Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica

Realizzazione di una applicazione Android mobile per la gestione di un sistema di smart parking

Tesi di laurea triennale

Relatore

Prof.Davide Bresolin

Laureando

Davide Sut

Anno Accademico 2022-2023

Sommario

Il presente documento descrive il lavoro da me svolto durante il periodo di stage, che ha avuto la durata di circa trecento ore, presso la sede di Padova dell'azienda Sync Lab s.r.l., nel periodo che va dal 14/11/2022 al 13/01/2023.

Lo scopo del progetto di stage era la realizzazione della parte mobile di una web app che si occupa di gestire un sistema di controllo parcheggi auto.
Il mio compito era quindi quello di collaborare nella fase di sviluppo del front-end, effettuando l'analisi, la progettazione e la codifica delle sue unità.

Durante il tirocinio è stato previsto un periodo di formazione volto allo studio di nuove tecnologie in ambito di sviluppo web e mobile; in particolare, ho studiato il framework Angular e il toolkit Ionic.

In questo documento vengono descritte le fasi di analisi, progettazione e codifica dell'applicativo software realizzato, spiegando anche il funzionamento delle tecnologie e framework utilizzati.

"Ognuno è un genio. Ma se si giudica un pesce dalla sua abilità di arrampicarsi sugli alberi, passerà tutta la sua vita a credersi stupido."

— Anonimo (attr. Albert Einstein)

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Prof. Davide Bresolin, relatore della mia tesi, per la disponibilità e professionalità offertami durante la stesura del documento.

Ringrazio l'Ing. Fabio Pallaro, mio tutor aziendale e manager sede Sync Lab, e il dott. Daniele Zorzi, responsabile del progetto di stage, per l'opportunità fornitami e il supporto dato durante l'intera attività di tirocinio.

Desidero ringraziare con affetto i miei genitori, Deborah e Paolo, per il loro amore infinito e per essermi stati vicini in ogni momento, sostenendo a pieno le mie scelte e motivandomi a dare sempre il meglio, senza mai mollare.

Ringrazio mia sorella Lucia, perché, nonostante i normali litigi e incomprensioni, è sempre stata l'unica a sapere come farmi stare bene, facendomi divertire con i suoi numerosi scherzi e giochi, ma anche standomi accanto e confortandomi in tutti momenti difficili che abbiamo condiviso, dimostrandomi, a modo suo, un grande amore fraterno.

Ringrazio la mia gatta Stella, per tutte le giornate di gioco, la compagnia e il conforto emotivo che mi ha regalato.

Un ringraziamento speciale va ai miei nonni, Luciana e Giuseppe, che mi hanno sempre sostenuto e aiutato anche nei momenti di sconforto, senza i loro insegnamenti non sarei la persona che sono ora.

Ringrazio mia nonna Maria, per il suo grande affetto e per tutti i pranzi e ritrovi familiari che ha organizzato, preoccupandosi sempre di non far mancare mai nulla a nessuno.

Ringrazio mio nonno Bruno, anche se non è più presente tra noi, per tutte le bellissime giornate passate assieme e per il suo grande cuore.

Ci tengo a ringraziare, inoltre, tutti i numerosi parenti non citati in precedenza, per il loro sostegno e affetto.

Ringrazio il mio caro amico Andrea, per tutte le divertenti serate passate assieme e perché mi è sempre stato vicino, aiutandomi a superare anche i momenti più difficili.

Infine vorrei ringraziare tutti i miei amici, in particolare Arianna, Alessandro e Davide, per aver alleggerito le mie giornate e per essere sempre stati presenti quando ne avevo bisogno.

Padova, Febbraio 2023 Davide Sut

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

In questo capitolo verrà introdotta l'attività di stage con una panoramica sull'azienda e sul progetto di tirocinio scelto.

1.1 L'azienda

Sync Lab è un'azienda italiana che nasce come software house nel 2002 a Napoli. Si è affermata nel mercato ICT (*Information and Communication Technologies*) ed è riuscita a trasformarsi rapidamente in *System Integrator*.

Opera nei settori della sanità, dell'industria, dell'energia, delle telecomunicazioni, della finanza e dei trasporti.
Alcuni tra i clienti più rilevanti sono: Sky, Eni, Enel, UniCredit, Poste Italiane, InfoCert, Tim, Vodafone, Intesa SanPaolo e Trenitalia.

L'azienda ha inoltre iniziato ad operare in nuovi ambiti, come ad esempio la sicurezza informatica, l'IoT (*Internet of Things*) e il mobile.
Sync Lab ha attualmente più di 150 clienti diretti e finali, con un numero di dipendenti superiore ai 300 divisi nelle 6 sedi italiane: Napoli, Roma, Milano, Padova, Verona e Como.

Figura 1.1: Logo dell'azienda Sync Lab

1.2 Scelta dell'azienda

Sono venuto a conoscenza delle proposte di stage di Sync Lab grazie all'evento Stage-IT 2022, tenutosi online lo scorso marzo tramite il popolare software Zoom. Quest'evento, promosso dall'Università degli Studi di Padova e Assindustria Venetocentro, ha lo scopo di facilitare l'incontro tra le aziende e gli studenti delle facoltà di Informatica, Ingegneria Informatica e Statistica che sono interessati a progetti di tirocinio in ambito IT (*Information Technology*).

Ho partecipato all'incontro proposto da Sync Lab e presieduto dall'ingegnere Fabio Pallaro, il quale ha introdotto i principali progetti di tirocinio proposti e risposto chiaramente alle domande in merito.
Inoltre ha espresso la volontà, da parte dell'azienda, di cercare sempre di accontentare gli stagisti nelle loro preferenze riguardo all'ambito dell'informatica che vorrebbero approfondire.

Ho scelto Sync Lab in quanto sono rimasto colpito positivamente dalle proposte di stage, che mi avrebbero permesso di approfondire l'ambito del web e del mobile, ma anche dalla chiarezza espositiva e disponibilità dimostrate.

1.3 Introduzione al progetto

Il progetto di stage consiste nell'effettuare il porting su mobile di un'applicazione web già esistente che gestisce un sistema di smart parking.
Quest'applicazione dispone di un front-end sviluppato con il framework Angular che interagisce con un back-end sviluppato con il framework Spring.

L'idea del progetto "Smart Parking" nasce dal voler facilitare un utente che ha bisogno di un posto auto all'interno di un parcheggio cittadino, senza fargli perdere tempo nel cercare un posto libero.
La web app esistente, infatti, mostra una mappa con le disponibilità in tempo reale dei posti auto all'interno delle varie zone di parcheggio.

L'azienda ha deciso di evolvere questo progetto tramite la realizzazione di un'applicazione mobile che disponesse delle stesse funzionalità della web app esistente, in modo da agevolare ulteriormente l'utente, che potrà così usufruire delle funzionalità tramite l'utilizzo di un'applicazione nativa direttamente sul suo smartphone.

Il progetto di stage prevede quindi l'utilizzo delle tecnologie Angular e Ionic per creare il front-end di un'applicazione mobile ibrida, cioè sviluppata con tecnologie web, ma che può essere comunque eseguita all'interno di un'applicazione nativa mobile.
In questo modo si possono approfondire sia le tematiche dello sviluppo web, sia quelle legate allo sviluppo mobile.

Il progetto "Smart Parking" si sta inoltre evolvendo in "Smart City", tramite l'aggiunta di funzionalità per la gestione di sensori ambientali.

Queste nuove funzionalità non sono state considerate per il porting su mobile, in quanto non ancora completate.

1.4 Organizzazione del testo

1.4.1 Struttura del documento

Il primo capitolo contiene un'introduzione all'attività di stage.

Il secondo capitolo descrive l'attività di stage, elencando gli obiettivi, la pianificazione, i possibili rischi e le tecnologie organizzative utilizzate.

Il terzo capitolo approfondisce le fasi di analisi e progettazione dell'applicativo software.

Il quarto capitolo approfondisce la fase di codifica e la validazione finale dell'applicativo software.

Il quinto capitolo presenta un resoconto conclusivo sull'attività di stage.

1.4.2 Convenzioni tipografiche

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 in seguito alla loro prima occorrenza nel testo;
- i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*.

Capitolo 2

Descrizione dello stage

In questo capitolo verrà presentata l'attività di stage, descrivendo gli obiettivi fissati, la pianificazione del lavoro e gli strumenti organizzativi utilizzati.

2.1 Obiettivi

Gli obiettivi per il progetto di stage consistono nell'acquisire le competenze necessarie per lo sviluppo dell'applicazione software in autonomia e seguendo il programma previsto.
Inoltre è richiesta l'implementazione di almeno l'80% delle funzionalità attese, arrivando desiderabilmente al 100%.
Infine sarà un valore aggiunto capacità di collaborare in modo significativo all'interno del gruppo di lavoro, durante le fasi di progettazione delle interfacce.

Notazione

Farò 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

Abbiamo previsto lo svolgimento dei seguenti obiettivi:

* Obbligatori
  - *O01*: Acquisizione delle competenze necessarie allo sviluppo del progetto;
  - *O02*: Capacità di raggiungere gli obiettivi richiesti in autonomia seguendo il cronoprogramma;
  - *O03*: Portare a termine le implementazioni previste con una percentuale di superamento pari all' 80% (porting della piattaforma su Android per una copertura delle funzionalità all' 80%).
* Desiderabili
  - *D01*: Portare a termine le implementazioni previste con una percentuale di superamento pari al 100% (porting della piattaforma su Android per una copertura delle funzionalità al 100%).
* Facoltativi
  - *F01*: Apportare un valore aggiunto al gruppo di lavoro durante le fasi di progettazione delle interfacce.

2.2 Pianificazione

Lo stage che ho svolto ha avuto la durata di **312 ore** lavorative, che corrispondono a circa 8 settimane.
Per migliorarne l'efficienza è stato diviso in due parti: la prima di formazione autonoma sulle tecnologie da utilizzare e la seconda di effettivo sviluppo delle funzionalità dell'applicativo software. Entrambe le parti hanno avuto la durata di circa un mese effettivo di lavoro.

La ripartizione delle attività settimanali è stata la seguente:

* **Prima Settimana - Formazione (40 ore)**
  - Incontro con persone coinvolte nel progetto per discutere i requisiti e le richieste relativamente al sistema da sviluppare; (4 ore)
  - Verifica credenziali e strumenti di lavoro assegnati; (4 ore)
  - Presa visione dell'infrastruttura esistente; (4 ore)
  - Formazione sulle tecnologie adottate; (8 ore)
  - Ripasso concetti Web (Servlet, servizi Rest, Json ecc.); (10 ore)
  - Studio tool StopLight. (10 ore)

* **Seconda Settimana - Formazione (40 ore)**
  - Ripasso Javascript; (16 ore)
  - Studio framework Angular. (24 ore)
* **Terza Settimana - Formazione (40 ore)**
  - Termine studio framework Angular; (16 ore)
  - Realizzazione prototipale in Angular di maschere di login e *CRUD* semplice a scopo didattico. (24 ore)
* **Quarta Settimana - Formazione (40 ore)**
  - Studio toolkit Ionic; (24 ore)
  - Utilizzo di Ionic per portare su mobile Android il prototipo realizzato nella settimana precedente. (16 ore)
* **Quinta Settimana - Sviluppo (40 ore)**
  - Studio del progetto Smart Parking ed analisi impatti per il porting a Ionic; (16 ore)
  - Modifiche al front end per un corretto porting su mobile. (24 ore)
* **Sesta Settimana - Sviluppo (40 ore)**
  - Sviluppo opportune modifiche per il porting su mobile. (40 ore)
* **Settima Settimana - Sviluppo (40 ore)**
  - Sviluppo opportune modifiche per il porting su mobile. (40 ore)
* **Ottava Settimana - Conclusione (32 ore)**
  - Termine sviluppi e collaudi finali. (32 ore)

2.3 Analisi dei rischi

Durante la fase iniziale sono stati individuati alcuni possibili rischi a cui si potrà andare incontro nel percorso di tirocinio.
Abbiamo quindi proceduto a elaborare le possibili soluzioni per far fronte a tali rischi.

1. Tecnologie sconosciute

Descrizione: le tecnologie che devo utilizzare per lo sviluppo e realizzazione del progetto potrebbero essermi parzialmente o completamente sconosciute.
Soluzione: studio autonomo delle tecnologie, seguendo le risorse fornite e segnalate dall'azienda.

2. Difficoltà nel recarsi in azienda

Descrizione: a causa della distanza tra l'azienda e la mia residenza, potrebbe risultare difficile raggiungere la sede aziendale.
Soluzione: organizzazione della settimana lavorativa in un giorno di lavoro in presenza e i rimanenti quattro in smart-working.

3. Monitoraggio del lavoro

Descrizione: a causa del lavoro prevalentemente in smart-working, non sarà sempre possibile avere un confronto diretto con il tutor aziendale.
Soluzione: utilizzo di strumenti organizzativi per condividere i progressi e risolvere eventuali problemi.

2.4 Strumenti organizzativi

Per organizzare il lavoro svolto e mantenere un tracciamento costante dei progressi sono stati utilizzati i seguenti strumenti software:

Trello

Trello è uno strumento online che consente ai gruppi di lavoro di gestire visivamente qualsiasi tipo di progetto, flusso o monitoraggio dei task. Possiede un piano gratuito che offre tutte le funzionalità di base per piccoli progetti e gruppi di lavoro.
Per il progetto di stage, abbiamo creato una *dashboard* contenente le attività divise per settimana.

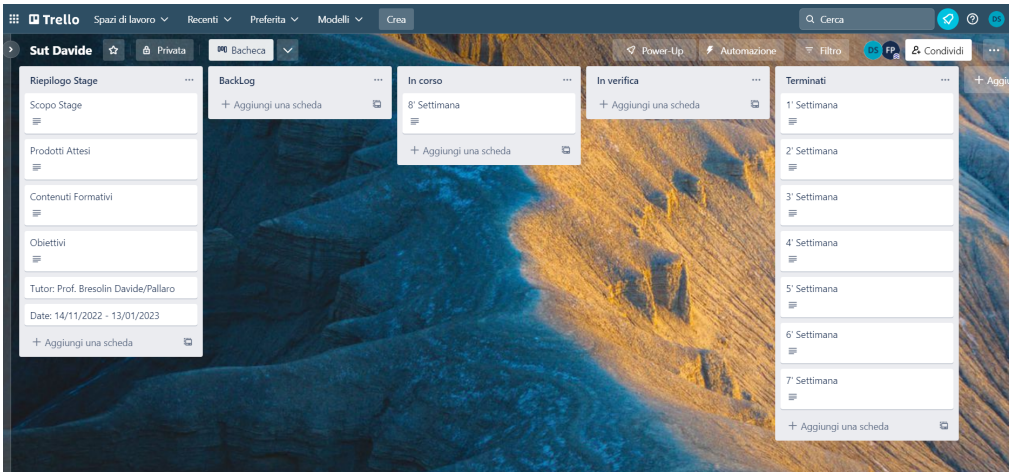
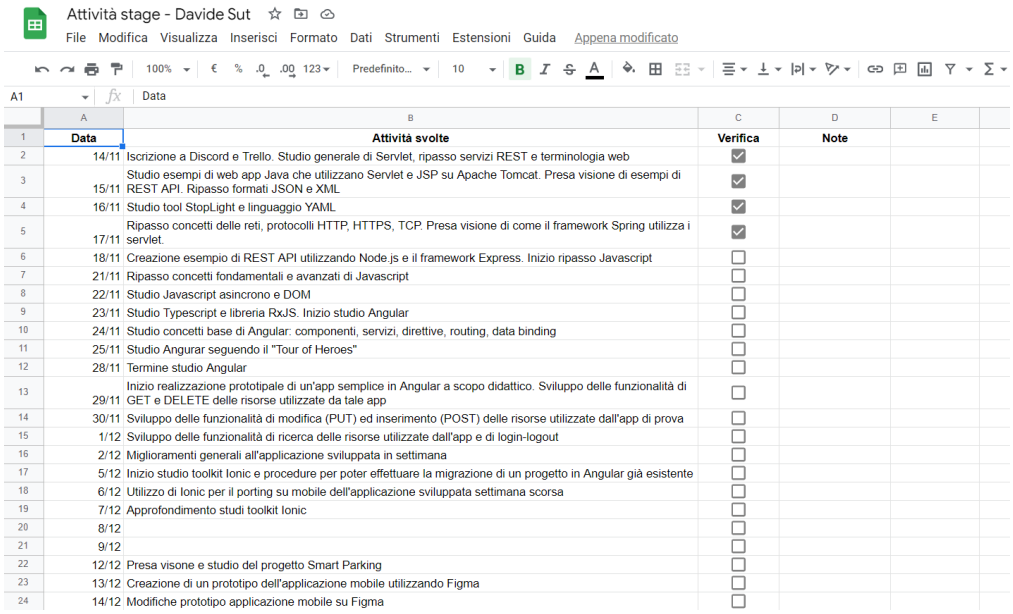


Figura 2.1: Dashboard di Trello

Google Sheets

Google Sheets è uno strumento online per la gestione di fogli di lavoro condivisi e fa parte della suite gratuita Google Docs Editors offerta da Google.
Abbiamo creato un foglio di lavoro in cui riportavo giornalmente le attività svolte.


Attività stage - Davide Sut

File Modifica Visualizza Inserisci Formato Dati Strumenti Estensioni Guida Appena modificato

A1 Data

	Data	Attività svolte	Verifica	Note
2	14/11	Iscrizione a Discord e Trello. Studio generale di Servlet, ripasso servizi REST e terminologia web	☑	
3	15/11	Studio esempi di web app Java che utilizzano Servlet e JSP su Apache Tomcat. Presa visione di esempi di REST API. Ripasso formati JSON e XML	☑	
4	16/11	Studio tool StopLight e linguaggio YAML	☑	
5	17/11	Ripasso concetti delle reti, protocolli HTTP, HTTPS, TCP. Presa visione di come il framework Spring utilizza i servlet.	☑	
6	18/11	Creazione esempio di REST API utilizzando Node.js e il framework Express. Inizio ripasso Javascript	☐	
7	21/11	Ripasso concetti fondamentali e avanzati di Javascript	☐	
8	22/11	Studio Javascript asincrono e DOM	☐	
9	23/11	Studio Typescript e libreria RxJS. Inizio studio Angular	☐	
10	24/11	Studio concetti base di Angular: componenti, servizi, direttive, routing, data binding	☐	
11	25/11	Studio Angurar seguendo il "Tour of Heroes"	☐	
12	28/11	Termine studio Angular	☐	
13	29/11	Inizio realizzazione prototipale di un'app semplice in Angular a scopo didattico. Sviluppo delle funzionalità di GET e DELETE delle risorse utilizzate da tale app	☐	
14	30/11	Sviluppo delle funzionalità di modifica (PUT) ed inserimento (POST) delle risorse utilizzate dall'app di prova	☐	
15	1/12	Sviluppo delle funzionalità di ricerca delle risorse utilizzate dall'app e di login-logout	☐	
16	2/12	Miglioramenti generali all'applicazione sviluppata in settimana	☐	
17	5/12	Inizio studio toolkit Ionic e procedure per poter effettuare la migrazione di un progetto in Angular già esistente	☐	
18	6/12	Utilizzo di Ionic per il porting su mobile dell'applicazione sviluppata settimana scorsa	☐	
19	7/12	Approfondimento studi toolkit Ionic	☐	
20	8/12		☐	
21	9/12		☐	
22	12/12	Presa visone e studio del progetto Smart Parking	☐	
23	13/12	Creazione di un prototipo dell'applicazione mobile utilizzando Figma	☐	
24	14/12	Modifiche prototipo applicazione mobile su Figma	☐	

Figura 2.2: Foglio di Google Sheets

Discord

Discord è un servizio di comunicazione vocale, video e testuale in cui è possibile creare spazi dedicati alla comunicazione di gruppo, chiamati *server*.
Sono stato inserito nel server aziendale, in modo da essere sempre aggiornato su eventuali comunicazioni di sevizio. Inoltre, l'ho usato per le comunicazioni brevi ed immediate con il gruppo di lavoro.

Figura 2.3: Discord

Google Calendar

Google Calendar è uno strumento per la gestione di calendari online.
L'ho usato per inserire le mie presenze in sede nel calendario dell'azienda e per restare aggiornato sulle disponibilità del mio gruppo di lavoro.

Figura 2.4: Google Calendar

Capitolo 3

Analisi e progettazione

In questo capitolo, inizialmente decriverò le tecnologie utilizzate durante il progetto, per poi illustrare la fase di analisi e progettazione dell'applicazione mobile Android.

3.1 Tecnologie e strumenti

Il progetto software è stato sviluppato utilizzando i linguaggi HTML, CSS, TypeScript, tramite i framework Angular, Ionic e il runtime JavaScript Node.js.
Inoltre ho utilizzato alcune funzionalità delle librerie Leaflet, RxJs e Angular Material.
Come strumenti di sviluppo ho usato Figma, Visual Studio Code, Android Studio, Git e GitHub.

Angular

Angular è un framework JavaScript open-source per la creazione di applicazioni web dinamiche tramite un'architettura modulare basata su piccole unità riutilizzabili, chiamate *Componenti*. Questo permette di strutturare al meglio l'applicazione e di avere un elevato riutilizzo e manutenibilità del codice.
Il framework è stato sviluppato con l'obiettivo di creare web app che funzionassero alla perfezione sia su dispositivi mobili che desktop, garantendo un'esperienza utente ottimale. Tiene in considerazione quindi tutte le caratteristiche di dispositivi come smartphone e tablet, che presentano prestazioni limitate e dimensioni dello schermo ridotte.
Supporta anche il *server-side rendering*, che consente di velocizzare ulteriormente il caricamento e avvio di un'applicazione.
Abbiamo scelto Angular in quanto è uno dei framework più popolari sul mercato e garantisce una base solida per la creazione di web app moderne.

Figura 3.1: Logo di Angular

Ionic

Ionic è un toolkit open-source per lo sviluppo di applicazioni ibride, cioè scritte con tecnologie web, ma che possono essere eseguite all'interno di un'applicazione nativa mobile.
Ionic offre una vasta libreria di componenti per l'interfaccia grafica, ottimizzati per mobile e può integrarsi con i maggiori framework front-end per lo sviluppo di applicazioni web, come ad esempio Angular.
Tramite la tecnologia *Capacitor*, sviluppata dallo stesso team di Ionic, consente di generare applicazioni native Android o iOS che incorporano l'applicazione web.
Abbiamo scelto Ionic in quanto ci permette di utilizzare Angular, sfruttando quindi tutti i suoi vantaggi, per lo sviluppo di un'unica applicazione mobile ibrida che esegue nativamente su dispositivi Android.

Figura 3.2: Logo di Ionic

TypeScript

TypeScript è un linguaggio di programmazione open-source che estende il linguaggio JavaScript aggiungendo il supporto per il controllo statico dei tipi. Questo risolve i problemi della tipizzazione dinamica debole del linguaggio JavaScript, che può causare errori difficili da analizzare a runtime.
Per poter eseguire, il codice TypeScript viene tradotto da un *transpiler* in codice JavaScript standard. I file JavaScript quindi possono essere eseguiti senza alcun problema all'interno di applicazioni TypeScript.
TypeScript è inoltre il linguaggio consigliato per lo sviluppo di applicazioni in Angular, quindi l'ho usato per sviluppare la parte logica dell'applicazione.

Figura 3.3: Logo di TypeScript

Node.js

Node.js è un runtime JavaScript gratuito ed open-source costruito sul motore V8 del browser web Google Chrome.
Node.js permette l'esecuzione di applicazioni JavaScript direttamente sul server, senza necessitare di un browser web. Questo consente agli sviluppatori di effettuare l'hosting dell'applicazione web localmente, senza la necessità di imparare nuovi linguaggi di programmazione lato server.
Viene utilizzato da Angular per la gestione delle dipendenze e premette l'installazione del tool da riga di comando *Angular CLI*, che dispone di varie funzioni che facilitano la gestione del progetto software, come ad esempio effettuare la build e creare un template per un nuovo componente.

Figura 3.4: Logo di Node.js

Figma

Figma è uno strumento gratuito per il design di interfacce utente.
Dispone di molte funzionalità, tra cui la generazione di un prototipo in cui si può avere un riscontro immediato sul cambiamento delle viste, causato dall'interazione dell'utente.
Abbiamo scelto Figma in quanto offre funzionalità professionali e gratuite per il design dell'applicazione mobile Android.

Figura 3.5: Logo di Figma

Visual Studio Code

Visual Studio Code è un IDE (Integrated Development Environment), ovvero un software per lo sviluppo di applicazioni software.
Ho scelto Visual Studio Code perché dispone di un vasto catalogo di estensioni esterne installabili e specifiche per qualsiasi tipo di applicazione, framework o linguaggio di programmazione.

Figura 3.6: Logo di Visual Studio Code

Android Studio

Android Studio è un IDE specifico per lo sviluppo di applicazioni mobile Android. L'ho utilizzato per effettuare il testing dell'applicazione nativa tramite la sua funzionalità di emulazione di un sistema Android.

Figura 3.7: Logo di Android Studio

Git

Git è un DVCS (Distributed Version Control System), ovvero un software per il versionamento del codice sorgente e della documentazione, organizzati all'interno di una *repository*.
Abbiamo scelto Git in quanto è il più popolare strumento di questo tipo ed è diventato ormai uno standard.

Figura 3.8: Logo di Git

GitHub

GitHub è una piattaforma web e cloud-based che permette di condividere facilmente il codice sorgente e la documentazione di un progetto software, utilizza Git come tecnologia di base.
Abbiamo scelto GitHub in quanto è il più popolare strumento di collaborazione che utilizza Git.

Figura 3.9: Logo di GitHub

HTML

HTML (Hyper Text Markup Language) è un linguaggio di markup utilizzato per descrivere la struttura di una pagina web.
Il web browser interpreta il codice scritto in HTML e produce una pagina web che viene presentata all'utente.

Figura 3.10: Logo HTML5

CSS

CSS (Cascading Style Sheets) è un linguaggio utilizzato per definire lo stile grafico di una pagina web, descrive quindi al browser come gli elementi HTML devono essere presentati.

Figura 3.11: Logo CSS3

Leaflet

Leaflet è una libreria JavaScript open-source per la creazione di mappe interattive che si adattano anche ai dispositivi mobile.
Dispone di molti plugin che ne estendono le funzionalità.
Abbiamo scelto Leaflet perché è la libreria più popolare e completa per la creazione di una mappa satellitare interattiva.

Figura 3.12: Logo di Leaflet

Angular Material

Angular Material è una libreria di componenti UI per Angular.
Ho scelto Angular Material in quanto presenta una vasta scelta ed è facilmente integrabile all'interno di applicazioni Angular.

Figura 3.13: Logo Angular Material

RxJs

RxJs è una libreria JavaScript per la programmazione asincrona.
Ho scelto di utilizzarla per gestire le parti di codice asincrone, data la sua popolarità di utilizzo.

Figura 3.14: Logo di RxJs

3.2 Architettura di Angular

Un'applicazione Angular è composta dai seguenti elementi principali:

Moduli

Un modulo in Angular racchiude un'insieme di componenti e ne fornisce un contesto di compilazione.
Ogni applicazione Angular possiede un modulo di root, da cui viene effettuata la procedura di *bootstrap* che consente l'avvio dell'applicazione stessa. Questo modulo speciale viene chiamato convenzionalmente *AppModule*.
Inoltre è possibile creare altri moduli funzionali, che gestiscono parti diverse dell'applicativo software. Questo tipo di approccio aiuta nello sviluppo di applicazioni complesse ed è orientato alla riusabilità del codice e favorisce il cosiddetto *lazy-loading*, che minimizza i tempi di caricamento all'avvio della web app.

Componenti

Un componente in Angular è semplicemente una classe TypeScript che contiene la logica e i dati dell'applicazione. Ogni componente è associato ad un template HTML,

che definisce una vista, ovvero un'insieme di elementi che verranno visualizzati nello schermo. In questi template è possibile utilizzare un linguaggio di markup di Angular, che combinato assieme al codice HTML, permette di generare dinamicamente i componenti del DOM (Document Object Model).
In particolare, è possibile usare le direttive per introdurre costrutti logici e il data binding per "connettere" i dati dell'applicazione al DOM.
Ogni applicazione Angular possiede almeno un componente di root, che viene utilizzato come base dell'applicazione, chiamato convenzionalmente *AppComponent*.

Servizi

Un servizio in Angular è una classe TypeScript utilizzata per condividere, tra più componenti, dati o business logic, non collegati direttamente alle viste.
Ogni servizio può quindi essere iniettato come dipendenza (*Dependency Injection*), all'interno di uno o più componenti. Questo rende il codice dei componenti più leggero ed efficiente, in quanto non dovrà occuparsi di effettuare il fetch dei dati e i controlli sull'input, ma lo delegherà ai servizi.

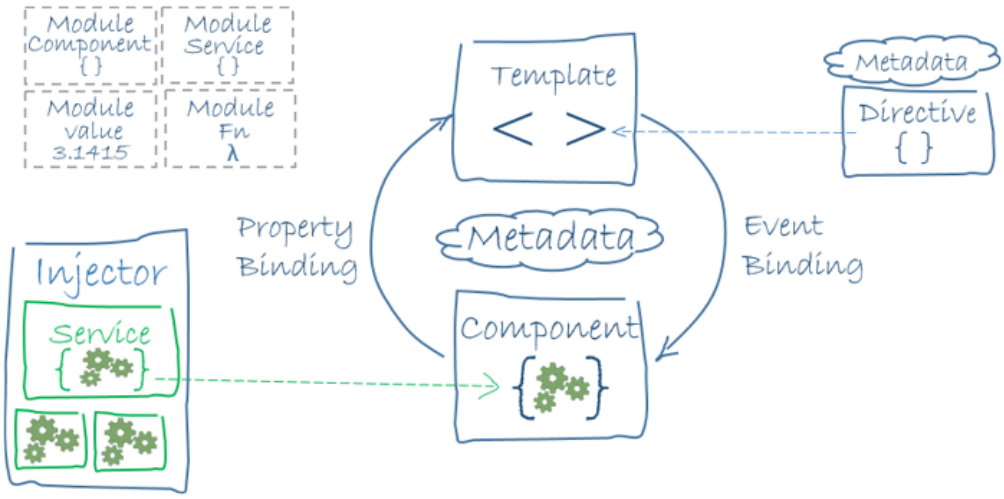


Figura 3.15: Architettura di Angular

3.3 Architettura dell'applicazione

L'architettura dell'applicazione mobile segue gli standard dati da Angular e Ionic. In Ionic esiste inoltre il concetto di pagina, ovvero un componente che agisce come un'intera vista e che ha un suo modulo dedicato. Una pagina può avere anche altri componenti annidati.

Ho deciso di strutturare l'applicazione in pagine: ogni pagina deve essere quindi definita all'interno del suo modulo specifico, così da implementare il *lazy loading*

delle parti che la compongono.
Se un componente viene utilizzato solamente in una pagina, allora deve essere definito all'interno dello stesso modulo della pagina di cui fa parte.
Se, invece, viene riutilizzato da più pagine, allora il componente deve essere definito all'interno di un modulo separato.

I file di template HTML e i fogli di stile CSS vengono collocati all'interno della stessa cartella del componente che li utilizza. I file dei servizi, invece, vengono collocati al di fuori, in una cartella apposita, in quanto possono essere utilizzati da più componenti.

La parte back-end dell'applicazione non viene utilizzata per questo progetto, quindi i dati relativi ai sensori di parcheggio, che vengono utilizzati dall'app, sono letti da un file json contenente dei valori che simulano un sistema di back-end reale.

3.4 Analisi funzionalità web app esistente

Prima di procedere con la progettazione dell'applicazione mobile Android, ho scelto di fare un'analisi delle funzionalità della web app già esistente, sviluppata per lo stesso macro-progetto software. In questo modo ho potuto capire al meglio le funzionalità da implementare per effettuare il porting su mobile.

Il sito web si presenta con il logo e una breve descrizione del progetto, segue una mappa satellitare, con cui l'utente può interagire zoomando avanti e indietro nella zona d'interesse.
Sotto alla mappa è invece presente la lista dei sensori di parcheggio installati, raggruppati per indirizzo. La posizione dei sensori viene anche indicata nella mappa stessa, con dei marcatori di colore diverso in base alla disponibilità del parcheggio. Cliccando su uno dei marcatori nella mappa o su una voce della lista dei sensori, l'utente visualizzerà con più precisione il punto in cui è situato il parcheggio selezionato e verrà aperto un pop-up, sopra al marcatore, che presenterà ulteriori informazioni, tra cui l'id del parcheggio.

Segue poi la lista dei sensori ambientali, che però non necessitano di essere riportati nell'applicazione mobile, in quanto lo scopo è quello di avere un'app dedicata solamente alla gestione di un sistema di parcheggi.
Sulla parte finale del sito possiamo notare i nomi degli sviluppatori del progetto software.

Da quest'analisi emergono quindi due parti principali della web app, ovvero la mappa satellitare e la lista dei sensori di parcheggio. Ho deciso quindi che queste sarebbero state le funzionalità di cui verrà effettuato il porting su mobile.

La figura 3.16 mostra come si presenta l'applicazione web già esistente per il progetto "Smart City".

Figura 3.16: Sito web progetto Smart City

3.5 Design applicazione mobile

Dopo aver analizzato le funzionalità richieste per lo sviluppo dell'applicazione mobile, ho deciso di creare un mock-up di quelle che sarebbero potute essere le varie pagine, così da poter avere un riscontro da parte del capo progetto, prima di

scrivere il codice effettivo.

La figura 3.17 mostra l'idea per la pagina di login, la quale servirà soltanto ad accedere alle funzionalità dell'applicativo software, senza una vera e propria procedura di autenticazione da parte dell'utente.

Figura 3.17: Mock-up pagina di login

Per la parte centrale dell'applicazione ho deciso di seguire le linee guida standard per il design di un'applicazione mobile a più pagine, dividendo ogni pagina in 3 parti:

* Header: parte superiore contenente il nome e il logo dell'applicazione.
* Contenuto: parte centrale in cui risiede il contenuto vero e proprio della pagina.
* Barra di navigazione: parte inferiore che contiene la lista delle pagine disponibili.

L'header è l'unica parte che rimane sempre uguale per ogni pagina, mentre il contenuto e la barra inferiore cambiano in base alla pagina selezionata dall'utente.

Ho deciso dunque di separare le funzionalità di visualizzazione della mappa e della lista dei sensori in due pagine diverse, accessibili "toccando" l'icona e scritta corrispondenti dalla barra in basso.

Le figure 3.18 e 3.19 mostrano i prototipi di design che ho realizzato per le due pagine.

Figura 3.18: Mock-up pagina: lista dei sensori

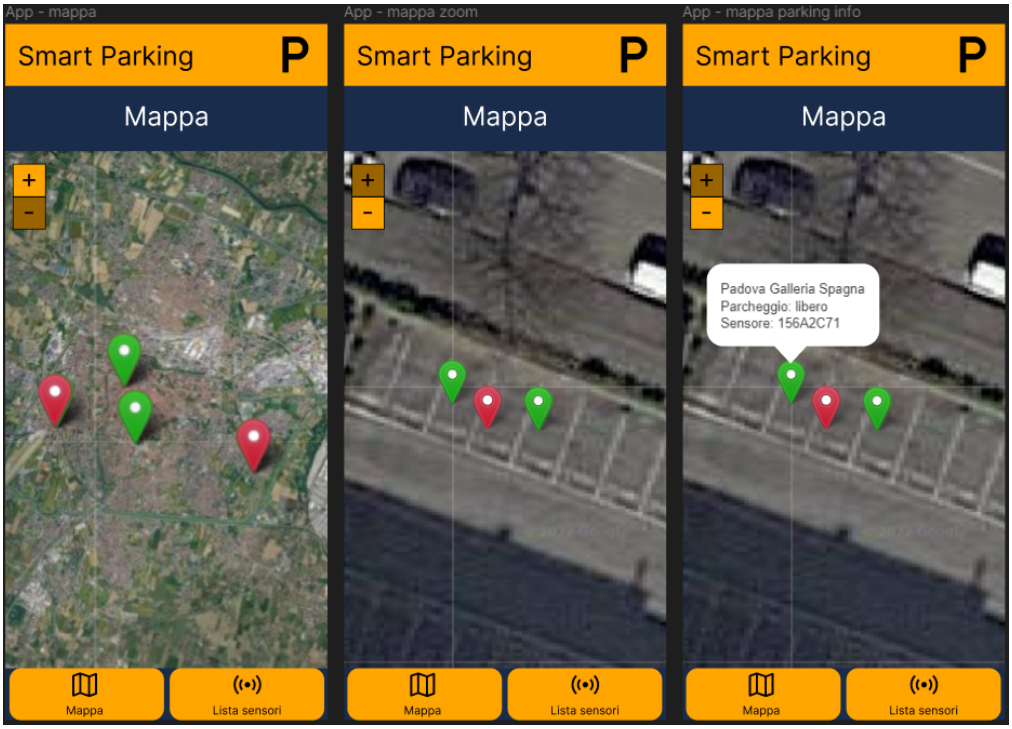


Figura 3.19: Mock-up pagina: mappa satellitare

Lo strumento che ho utilizzato per creare i mock-up, ovvero Figma, permette di visualizzare una demo dell'applicazione con la possibilità di interazione da parte

dell'utente, riuscendo così ad avere un feedback immediato sui cambiamenti delle viste, senza però averle ancora implementate.

Capitolo 4

Codifica

In questo capitolo inizialmente illustrerò la fase di analisi della portabilità del codice della web app già esistente, per poi descrivere l'implementazione dell'applicazione mobile Android.

4.1 Analisi portabilità del codice della web app già esistente

Prima di procedere con l'implementazione delle funzionalità dell'applicazione mobile, abbiamo deciso di effettuare un'analisi della portabilità del codice della web app già esistente. In questo modo si sfruttano i vantaggi della riusabilità del codice, in quanto non lo si deve completamente riscrivere, ma lo si adatta al diverso ambiente di sviluppo.
Inoltre le tecnologie che abbiamo scelto di utilizzare per la realizzazione dell'applicazione mobile Android facilitano il porting, dato che hanno molti elementi in comune con quelle utilizzate per sviluppare la parte front-end dell'applicazione web.

Ho preso in esame il codice che gestisce le funzionalità di interazione con la mappa satellitare e di visualizzazione della lista dei sensori di parcheggio.

Da quest'analisi è emerso che la parte di software che gestisce le funzionalità della mappa satellitare è completamente riadattabile ed utilizzabile, seppur con qualche modifica minore.
Anche la parte che si occupa della visualizzazione della lista dei sensori di parcheggio potrebbe essere facilmente portabile, ma ho deciso invece di riscriverla completamente, in modo da utilizzare gli strumenti di UI forniti dal toolkit Ionic, che sono ottimizzati per i dispositivi mobile.

4.2 Design Pattern e tecniche utilizzati

Singleton

Design pattern creazionale che assicura di utilizzare una sola istanza di una determinata classe.
Viene utilizzato da Angular per assicurare che venga creata una sola istanza dei servizi forniti alla radice dell'applicazione.
Inoltre, grazie a questo design pattern, un servizio può cambiare il proprio stato in seguito ad una richiesta da parte di un componente e condividere questo cambiamento con gli altri componenti che lo utilizzano.

La figura 4.1 mostra il diagramma UML che identifica la struttura del design pattern Singleton.

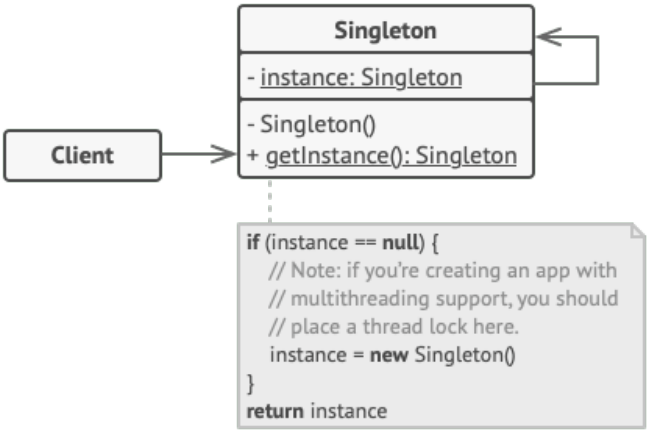


Figura 4.1: Singleton design pattern

Observer

Design pattern comportamentale che permette ad alcuni oggetti di essere notificati appena avviene un cambiamento allo stato di un altro oggetto, che stanno osservando.
L'ho utilizzato per gestire le chiamate HTTP verso il back-end, che possono avere tempi di risposta variabili, tramite i concetti di *Observable*, *Subject* e *Subscription* della libreria JavaScript RxJs.

La figura 4.2 mostra il diagramma UML che identifica la struttura del design pattern Observer.

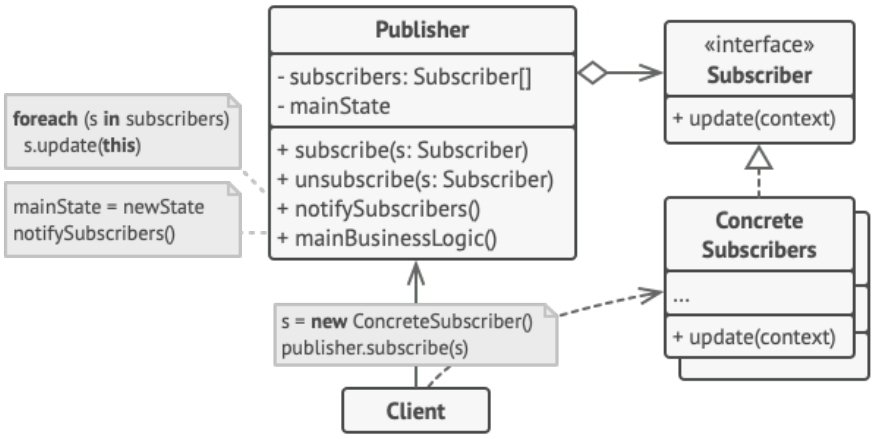


Figura 4.2: Observer design pattern

Dependency Injection

Linea guida importante per lo sviluppo di applicazioni a larga scala.
Angular la utilizza per realizzare la scalabilità e l'efficienza delle applicazioni.
In ogni componente, nel costruttore vengono dichiarate le dipendenze, tipicamente servizi, che vengono fornite, in fase di inizializzazione, da un sistema esterno tramite un *injector*.

Lazy Loading

Tecnica di ottimizzazione, utilizzata principalmente da applicazioni web, che consiste nel caricare solamente le parti dell'applicazione richieste dall'utente, posticipando quindi il caricamento delle rimanenti a quando sarà necessario. In questo modo vengono ridotti i tempi di caricamento all'avvio della web app.
L'ho utilizzata con Angular per gestire il caricamento delle varie pagine e componenti presenti nell'applicazione mobile.

4.3 Implementazione

L'applicazione è composta da 3 pagine diverse: login, mappa e lista sensori.

4.3.1 Pagine

Pagina di login

La pagina di login è la prima che viene visualizzata all'avvio dell'applicazione.
Non viene implementata una funzionalità di login vera e propria, in quanto questa

pagina ha il solo scopo di dare il benvenuto all'utente con il titolo dell'applicazione e il logo dell'azienda che l'ha sviluppata.
Presenta un pulsante al centro che permette all'utente di accedere a tutte le funzionalità dell'applicazione.

La figura 4.3 mostra la pagina di login in un dispositivo Android.

Figura 4.3: Pagina di login

Per realizzare la pagina di login è stato utilizzato il seguente componente:

* LoginPage

Pagina mappa satellitare

Questa pagina permette di visualizzare lo stato dei sensori all'interno di una mappa satellitare di Padova.
Viene visualizzata non appena l'utente ha effettuato il login e anche cliccando la voce a sinistra della barra di navigazione in basso, che contiene un'icona di una mappa con la scritta "Mappa" posta al di sotto di essa.
L'utente può effettuare lo zoom in avanti o indietro sulla mappa, in modo da visualizzare l'area d'interesse più nel dettaglio. Cliccando su un marcatore, vengono visualizzate le informazioni relative al sensore di parcheggio associato: la via, la disponibilità e l'id del sensore.

La figura 4.4 mostra la pagina relativa alla mappa satellitare in un dispositivo Android.

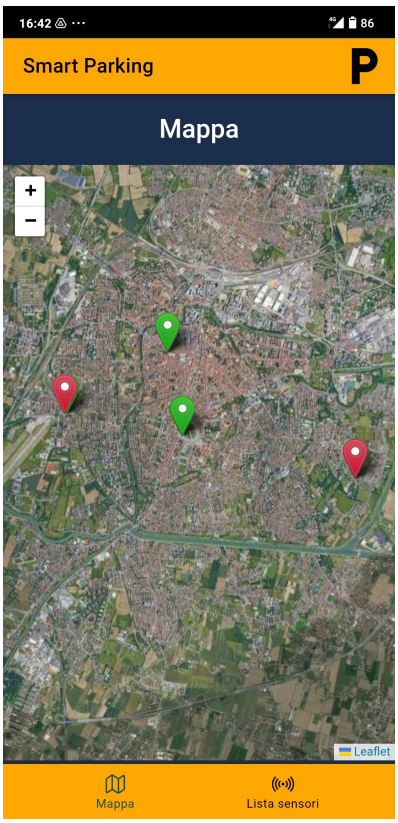


Figura 4.4: Pagina mappa satellitare

Per realizzare la pagina della mappa sono stati utilizzati i seguenti componenti e servizi:

* TabsPage
* MapPage
* HeaderBarComponent
* SensorService

Pagina lista dei sensori di parcheggio

Questa pagina permette di visualizzare la lista dei sensori presenti.
Viene presentata quando l'utente clicca la voce a destra della barra di navigazione in basso, la quale contiene un'icona di un sensore con la scritta "Lista sensori" posta al di sotto di essa.
La pagina presenta una lista di località di Padova, che l'utente può espandere cliccando su una qualunque voce, in modo da mostrare l'id e lo stato di tutti i sensori di parcheggio installati in quella località. Cliccando su un sensore di parcheggio, inoltre, si viene portati nella pagina della mappa satellitare, in cui viene

mostrata la posizione esatta e ulteriori dettagli del sensore.

La figura 4.5 mostra la pagina relativa alla lista dei sensori in un dispositivo Android.

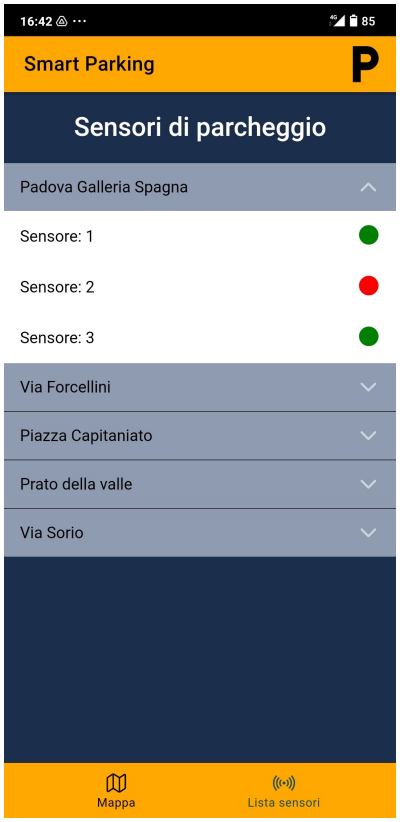


Figura 4.5: Pagina lista dei sensori di parcheggio

Per realizzare la pagina con la lista dei sensori sono stati utilizzati i seguenti componenti e servizi:

* TabsPage
* SensorPage
* HeaderBarComponent
* SensorService

4.3.2 Componenti

AppComponent

Componente di root che rappresenta l'intera applicazione.
Non dispone di metodi in quanto viene utilizzato solamente come base per il routing dell'app, ovvero è un componente che serve per la gestione della visualizzazione delle varie pagine, mostrando sempre quella selezionata dall'utente.

LoginPage

Componente che rappresenta l'intera pagina di login.
Non ha metodi in quanto è una pagina statica, che ha il solo scopo di permettere l'accesso alle funzionalità dell'applicazione.
Quando l'utente clicca il pulsante "Entra", viene sfruttato il routing di Ionic, il quale cambia la pagina visualizzata mostrando la pagina contenente la mappa satellitare.

HeaderBarComponent

Componente che rappresenta l'header per la pagina della mappa satellitare.
Non dispone di metodi in quanto è un componente statico che contiene il titolo dell'applicazione a sinistra e un'icona di parcheggio a destra.

TabsPage

Componente padre che gestisce le pagine: mappa satellitare e lista sensori.
Viene usato per creare una falsa pagina che ha il solo scopo di visualizzare una o l'altra pagina in base all'interazione dell'utente con la barra inferiore. Di default viene visualizzata la pagina della mappa satellitare.
Questa configurazione è disponibile in automatico creando un progetto Ionic con il template "tabs".
TabsPage non presenta metodi, in quanto serve solamente ad utilizzare il routing per visualizzare le pagine figlie e passare facilmente da una all'altra.

MapPage

Componente che rappresenta il contenuto della pagina della mappa satellitare.
Viene usato per la gestione della mappa satellitare e delle interazioni da parte dell'utente con essa.
Dispone dei seguenti metodi:

* **ngOnInit**: viene chiamato all'inizializzazione del componente e si occupa di creare la mappa ed effettuare il fetch dei dati relativi ai sensori di parcheggio mediante il servizio SensorService.

* **getInfo**: metodo che crea il testo con le informazioni relative ad un sensore di parcheggio dato in input.

* **showMarker**: metodo che fa aprire, nella mappa, il pop-up del marcatore associato al sensore dato in input.

SensorPage

Componente che rappresenta il contenuto della pagina della lista dei sensori di parcheggio.

Viene usato per la gestione della lista dei sensori e delle interazioni da parte dell'utente con essa.
Dispone dei seguenti metodi:

* **ngOnInit**: viene chiamato all'inizializzazione del componente e si occupa di creare la lista dei sensori, utilizzando il servizio SensorService.

* **groupByAddress**: metodo che raggruppa per indirizzo la lista dei sensori data in input.

* **showMarker**: metodo che cambia pagina visualizzata, portando l'utente alla pagina della mappa e fa aprire il pop-up del marcatore associato al sensore dato in input.

4.3.3 Servizi

SensorService

Servizio che viene utilizzato per gestire i dati dei sensori di parcheggio.
Dispone dei seguenti metodi:

* **emitChange**: metodo usato per far comunicare la pagina della mappa con la pagina dei sensori, viene chiamato quando un'azione in una delle due pagine deve portare un risultato anche sull'altra pagina. Per fare ciò viene utilizzato un Observable: una pagina emette il valore e l'altra resta in ascolto, eseguendo un'azione non appena lo riceve.

* **getAllParkingAreas**: metodo che effettua una chiamata http GET al back-end dell'applicazione per ricevere i dati dei sensori di parcheggio. Per questo progetto il back-end consiste in un file .json contenente alcuni dati statici, ma comunque realistici.

4.4 Problemi riscontrati

4.4.1 Colori sbagliati con modalità scura

In fase di testing dell'applicazione mi sono accorto che in alcuni dispositivi mobile i colori dell'app erano sbagliati.
Dopo un'analisi più attenta ho capito che questo era dovuto alla presenza della modalità scura nei dispositivi mobile in questione.
Per risolvere il problema abbiamo deciso di disabilitare l'uso della modalità scura da parte della nostra applicazione all'interno dei dispositivi che evidenziano questo problema.

4.4.2 Visualizzazione mappa satellitare

In fase di sviluppo sono stati riscontrati alcuni problemi di visualizzazione della mappa satellitare nei dispositivi mobile.

Il primo problema consisteva nell'impossibilità di riuscire a caricare la mappa satellitare, lasciando così una schermata vuota.
Dopo aver effettuato alcuni test, ho capito che si trattava di un problema di sicurezza. Ho dovuto quindi cambiare la fonte da cui venivano prese le immagini della mappa satellitare, in quanto utilizzava il protocollo HTTP, che viene bloccato di default dai sistemi Android perché ritenuto poco sicuro, in una fonte che utilizza il più sicuro protocollo HTTPS.

In seguito è stato riscontrato un problema di caricamento parziale della mappa satellitare: l'applicazione non riusciva a caricare alcune aree.
Dopo un'analisi più accurata e vari test, ho capito che il problema era dovuto ad non completo caricamento delle risorse necessarie alla corretta creazione della mappa.
Per risolvere il problema ho utilizzato la funzione *setTimeout* di RxJs, in modo da bloccare l'esecuzione del codice per un piccolo periodo di tempo e permettere così il corretto caricamento della mappa satellitare.

4.5 Validazione e collaudo

Durante il periodo di stage, una volta a settimana, ho partecipato alla riunione di allineamento con tutti i membri che hanno seguito il progetto. Durante queste riunioni ho presentato i cambiamenti più significativi e le problematiche riscontrate. In questo modo ho potuto intervenire in tempi brevi, qualora fossero state richieste alcune modifiche al codice da parte del tutor aziendale.

Nell'ultima settimana di stage ho presentato l'applicazione mobile Android finale al tutor aziendale, che ne ha collaudato tutte le funzionalità. Ricevuta la sua approvazione finale, il codice del progetto è stato consegnato in una chiavetta USB, in modo da poter essere installato facilmente in una macchina aziendale.

Capitolo 5

Conclusioni

In questo capitolo verrà riportato il resoconto sugli obiettivi raggiunti e le conoscenze acquisite; infine vi sarà una valutazione personale dello stage svolto.

5.1 Raggiungimento degli obiettivi

Lo stage si è svolto rispettando tutti gli obiettivi obbligatori, desiderabili e facoltativi previsti dal piano di lavoro e descritti nella sezione 2.1.

La tabella 5.1 riassume tali obiettivi e ne descrive lo stato.

ID Obiettivo	Descrizione	Stato
O01	Acquisizione delle competenze necessarie allo sviluppo del progetto	Soddisfatto
O02	Capacità di raggiungere gli obiettivi richiesti in autonomia seguendo il cronoprogramma	Soddisfatto
O03	Portare a termine le implementazioni previste con una percentuale di superamento pari all' 80%	Soddisfatto
D01	Portare a termine le implementazioni previste con una percentuale di superamento pari al 100%	Soddisfatto
F01	Apportare un valore aggiunto al gruppo di lavoro durante le fasi di progettazione delle interfacce	Soddisfatto

Tabella 5.1: Resoconto raggiungimento obiettivi

5.2 Conoscenze acquisite

Il progetto di stage mi ha permesso di maturare alcune conoscenze di base nell'ambito dello sviluppo web e mobile, tramite l'utilizzo di framework popolari e con molte potenzialità come Angular e Ionic.
Ora sono in grado di costruire un'applicazione web completa e moderna, combinando le conoscenze di back-end, ottenute nello sviluppo del progetto del corso di Ingegneria del Software, con le conoscenze di front-end acquisite da quest'esperienza di stage. Inoltre, sono in grado di riuscire ad adattare facilmente tale applicazione ai dispositivi mobile.

Ho imparato a consultare le risorse e la documentazione, per gli strumenti software che ho utilizzato, in maniera efficiente ed efficace, in modo da riuscire a risolvere, in tempi brevi, i problemi che si sono presentati.

In questi due mesi di stage ho acquisito competenze su alcune dinamiche aziendali, come ad esempio le scadenze da rispettare e la pianificazione del lavoro, con relativi controlli settimanali sulla qualità del prodotto.
Ho potuto assistere ad alcune riunioni organizzative, svolte dal capo progetto, di progetti software a larga scala. Questo mi ha permesso di avere una panoramica ancora migliore delle dinamiche lavorative in un'azienda informatica al giorno d'oggi.

5.3 Valutazione personale

Quest'esperienza di stage è stata positiva, in quanto mi ha permesso di entrare in un ambiente di lavoro reale e competitivo, che ha messo alla prova le mie conoscenze e capacità, evidenziando i miei punti di forza, ma anche eventuali mancanze o limiti. Questo mi ha portato ad avere una grande crescita personale e professionale, dandomi le basi su cui lavorare in futuro per affrontare al meglio le situazioni lavorative che si presenteranno.
Inoltre mi ha permesso di crescere nella gestione dei rapporti, lavorativi e non, con le altre persone presenti nell'azienda.

Durante lo sviluppo dell'applicazione, il tutor esterno mi ha lasciato una completa autonomia, pur mettendosi sempre a disposizione, e questo mi ha consentito di valutare ed effettuare alcune scelte implementative, che mi hanno portato, quindi, ad avere una forte responsabilità sulla buona riuscita del progetto.

L'ambiente lavorativo è sempre stato adeguato e volto all'apprendimento.
In particolare, nel corso degli incontri settimanali, il tutor esterno ha sempre ascoltato e sostenuto le mie idee, dandomi qualche consiglio sul materiale formativo da consultare per implementarle al meglio.

Le tecnologie utilizzate sono risultate molto interessanti e di piacevole apprendimento, in quanto dispongono di un'ottima documentazione e varie risorse da consultare. Sono contento di essere riuscito ad ottenere una buona familiarità, seppur ancora di base, con esse.

Infine, ritengo che in generale quest'esperienza sia di fondamentale importanza, perché mette alla luce le conoscenze acquisite durante il corso di studi e permette di applicarle in un contesto lavorativo reale, preparando così lo studente a quello che potrebbe essere il suo futuro professionale.

Bibliografia

Siti web consultati

CSS Introduction. URL: https://www.w3schools.com/css/css_intro.asp.

HTML Introduction. URL: https://www.w3schools.com/html/html_intro.asp.

Introduction to Angular concepts. URL: https://angular.io/guide/architecture.

Introduction to Node.js. URL: https://nodejs.dev/en/learn/.

Introduzione a TypeScript. URL: https://www.mrw.it/javascript/introduzione-typescript_12482.html.

Introduzione ad Angular. URL: https://www.mrwebmaster.it/javascript/introduzione-angular_12716.html.

Lazy Loading in Angular. URL: https://angular.io/guide/lazy-loading-ngmodules.

Observer design pattern. URL: https://refactoring.guru/design-patterns/observer.

Singleton design pattern. URL: https://refactoring.guru/design-patterns/singleton.

Sito ufficiale Android Studio. URL: https://developer.android.com/studio.

Sito ufficiale Angular. URL: https://angular.io.

Sito ufficiale Angular Material. URL: https://material.angular.io.

Sito ufficiale GitHub. URL: https://github.com.

Sito ufficiale Ionic. URL: https://ionicframework.com.

Sito ufficiale Leaflet. URL: https://leafletjs.com.

Sito ufficiale Node.js. URL: https://nodejs.org/en/.

Sito ufficiale RxJs. URL: https://rxjs.dev.

Sito ufficiale Sync Lab s.r.l. URL: https://www.synclab.it.

Sito ufficiale Visual Studio Code. URL: https://code.visualstudio.com.

Understanding dependency injection. URL: https://angular.io/guide/dependency-injection.