

Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

CORSO DI LAUREA IN INFORMATICA



Migrazione e analisi comparativa di un  
back-end per un servizio di smart parking

*Tesi di laurea*

*Relatore*

Prof. Paolo Baldan

*Laureando*

Andrea Volpe

---

ANNO ACCADEMICO 2021-2022



Lorem ipsum dolor sit amet, consectetur adipiscing elit.

— Oscar Wilde

Dedicato a ...



# Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di circa trecento ore, dal laureando Pinco Pallino presso l'azienda Azienda S.p.A. Gli obbiettivi da raggiungere erano molteplici.

In primo luogo era richiesto lo sviluppo di ... In secondo luogo era richiesta l'implementazione di un ... Tale framework permette di registrare gli eventi di un controllore programmabile, quali segnali applicati Terzo ed ultimo obbiettivo era l'integrazione ...



*“Life is really simple, but we insist on making it complicated”*

— Confucius

# Ringraziamenti

*Innanzitutto, vorrei esprimere la mia gratitudine al Prof. NomeDelProfessore, 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 2022*

Andrea Volpe





# Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	L'azienda . . . . .	1
1.2	Scelta dell'azienda . . . . .	1
1.3	Introduzione al progetto . . . . .	2
1.4	Problematiche riscontrate . . . . .	3
1.5	Soluzione scelta . . . . .	3
1.6	Descrizione del prodotto ottenuto . . . . .	4
1.7	Tecnologie utilizzate . . . . .	5
1.8	Organizzazione del testo . . . . .	6
<b>2</b>	<b>Analisi dei requisiti</b>	<b>7</b>
<b>3</b>	<b>Progettazione</b>	<b>9</b>
<b>4</b>	<b>Analisi dei requisiti</b>	<b>11</b>
<b>5</b>	<b>Conclusioni</b>	<b>13</b>
<b>A</b>	<b>Appendice A</b>	<b>15</b>
	<b>Bibliografia</b>	<b>19</b>

**Elenco delle figure**

**Elenco delle tabelle**

# Capitolo 1

## Introduzione

### 1.1 L'azienda

Sync Lab nasce a Napoli nel 2002 come software house ed è rapidamente cresciuta nel mercato dell'Information and Communications Technology (ICT) G. A seguito di una maturazione delle competenze tecnologiche, metodologiche ed applicative nel dominio del software, l'azienda è riuscita rapidamente a trasformarsi in System Integrator conquistando significative fette di mercato nei settori mobile, videosorveglianza e sicurezza delle infrastrutture informatiche aziendali. Attualmente, Sync Lab ha più di 150 clienti diretti e finali, con un organico aziendale di 300 dipendenti distribuiti tra le 6 sedi dislocate in tutta Italia. Sync Lab 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.



### 1.2 Scelta dell'azienda

Sono venuto a conoscenza dell'azienda Sync Lab grazie al progetto d'ingegneria del software, dove l'azienda è stata la proponente del mio progetto.

Sono venuto a conoscenza del progetto di stage di Sync Lab grazie all'evento stage-it 2022. L'evento promosso da Assindustria Venetocentro in collaborazione con l'Università di Padova per favorire l'incontro tra aziende con progetti innovativi in ambito IT e studenti dei corsi di laurea in Informatica, Ingegneria informatica e Statistica.

### 1.3 Introduzione al progetto




Lo scopo del progetto di stage consiste nell'effettuare la migrazione di un servizio di API REST lato back-end realizzato da un precedente studente tirocinante con il framework Spring in un servizio di API REST lato back-end realizzato con un diverso framework, chiamato NestJS. La migrazione viene fatta per effettuare un'analisi comparativa tra i due servizi, in modo da valutarne le caratteristiche e decidere quale dei due meglio si adatta alle esigenze del progetto.

Il progetto consiste nella realizzazione di una webapp che si occupa di gestire un sistema di controllo parcheggi auto. Il sistema va ad interrogare una base di dati contenente l'informazione inerente allo stato di alcuni sensori di parcheggio fornendo la visualizzazione dei posti liberi/occupati all'interno di una mappa.

L'idea del progetto consiste nel agevolare il client dell'applicativo, in quanto può venire a conoscenza della disponibilità di un parcheggio prima di entrarci, evitando quindi spostamenti inutili nel caso il parcheggio sia pieno.

E' prevista poi la realizzazione di una sezione dedicata ai manutentori, per verificare lo stato dei sensori, facilitando quindi il processo di manutenzione.

Il progetto è formato da una parte di front-end, realizzata con il framework Angular e una parte di back-end che consiste di un servizio di REST API, realizzato in due versioni: una con il framework Spring e una con il framework NestJS.

	Parcheggio libero
	Parcheggio occupato
	Sensore ambientale



## 1.4 Problematiche riscontrate

Problematiche dovute alla mancanza di conoscenza delle tecnologie:

- \* Architettura a microservizi: avevo solo una conoscenza basilare della tecnologia, grazie al corso d'ingegneria del software ma non sufficiente per sviluppare il progetto.
- \* Framework Spring: la conoscenza di questo framework era completamente assente ed era importante conoscerlo per poter comprendere con chiarezza il software esistente di cui doveva essere effettuata la migrazione.
- \* Framework Node.js e NestJS: la conoscenza di questi due framework era completamente assente era di fondamentale importanza conoscerli per poter implementare il servizio di REST API lato back-end richiesto.

Problematiche a livello architetturale:

- \* La quantità di REST API da migrare era troppo elevata per il tempo a disposizione.
- \* Il database in uso si è rivelato non essere in forma normale e di conseguenza dava problemi come la ridondanza dei dati.

## 1.5 Soluzione scelta

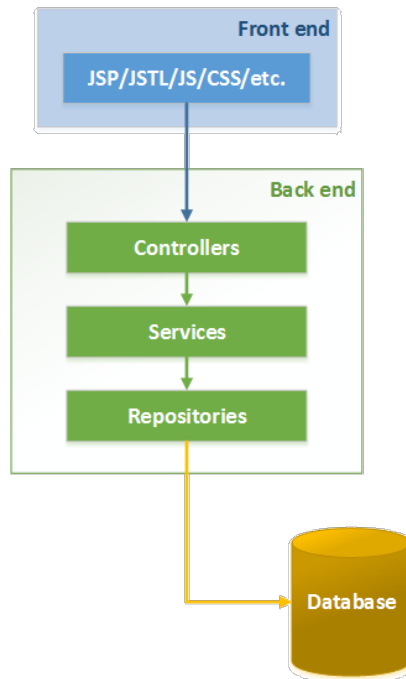
E' stato scelto di sviluppare con un architettura di tipo layered architecture. Questo è uno degli stili architetturali più utilizzati quando si sviluppa un monolite. L'idea dietro a questa architettura è che i moduli con funzionalità simili sono organizzati in livelli orizzontali. Quindi ogni livello svolge uno specifico ruolo nell'applicazione.

La layered architecture astrae la visione del sistema nel suo insieme, fornendo dettagli sufficienti per comprendere ruoli e le responsabilità dei singoli livelli e le relazioni che intercorrono tra loro.

La motivazione che ha portato alla scelta di questo stile architetturale è un'analisi fatta che ha rivelato la layered architecture adattarsi molto bene al servizio di REST API che si voleva andare a realizzare ed inoltre il fatto che molti framework per lo sviluppo di applicativi back-end si basano su esso, tra cui Spring e NestJS, che sono fondati sul pattern controller-service-repository. Un pattern che sfrutta la layered architecture, creando tre diversi livelli:

- \* controller: è il livello più alto ed è l'unico responsabile dell'esposizione delle funzionalità in modo che possano essere consumate da entità esterne.
- \* service: livello centrale, gestisce tutta la business logic.
- \* repository: livello più basso, è responsabile di salvare e recuperare i dati da un sistema di persistenza, come un database.

Questa struttura viene utilizzata per effettuare una buona separazione delle responsabilità. L'architettura usata da Spring e NestJS ha portato a sceglierli come framework



per realizzare la parte back-end del progetto.

Non è prevista la creazione di un sistema di autenticazione per l'uso delle API, in quanto un altro studente tirocinante si stava occupando della creazione di questa parte.

## 1.6 Descrizione del prodotto ottenuto

Al momento è disponibile un back-end contenente le REST API sviluppate in NestJS, utilizzabile, in quanto non potendo migrare l'intero set di REST API disponibili in Spring, come preventivato, sono state sviluppate tutte le REST API più importanti per effettuare le operazioni CRUD più comuni.

Le REST API espongono un'interfaccia compatibile con quello che ormai è uno standard per la comunicazione con servizi di tipo REST. Ovvero per comunicare con le REST API bisogna fare delle richieste HTTP a degli specifici endpoint con i seguenti metodi HTTP:

- \* GET: per ottenere delle risorse dal servizio REST
- \* POST: per creare una nuova risorsa nel servizio REST
- \* PUT: per modificare una risorsa nel servizio REST
- \* DELETE: per eliminare una risorsa dal servizio REST

E' presente poi un servizio schedato che ogni due minuti in maniera autonoma va a fare il polling da un file XML online, contenente gli stati aggiornati dei sensori.

Questo servizio registra poi le variazioni, rispetto al polling precedente, nel servizio di persistenza.

Il file XML viene scritto e gestito dai produttori dei sensori di parcheggio, quindi non è compito di questo progetto gestirne il funzionamento. Il funzionamento di questo file è comunque abbastanza banale, in quanto ad ogni variazione di stato il sensore di parcheggio va semplicemente ad aggiornare il record a lui associato all'interno del file.

## 1.7 Tecnologie utilizzate

### Git

E' uno degli strumenti di controllo di versionamento più utilizzati. Facilita la collaborazione di più sviluppatori nella realizzazione di un progetto e permette con semplicità di spostarsi tra varie versioni del software realizzate. Nel progetto è stato utilizzato con il workflow Gitflow.

### Visual Studio Code

E' un editor di codice sorgente sviluppato da Microsoft che aiuta molto lo sviluppatore durante la fase di sviluppo del codice in quanto evidenzia le parole chiave, segnala errori di scrittura, suggerisce snippet di codice. Possiede una grande libreria di estensioni facilmente installabili, per renderlo compatibile con praticamente qualsiasi linguaggio di programmazione.

### Postman

E' un'applicazione che viene utilizzata solitamente per testare API. E' un client HTTP che testa richieste HTTP, utilizzando una GUI, attraverso la quale otteniamo diversi tipi di risposta in base alle API che andiamo ad interrogare.

### Stoplight

E' una piattaforma per disegnare API. Grazie a questo strumento è possibile documentare in maniera rigorosa e su uno spazio in cloud un set di API. La piattaforma permette di specificare varie informazioni per ogni API, tra cui endpoint, parametri in ingresso attesi, possibili risposte con status code associato. Questo strumento è molto utile per gli sviluppatori front-end che devono chiamare le API di un servizio back-end, soprattutto grazie alla funzionalità che permette di effettuare il mock della risposta di un'API.

### TypeScript

E' un superset di JavaScript, che aggiunge tipi, classi, interfacce e moduli opzionali al JavaScript tradizionale. Si tratta sostanzialmente di una estensione di JavaScript. TypeScript è un linguaggio tipizzato, ovvero aggiunge definizioni di tipo statico: i tipi consentono di descrivere la forma di un oggetto, documentandolo meglio e consentendo a TypeScript di verificare che il codice funzioni correttamente.

### Node.js

E' un framework per realizzare applicazioni Web in JavaScript, permettendoci di utilizzare questo linguaggio, tipicamente utilizzato nella client-side, anche per la scrittura

di applicazioni server-side. La piattaforma è basata sul JavaScript Engine V8, che è il runtime di Google utilizzato anche da Chrome e disponibile sulle principali piattaforme, anche se maggiormente performante su sistemi operativi UNIX-like.

#### NestJS

E' un framework per la creazione di applicazioni lato server Node.js efficienti e scalabili. Utilizza JavaScript ma è costruito con e supporta completamente TypeScript. Aggiunge un livello di astrazione al framework Express, che a sua volta aggiunge astrazione al framework Node.js. Di conseguenza NestJS utilizza Node.js per eseguire il codice JavaScript prodotto dal codice TypeScript compilato.

#### Spring

Spring è un framework leggero, basato su Java. Questo framework integra soluzioni a vari problemi tecnici che si presentano con alta frequenza durante lo sviluppo software. Spring si basa su due design pattern fondamentali che sono l'Inversion of Control e Dependency Injection.

#### PostgreSQL

Chiamato anche Postgres, è un sistema di database relazionale a oggetti (ORDBMS), open source e gratuito. Le principali caratteristiche di Postgres sono affidabilità, integrità dei dati, funzionalità ed estensibilità, oltre alla propria community open source che gestisce, aggiorna e sviluppa soluzioni performanti e innovative.

## 1.8 Organizzazione del testo

**Il secondo capitolo** describe ...

**Il terzo capitolo** approfondisce ...

**Il quarto capitolo** approfondisce ...

**Il quinto capitolo** approfondisce ...

Riguardo la stesura del testo, relativamente al 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*<sup>[g]</sup>;
- \* i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*.



## Capitolo 2

# Analisi dei requisiti



## Capitolo 3

# Progettazione



## Capitolo 4

# Analisi dei requisiti



Capitolo 5

Conclusioni





Appendice A

Appendice A

Citazione

---

Autore della citazione







# Bibliografia

## Riferimenti bibliografici

James P. Womack, Daniel T. Jones. *Lean Thinking, Second Editon*. Simon & Schuster, Inc., 2010.

## Siti web consultati

*Manifesto Agile*. URL: <http://agilemanifesto.org/iso/it/>.