Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica

Realizzazione di un microservizio con Spring per un'applicazione per la gestione di attività sportive

Tesi di laurea triennale

Relatore

Prof.Paolo Baldan

Laureando

Lorenzo Matterazzo

Anno Accademico 2020-2021

Realizzazione di un microservizio con Spring per un'applicazione per la gestione di attività sportive
Tesi di laurea triennale

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di 320 ore, dal laureando Lorenzo Matterazzo presso l'azienda Sync Lab S.r.l., situata a Padova. Lo *stage* ha avuto come argomento principale l'implementazione di nuove funzionalità nel contesto dell'applicazione ***SportWill***, una *web app* che dà modo all'utente di divulgare la sua intenzione (*will*[g]) di effettuare un'attività sportiva. La piattaforma fa vedere tutto a tutti, rendendo troppo caotica la fruizione, quindi l'esigenza era di dare la possibilità all'utente di creare uno o più gruppi a cui utenti "amici" possano unirsi, vedendo quindi solo le *will* di gruppo. Le attività svolte nel corso dello *stage* sono due:

* la prima è un insieme di attività che sono legate al *backend* della *web app*, come lo sviluppo di tre microservizi[g] mediante il framework[g] Spring[g] Java. In particolare:

  1. il primo microservizio consente l'effettuazione delle funzionalità Create Read Update Delete (CRUD) per la gestione dei gruppi e la visualizzazione delle *will* degli utenti appartenenti allo stesso gruppo;
  2. il secondo è l'implementazione di un API Gateway[g], che permette di esporre le Application Program Interface (API) dei vari servizi presenti in un unico punto di accesso;
  3. il terzo è l'implementazione di un Eureka Server[g] che contiene le informazioni di tutti i servizi che si registrano nel suo server.

  Oltre all'implementazione di nuovi microservizi, quelli esistenti sono stati modificati affinchè le *will* possano essere visualizzate o da tutti gli utenti oppure solo dagli utenti appartenenti agli stessi gruppi. Ultimate le attività lato *backend*, è stata effettuata la containerizzazione[g] di tutti i microservizi su Docker.

* La seconda attività è stata la modifica del *frontend*, mediante il framework Angular[g], per adeguarla alle nuove funzionalità del *backend*.

L'esito dello *stage* è stato molto positivo: le attività obbligatorie e facoltative sono state portate a termine con successo abbastanza facilmente e con un un po' di anticipo che mi ha permesso di implementare anche la parte *frontend* dell'applicazione.

"Life is really simple, but we insist on making it complicated"

— Confucius

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Prof. Paolo Baldan, relatore della mia tesi, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro.

Desidero ringraziare con affetto i miei genitori per il sostegno, il grande aiuto e per essermi stati vicini in ogni momento durante gli anni di studio.

Ho desiderio di ringraziare poi i miei amici per tutti i bellissimi anni passati insieme e le mille avventure vissute.

Padova, Dicembre 2021 Lorenzo Matterazzo

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

1.1 Convenzioni tipografiche

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

* gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
* per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: parola[g];
* i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*.

1.2 L'azienda

Sync Lab S.r.l. nasce a Napoli nel 2002 come *software house* ed è rapidamente cresciuta nel mercato dell'Information and Comunications Tecnology (ICT), tramutatasi in *System Integrator* e conquistando significative fette di mercato nei settori *mobile*, videosorveglianza e sicurezza delle infrastrutture informatiche aziendali.
Attualmente, Sync Lab S.r.l. ha più di 150 clientidiretti e finali, con un organico aziendale di 200 dipendenti distribuiti tra le 5 sedi dislocate in tutta Italia.
Sync Lab S.r.l. si pone come obiettivo principale quello di supportare il cliente nella realizzazione, messa in opera e governance di soluzione IT, sia dal punto di vista tecnologico, sia nel governo del cambiamento organizzativo.

Figura 1.1: Logo dell'azienda

1.3 Lo stage proposto

"Lo sport dà il meglio di sé quando ci unisce."
Frank Deford

SportWill permette agli utenti di condividere le proprie *will*, e gli utenti che hanno intenzione di unirsi in questa attività vi possono partecipare.
Dal momento che allo stato dell'arte attuale non esiste ancora la suddivisione delle *will* per gruppi, lo *stage* proposto da Sync Lab S.r.l. consiste nell'integrare alla piattaforma già esistente la suddivisione delle visualizzazioni delle *will* solo agli utenti che appartengono agli stessi gruppi.
Gli obbiettivi da raggiungere nel corso dello *stage* sono principalmente due:

* sviluppo di un microservizio utilizzando il framework Spring Java per la creazione dei gruppi;
* modifica dei microservizi esistenti affinchè permettano la visualizzazione solo delle *will* di utenti appartenenti allo stesso gruppo.

Non è richiesta la modifica del *frontend* in modo da adeguarlo alle nuove funzionalità del *backend*, a meno che non rimanga tempo da investire su questa attività.

1.4 Strumenti utilizzati

1.4.1 Visual Studio Code

Visual Studio Code è un editor di codice sorgente sviluppato da Microsoft per Windows, Linux e macOS. Include il supporto per debugging, un controllo per Git integrato, Syntax highlighting, IntelliSense, Snippet e refactoring del codice.
Punto di forza di Visual Studio Code sono le estensioni grazie alle quali è possibile ampliare notevolmente le funzionalità del programma.
Le estensioni utilizzate nel corso dello *stage* sono:

* **GitLens**: permette di ampliare le funzionalità di Git integrate in Visual Studio Code;
* **Spring Boot Extension Pack**: raccolta di estensioni per lo sviluppo con Spring Boot Application;
* **W3C Web Validator**: permette di controllare la validità del *markup* dei documenti html e css;
* **Web Accessibility**: permette di verificare l'accessibilità dei documenti html, evidenziando gli elementi che si potrebbe prendere in conisderazione di cambiare e dando suggerimenti su come potrebbe modificato;
* **Docker Extension Pack**: raccolta di estensioni per lo gestione dei *container* Docker, Docker images, Dockerfile e file Docker-compose;
* **Angular Extension Pack**: raccolta di estensioni per lo sviluppo con Angular;
* **Extension Pack for Java**: raccolta di estensioni popolari che possono aiutare a scrivere, testare e fare il *debugging* di applicazioni Java.

1.4.2 Figma

Figma è un *tool* per la progettazione di interfacce, che si rivolge principalmente ai *web designer* che hanno bisogno di un software studiato appositamente per realizzare il *design* di siti web e applicazioni.
Nel contesto dello *stage* è stato utilizzato per la realizzazione delle pagine web per la visualizzazione, creazione e modifica di un gruppo.

1.4.3 Git

Software di versionamento utile a tracciare modifiche e cambiamenti di insiemi di file.

1.4.4 Postman

Si tratta di uno strumento che permette di eseguire richieste HTTP ad un *server* di *backend*. Quando si lavora con un altro sviluppatore *backend* è possibile condividere le API, ma la sua vera forza è quella di farci sapere tutto di una richiesta HTTP.

1.4.5 DbVisualizer

DbVisualizer è uno strumento multi-database intuitivo e ricco di funzionalità per sviluppatori, analisti e amministratori di database, che fornisce un'unica, potente interfaccia su un'ampia gamma di sistemi operativi. Grazie alla sua interfaccia chiara e facile da usare, DbVisualizer si è dimostrato uno strumento di database molto conveniente, che funziona su tutti i principali sistemi operativi e supporta molte varietà di database.

1.5 Prodotto ottenuto

Al termine dello *stage* le integrazioni delle funzionalità con la *web-app* sono state realizzata con successo. Tutte le chiamate alle API sono state testate e sono state integrate anche nel *frontend*. L'integrazione delle nuove funzionalità permettono alla *web-app* di:

* visualizzare le *will* appartenenti agli utenti che partecipano agli stessi gruppi;
* visualizzare le *will* con visibilità globale (quindi che non sono visibili solo agli utenti che partecipano agli stessi gruppi);
* visualizzare i gruppi a cui partecipa un utente;
* visualizzare e modificare i gruppi creati da un utente;
* visualizzare e cercare i gruppi;
* creare nuovi gruppi.

1.6 Accessibilità

Durante la realizzazione del sito è stata resa la navigazione più efficace attraverso l'uso di accesskey.
Le immagini sono tutte marcate con gli appositi *tag alt*, che sono stati lasciati vuoti nel caso servissero solo per il *layout*.
È presente una barra di navigazione che aiuta a navigare nel sito.
Ogni *link* è stato reso distinguibile da ogni altro elemento tramite appositi CSS, *hover* e *visited*, in modo da aiutare l'utente ad orientarsi.
Inoltre, per evitare *link* circolari, nella barra di navigazione le pagine non contengono *link* che navigano alla pagina corrente.
I form contengono dei tag label per ogni input.
Non sono stati aggiunti *tag optgroup* o *fieldset* in quanto sono utili nel caso di *form* molto grandi, ma essendo presenti solo *form* di piccole dimensioni è stato ritenuto non necessario.
Sono presenti gli attributi *accesskey* con chiave uguale alla prima lettera della parola del *tag label* associato per migliorare l'accessibilità alle *form* da tastiera senza l'uso del *mouse*. Sono presenti degli aiuti contestuali che mostrano gli errori nel caso fossero presenti. Non sono stati aggiunti *tabindex* nei *form* dato che l'ordine di tabulazione è già corretto.
Sono stato evitati i tag ed attributi deprecati.
È presente del testo nascosto utile agli utenti con disabilità visive, come il *link* con la funzione di saltare al contenuto, ovvero di permettere di non far leggere allo *screen reader* la barra di navigazione, passando direttamente al contenuto, ed è presente all'inizio della navigazione per segnalare che le scorciatoie da tastiera sono attive.
Inoltre per migliorare la navigazione dello *scroll*, viene mostrato un pulsante in basso a destra dello schermo che, se un utente lo clicca, viene effettuato uno *scroll* verso l'alto fino all'inizio della pagina.
Il pulsante è un *link* con testo nascosto e come immagine di *background* una freccia, in modo che lo *screen reader* riesca comunque a leggere il testo. Le parole in lingua straniera sono state racchiuse dentro un *tag* con l'attributo *lang*, in modo da permettere allo *screen reader* di leggere la parola correttamente.

1.7 Organizzazione del testo

Il secondo capitolo descrive l'analisi preventiva dei rischi, gli obiettivi dello *stage* e la pianificazione delle ore di lavoro.

Il terzo capitolo approfondisce l'analisi dei requisiti del prodotto.

Il quarto capitolo approfondisce la fase di progettazione e codifica.

Il quinto capitolo approfondisce l'accessibilità e la fase di verifica e validazione.

Il sesto capitolo contiene un'analisi del lavoro svolto e le conclusioni tratte.

Capitolo 2

Descrizione dello stage

In questo capitolo è presente un'analisi preventiva dei rischi che potevano venire riscontrati durante lo svolgimento dello stage, la lista degli obiettivi da raggiungere e la pianificazione delle ore di lavoro.

2.1 Analisi preventiva dei rischi

Qui vengono analizzati i rischi emersi durante la fase di analisi iniziale a cui è possibile incombere. Per ogni rischio individuato, si è proceduto ad elaborare un piano di contingenza per far fronte a tali rischi.

1. Inesperienza tecnologica
Descrizione: le tecnologie da utilizzare sono nuove o esplorate parzialmente, il che può portare alla nascita di problemi operativi.
Piano di contingenza: le attività che richiedono maggior tempo oppure con un livello di capacità tecniche elevate saranno trattate per prime, in modo da migliorarle incrementalmente durante il periodo di stage.

2. Problematiche *software* di supporto
Descrizione: il *computer* in cui si sviluppa il prodotto potrebbe guastarsi, e nel caso accadesse potrebbe causare gravi ritardi.
Piano di contingenza: ad ogni *commit* effettuato è necessario aggiornare la *repository* remota. Inoltre deve essere possibile replicare velocemente l'ambiente di lavoro in un altro *computer* in modo da tornare operativi nel minor tempo possibile. Un modo per ripristinare velocemente le impostazioni di lavoro è la creazione di una cartella da versionare chiamata `.vscode` in cui inserire:

- il file `settings.json`, in cui salvare le impostazioni dell'editor;
- il file `extensions.json`, in cui aggingere gli id di ogni estensione utilizzata in Visual Studio Code;
- Il file `launch.json`, che viene usato per configurare il `debugger` in Visual Studio Code.

.

3. Tempistiche
Descrizione: il tempo di apprendimento di nuove tecnologie potrebbe portare a

ritardi sulle scadenze previste. I ritardi verranno individuati nel caso in cui il lavoro da effettuare si scostasse dalla pianificazione presente su Trello[g].
Piano di contingenza: appena si rilevano difficoltà o scostamenti rispetto al piano di lavoro, si dovrà avvisare tempestivamente il *tutor* aziendale, con cui ci si potrà confrontare, e solo in caso estremo rimandare le scadenze prefissate..

4. Impegni personali
Descrizione: è possibile che vi siano degli impegni personali da adempiere e che di conseguenza abbia meno tempo da poter dedicare allo sviluppo del progetto..
Piano di contingenza: gli incarichi con le relative scadenze sono stati predisposti nel rispetto degli eventuali impegni personali. In caso di imprevisti, bisognerà immediatamente contattare il *tutor* aziendale..

5. Interpretazione errata o non sufficiente dei requisiti
Descrizione: Dopo una prima analisi dei requisiti, è possibile che si noti la necessità di modificare o aggiungere nuovi requisiti in un secondo momento..
Piano di contingenza: Due volte a settimana verrà fatto il punto della situazione con il *tutor* aziendale. Nel caso si notassero assenze nei requisiti, si procederà alla sua conseguante analisi e si deciderà come procedere in maniera da limitare un eventuale rallentamento sulla di sviluppo..

2.2 Requisiti e obiettivi

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 dal riconoscibile valore aggiunto;
* *F* per i requisiti facoltativi, rappresentanti valore aggiunto non strettamente competitivo.

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

Obiettivi fissati

Si prevede lo svolgimento dei seguenti obiettivi:

* Obbligatori
  - *O01*: Acquisizione competenze sulle tematiche sopra descritte;
  - *O02*: Capacità di raggiungere gli obiettivi richiesti in autonomia seguendo il cronoprogramma;
  - *O03*: Portare a termine l'implementazione dei microservizi richiesti con una percentuale di superamento pari al 80.
* Desiderabili

  - *D01*: Portare a termine l'implementazione dei microservizi richiesti con una percentuale di superamento pari al 100.

* Facoltativi
  - *F01*: Utilizzo della containerizzazione per portare tutti i microservizi su Docker.

2.3 Pianificazione del lavoro

Pianificazione settimanale

* **Prima Settimana (40 ore)**
  - Incontro con persone coinvolte nel progetto per discutere i requisiti e le richieste relativamente al sistema da sviluppare;
  - Verifica credenziali e strumenti di lavoro assegnati;
  - Ripasso Java Standard Edition e tool di sviluppo (IDE ecc.);
  - Studio teorico dell'architettura a microservizi: passaggio da monolite a microservizi con pro e contro;
  - Ripasso principi della buona programmazione (SOLID, CleanCode);
  - Ripasso Java Standard Edition.
* **Seconda Settimana - (40 ore)**
  - Studio teorico dell'architettura a microservizi: passaggio da monolite ad architetture a microservizi;
  - Studio teorico dell'architettura a microservizi: Api Gateway, Service Discovery e Service Registry, Circuit Breaker e Saga Pattern;
  - Studio Spring Core/Spring Boot.
* **Terza Settimana - (40 ore)**
  - Studio servizi REST e framework Spring Data REST;
  - Studio ORM, in particolare il framework Spring Data JPA.
* **Quarta Settimana - (40 ore)**
  - Studio ORM, in particolare il framework Spring Data JPA.;
* **Quinta Settimana - (40 ore)**
  - Studio della piattaforma SportWill esistente;
  - Analisi nuova funzionalità da implementare.
* **Sesta Settimana - (40 ore)**
  - Implementazione del nuovo servizio.
* **Settima Settimana - (40 ore)**
  - Implementazione del nuovo servizio.
* **Ottava Settimana - Conclusione (40 ore)**
  - Considerazioni e collaudi finali.

Ripartizione ore

La pianificazione, in termini di quantità di ore di lavoro, sarà così distribuita:

Durata in ore	Descrizione dell'attività
160	**Formazione sulle tecnologie**
18	*Studio Java Standard Edition e tool di sviluppo*
18	*Studio architettura a microservizi*
4	*Ripasso dei principi della buona programmazione (SOLID, Clean-Code)*
10	*Studio teorico dell'architettura a microservizi: passaggio da monolite ad architetture a microservizi*
15	*Studio teorico dell'architettura a microservizi: Api Gateway, Service Discovery e Service Registry, Circuit Breaker e Saga Pattern*
15	*Studio Spring Core/Spring Boot*
20	*Studio servizi REST e framework Spring Data REST*
60	*Studio ORM, in particolare il framework Spring Data JPA*
40	**Definizione architettura di riferimento e relativa documentazione**
14	*Analisi del problema e del dominio applicativo*
22	*Progettazione della piattaforma e relativi test*
4	*Stesura documentazione relativa ad analisi e progettazione*
80	**Implementazione del nuovo servizio**
40	**Collaudo Finale**
30	*Collaudo*
6	*Stesura documentazione finale*
2	*Incontro di presentazione della piattaforma con gli stakeholders*
2	*Live demo di tutto il lavoro di stage*
Totale ore	**320**

Capitolo 3

Analisi dei requisiti

Il presente capitolo descrive in maniera dettagliata requisiti e casi d'uso individuati durante l'analisi del progetto di stage.

3.1 Casi d'uso

Attori pricipali

Nell'analisi del progetto di *stage* è emerso solo un attore, ovvero l'**utente autenticato**, attore che indica un utente che ha effettuato l'autenticazione all'interno dell'applicazione web. Ha quindi la possibilità di vedere tutte le informazioni sui gruppi e sulle *will* che sarebbero altrimenti inaccessibili.

Elenco dei casi d'uso

Per lo studio dei casi di utilizzo del prodotto sono stati creati dei diagrammi. I diagrammi dei casi d'uso (in inglese *Use Case Diagram*) sono diagrammi di tipo Unified Modeling Language (UML) dedicati alla descrizione delle funzioni o servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso.
I casi d'uso che riportati in seguito descrivono solo le funzionalità che dovranno essere implementate, senza quindi descrivere quelle già presenti nella *web-app*.

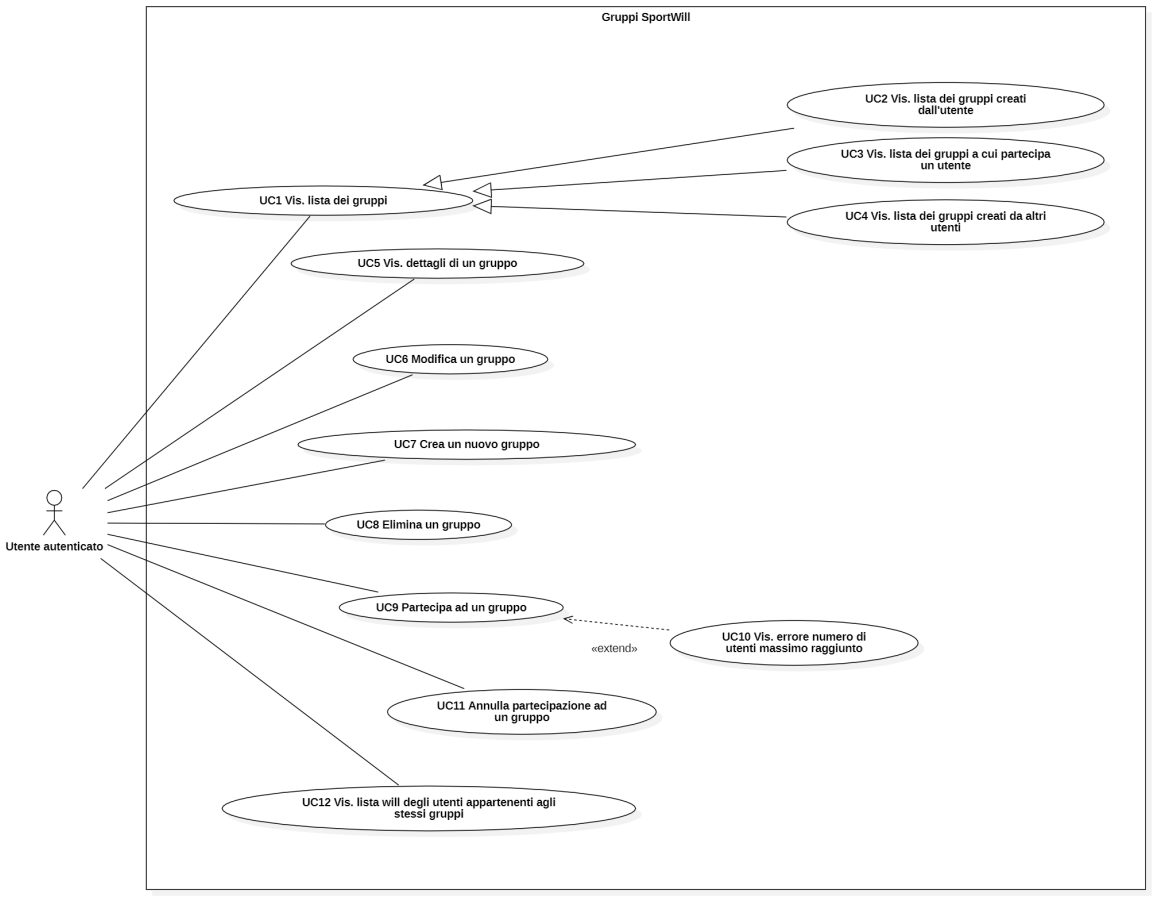

Figura 3.1: Diagramma generale dei casi d'uso

UC1: Visualizzazione lista dei gruppi

Attori Principali: Utente autenticato.

Precondizioni: L'utente ha aperto la *web-app* ed è autenticato.

Descrizione: L'utente vuole visualizzare la lista dei gruppi.

Postcondizioni: L'utente ha visualizzato la lista dei gruppi.

Scenario principale:

1. L'utete visualizza la lista dei gruppi.

UC1.1:

UC1.1.1:

UC1.1.2:

UC1.1.3:

UC2: Prova1

Attori Principali: Sviluppatore applicativi.

Precondizioni: Lo sviluppatore è entrato nel plug-in di simulazione all'interno dell'IDE.

Descrizione: La finestra di simulazione mette a disposizione i comandi per configurare, registrare o eseguire un test.

Postcondizioni: Il sistema è pronto per permettere una nuova interazione.

UC2.1:

UC2.1.1:

UC2.2:

UC2.3:

UC2.3.1:

UC2.3.2:

3.2 Tracciamento dei requisiti

Da un'attenta analisi dei requisiti e degli use case effettuata sul progetto è stata stilata la tabella che traccia i requisiti in rapporto agli use case.
Sono stati individuati diversi tipi di requisiti e si è quindi fatto utilizzo di un codice identificativo per distinguerli.
Il codice dei requisiti è così strutturato R(F/Q/V)(N/D/O) dove:

R = requisito

F = funzionale

Q = qualitativo

V = di vincolo

N = obbligatorio (necessario)

D = desiderabile

Z = opzionale

Nelle tabelle 3.1, 3.2 e 3.3 sono riassunti i requisiti e il loro tracciamento con gli use case delineati in fase di analisi.

Tabella 3.1: Tabella del tracciamento dei requisti funzionali

Requisito	Descrizione	Use Case
RFN-1	L'interfaccia permette di configurare il tipo di sonde del test	UC1

Tabella 3.2: Tabella del tracciamento dei requisiti qualitativi

Requisito	Descrizione	Use Case
RQD-1	Le prestazioni del simulatore hardware deve garantire la giusta esecuzione dei test e non la generazione di falsi negativi	-

Tabella 3.3: Tabella del tracciamento dei requisiti di vincolo

Requisito	Descrizione	Use Case
RVO-1	La libreria per l'esecuzione dei test automatici deve essere riutilizzabile	-

Capitolo 4

Progettazione e codifica

Breve introduzione al capitolo

4.1 Tecnologie e strumenti

Di seguito viene data una panoramica delle tecnologie e strumenti utilizzati.

Tecnologia 1

Descrizione Tecnologia 1.

Tecnologia 2

Descrizione Tecnologia 2

4.2 Ciclo di vita del software

4.3 Progettazione

Namespace 1

Descrizione namespace 1.

Classe 1: Descrizione classe 1

Classe 2: Descrizione classe 2

4.4 Design Pattern utilizzati

UC2.3.2

4.5 Codifica

Capitolo 5

Verifica e validazione

Capitolo 6

Conclusioni

6.1 Consuntivo finale

6.2 Raggiungimento degli obiettivi

6.3 Conoscenze acquisite

6.4 Valutazione personale

Appendice A

Appendice A

Citazione

Autore della citazione

Bibliografia