Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica

Sviluppo web di una sezione per aziende del portale di landing page per prodotti

Tesi di laurea

Relatore

Prof.ssa Ombretta Gaggi

Laureando

Noemi Liva

1237380

Anno Accademico 2022-2023

Dedicata a papà Domenico,
perchè in ogni mio gesto c'è una parte di te.

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, dalla laureanda Liva Noemi presso l'azienda S.AI Synthema Artificial Intelligence.
Il tirocinio si è svolto durante i mesi di maggio, giugno e luglio 2023 e ha avuto la durata di 320 ore. L'obiettivo dello stage è stato lo sviluppo web di una sezione dedicata alle aziende produttrici su un portale di landing page per prodotti.
L'elaborato ha lo scopo di illustrare:

* Il contesto aziendale dove è stato svolto lo stage e un'introduzione al progetto (Capitolo 1);
* La descrizione del progetto sviluppato in ogni suo aspetto (Capitolo 2);
* Gli strumenti e le tecnologie utilizzate (Capitolo 3);
* I dettagli per quanto riguarda la fase di progettazione e codifica, e quella di verifica e validazione(Capitolo 4 e 5);
* Una valutazione finale sull'esperienza e le competenze acquisite (Capitolo 6).

Ringraziamenti

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

A mio fratello,
che pur non sapendo neanche che cosa faccio, non ha mai dubitato che ci sarei riuscita.

A mia madre,
per avermi ricordato che il mio valore è completamente indipendente da ogni valutazione, per avermi convinto a non mollare e per guidarmi ogni giorno verso la felicità.

A Mary,
che crede in me più di me stessa e mi è rimasta accanto quando nessun altro ne sarebbe stato in grado.

A Susanna,
alla certezza di avere una persona accanto qualunque cosa succeda, al bene incondizionato, alle risate spontanee e alla profondità d'animo.

A Luca,
che attraverso il suo amore è riuscito a farmi amare me stessa, che mi ha preso per mano è mi ha mostrato che bella che può essere la vita.

A tutti i miei amici,
per aver acceso un filo di luce quando da sola non ne sarei stata in grado.

Padova, Settembre 2023 Noemi Liva

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Questo capitolo descrive la realtà aziendale e il suo ambito di interesse.

1.1 L'azienda

Synthema Artificial Intelligence (S.AI)[1] è una *start-up* innovativa che si occupa di ricerca, progettazione, sviluppo, commercializzazione e manutenzione di prodotti e servizi ad alto valore tecnologico.
Questi servizi sono basati sull'*Internet of Things* e su tecniche di Intelligenza Artificiale (IA)[g]per l'analisi integrata e la comprensione di dati multimodali da fonti eterogenee quali il linguaggio naturale sia scritto che parlato, l'audio, le immagini, i video e i dati generati da sensori. Inoltre le IA si occupano anche della gestione dei *workflow*, sia nel settore pubblico che privato.
S.AI ha ereditato l'esperienza decennale di SyNTHEMA srl in progetti di ricerca europei nelle aree del web crawling[g], Natural Language Processing (NLP)[g](sia scritto che parlato), IA e *machine learning* e tecnologie semantiche.

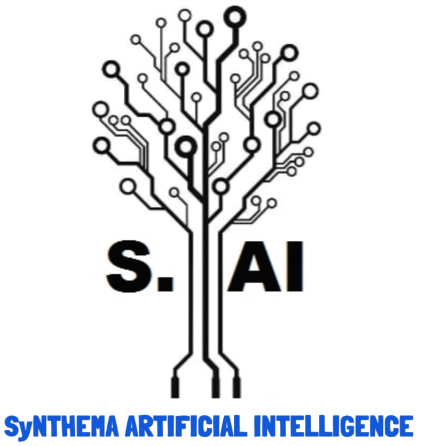

Figura 1.1: Logo synthema artificial intelligience

[1] *S-AI: Synthema Artificial Intelligence.* URL: https://www.s-ai.it/.

1.1.1 Ricerca e sviluppo

S.AI è attualmente impegnata nella preparazione di proposte di progetto nelle aree dell'*emotion recognition* e *cognitive computing*. Il team di ricerca di S.AI ha partecipato come WP leader in diversi progetti europei, tra i quali:

* KYOTO *(Knowledge Yielding Ontologies for Transition-based Organization)*;
* MOSAIC *(Multi-Modal Situation Assessment & Analitics Platform)*;
* CAPER *(Collaborative information, Acquisition, Processing, Exploitation and Reporting for the prevention of organised crime)*;
* SAVAS *(Sharing Audio Visual language resources for Automatic Subtitling)*.

1.2 Panoramica dello stage

L'azienda S.AI si occupa di sviluppare software e tecnologie legate al mondo del processamento del linguaggio naturale (NLP).
Per uno dei progetti che l'azienda porta avanti con BPK, azienda che opera nel settore del *packaging*, è stato sviluppato il prototipo di portale informativo sui prodotti alimentari e sul loro imballo. Mediante la scansione di un *QR Code* o di un codice a barre o attraverso la digitazione di un codice presente sulla confezione di un prodotto, l'utente della piattaforma può raggiungere una pagina contenente informazioni aggiuntive sui prodotti quali la descrizione, le specifiche e le recensioni.
Per lo stesso progetto si è reso necessario lo sviluppo di una sezione del portale dedicata alle aziende produttrici, per permettere di aggiungere informazioni sui loro prodotti e sui marchi e di accedere ad analisi e statistiche relative ai loro prodotti.
Lo stage si è basato sull'inserimento della studentessa nel team di ricerca e sviluppo che si occupava di questo progetto. Sono stata formata nel campo dello sviluppo web, utilizzando i framework React e Redux.
Durante lo stage è stato realizzato un prototipo della sezione dedicata alle aziende della piattaforma, che serve come base per il suo sviluppo futuro.

I principali argomenti trattati sono stati:

* realizzazione di un'applicazione web;
* tecnologie per lo sviluppo software (git, continuos integration, ...);
* realizzazione di componenti React;
* gestione dello stato con Redux.

1.3 Organizzazione del testo

Il documento sarà sviluppato nel seguente modo:

* Il primo capitolo effettua una breve introduzione all'azienda e al lavoro effettuato durante lo stage curricolare presso Sync Lab.
* Il secondo capitolo descrive il progetto svolto nella sua interezza, con i suoi requisiti e i suoi vincoli.
* Il terzo capitolo descrive tutti gli strumenti e le tecnologie utilizzate per questo progetto.
* Il quarto capitolo approfondisce la fase di progettazione e la codifica del prodotto.
* Il quinto capitolo approfondisce la fase di verifica e collaudo spiegando l'approccio usato durante i test.
* Il sesto capitolo contiene un'analisi del lavoro svolto e le conclusioni tratte.

1.3.1 Convenzioni tipografiche

Per quanto riguarda 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 vengono definiti nel glossario, situato alla fine del presente documento;
* per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: termine[g];
* i termini in lingua straniera o facenti parte del gergo tecnico sono evidenziati con il carattere corsivo.

Capitolo 2

Il Progetto

Il capitolo descrive l'applicativo, gli obbiettivi e i vincoli per lo sviluppo.

2.1 L'applicativo

Al momento dell'inizio dello stage, l'applicativo illustrato al paragrafo 1.2 era in grado di scansionare un prodotto e mostrare le sue informazoni basilari, quali ingredienti e valori nutrizionali.
Per agevolare l'utilizzo era presente anche una barra di ricerca, dove è possibile inserire il nome del prodotto se si cerca qualcosa di specifico, o un ingrediente, per avere la lista di tutti i prodotti che lo contengono. Durante il tirocinio, si è voluto portare questa versione primaria di applicativo a una versione stabile e aggiornata, aggiungendo molte funzionalità utili agli utenti e anche alle aziende produttrici, come le statistiche del prodotto in base alle recensioni lasciate, o la possibilità di aggiungere informazioni su un prodotto e sul suo *packaging*, il tutto chiaramente una volta registrati e autenticati nell'applicazione.
FoodAdvisor attuale nome dell'applicazione con logo presente nell'immagine 2.1, offre tutte le informazioni di cui l'utente ha bisogno per fare scelte alimentari consapevoli. Con questo servizio, si può quindi scoprire tutto su ingredienti e *packaging* dei prodotti, nonché condividere le proprie esperienze con altri consumatori.

Figura 2.1: Logo FoodAdvisor

2.2 Obiettivi

Sulla base della durata massima di 320 ore prevista per lo stage, assieme al tutor aziendale ho stilato un piano di lavoro e coerentemente ho concordato gli obiettivi che si aspettavano di veder raggiunti al termine del rapporto lavorativo. Gli obbiettivi riguardano un prototipo di un' applicazione, che deve consentire all'utente di:

* registrarsi e autenticarsi;
* associare il proprio profilo a un'azienda;
* associare marchi ad un'azienda;
* associare prodotti ad un'azienda;
* accedere alle informazioni e alle statistiche su un prodotto;
* accedere alle informazioni e alle statistiche su un marchio;
* aggiungere informazioni (per es. informazioni commerciali e promozionali) su un prodotto;
* aggiungere informazioni (per es. informazioni commerciali e promozionali) su un marchio;

Ci si attende anche che:

* il progetto sia pubblicato in un *repository git* aziendale, rispettando gli standard aziendali per la scrittura del codice e per i *commit*;
* sia presente la documentazione per la compilazione e configurazione del progetto realizzato;
* siano presenti i test di unità delle componenti realizzate;
* sia presente la documentazione delle componenti e delle funzioni realizzate.

2.2.1 Standard aziendali

Gli standard aziendali per quanto riguarda i *commit* e le merge request sono diversi da azienda ad azienda, per quanto riguarda S.AI sono i seguenti:

* quando si scrive il codice non bisogna correggere errori che non riguardano la *issue* che si sta svolgendo, ne verrà creata una nuova di correzione, in modo tale che i commit riguardano solo la task attuale;
* per ogni task viene creato un nuovo branch su gitlab, e una volta completata verrà attuata una *merge request*;
* bisogna scrivere un *commit* per ogni modifica fatta, seguendo questa convenzione: "Nome del file: modifica";
* i singoli *commit* essendo su singoli file, possono non funzionare correttamente, basta che al momento della merge request sia tutto funzionante;
* se vengono create nuove componenti per una sola funzione possono essere racchiuse in un singolo *commit*;
* anche quando si eliminano dei file è necessario il *commit*;

2.3 Requisiti

I requisiti stipulati assieme all'azienda sono rappresentati in questa tabella:

	Obbligatori
OB01	Possibilità di creare e configurare un'azienda
OB02	Possibilità di creare e configurare prodotti e marchi dell'azienda
OB03	Possibilità di visualizzare e ricercare marchi e prodotti dell'azienda
OB04	Possibilità di visualizzare statistiche e analisi per prodotti e marchi
	Desiderabili
DE01	Possibilità di associare più utenti ad una azienda
DE02	Possibilità di caricare informazioni relative a prodotti e marchi dirette all'utente finale
DE03	Possibilità per l'utente finale di vedere le informazioni caricate dal produttore
	Opzionali
OP01	Possibilità di avere ruoli separati per la gestione delle aziende
OP02	Possibilità di avere un'interfaccia amministrativa per l'autorizzazione di alcune operazioni (per es. associare marchi e prodotti ad un'azienda)

Tabella 2.1: Tabella dei requisiti

2.4 Vincoli

L'applicativo dovrà inoltre rispettare i seguenti punti:

* la componente *back-end* dell'applicativo deve essere sviluppata in linguaggio Python utilizzando il framework[g]Django;
* la componente *front-end* dell'applicativo deve essere sviluppata utilizzando il *framework* React;
* l'interfaccia deve essere accattivante e di semplice utilizzo. Questo aspetto è di rilevante importanza, soprattutto per sistemi complessi come quello in questione;
* l'applicazione dovrà seguire le linee guida di accessibilità Web Content Accessibility Guidelines[g]2.0.

2.4.1 WCAG 2.0

Le *Web Content Accessibility Guidelines* 2.0[1] contengono regole studiate per rendere i contenuti del *web* maggiormente accessibili, ampliando l'utenza anche a persone con disabilità, tra cui la cecità e l'ipovisione, la sordità e la perdita di udito, ridotte capacità di movimento e combinazioni di queste.
Le persone ed organizzazioni che utilizzano le WCAG possono essere molto diverse tra loro e comprendono *web designer* e sviluppatori, legislatori, aziende, insegnanti e studenti, per questo motivo le WCAG sono state realizzate con una struttura gerarchica in modo da essere comprensibili a tutti gli utenti.
Quest' ultima è formata come segue:

* **Principi:** al livello superiore, vengono definiti i quattro principi che fanno da pilastri all'accessibilità del *web* ovvero la percepibilità, l'utilizzabilità, la comprensibilità e la robustezza.

* **Linee guida:** dai quattro principi discendono le linee guida. Le 12 linee guida forniscono gli obiettivi di base su cui gli autori dovrebbero lavorare per rendere il contenuto più accessibile agli utenti con diverse disabilità.

* **Criteri di successo:** per ogni linea guida, vengono forniti criteri di successo verificabili per consentire l'utilizzo delle WCAG2.0, al fine di soddisfare le esigenze dei diversi gruppi e situazioni, vengono definiti tre livelli di conformità: A (minimo), AA e AAA (massimo).

* **Tecniche sufficienti e consigliate:** per ciascuna linea guida e criterio di successo, sono documentate una serie di tecniche.
  Le tecniche sono informative e ricadono in due categorie: sufficienti per soddisfare il criterio di successo e consigliate. Le seconde vanno oltre ciò che viene richiesto da ciascun singolo criterio di successo e consentono agli autori di rispettare le linee guida ad un livello più elevato.

Primo principio - Percepibilità

Le informazioni e i componenti dell'interfaccia utente devono essere presentati agli utenti in modo che possano essere percepiti.

* Alternative testuali: fornire alternative testuali per qualsiasi contenuto non di testo in modo che questo possa essere trasformato in altre forme fruibili secondo le necessità degli utenti come stampa a caratteri ingranditi, Braille, sintesi vocale, simboli o un linguaggio più semplice.

* Tipi di media temporizzati: fornire alternative per i tipi di media temporizzati.

* Adattabilità: creare contenuti che possano essere rappresentati in modalità differenti (ad esempio, con *layout* più semplici), senza perdere informazioni o la struttura.

* Distinguibilità: rendere più semplice agli utenti la visione e l'ascolto dei contenuti, separando i contenuti in primo piano dallo sfondo.

[1] *WCAG 2.0.* URL: https://www.w3.org/TR/WCAG20/.

Secondo principio - Utilizzabilità

I componenti e la navigazione dell'interfaccia utente devono essere utilizzabili.

* Accessibilità da tastiera: rendere disponibili tutte le funzionalità tramite tastiera.
* Adeguata disponibilità di tempo: fornire agli utenti tempo sufficiente per leggere ed utilizzare i contenuti.
* Convulsioni: non sviluppare contenuti che possano causare attacchi epilettici.
* Navigabilità: fornire delle funzionalità di supporto all'utente per navigare, trovare contenuti e determinare la propria posizione.

Terzo principio - Comprensibilità

Le informazioni e le operazioni dell'interfaccia utente devono essere comprensibili.

* Leggibilità: rendere il testo leggibile e comprensibile.
* Prevedibilità: creare pagine *web* che appaiano e che siano prevedibili.
* Assistenza nell'inserimento: aiutare gli utenti ad evitare gli errori ed agevolarli nella loro correzione.

Quarto principio - Robustezza

Il contenuto deve essere abbastanza robusto per essere interpretato in maniera affidabile mediante una vasta gamma di programmi utente, comprese le tecnologie assistive.

* Compatibilità: garantire la massima compatibilità con i programmi utente attuali e futuri, comprese le tecnologie assistive.

Capitolo 3

Strumenti e tecnologie

In questa sezione è presente una breve descrizione dei linguaggi e delle applicazioni utilizzati durante la fase di sviluppo.

3.1 Linguaggi

3.1.1 HTML

HyperText Markup Language (HTML) dalla traduzione letterale, linguaggio a marcatori per ipertesti, è un linguaggio di *markup*[g], che descrive le modalità di impaginazione, formattazione o visualizzazione grafica (*layout*) del contenuto, testuale e non, di una pagina web attraverso *tag* di formattazione.

Figura 3.1: Logo HTML5

L'HTML supporta l'inserimento di script e oggetti esterni quali immagini o filmati.
Per il progetto abbiamo utilizzato **HTML 5**[1] (logo in figura 3.1).
Le novità introdotte dall'HTML5 sono finalizzate soprattutto a migliorare il disaccoppiamento tra la struttura logica di una pagina web (definita appunto dal *markup*), e la sua rappresentazione, gestita tramite gli stili CSS per adattarsi alle nuove esigenze di comunicazione e pubblicazione all'interno di Internet.

L'HTML5 ha introdotto le seguenti modifiche:

* sono fissate in modo rigido le regole per la struttura del testo in capitoli, paragrafi e sezioni;

[1] *HTML 5*. URL: https://www.html.it/guide/guida-html5/.

* vengono deprecati o eliminati alcuni elementi che hanno dimostrato scarso o nessun utilizzo effettivo;
* vengono estesi a tutti i tag una famiglia di attributi, finalizzati all'accessibilità, finora previsti solo per alcuni tag;
* è introdotta la geo localizzazione, dovuta ad una forte espansione di sistemi operativi mobili;
* possibilità di utilizzare alcuni siti offline;
* sono introdotti elementi specifici per la resa di contenuti multimediali (tag video e audio);
* introdotto il tag Canvas che permette di utilizzare JavaScript per creare grafica, animazioni e bitmap.

3.1.2 CSS

Cascading Style Sheets (CSS)[2] il cui logo è illustrato in figura 3.2, è un linguaggio usato per definire la formattazione di documenti HTML, XHTML e XML. Le regole per comporre il CSS sono contenute in un insieme di direttive emanate a partire dal 1996 dal *World Wide Web Consortium (W3C)*[g][3] . Il CSS separa i contenuti delle pagine HTML dalla loro formattazione o *layout* e permettono una programmazione più chiara e facile da utilizzare, sia per gli autori delle pagine stesse sia per gli utenti, garantendo contemporaneamente anche il riutilizzo di codice ed una sua più facile manutenzione.

L'introduzione del CSS si è resa necessaria per separare i contenuti dalla formattazione e permettere una programmazione più chiara e facile da utilizzare, sia per gli autori delle pagine HTML che per gli utenti. Inoltre avendo dei fogli di stile separati, a seconda del *device* nel quale viene visualizzata la pagina può essere impostato un foglio di stile specifico, per permettere migliore usabilità, efficienza e resa grafica.
Il supporto completo e corretto delle specifiche CSS non è offerto da nessun browser attuale. Tuttavia esistono browser che si avvicinano molto a questo risultato ed altri che invece ne sono molto lontani.

Figura 3.2: Logo CSS

[2] *CSS*. URL: https://it.wikipedia.org/wiki/CSS.
[3] *W3C*. URL: https://www.w3.org/.

3.1.3 PHP

Hypertext Preprocessor (PHP)[4] (logo in figura 3.3) è un linguaggio di *scripting*[g] *server-side*[g] utilizzato per la creazione di pagine web dinamiche integrate efficacemente con i database. Attualmente è adoperato nello sviluppo di applicazioni web lato server, ma può essere usato anche per scrivere *script* a riga di comando o applicazioni *stand-alone* con interfaccia grafica.

Figura 3.3: Logo PHP

Questo linguaggio interagisce con un'ampia gamma di database tra cui MySQL, MSSQL, PostgreSQL, dBase e Hyperwave. Inoltre è multipiattaforma ovvero può essere utilizzato sui principali sistemi operativi, inclusi Linux, Microsoft Windows, MacOS X, e supporta la maggior parte dei server web esistenti.
PHP permette dunque di creare pagine dinamiche e, pur disponendo di una sintassi piuttosto semplice, è comunque in grado di offrire moltissime funzionalità, tra cui:

* raccogliere dati da un modulo di contatto;
* creare mailing list;
* mandare/ricevere cookies;
* gestire e caricare file;
* creare sezioni di registrazione e di *login* per aree riservate;
* effettuare la cifratura/decifratura di dati.

[4] *PHP*. URL: https://www.php.net/.

3.1.4 Javascript

Javascript[5] il cui logo è rappresentato in figura 3.4, è un linguaggio di *scripting* orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione web *client-side*[g]per la creazione di effetti dinamici interattivi innescati da azioni dell'utente (ad esempio: movimento del mouse).

Figura 3.4: Logo Javascript

Un aspetto importante di Javascript è che il codice viene eseguito direttamente sul *client* e non sul *server*. Il vantaggio è che, anche con la presenza di script particolarmente complessi, il server non viene sovraccaricato a causa delle richieste dei client.

JavaScript riesce a creare HTML dinamico che:

* può modificare tutti gli elementi, attributi e stili CSS e HTML della pagina;
* è in grado di reagire a tutti gli eventi della pagina;
* può controllare i valori nei campi di input, nascondere o visualizzare determinati elementi, rimpiazzare delle immagini e adattare valori di layout.

JavaScript nato come supporto ad HTML, ricopre ancora oggi questo ruolo, specie con l'avvento dello standard HTML5.
Esistono essenzialmente tre modi per inserire codice JavaScript in una pagina HTML:

* inserire codice *inline*;
* scrivere blocchi di codice nella pagina attraverso il *tag* <script></script>;
* importare file con codice JavaScript esterno nel seguente modo: <script src="fileesterno.js"></script>.

JavaScript è uno dei linguaggi di programmazione più usati al mondo, l' enorme diffusione è dovuta principalmente al fiorire di numerose librerie nate allo scopo di semplificare la programmazione sul browser, per questo progetto verrà usata React (noto anche come React.js o ReactJS) una libreria *open-source*, *front-end* per la creazione di interfacce utente.

[5] *JavaScript*. URL: https://it.wikipedia.org/wiki/JavaScript.

3.1.5 Python

Python[6] è un linguaggio ad alto livello[g], orientato a oggetti, adatto, tra gli altri usi, per sviluppare applicazioni distribuite, *scripting*, computazione numerica e *system testing*, il suo logo è presente nella figura 3.5.
È un linguaggio multi-paradigma che ha tra i principali obiettivi: dinamicità, semplicità e flessibilità. Supporta il paradigma *object oriented*, la programmazione strutturata e molte caratteristiche di programmazione funzionale.
Le caratteristiche più immediatamente riconoscibili di Python sono le variabili non tipizzate e l'uso dell'indentazione per la sintassi delle specifiche, al posto delle più comuni parentesi.
Altre caratteristiche distintive sono l'overloading di operatori e funzioni tramite delegati, la presenza di un ricco assortimento di tipi e funzioni di base e librerie *standard*, sintassi avanzate quali *slicing* e *list comprehension*.

Figura 3.5: Logo Python

3.2 Tecnologie

3.2.1 MySQL

MySQL[7] (logo presente in figura 3.6) è un database relazionale composto da un *client* a riga di comando e un server. È un software libero rilasciato con doppia licenza. È possibile scaricare il DBMS dal sito di riferimento MySQL AB. Il sito di MySQL AB contiene un'ampia documentazione sul DBMS, nello specifico spiega come installarlo.

Figura 3.6: Logo MySQL

[6] *Phthon*. URL: https://it.wikipedia.org/wiki/Python.
[7] *MySQL*. URL: https://www.mysql.com/it/.

La versione utilizzata è la 5.6 l'ultima pubblicata in produzione. Sono molte le nuove funzionalità della versione 5.6:

* la gestione dei microsecondi e dei millisecondi nei *datatype* temporali e nei *timestamp*;
* la possibilità di controllare i dati della Host Cache e i relativi errori;
* nuove viste, utili per il monitoraggio ed il *tuning* della base dati, nel *Performance* Schema e nell'*Information* Schema;
* la possibilità di escludere alcune *directory* dalla ricerca;
* l'utilizzo di ricerche testuali (FULLTEXT SEARCH) sull'Engine InnoDB;
* Molte utili estensioni che migliorano la sicurezza della base dati.

3.2.2 React

React[8] è una libreria *open-source, front-end*, JavaScript per la creazione di interfacce utente. Viene mantenuto da Meta (Facebook) e da una comunità di singoli sviluppatori e aziende, il suo logo è rappresentato nella figura 3.7.

Figura 3.7: Logo React

React può essere utilizzato come base nello sviluppo di applicazioni a pagina singola ma è utilizzabile anche su mobile tramite *React Native*, una libreria che traduce i componenti React in componenti nativi (iOS e Android). Tuttavia, React si occupa solo del rendering dei dati, pertanto la creazione di applicazioni React richiede generalmente l'uso di librerie aggiuntive per lo *state management* e il *routing*, una di queste è Redux, che è stata utilizzata per lo sviluppo di questo progetto.
React consente di sviluppare applicazioni dinamiche che non necessitano di ricaricare la pagina per visualizzare i dati modificati. Inoltre nelle applicazioni React le modifiche effettuate sul codice si possono visualizzare in tempo reale, permettendo uno sviluppo rapido, efficiente e flessibile delle applicazioni web.

[8] *React.* URL: https://react.dev/.

3.2.3 Material UI

Material UI[9] è una libreria *open source* di React e implementa i *material design* di Google. Material UI presenta opzioni di personalizzazione che semplificano l'implementazione del sistema di progettazione. Questo comporta la possibilità di personalizzare i componenti presenti con molta facilità.
Per esempio, questi sono i layout dei bottoni che mette a disposizione Mui:

* <Button variant="text">Text</Button>
* <Button variant="contained">Contained</Button>
* <Button variant="outlined">Outlined</Button>

Che sono illustrati nella figura 3.8, ed è possibile personalizzarli aggiungendo del codice di stile, cambiando il colore, la forma e ogni aspetto possibile, importando il codice CSS come parametro, in questo modo:

$$< Buttonvariant = "contained" sx = \{\{border :' 2pxsolid1976d2', borderRadius :' 30px',$$

$$backgroundColor :' white', color :' \#1976d2'\}\} > contained < /Button >$$

Il risultato è raffigurato nella figura 3.8 a destra.

Figura 3.8: Bottoni Mui

Vantaggi:

* Programmare più velocemente: è possibile concentrarsi su altro rispetto che sull'interfaccia utente, in quanto è tutto ben preimpostato.
* Esteticamente piacevole: ogni componente dell'interfaccia utente del materiale è stata realizzata per cercare di soddisfare gli *standard* di forma e funzione, proponendo più opzioni di *layout*.
* Personalizzazione: la libreria include un ampio set di funzionalità di personalizzazione intuitive.
* Collaborazione tra team: l'esperienza di sviluppo intuitiva dell'interfaccia riduce la barriera all'ingresso per gli sviluppatori back-end e i progettisti meno tecnici, consentendo ai team di collaborare in modo più efficace. I kit di progettazione semplificano il flusso di lavoro e aumentano la coerenza tra designer e sviluppatori.
* Scelto da migliaia di organizzazioni: Material UI ha la più grande *community* di UI nell'ecosistema React. La sua storia risale al 2014 e può contare sul supporto della *community* per gli anni a venire.

[9] *MUI*. URL: https://mui.com/.

3.2.4 Redux

Redux[10] è una libreria JavaScript *open source* per la gestione e la centralizzazione dello stato dell'applicazione. È più comunemente usato con librerie come React o Angular per la creazione di interfacce utente.
Geekandjob.com descrive Redux come "un contenitore di stati prevedibili per le applicazioni JavaScript". Nasce per risolvere la gestione dello stato ovvero l'insieme delle condizioni interne in uno specifico istante che determinano il risultato delle interazioni con l'esterno.
In altre parole, lo stato di un'applicazione è l'insieme delle informazioni che determinano l'*output* in corrispondenza di un dato *input* in uno specifico istante.
La soluzione proposta da Redux si basa su tre principi fondamentali:

* esiste una singola fonte di verità: lo stato dell'intera applicazione è memorizzato in un unico oggetto;
* lo stato è in sola lettura: non è possibile modificare direttamente le informazioni memorizzate nello stato dell'applicazione, l'unico modo per farlo è tramite azioni esplicite;
* le modifiche allo stato vanno fatte con funzioni pure: lo stato corrente viene sostituito da un nuovo stato generato da funzioni esclusivamente in base ad una azione ed allo stato precedente.

Figura 3.9: Logo Redux

[10] *Redux*. URL: https://redux.js.org/.

3.3 Applicazioni

3.3.1 Jira

Jira[11] (logo rappresentato in figura 3.9) è una *suite* di software proprietari per il tracciamento delle segnalazioni sviluppato da Atlassian, che consente il *bug tracking* e la gestione dei progetti sviluppati con metodologie agili.
Consente alle aziende di risparmiare tempo e risorse e di migliorare la loro efficienza complessiva. Offre inoltre diverse funzionalità fra cui:

* schede Scrum;
* flussi di lavoro personalizzati;
* etichettatura flessibile dei problemi;
* tracciamento dei *bag*.

In questo modo permette di rendere più semplice la gestione del lavoro.

3.3.2 Slack

Slack[12] , il cui logo è presente in figura 3.9, è un software che rientra nella categoria degli strumenti di collaborazione aziendale utilizzato per inviare messaggi in modo istantaneo ai membri del team. Viene utilizzato da S.AI in quanto è un'azienda che lavora principalmente in *smart working* ed ha quindi bisogno di un mezzo di comunicazione organizzato ed efficiente.

3.3.3 Gitlab

GitLab[13] è una piattaforma web di gestione di *repository* Git, il cui logo è presente in figura 3.9. Un progetto *open source* che permette di salvare il proprio codice in *repository* ad accesso pubblico o privato.
Come altri sistemi di controllo di versione, GitLab consente operazioni di *pull*, *push* e *merge*. In questo modo è possibile collaborare con altri programmatori pur lavorando in modo autonomo, senza generare conflitti.

Figura 3.10: Logo applicazioni

[11] *Jira*. URL: https://www.atlassian.com/it/software/jira.
[12] *Slack*. URL: https://slack.com/intl/it-it.
[13] *Gitlab*. URL: https://about.gitlab.com/.

Capitolo 4

Sviluppo del progetto

Il seguente capitolo ha lo scopo di descrivere lo sviluppo del progetto, attraverso le fasi di analisi e di codifica, verrà spiegato nel dettaglio come è stata progettata l'applicazione.

4.1 Analisi

Dopo una prima parte di stage dove ho studiato le tecnologie esposte nel capitolo precedente, ho iniziato effettivamente a pensare come realizzare il prodotto.
Analizzando la richiesta, ciò che dovevo produrre era una rappresentazione grafica delle aziende all'interno dell'applicazione, con la possibilità di associare utenti e prodotti.
Il primo requisito analizzato, riportato nella Tabella 2.1, è stato **OB01**: possibilità di creare e configurare un'azienda.
La prima domanda che ci siamo posti è stata: "come deve essere configurata un'azienda? O meglio quali campi deve avere?", dopo una discussione avvenuta tra me e il tutor aziendale abbiamo optato per inserire solo le informazioni necessarie ovvero:

* id dell'azienda e quindi chiave univoca;
* nome;
* *slug* ovvero una forma leggibile e valida per l'URL di una pagina web;
* lista degli utenti;
* lista dei prodotti;
* lista dei marchi;

Le aziende vengono create da un utente e al momento della creazione viene generato automaticamente l'id e viene richiesto il nome dell'azienda, gli altri campi sono quindi opzionali. E' chiaro però che una volta creata l'azienda, la sua lista degli utenti avrà un componente (il creatore).
Un utente può fare parte di più aziende all'interno dell'applicazione, e non c'è un numero limite delle aziende che si possono creare.
Per questioni di leggibilità ho scelto di dividere l'analisi in tre sezioni, l'analisi sugli utenti, quella sui marchi e quella sui prodotti.

4.1.1 Utenti

Il campo delle aziende "lista degli utenti" non può essere vuoto in quanto avrà sicuramente al suo interno il creatore, per cui abbiamo ritenuto necessario analizzare come passo successivo la lista degli utenti (**DE01**:"Possibilità di associare più utenti ad un'azienda"). Siamo partiti analizzando i dati già presenti all'interno dell'applicazione, per quanto riguarda gli utenti, per capire quali di questi fossero utili per il nostro scopo.
Un utente dell'applicazione è formato dai seguenti campi:

* id utente (chiave univoca);
* foto profilo;
* username;
* email;
* nome e cognome;
* data di nascita;
* genere;
* indirizzo;
* reddito familiare;
* stato civile;
* livello di istruzione;

Questi dati sono utili per poter classificare meglio gli utenti o per visualizzare le statistiche sui prodotti; ma mostrarli tutti all'interno di un'azienda oltre a invadere la *privacy* degli utenti, creerebbe anche molta confusione visiva. Abbiamo quindi scelto di mostare solo questi campi:

* foto profilo;
* username;
* id (non visibile ma presente);
* email;

Abbiamo poi aggiunto la possibilità che un utente sia amministratore dell'azienda, attraverso il campo *isAdmin* e possa quindi svolgere azioni riservate come eliminare l'azienda, modificare i campi dell'azienda, rimuovere e aggiungere utenti, o eliminare e modificare prodotti. Queste azioni ricoprono i requisiti **OP01**: "Possibilità di avere ruoli separati per la gestione delle aziende" e **OP02**: "Possibilità di avere un'interfaccia amministrativa per l'autorizzazione di alcune operazioni (per es. associare marchi e prodotti ad un'azienda)". Alcune scelte architetturali che abbiamo fatto riguardo questo campo sono:

* il creatore dell'azienda ha il campo *isAdmin* attivo, quindi è un amministratore;
* all'interno di un'azienda deve essere presente almeno un amministratore;

- all'interno di un'azienda possono esserci più amministratori;
- un utente che è amministratore non può essere eliminato dall'azienda;
- un amministratore può rendere un altro utente amministratore, e può anche rendere un amministratore un utente semplice.

La figura 4.1 raffigura la sezione degli utenti di un'azienda (in questo caso l'azienda si chiama "org1"), e rappresenta ciò che vede un utente amministratore che può quindi modificare gli elementi presenti attraverso l'icona della matita, eliminare cliccando sul cestino o aggiungere utenti con l'apposito pulsante.
Per rendere l'applicazione accessibile ogni icona ha un testo alternativo che compare nel momento in cui il cursore è posizionato sull'icona.

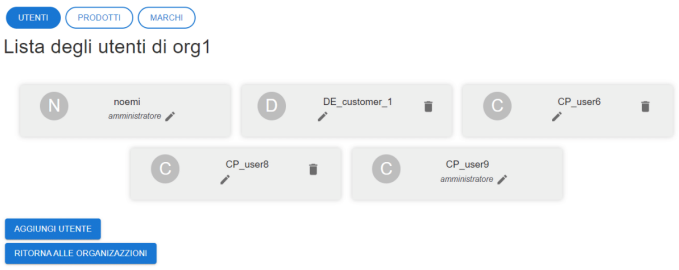

Figura 4.1: Utenti di un'azienda

Per la parte grafica abbiamo deciso di tenere i colori dell'applicazione già presente (blu e viola) e di utilizzare un grigio chiaro come sfondo delle "*card*" degli utenti per rendere ben chiara la selezione di utente rispetto all'altro.
Abbiamo utilizzato lo stesso *layout* anche per i marchi e i prodotti delle aziende, per mantenere coerenza.

4.1.2 Marchi

Per quanto riguarda i marchi, abbiamo iniziato dall'obbiettivo **OB02**: "Possibilità di creare e configurare prodotti e marchi dell'azienda", per poi passare all' **OB03**: "Possibilità di visualizzare e ricercare marchi e prodotti dell'azienda", chiaramente concentrandoci solo sulla parte dei marchi.
Nell'applicazione non erano presenti i marchi e li abbiamo quindi creati da zero. Si è scelto di formarli con i seguenti campi:

- id del marchio, chiave univoca;
- nome del marchio, variabile testuale;
- testo libero, utilizzato se il marchio ha uno slogan, o se l'azienda vuole dare qualche informazione specifica.

Gli utenti amministratori hanno la possibilità di aggiungere marchi tramite l'apposito pulsante, e come per gli utenti eliminare il singolo marchio con l'icona del cestino, o modificarne le informazioni con l'icona della matita. Le informazioni modificabili sono il nome e il testo libero.

Abbiamo preso queste scelte riguardo ai marchi:

* un'azienda può non avere marchi associati, in questo caso nella sezione dedicata ai marchi si visualizza la scritta "non ci sono marchi associati a questa azienda" e il tasto di aggiunta di un marchio (solo se l'utente è amministratore);
* al momento della creazione del marchio, viene richiesto il nome del brand che si vuole aggiungere, come campo obbligatorio; e il testo aggiuntivo che invece è opzionale;
* il marchio non avrà al suo interno la lista dei prodotti, ma abbiamo scelto di mettere all'interno dei prodotti la lista dei marchi.

Nella figura 4.2 è rappresentata la sezione dei marchi di un'azienda.

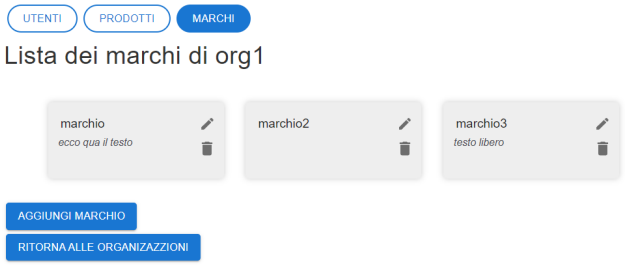

Figura 4.2: Marchi di un'azienda

4.1.3 Prodotti

Per quanto riguarda i prodotti, abbiamo seguito lo stesso procedimento dei marchi, iniziando quindi da **OB02**: "Possibilità di creare e configurare prodotti e marchi dell'azienda", per poi passare all' **OB03**: "Possibilità di visualizzare e ricercare marchi e prodotti dell'azienda", spostando la nostra attenzione sui prodotti.
A differenza dei marchi, all'interno dell'applicazione erano già presenti i prodotti che però erano composti da una ventina di campi non utili alla sezione delle aziende, come allergeni, calorie, ecc. Si è quindi scelto di creare una nuova classe derivata dai prodotti standard che ha i seguenti campi:

* codice EAN, ovvero un codice a barre utilizzato per l'identificazione univoca di prodotti destinati al consumatore finale;
* immagine del prodotto, se il prodotto non ha un'immagine appare un'immagine di caricamento;
* nome del prodotto;
* lista dei marchi del prodotto, se non ha associato nessun marchio, non viene mostrato nulla.

Le informazioni sono state disposte come da figura 4.3.

Figura 4.3: Prodotti di un'azienda

Per aggiungere un nuovo prodotto all'azienda bisogna inserire il codice EAN ed il nome, come con le altri classi, solo un organizzatore può aggiungere un nuovo prodotto.
Sono inoltre presenti le funzioni di rimozione di un prodotto e di modifica, all'interno della modifica è possibile cambiare solamente il nome del prodotto. E' presente anche una funzionalità aggiuntiva ovvero l'inserimento di alcune informazioni (**DE02**: "Possibilità di caricare informazioni relative a prodotti e marchi dirette all'utente finale") attraverso l'icona con il simbolo + che come tutti gli altri pulsanti è dotata del testo alternativo "aggiungi informazioni".
Dalla sezione dedicata ai prodotti delle aziende si possono anche vedere le statistiche e le analisi dei singoli prodotti (**OB04**: "Possibilità di visualizzare statistiche e analisi per prodotti e marchi"), cliccando su un prodotto apparirà una pagina di selezione che allo stato attuale ha solo due scelte di grafico, l'istogramma e il *wordcloud* che si basano sulle recensione lasciate dagli utenti, e sulle informazioni personali che gli utenti che hanno recensito hanno inserito nel loro profilo. Io mi sono dedicata allo sviluppo degli istogrammi. E abbiamo scelto di mostrare i seguenti filtri:

* generale (1 stella, 2 stelle, 3 stelle, 4 stelle, 5 stelle);
* età (<18, 18-24, 25-34, 35-44, 45-54, 55-64, >64);
* genere (Maschio, Femmina, Altro);
* titolo di studi(Scuola primaria, Scuola secondaria, Laurea triennale, Laurea magistrale, Master);
* stato sociale (Single, Sposato, Divorziato, Vedovo);

Se non viene selezionato nessuno di questi filtri quello di default è "generale", e viene mostrato un solo istogramma, come si può notare in figura 4.4, che ha come ascisse il numero di stelle e come ordinate il numero totale di recensioni per quelle stelle. Una volta selezionato un altro filtro, appariranno 3 istogrammi diversi, il primo è un

Figura 4.4: Istogramma di un prodotto

istogramma a pila, rappresentato in figura 4.5, l'ascissa corrisponde al numero di stelle e le ordinate invece rappresentano il numero di recensioni suddivise per colore in base al filtro (in questo caso il genere) e si può quindi vedere fra le persone che hanno votato quattro stelle quante erano maschi e quante invece femmine.

Figura 4.5: Istogramma a pila di un prodotto

Gli altri due sono istogrammi che hanno come ascissa i valori del filtro, in questo caso Maschio, Femmina, Altro, e come ordinata uno presenta il numero totale di recensioni per categoria e l'altro la media delle stelle per categoria, quindi si può vedere per esempio l'indice di gradimento delle persone single in base alla media di stelle che hanno messo.

4.2 Progettazione

Il software FoodAdvisor si configura come una web-app. Il sito è modulare ovvero composto da un insieme di servizi. A seconda della propria sottoscrizione, l'utente ha accesso a un sottoinsieme di questi servizi. Il sistema è composto da un *frontend* e un *backend*. Il *backend* è scritto in Python 3, supportato dal *framework* Django per un agevole sviluppo web.
Il *frontend* è scritto in Javascript che fa uso delle librerie sopracitate React e Redux, le quali consentono il regolamento del flusso di esecuzione e l'organizzazione del codice in piccole componenti riusabili. La comunicazione tra *frontend* e *backend* è costituita da chiamate Representational state transfer (REST)[g]attraverso l'utilizzo di Django REST framework.

4.2.1 Architettura Django

Django utilizza una versione particolare dell'architettura MVC (*Model View Controller*) che prende il nome di **MTV**[1] ovvero *Model Template View*. Questa architettura fa in modo che il *Controller* risulti in realtà il *framework* in sé, le *View* non scelgano come i dati vadano mostrati bensì quali dati mostrare e il come mostrarli viene in realtà deciso nei *Template*.
In figura 4.6 è presente lo schema dell'architettura.

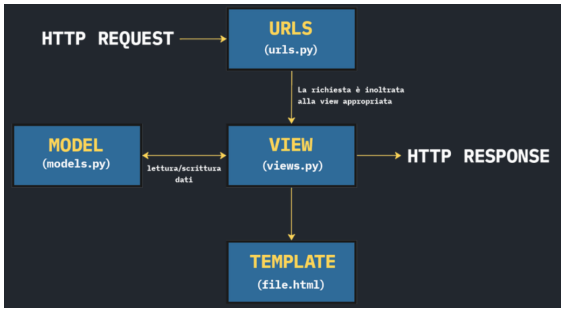

Figura 4.6: Architettura MVT

Nel momento in cui viene effettuata una richiesta HTTP essa viene confrontata con gli URL. La richiesta viene quindi inoltrata alla *View* che la gestisce producendo una risposta HTTP che poi verrà formattata dai *template*, il *Model* fornisce la logica per accedere ai dati e convertirli in tabelle nel *database*.
Riassumendo quindi:

* **M - Model**: "*data access layer*" - Definisce la struttura delle entità del sito, tradotte poi in tabelle nel *database*.
* **T - Template**: "*presentation layer*" - Descrive come i dati vadano mostrati nelle pagine web.
* **V - View**: "*business logic layer*" - Gestisce le richieste e le risposte HTTP, dispone della logica per sapere a quali dati accedere tramite i *model* e delega la formattazione della risposta ai *template*.

[1] *Architettura django*. URL: https://www.programmareinpython.it/blog/che-cosa-rende-django-speciale-miglior-web-framewo/.

4.2.2 Architettura React

Il progetto di cui mi sono occupata, e in generale i progetti fatti in React, hanno una cartella sorgente (src) in cui sono elencati tutti i file e le cartelle essenziali, qui sotto elencati:

* *assets*: contiene tutti i file statici del progetto, come logo, caratteri, immagini e favicon.

* *Components*: contiene le varie componenti create, questa cartella è a sua volta nidificata all'interno, per esempio se servono più componenti per il profilo, queste vengono racchiuse dentro una singola cartella.

* *Features*: contiene il codice che consente di interagire con risorse API.

* *Hooks*: contiene codici e logica che possono essere riutilizzati su più componenti.

* *Utils*: contiene frammenti di funzioni riutilizzabili per eseguire attività rapide.

* *App*: contiene il componente principale dell'applicazione React ovvero il file App.js, che collega tutti i componenti e le viste, la configurazione del file, e i suoi test.

* *Test*: invece contiene gli *snapshot*, inseriti in un'apposita cartella, e i test delle componenti, divise nelle stesse cartelle presenti all'interno di *components*.

* *Index.jsx*: è il punto di ingresso dell'applicazione React.

* *Index.css*: è il file CSS principale e globale per l' applicazione. Qualsiasi stile di scrittura in questo file è stato applicato a tutto il progetto.

React Best Practices

Le buone pratiche di React, consigliate nel sito ufficiale della libreria e dalla community, sono svariate e consentono di creare interfacce utente con codice corretto, leggibile, modulare e manutenibile. Le *best practices* utilizzate in particolare nel progetto in questione sono le seguenti:

* scomporre la GUI in una gerarchia di component, seguendo il principio di singola responsabilità tipico della programmazione ad oggetti;

* identificare lo stato minimo indispensabile necessario per ottenere tutte le funzionalità desiderate e decidere in quale component farlo giacere. Il principio base, valido in generale per qualsiasi ambito della programmazione, consiste nel *DRY* (ovvero *«Don't Repeat Yourself»*);

* utilizzare la destrutturazione degli oggetti, soprattutto nel caso di quelli passati tramite *props*, in modo da rendere il codice più snello e leggibile;

* utilizzare i *functional component* e i *react hooks* in quanto consentono una gestione più rapida e leggibile dello stato rispetto ai *class component*, anche se dal punto di vista di React le due tipologie di component sono equivalenti.

4.2.3 Metodo Agile

L'azienda S.AI - *Synthema Artificial Intelligience* utilizza nella sua quotidianità un approccio agile alla progettazione e codifica del software. I metodi agili sono una famiglia di metodi di sviluppo che hanno in comune:

* rilasci frequenti del prodotto sviluppato;
* collaborazione continua del team di progetto col cliente;
* documentazione di sviluppo ridotta;
* valutazione sistematica e continua di valori e rischi dei cambiamenti.

E si fondano su 4 principi base:

* concentrarsi su individui e interazioni piuttosto che su processi e strumenti;
* è meglio un software funzionante piuttosto che una documentazione completa;
* la collaborazione col cliente è meglio della negoziazione contrattuale;
* reagire al cambiamento è meglio che seguire un piano.

Nello specifico l'azienda utilizza il metodo ***Scrum***[2], illustrato nella figura 4.7, che fornisce un modello di valori, ruoli e linee guida per aiutare il *team* a concentrarsi sull'iterazione e sul miglioramento continuo.

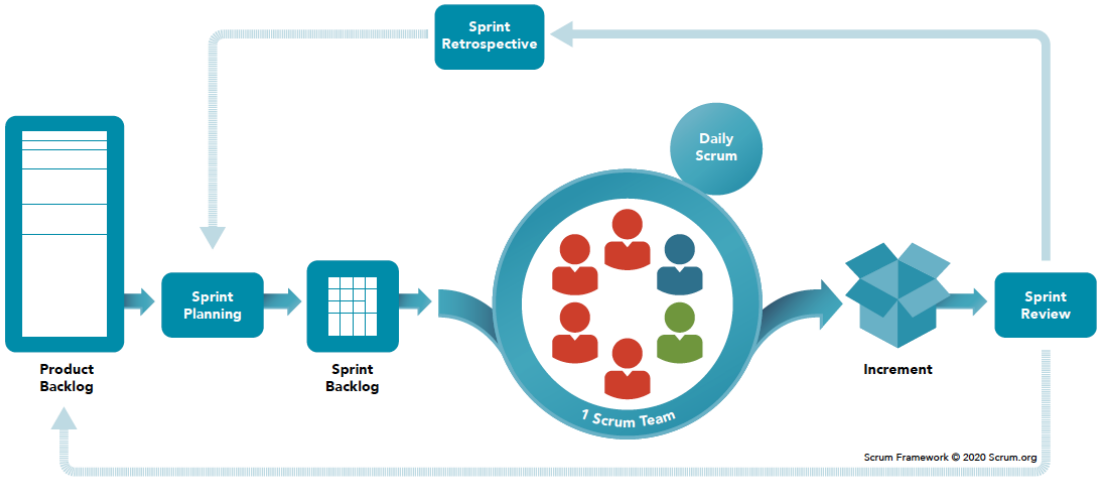

Figura 4.7: Modello SCRUM

[2] *Modello Scrum.* URL: https://www.agileway.it/scrum-metodologia-agile/.

Come funziona scrum?

In genere, *Scrum* è strutturato in *sprint*, che di solito sono sessioni di lavoro della durata di due settimane, al termine delle quali ci si aspetta un risultato finale. S-AI ha deciso invece di fare *sprint* di una settimana soltanto, dividendo quindi le attività del *backlog*[g]in sottoattività più ridotte in modo da poter essere completate in una settimana.
Analizziamo più nel dettaglio come funziona *scrum*, facendo riferimento alla figura 4.7:

* **Organizzare il *backlog***: il *team leader* identificherà le attività da svolgere a partire dal *product backlog*, che è l'elenco di ciò che deve essere fatto. Questo deve essere chiaramente documentato in un unico posto, per esempio in uno strumento di gestione dei progetti, nel nostro caso Jira (cap.3.3.1).

* **Pianificazione dello *sprint***: si valuterà a quale attività del *backlog* si dedicherà il tuo *team* durante quello specifico *sprint*.

* **Inizio dello *sprint Scrum***: durante lo *sprint*, il *team* lavorerà alle attività del backlog identificate durante la sessione di pianificazione dello *sprint*.

* **Riunioni giornaliere**: il metodo consiglia un incontro della durata di 15 minuti con il *team Scrum* ogni giorno. Queste riunioni servono per fare il punto su ciò a cui state lavorando e valutare eventuali blocchi imprevisti in cui potreste esservi imbattuti.

* **Presentazione del lavoro durante la revisione dello *sprint***: una volta terminato lo *sprint Scrum*, il *team* dovrebbe riunirsi per una revisione del lavoro svolto, durante la quale presenterà il lavoro fatto per l'approvazione o l'ispezione delle parti interessate.

* **Confronto e riflessione durante la retrospettiva**: al termine dello *sprint*, bisogna trovare del tempo per discutere di come è andato e cosa potrebbe essere migliorato in futuro. Ricordando che *Scrum* crede in un processo di miglioramento continuo, quindi non bisogna esitare a provare nuovi processi o rielaborare strategie che sembrano meno efficaci durante il prossimo *sprint*.

4.3 Codifica

Per lo sviluppo di questo applicativo ho utilizzato Visual Studio Code, un *IDE cross-platform* molto leggero e dinamico che permette di lavorare con una grande quantità di linguaggi.

4.3.1 Componenti React

Il codice sorgente è organizzato in moduli indipendenti e potenzialmente riutilizzabili; in altre parole è costituito da componenti. I componenti sono blocchi di codice che vengono costruiti dinamicamente a partire da due fattori: dati immutabili forniti in ingresso e stato interno. I primi sono definiti come *properties*, o più comunemente *props*. Le *props* sono caratterizzate da nomi univoci e possono assumere la forma di qualsiasi tipo di dato definito da Javascript. All'interno del componente si può accedere al loro valore in sola lettura, in quanto tutti i componenti React devono comportarsi come funzioni pure rispetto alle proprie *props*.
Tutti i componenti react venivano creati nell'apposita cartella "*components*" e il nome che identifica il componente è univoco e solitamente è caratterizzato dall'iniziale maiuscola. Il codice è diviso in tre blocchi principali: le prime righe segnalano le librerie e i componenti importati necessari alla definizione del componente, raffigurati in figura 4.8; il secondo racchiude delle istruzioni Javascript per l'elaborazione dei dati; infine, il terzo rappresenta il contenuto ritornato e visualizzato effettivamente nella pagina, definito in JSX.

```
import { PropTypes } from 'prop-types';
import { useState } from 'react';
import { useNavigate, useParams } from 'react-router-dom';
import { useTranslation } from 'react-i18next';
import { useDispatch } from 'react-redux';
import { Typography, Button, Grid } from '@mui/material';
import { Delete as DeleteIcon, Edit as EditIcon } from '@mui/icons-material';
import { useGetProfileQuery } from '../features/api/rest/apiProfileSlice';
```

Figura 4.8: Componente React - primo blocco/*import*

Analizziamo le componenti che possiamo trovare all'interno del **primo blocco**:

* la funzione ***Proptypes*** serve ad assicurarsi che i dati ricevuti dalle *props*[g]siano del tipo giusto. Il suo utilizzo è raffigurato nell'immagine 4.9.

```
import { PropTypes } from 'prop-types';
...
function UserCard({ username, isadmin, id}) {
  return (
  );
}

UserCard.propTypes = {
  id: PropTypes.number,
  username: PropTypes.string,
  isadmin: PropTypes.bool,
};
export default UserCard;
```

Figura 4.9: Funzionamento Proptypes

* **Gli *hook* di react**, in questo caso "*useState*", sono funzioni che permettono di "ancorare" all'interno di *React state* e *lifecycle* da componenti funzione. Per la realizzazione di questo progetto sono stati usati "*useEffect*" e "*useState*". Il primo aggiunge la possibilità di eseguire operazioni di tipo *side effect*, che possono quindi alterare altri componenti e non possono essere eseguite durante la renderizzazione da componenti funzione. Il secondo permette invece di mantenere uno stato locale all'interno dei componenti funzioni, prima presente solo all'interno delle classi.
* **Gli *hook* di *react-router***: all'interno del progetto ne abbiamo utilizzati prevalentemente due. Il primo è "*useNavigate*" che restituisce una funzione che ti consente di navigare a livello di codice in due modi, o passando un "*To value*" (stesso tipo di <Link to>), oppure passando il delta che vuoi inserire nello *stack* della cronologia. Ad esempio, *navigate*(-1) equivale a premere il pulsante Indietro. Il secondo *hook* utilizzato è "*useParams*" che restituisce un oggetto contenente un *set* di coppie chiave/valore, impiegate per la determinazione della corrispondenza tra *path* del componente e URL del *browser*, ed è stato utile per esempio per rendere dinamica la selezione del filtro degli istogrammi.
* ***UseTranslation*** da *react-i18next*: permette la traduzione dinamica, con l'utilizzo di chiavi a cui vengono associati più valori, uno diverso per ogni lingua. Per esempio come si vede in figura 4.10, a "*admin*" è stato associato "*admin*" e "amministratore", di conseguenza sarà possibile creare un file di traduzione per ogni lingua desiderata, senza cambiare il codice sorgente che riporterà solo le chiavi.

Figura 4.10: Funzionamento UseTranslation

* ***UseDispatch*** che è l'*Hook* fornito da Redux per simulare il metodo dei *reducer* ovvero delle funzioni pure che modificano lo *state tree*.
* Alcune **componenti dalla libreria MUI** per realizzare le parti dell'interfaccia utente, tra cui bottoni e spinner per il caricamento.
* Alcune **immagini e icone** sempre per la parte grafica dell'applicazione, in questo caso vengono importate anche loro da MUI *material*.
* Gli ***endpoint***[g] presi dai file API utilizzati all'interno della componente.
* **Altri componenti**: come accennato all'inizio del paragrafo, ogni componente si presenta come un blocco indipendente e isolato dal resto del codice. Per poter essere utilizzata e inclusa in altre componenti è necessario esplicitare le istruzioni di export, ove definite, e import dove necessarie. Come si vede in figura 4.10 per quanto riguarda "*UserCard*".

Per il **secondo blocco**, analizziamo il caso di "*OrganizationCard*", mostrato in figura 4.11. Questa componente è stata creata per la rappresentazione visiva di una singola azienda:

```
function OrganizationCard({ name, id }) {
  const { t } = useTranslation();
  const navigate = useNavigate();
  const [deleteorg] = useDestroyOrganizationMutation();

  const deletsubmit = () => {
    deleteorg(id);
  };
  const editorg = () => {
    navigate(`/profile/editorganization/${id}`);
  };
  const gotouser = () => {
    navigate(`/profile/userorganization/${id}`);
  };
```

Figura 4.11: Componente React - secondo blocco

In questo blocco, oltre alla dichiarazione della funzione componente, troviamo le dichiarazioni di tutti i metodi che verranno usati dal componente:

* *UseTranslation* e *useNavigate* come accennato in precedenza servono rispettivamente per la traduzione della pagina e per navigare all'interno dell'applicazione.
* La ridefinizione di un *endpoint* importato nel blocco precedente dai file API; in questo caso l'*endpoint* che si occupa di eliminare un'organizzazione dal *database*.
* I metodi creati per le azioni necessarie al componente: *deletesubmit* che si occupa di richiamare l'*endpoint* passando l'id dell'organizzazione, *gotouser* che entra nella sezione degli utenti dell'azienda, *editorg* che naviga in una pagina di modifica dell'azienda, dove è possibile modificare il nome e lo *slug*.

Tutte queste funzionalità saranno richiamate all'interno del **terzo blocco**.
Continuiamo quindi l'analisi di "*OrganizationCard*", passando al contenuto ritornato e visualizzato effettivamente nella pagina, riportato nella figura 4.12:

```
return (
  <Card className="Organization_Card">
    <CardContent sx={{ p: 0, mt: 1 }} onClick={gotouser}>
      <Typography sx={{}} variant="body1" color="text.primary">
        {name}
      </Typography>
    </CardContent>
    <Grid item xs={12} sx={{ display: 'flex' }}>
      <Button type="submit" variant="contained" color="primary" sx={{ ml: 2, margin: '3px' }} onClick={editorg}>
        {t('common.actions.edit')}
      </Button>
      <Button type="submit" variant="contained" color="secondary" sx={{ ml: 2, margin: '3px' }} onClick={deletsubmit}>
        {t('common.actions.delete')}
      </Button>
    </Grid>
  </Card>
);
```

Figura 4.12: Componente React - terzo blocco

In questo caso e in tutti i casi in cui si tratta di *card* verrà ritornata proprio questa classe, che avra al suo interno la classe *CardContent* dove in questo esempio viene messo il nome dell'azienda. Normalmente questa classe può essere utilizzata come *link* per esempio per passare alla sezione degli utenti.

Dentro la *Card* ma fuori da *CardContent* verrano poi inserite le azioni della carta, in questo caso contraddistinte dai due bottoni "modifica" ed "elimina" che useranno rispettivamente le funzioni *editorg* e *deletesubmit*. Il risultato finale è rappresentato nella figura 4.13.

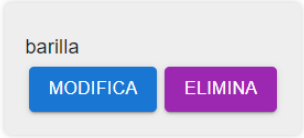

Figura 4.13: Rappresentazione di *OrganizationCard*

Questa componente è stata importata all'interno del primo blocco della componente *Organizations* tramite il seguente codice: *"import OrganizationCard from './OrganizationCard';"*.
Una volta ottenuta la lista di tutte le organizzazioni di un utente (nel codice sottostante è chiamata *organizations*), si ricavano il nome e l'id delle singole (rispettivamente *organization.name* e *organization.id*), per poi utilizzare *OrganizationCard* in modo da visualizzare la lista delle organizzazioni come da figura 4.14:

```
organizations.map((organization) => (
<OrganizationCard name=organization.name id=organization.id />
))
```

Figura 4.14: Rappresentazione delle aziende di un utente

La stessa logica viene usata per le sezioni dedicate a una singola azienda descritte nel capitolo 4.1 quindi per quanto riguarda gli utenti ci sarà una componente *UserCard* che viene richiamata dentro la componente *UserOrganization* e allo stesso modo *BrandCard* e *ProductOrganizationCard* vengono usate in *BrandOrganization* e *ProductOrganization*.

4.3.2 Creazione dei grafici

Per creare i grafici è stata utilizzata la libreria ***chartjs 2***[3] di React che dà la possibilità di creare grafici con facilità, e personalizzarne tutti gli aspetti possibili.
Ad esempio, è possibile modificare il colore e lo spessore dei bordi delle barre dei grafici, le descrizioni e la legenda cambiando il colore e la dimensione del carattere, aggiungere o eliminare una serie di dati con molta facilità.
Il tipo di grafico da me usato è il grafico a barre che possiede le seguenti *props*:

* *Data*: ovvero i dati che andranno inseriti nel grafico, di tipo *ChartData <"bar", TData, TLabel>*. Per esempio nel caso di filtro "generale" i dati passati sono quelli presenti in figura 4.15, dove "*starArray*" è un *array* contente il numero di recensioni per ogni stella di un determinato prodotto. Quindi se "*starArray*" è uguale a 0, 2, 6, 4, 0 significa che il prodotto ha zero recensioni da 1 e 5 stelle, 2 con 2 stelle, 6 con tre stelle e 4 con quattro.

```
{
  labels: ['1 stella', '2 stelle', '3 stelle', '4 stelle', '5 stelle'],
  datasets: {
    label: 'numero di recensioni',
    data: starArray,
    backgroundColor: 'rgb(255, 99, 132)',
  },
},
```

Figura 4.15: Esempio del tipo DataChart

  Come si può notare dall'immagine, all'interno dei dati sono presenti anche la legenda e i valori degli assi.

* *Options*: contiene tutte le opzioni del grafico per esempio la scala, le dimensioni del grafico, il *font* dei testi e altre.

* *Plugin*: contiene i *plugin* ovvero le estensioni, parti di codice per personalizzare ulteriormente un grafico, per esempio aggiungendo un tema predefinito .

* *Redraw*: una variabile booleana (di *default* falsa) per scegliere se ricaricare e ridisegnare il grafico a ogni aggiornamento.

* *DatasetIdKey*: una stringa il cui valore standard è '*label*' per dare un nome chiave al set di dati.

* *UpdateMode*: che prende dei valori definiti: "*resize*", "*reset*", "*none*", "*hide*", "*show*", "*normal*", "*active*" che indicano le transizioni e configurazioni che potrebbero esserci.

[3] *chartJs-2*. URL: https://react-chartjs-2.js.org.

Selezione dei dati

I dati utilizzati per creare i grafici sono stati selezionati da un unico *endpoint*: ***"fetchAdvancedReview"***, che richiedeva i seguenti input: ean ovvero il codice EAN del prodotto del quale si volevano vedere le statistiche (obbligatorio), e i dati dei filtri, che potevano essere inseriti oppure no:

* "*education*" per il livello di educazione, numeri da 0 a 4 in base al livello;
* "*gen*" per il genere, M per maschio, F per femmine e O per altro;
* "*status*" per lo stato civile, S di *single*, M di *marital*, D di *divorced* e W di *widowed*;
* "*lt*" e "*gt*" per l'eta, che erano rispettivamente *less then* e *greater then*, lt richiedeva quindi un valore massimo e gt un valore minimo.

Il risultato della selezione aveva poi i campi *star1-star5* in modo che si potesse selezionare ulteriormente il valore.
Ma vediamo meglio il funzionamento, partendo dall'esempio più semplice, quello "generale" dove come spiegato nel capitolo 4.1.3 si visualizza un solo grafico a barre. I dati al suo interno sono stati ricavati in questo modo:

1. E' stata definita una variabile con all'interno tutte le recensioni del prodotto:

$$const\{data : review = []\} = useFetchAdvancedReviewQuery(\{ean\});$$

2. E' stato definito l'*array* con i dati divisi per il numero di stelle:

$$conststarArray = [review?.star1||null, review?.star2||null, review?.star3||null,$$
$$review?.star4||null, review?.star5||null];$$

NB: ogni valore viene selezionato in modo tale che se un prodotto non ha recensioni per quel numero di stella, non ci sia un errore 404, ma venga assegnato 0.

Passiamo ora agli altri filtri, che hanno un procedimento più complesso, dal momento che oltre alla divisione per stelle è necessaria anche quella per categoria, ad esempio nel caso dello stato civile abbiamo *single*, sposati, divorziati e vedovi.
Analizziamo quindi questo esempio:

1. A differenza del filtro "generale" dove si selezionano tutte le recensioni attraverso la *query*, per gli altri filtri è necessario fare un *array* per ogni categoria:

$$const\{data : StatusS = []\} = useFetchAdvancedReviewQuery(\{ean, marital :' S'\});$$

$$const\{data : StatusM = []\} = useFetchAdvancedReviewQuery(\{ean, marital :' M'\});$$

$$const\{data : StatusD = []\} = useFetchAdvancedReviewQuery(\{ean, marital :' D'\});$$

$$const\{data : StatusW = []\} = useFetchAdvancedReviewQuery(\{ean, marital :' W'\});$$

2. Questi *array* verranno poi divisi singolarmente per stelle come nel caso precedente:

$$StatusArrayS = [StatusS?.star1||null, StatusS?.star2||null, StatusS?.star3||null,$$
$$StatusS?.star4||null, StatusS?.star5||null];$$
$$StatusArrayM = [StatusM?.star1||null, StatusM?.star2||null, StatusM?.star3||null,$$
$$StatusM?.star4||null, StatusM?.star5||null];$$
$$StatusArrayD = [StatusD?.star1||null, StatusD?.star2||null, StatusD?.star3||null,$$
$$StatusD?.star4||null, StatusD?.star5||null];$$
$$StatusArrayW = [StatusW?.star1||null, StatusW?.star2||null, StatusW?.star3||null,$$
$$StatusW?.star4||null, StatusW?.star5||null];$$

3. Vengono poi formati altri due *array*, uno che calcola la media delle stelle per categoria (*StatusAverage*) e una che calcola il numero totale di recensioni sempre per categoria (*StatusReview*):

$$StatusReview = [StatusS?.total_reviews||null, StatusM?.total_reviews||null,$$
$$StatusD?.total_reviews||null, StatusW?.total_reviews||null];$$
$$StatusAverage = [StatusS?.average||null, StatusM?.average||null,$$
$$StatusD?.average||null, StatusW?.average||null];$$

Una volta acquisiti tutti i dati necessari si è passato all'elaborazione dei tre grafici:

* Il primo, come detto nel capitolo precedente è un grafico a pila (figura 4.5) , ed è quindi il più complesso, perchè necessita della divisione per stelle secondo l'ascissa e per categoria secondo le ordinate. I dati passati sono rappresentati in figura 4.16, e si può notare che i *labels* generali sono le stelle (prima riga) e poi ogni categoria ha la propria, in questo caso *single*, sposato, divorziati e vedovi, in modo che si capisca per ogni persona che ha votato una stella a quale categoria appartiene.

```
labels: [t('1star'), t('2star'), t('3star'), t('4star'), t('5star')],
datasets: [
  {
    label: t('marital_status.single'),
    data: StatusArrayS,
    backgroundColor: 'rgb(0,128,128)',
  },
  {
    label: t('marital_status.married'),
    data: StatusArrayM,
    backgroundColor: 'rgb(0,0,128)',
  },
  {
    label: t('marital_status.divorced'),
    data: StatusArrayD,
    backgroundColor: 'rgb(0,0,255)',
  },
  {
    label: t('marital_status.widowed'),
    data: StatusArrayW,
    backgroundColor: 'rgb(0,255,255)',
  },
],
```

Figura 4.16: dati di istogramma a pila

* Il secondo grafico raffigura il numero totale di recensioni per categoria, avrà quindi nelle ascisse i valori dei filtri (*single*, sposati, divorziati e vedovi) e nelle ordinate il numero di recensioni. Nella figura 4.17 possiamo notare come sono passati i dati:

```
labels: [t('marital_status.single'), t('marital_status.married'), t('marital_status.divorced'), t('marital_status.widowed')],
datasets: [
  {
    label: t('charts.histogram.number_of_review'),
    data: statusReview,
    backgroundColor: 'rgb(0,128,128)',
  },
],
```

Figura 4.17: dati di istogramma con numero totale di recensioni

* Il terzo e ultimo grafico raffigura la media delle recensioni per categoria; come il precedente presenta i valori dei filtri nell'asse delle x (*single*, sposati, divorziati e vedovi) e in quella delle y la media delle stelle, come da figura 4.18:

```
labels: [t('marital_status.single'), t('marital_status.married'), t('marital_status.divorced'), t('marital_status.widowed')],
datasets: [
  {
    label: t('charts.histogram.average_rating'),
    data: StatusAverage,
    backgroundColor: 'rgb(0,128,128)',
  },
],
```

Figura 4.18: dati di istogramma con media delle recensioni

4.3.3 Problematiche incontrate

Durante lo sviluppo dell'obbiettivo OB04: "Possibilità di visualizzare statistiche e analisi per prodotti e marchi" della Tabella 2.1, ho riscontrato dei problemi con l'utilizzo degli *hook* di React, nello specifico con l'***hook "useEffect"***[4]. Uno degli errori comparsi, è riportato nella figura 4.19. Questo *hook* serve quando una componente deve svolgere un'azione dopo il rendering.

```
▶ Warning: Cannot update a component (`HashRouter`) while rendering a   react-dom.development.js:86
different component (`ChartReview`). To locate the bad setState() call inside `ChartReview`, follow
the stack trace as described in https://reactjs.org/link/setstate-in-render
    at ChartReview (https://localhost:3000/static/js/bundle.js:3993:66)
    at RenderedRoute (https://localhost:3000/static/js/bundle.js:165691:5)
    at Outlet (https://localhost:3000/static/js/bundle.js:166096:26)
    at RequireAuth (https://localhost:3000/static/js/bundle.js:17563:93)
    at RenderedRoute (https://localhost:3000/static/js/bundle.js:165691:5)
    at Outlet (https://localhost:3000/static/js/bundle.js:166096:26)
    at main
    at https://localhost:3000/static/js/bundle.js:24988:66
    at Container (https://localhost:3000/static/js/bundle.js:69420:19)
    at div
    at Layout (https://localhost:3000/static/js/bundle.js:8042:5)
    at RenderedRoute (https://localhost:3000/static/js/bundle.js:165691:5)
    at Routes (https://localhost:3000/static/js/bundle.js:166181:5)
    at App
    at OpenTelemetryProvider (https://localhost:3000/static/js/bundle.js:9925:5)
    at Router (https://localhost:3000/static/js/bundle.js:166119:15)
    at HashRouter (https://localhost:3000/static/js/bundle.js:164340:5)
    at CookieConsentProvider (https://localhost:3000/static/js/bundle.js:112734:15)
    at LocalizationProvider (https://localhost:3000/static/js/bundle.js:77359:19)
    at InnerThemeProvider (https://localhost:3000/static/js/bundle.js:69689:70)
    at ThemeProvider (https://localhost:3000/static/js/bundle.js:69021:5)
    at ThemeProvider (https://localhost:3000/static/js/bundle.js:69708:5)
    at Provider (https://localhost:3000/static/js/bundle.js:161866:5)
    at Suspense
    at Index (https://localhost:3000/static/js/bundle.js:23470:68)
```

Figura 4.19: Errore useEffect

[4] *UseEffect*. URL: https://react.dev/reference/react/useEffect.

Per usare *useEffect* è necessario inserire due argomenti: una funzione di *callback*[g]che contiene il codice da eseguire quando il componente viene reso o il valore della dipendenza cambia e un *array* delle dipendenze che specifica i valori che devono essere monitorati per le modifiche. La funzione di *callback* verrà eseguita quando un valore qualsiasi di questo *array* cambia.
Nel mio caso, la visualizzazione dei grafici doveva avvenire solo al completamento degli *endpoint* che richiedevano i dati per popolare i grafici.
Vediamo alcuni errori comuni di *useEffect* e le loro soluzioni:

* ***Array* di dipendenze mancante**: Un errore comune è quello di dimenticare di includere un *array* di dipendenze come secondo argomento del comando *useEffect* che può causare comportamenti indesiderati, come un eccessivo *re-rendering* o dati non aggiornati.
  - Soluzione: Fornire sempre un *array* di dipendenze a *useEffect*, anche se è vuoto. Includere tutte le variabili o i valori da cui dipende l'effetto. Questo aiuta React a determinare quando l'effetto deve essere eseguito o saltato.
* ***Array* di dipendenze non corretto**: Se l'*array* di dipendenze non è definito con precisione, l'effetto potrebbe non funzionare quando le dipendenze previste cambiano.
  - Soluzione: Assicurarsi di includere tutte le dipendenze necessarie nell'*array* di dipendenze. Se l'effetto dipende da più variabili, includerle tutte per attivare l'effetto quando una qualsiasi delle dipendenze cambia.
* ***Loop* infiniti**: La creazione di un *loop* infinito può verificarsi quando l'effetto modifica uno stato o un oggetto che dipende dall'effetto stesso. Ciò comporta che l'effetto venga attivato ripetutamente, causando un eccessivo *re-rendering* e potenzialmente il blocco dell'applicazione.
  - Soluzione: Assicurarsi che l'effetto non modifichi direttamente una dipendenza inclusa nel suo *array* di dipendenze. Al contrario, creare variabili separate o usare altre tecniche di gestione dello stato per gestire le modifiche necessarie.
* **Trascurare la pulizia**: Trascurare la pulizia degli effetti collaterali può portare a perdite di memoria o a un inutile consumo di risorse. Non ripulire i *listener* di eventi, gli intervalli o le sottoscrizioni può causare un comportamento inaspettato, soprattutto quando il componente si smonta.
  - Soluzione: Fornire sempre una funzione di pulizia nella dichiarazione di ritorno dell' *hook useEffect*.

Capitolo 5

Verifica e validazione

In questo capitolo vengono descritte le attività di verifica svolte al fine di garantire una buona qualità del prodotto.

5.1 Test

Durante lo sviluppo dell'applicazione è stata effettuata una continua attività di verifica e validazione ai fini di poter garantire un prodotto sicuro e conforme agli standard di qualità dell'azienda.
La maggior parte della **verifica statica** sul codice è stata fatta con le estensioni di *Visual Studio Code*; l'azienda mi ha infatti consigliato di usare questo *editor* e di scaricare le seguenti estensioni:

* ESlint;
* *prettier - Code formatter*;
* intellicode.

Queste estensioni forniscono già in fase di scrittura una sicurezza notevole per evitare errori di battitura, parenterizzazione, incompatibilità di tipi o inizializzazioni errate di variabili segnalando instantaneamente l'errore e fornendone una possibile soluzione.
Elenco di seguito alcune regole riguardo allo stile di scrittura del codice che abbiamo mantenuto in tutti i file:

* quando ci sono delle parentesi, quella di apertura deve essere allineata alla riga di codice a cui fa riferimento, quella di chiusura deve essere su una nuova linea;
* rimuovere gli spazi superflui;
* utilizzare gli spazi bianchi dove possono migliorare la leggibilità del codice. Per esempio:
  - prima e dopo ogni parentesi;
  - prima e dopo il carattere di unione (+ nel caso di Javascript);
  - dopo ogni virgola;
  - prima e dopo ogni carattere matematico o di confronto.
* indentare correttamente per rendere chiari i nodi.

La **verifica dinamica** e quindi la correttezza del flusso dell'informazione è stata verificata tramite numerosi test effettuati in parallelo alla fase di sviluppo cercando di simulare ogni possibile azione. Ogni componente React ha quindi il suo test dedicato, una *task* non poteva essere completata se non erano presenti e aggiornati tutti i test dell'applicazione. Ogni singolo test doveva passare, come da immagine 5.1.

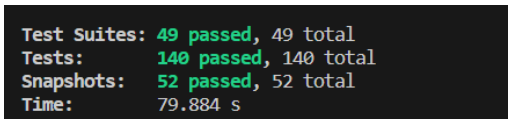

Figura 5.1: Risultato test

I test sono stati scritti grazie a *React Test Library* e *Jest*[1] che sono preconfezionati con l'app *Create React* e aderiscono al principio guida secondo cui le app di test dovrebbero assomigliare a come verrà utilizzato il software.
Inoltre venivano controllati anche gli *snapshot*, tradotto in italiano istantanea, ossia la "fotografia" di un oggetto in un determinato momento temporale. In particolare, lo *snapshot* in informatica riguarda lo stato del sistema IT in una fase in modo che se si vogliano fare delle variazioni e non vadano a buon fine si possa ritornare alla situazione precedente. Se quindi quanto scritto nel test, non combacia con lo *snapshot*, il test viene considerato fallito come mostrato in figura 5.2.

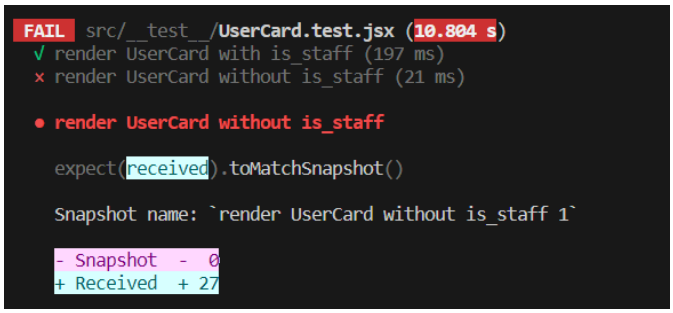

Figura 5.2: Snapshot fallito

Come si può notare, un singolo file può avere più test a lui dedicati, e possono fallire singolarmente. In questo caso le *UserCard* venivano mostrate correttamente se l'utente era amministratore e invece risultavano sbagliate quando non lo era. Questo dimostra che testando parallelamente alla codifica si capisce più velocemente dove è il problema e se tutto funziona correttamente.
Ogni nuova funzionalità dell'applicativo (e quindi *task*) è stata inoltre testata manualmente da due membri del *team* prima di essere confermata. Hanno verificato che l'applicativo facesse quanto richiesto, che l'utente venisse avvertito nel caso in cui inserisse dei campi sbagliati o facesse azioni non concesse (come provare a eliminare l'ultimo utente amministratore). Hanno controllato inoltre che tutti i link funzionassero correttamente e che l'applicazione rispettasse le WCAG 2.0 (descritte al capitolo 2.4.1) e che fosse quindi accessibile al maggior numero di utenti possibili.

[1] *Jest*. URL: https://jestjs.io/docs/jest-community.

5.2 Collaudo

Dopo aver implementato e superato i test definiti, l'applicazione è stata collaudata nei seguenti *web browser* :

* Google Chrome v. 111
* Firefox v.114
* Safari v.14
* Edge v.110

Sono stati effettuati quindi i test di accettazione, o collaudo, per accettarsi che il prodotto rispettasse i requisiti richiesti e non presentasse malfunzionamenti o anomalie particolari. Il collaudo è stato effettuato con esito positivo dal tutor aziendale e dagli altri membri del *team* su dispositivi con sistemi operativi sia Mac che Windows.

Capitolo 6

Conclusioni

Il lavoro svolto per la S.AI - Synthema artificial intelligience prevedeva in sintesi l'inserimento in un'applicazione preesistente (*FoodAdvisor*) di una sezione dedicata alle aziende produttrici, per permettere di aggiungere informazioni sui loro prodotti e sui marchi e di accedere ad analisi e statistiche relative ai loro prodotti. Inoltre ogni azienda doveva poter avere più utenti associati e con funzionalità diverse in base al loro grado all'interno di essa (amministratore o no).
Lo sviluppo è stato fatto rispettando gli *standard* interni dell'azienda elencati nel capitolo 2.2.1.

6.1 Analisi del lavoro e conoscenze aquisite

Come concordato con l'azienda ospitante, lo stage è stato effettuato da remoto, utilizzando gli appositi strumenti organizzativi senza riscontrare particolari problemi. Questa modalità di svolgimento del lavoro, anche se avrebbe potuto creare dei potenziali problemi logistici, mi ha permesso di guadagnare maggiore autonomia nell'apprendimento teorico e nell'organizzazione e svolgimento di attività.
Per quanto riguarda l'implementazione della piattaforma gli strumenti di sviluppo sono risultati più che sufficienti per portare a termine il lavoro. In particolare ho avuto la possibilità di approfondire l'IDE Visual Studio Code, il quale offre un'enorme quantità di estensioni utili per uno sviluppo software efficace ed efficiente, soprattutto riguardo all'auto-completamento del codice.
Infine riguardo ai prodotti attesi al termine del progetto, ovvero la parte dedicata alle aziende della piattaforma FoodAdvisor, i test di unità delle componenti realizzate e la documentazione delle componenti e delle funzioni realizzate, posso dire che sono risultati all'altezza delle aspettative da parte del capo progetto.
Per quanto riguarda gli standard aziendali sui comandi git sono stati rispettati tutti, e lo stage è stato un'occasione importante per apprendere al meglio il software git, che essendo utilizzato in moltissime aziende informatiche, sarà una competenza sicuramente spendibile in un futuro mondo lavorativo.

6.2 Raggiungimento degli obiettivi

Gli obiettivi prefissati all'inizio dello stage illustrati nel capitolo 2.3 risultano quasi completamente soddisfatti.
Di seguito viene illustrata la tabella riassuntiva.

	Obbligatori	
OB01	Possibilità di creare e configurare un'azienda	soddisfatto
OB02	Possibilità di creare e configurare prodotti e marchi dell'azienda	soddisfatto
OB03	Possibilità di visualizzare e ricercare marchi e prodotti dell'azienda	soddisfatto
OB04	Possibilità di visualizzare statistiche e analisi per prodotti e marchi	soddisfatto
	Desiderabili	
DE01	Possibilità di associare più utenti ad una azienda	soddisfatto
DE02	Possibilità di caricare informazioni relative a prodotti e marchi dirette all'utente finale	soddisfatto
DE03	Possibilità per l'utente finale di vedere le informazioni caricate dal produttore	non soddisfatto
	Opzionali	
OP01	Possibilità di avere ruoli separati per la gestione delle aziende	soddisfatto
OP02	Possibilità di avere un'interfaccia amministrativa per l'autorizzazione di alcune operazioni (per es. associare marchi e prodotti ad un'azienda)	soddisfatto

Tabella 6.1: Tabella di riepilogo dello stato degli obiettivi

Tutti gli obbiettivi obbligatori sono stati soddisfatti ma OB04 ha richiesto più tempo del previsto a causa di qualche problema riscontrato. Pertanto non sono riuscita a soddisfare tutti i requisiti desiderabili. L'azienda si ritiene comunque soddisfatta del mio risultato poichè oltre ad aver portato a termine i requisiti obbligatori ho rispettato tutte le regole e i tempi da loro imposti, ed ho presentato sempre codice chiaro e leggibile, senza bisogno di riscriverlo.

6.3 Valutazione personale

L'attività di stage oltre a farmi apprendere nuove tecnologie e approfondirne altre già conosciute, mi ha permesso di conoscere meglio le mie capacità nell'ambito lavorativo, permettendomi un miglioramento nella gestione delle tempistiche e nella capacità di lavorare in gruppo. Lavorare all'interno di un team molto giovane e dinamico, in cui ci si confronta quotidianamente e in maniera costruttiva per trovare soluzioni alternative e innovative ai problemi che inevitabilmente fanno parte di una realtà aziendale, è stato molto incoraggiante e motivante.
Ho imparato inoltre a comunicare in modo più efficace non solo le mie idee ma anche le mie difficoltà, in modo che altri potessero aiutarmi e consigliarmi.
Questo progetto di stage mi ha permesso di sperimentare, mettermi alla prova e utilizzare la mia creatività per realizzare un prodotto finale che incontrasse le aspettative dell'azienda.
Valuto dunque molto positivamente l'esperienza di stage svolta presso S.AI poiché mi ha permesso di entrare in contatto con un ambiente lavorativo reale, di mettere a frutto le mie conoscenze pregresse e svilupparne di nuove.

Glossario

BACKLOG Il *backlog* è una serie di operazioni in attesa di essere eseguite.. 30

CLIENT-SIDE In un'architettura software di tipo client-server fa riferimento a tutte le operazioni compiute dal client.. 14

ENDPOINT Una comunicazione *endpoint* è un nodo di comunicazione rilevabile la cui portata può essere variata per restringere o ampliare la zona di ricerca. Gli endpoint favoriscono uno strato di astrazione programmabile per cui sistemi software e/o sottosistemi possono comunicare tra di loro, inoltre i mezzi di comunicazione sono disaccoppiati dai sottosistemi di comunicazione. 32

FRAMEWORK Un framework (anglicismo che può essere tradotto come struttura o quadro strutturale) è un'architettura logica di supporto sulla quale un software può essere progettato e realizzato, spesso facilitandone lo sviluppo da parte del programmatore.. 7

FUNZIONE DI *CALLBACK* La funzione di *callback* è una funzione che viene passato come parametro a un'altra funzione. In particolare, quando ci si riferisce alla *callback* richiamata da una funzione, la *callback* viene passata come argomento ad un parametro della funzione chiamante. In questo modo la chiamante può realizzare un compito specifico (quello svolto dalla *callback*) che non è, molto spesso, noto al momento della scrittura del codice.. 39

IA L'intelligenza artificiale è una disciplina che studia se e in che modo si possano realizzare sistemi informatici intelligenti in grado di simulare la capacità e il comportamento del pensiero umano.. 51

LINGUAGGIO AD ALTO LIVELLO Un linguaggio di programmazione ad alto livello, in informatica, è un linguaggio di programmazione caratterizzato da una significativa astrazione dai dettagli del funzionamento di un calcolatore e dalle caratteristiche del linguaggio macchina. Il livello di astrazione definisce quanto sia di "alto livello" un linguaggio di programmazione.. 15

LINGUAGGIO DI *MARKUP* Un linguaggio di *markup* è un insieme di regole che descrivono i meccanismi di rappresentazione (strutturali, semantici o presentazionali) di un testo che, utilizzando convenzioni standardizzate, sono utili su più supporti.. 11

LINGUAGGIO DI *SCRIPTING* Linguaggio di programmazione interpretato destinato in genere a compiti di automazione del sistema operativo o delle applicazioni, o a essere usato nella programmazione web all'interno di pagine web. . 13

NLP In informatica con il termine *Natural Language Processing NLP* (ing. elaborazione del linguaggio naturale) si intendono algoritmi di intelligenza artificiale in grado di analizzare, rappresentare e quindi comprendere il linguaggio naturale. Le finalità possono variare dalla comprensione del contenuto, alla traduzione, fino alla produzione di testo in modo autonomo a partire da dati o documenti forniti in input.. 51

PROPS Le *props* sono tutti quegli oggetti, funzioni, stringhe, numeri, *array*, ecc. che vogliamo inviare ad un componente React. Le *props* devono essere *read-only*. Ciò sta a significare che un componente non deve mai modificare le *props* che riceve in *input*.. 31

REST Representational state transfer (REST) è uno stile architetturale per sistemi distribuiti. Il termine REST rappresenta un sistema di trasmissione di dati su HTTP senza ulteriori livelli. I sistemi REST non prevedono il concetto di sessione, ovvero sono *stateless*. L'architettura REST si basa su HTTP. Il funzionamento prevede una struttura degli URL ben definita che identifica univocamente una risorsa o un insieme di risorse e l'utilizzo dei metodi HTTP specifici per il recupero di informazioni (*GET*), per la modifica (*POST, PUT, PATCH, DELETE*) e per altri scopi (*OPTIONS*, ecc.).. 51

SERVER-SIDE In un'architettura software di tipo client-server fa riferimento a tutte le operazioni compiute dal server.. 13

WCAG Le *Web Content Accessibility Guidelines* sono parte di una serie di linee guida per l'accessibilità dei siti Web,fanno parte del World Wide Web Consortium (W3C). Gli sviluppatori di contenuti, di strumenti di sviluppo e di strumenti di valutazione dell'accessibilità possono seguire queste linee guida, per creare e valutare contenuti Web accessibili per dispositivi hardware o software limitati.. 7, 51

WEB CRAWLING Il *web crawling* (ing. scansione del web) è il processo di analisi di siti web utilizzato per indicizzarne tutti i contenuti. Un *crawler* (ing. colui che scansiona), è un software specializzato per prelevare tutto il contenuto di una pagina web e seguirne i vari link per analizzare siti web collegati o pagine secondarie. Molti crawler forniscono impostazioni per controllare e limitare l'esecuzione.
Il caso d'uso più comune dei crawler è l'utilizzo da parte dei motori di ricerca: le pagine trovate dal crawler vengono indicizzate per fornire risultati attinenti alla query di ricerca dell'utente che utilizza il servizio.. 1

W3C Il *World Wide Web Consortium* è un'organizzazione non governativa internazionale che ha come scopo quello di favorire lo sviluppo di tutte le potenzialità del *World Wide Web* e diffondere la cultura dell'accessibilità della Rete.. 51

Acronimi

CSS Cascading Style Sheets. 12

HTML HyperText Markup Language. 11

IA Intelligenza Artificiale. 1, 49

NLP Natural Language Processing. 1, 50

PHP Hypertext Preprocessor. 13

REST Representational state transfer. 27, 50

W3C World Wide Web Consortium. 12, 50

WCAG Web Content Accessibility Guidelines. 50

Bibliografia

Siti web consultati

Architettura django. URL: https://www.programmareinpython.it/blog/che-cosa-rende-django-speciale-miglior-web-framewo/ (cit. a p. 27).

chartJs-2. URL: https://react-chartjs-2.js.org (cit. a p. 35).

CSS. URL: https://it.wikipedia.org/wiki/CSS (cit. a p. 12).

Gitlab. URL: https://about.gitlab.com/ (cit. a p. 19).

HTML 5. URL: https://www.html.it/guide/guida-html5/ (cit. a p. 11).

JavaScript. URL: https://it.wikipedia.org/wiki/JavaScript (cit. a p. 14).

Jest. URL: https://jestjs.io/docs/jest-community (cit. a p. 42).

Jira. URL: https://www.atlassian.com/it/software/jira (cit. a p. 19).

Modello Scrum. URL: https://www.agileway.it/scrum-metodologia-agile/ (cit. a p. 29).

MUI. URL: https://mui.com/ (cit. a p. 17).

MySQL. URL: https://www.mysql.com/it/ (cit. a p. 15).

PHP. URL: https://www.php.net/ (cit. a p. 13).

Phthon. URL: https://it.wikipedia.org/wiki/Python (cit. a p. 15).

React. URL: https://react.dev/ (cit. a p. 16).

Redux. URL: https://redux.js.org/ (cit. a p. 18).

S-AI: Synthema Artificial Intelligence. URL: https://www.s-ai.it/ (cit. a p. 1).

Slack. URL: https://slack.com/intl/it-it (cit. a p. 19).

UseEffect. URL: https://react.dev/reference/react/useEffect (cit. a p. 38).

W3C. URL: https://www.w3.org/ (cit. a p. 12).

WCAG 2.0. URL: https://www.w3.org/TR/WCAG20/ (cit. a p. 8).