6) ;

4.1.3 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*.

4.1.4 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`.

4.1.5 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.

4.1.6 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.

4.1.7 Strumenti per l'analisi dei requisiti

TODO

4.1.8 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.

4.2 Tecnologie

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

4.2.1 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.

4.2.2 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 *AngularJS*, abbandonando l'utilizzo di JavaScript in favore di TypeScript (v. sezione 4.2.1), 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. 4.2.2);
- **Routing Modules:** moduli particolari che permettono di mappare richieste *URL* alla visualizzazione di *component* (v. sez. 4.2.2);
- **Components:** elemento base di una web application sviluppata in *Angular* (v. sezione 4.2.2);
- 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.

4.2.3 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*.

4.2.4 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.

4.3 Altre tecnologie

La seguente sezione ha lo scopo di fornire una visione generale ad alto livello delle tecnologie apprese ma non applicate all'interno del progetto.

4.3.1 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)

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, funzioni 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*.

4.3.2 MongoDb

4.3.3 Spring

Spring Data

Spring Data REST

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