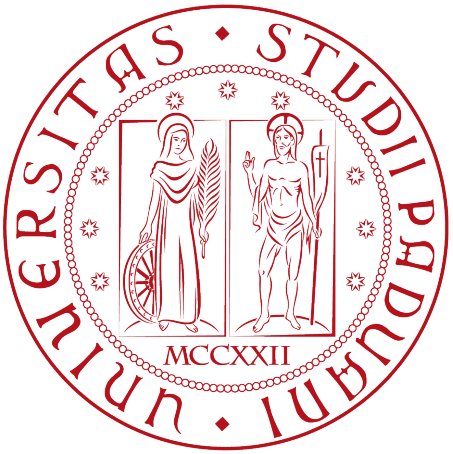
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica



Analisi di dati per la profilazione degli utenti e promozione di attività commerciali

Tesi di laurea

Relatore

Prof. Francesco Ranzato

Laureando

Amedeo Meggiolaro

Anno Accademico 2021-2022

Sommario

Questa tesi di laurea riporta i risultati ottenuti dal laureando Amedeo Meggiolaro presso l'azienda Unipiazza S.R.L. nel corso delle trecento ore di tirocinio formativo.
L'azienda necessitava di un algoritmo in grado di analizzare i dati degli utenti e proporgli delle attività commerciali a loro affini.
Per questo motivo è stato fatto un lavoro di analisi dei dati inerenti al database in modo da creare sei diversi profili di utente tipo, denominati personas, basati sui dati raccolti dagli utilizzatori del servizio fedeltà di Unipiazza.
In seguito è stato creato un algoritmo in grado di analizzare i dati di un singolo utente e di assegnargli un punteggio, in modo da poter verificare a quale profilo di persona fosse più attinente.
Grazie alla profilazione degli utenti, l'algoritmo è capace di promuovere specifiche attività commerciali a loro affini.

Ringraziamenti

Voglio esprimere la mia gratitudine al Prof. Francesco Ranzato, relatore della mia tesi, per la disponibilità dimostrata e il supporto fornitomi durante la stesura del lavoro.

Desidero ringraziare con affetto i miei genitori, i miei fratelli e Giulia per il loro importante sostegno durante questi anni di studio.

Desidero ringraziare infine i miei amici, in particolare Alessandro, per questi meravigliosi anni di università passati assieme.

Padova, Dicembre 2022 Amedeo Meggiolaro

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Questo capitolo fornisce un resoconto delle norme tipografiche utilizzate per la stesura del documento, una presentazione dell'ente ospitante e una descrizione del contenuto del documento suddivisa per capitoli.

1.1 Convenzioni tipografiche

Per quanto riguarda la stesura del testo relativa al documento, sono state adottate le seguenti convenzioni tipografiche:

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

1.2 L'ente ospitante

Figura 1.1: Logo Unipiazza S.R.L.

Unipiazza S.R.L. è una *Startup*[g] Innovativa Padovana che lavora nel campo dei Sistemi Fedeltà Digitali. La piattaforma fedeltà permette alle PMI di acquisire e analizzare dati sui propri clienti per farli tornare più spesso all'interno del punto vendita. Utilizzata in 300 locali su 9 principali città diverse da oltre 200 mila utenti, Unipiazza aiuta le attività locali fornendogli gli stessi strumenti digitali tipici della grande distribuzione. Grazie alla natura giovane dell'azienda, Unipiazza testa e sviluppa costantemente nuove funzionalità tramite i *feed-back*[g] del reparto commerciale, in contatto quotidiano con il team di sviluppo.

1.3 Organizzazione del testo

Il secondo capitolo descrive il progetto di stage, gli obiettivi stabiliti col tutor aziendale e la sua pianificazione.

Il terzo capitolo approfondisce gli strumenti e le tecnologie utilizzate durante la progettazione e lo sviluppo del progetto.

Il quarto capitolo espone l'analisi di dati fatta sul *database*[g] per ottenere i dati necessari al successivo sviluppo dell'*algoritmo*[g] .

Il quinto capitolo descrive la progettazione dell'*algoritmo* ottenuta sulla base dei dati raccolti in fase di analisi e il funzionamento dello stesso.

Il sesto capitolo tratta la verifica, analisi e validazione del codice prodotto e dell'*algoritmo* sviluppato.

Il settimo capitolo espone le conclusioni tratte in seguito al lavoro svolto durante il tirocinio formativo.

Capitolo 2

Descrizione dello stage

Il capitolo seguente contiene una descrizione generale del progetto di stage. Include inoltre gli obiettivi e la pianificazione stabiliti in accordo col tutor aziendale e revisionati dal relatore interno.

2.1 Introduzione al progetto

Vista la grande mole di dati raccolta quotidianamente nel *database* dell'azienda, Unipiazza S.R.L. necessitava di un *algoritmo* capace di analizzare i dati dei singoli utenti con lo scopo di proporre attività commerciali a loro affini.
Per questo motivo in una prima fase è stato effettuato un lavoro di profilazione degli utenti tramite l'analisi dei dati in modo tale da poter generare sei diversi profili, denominati *personas*, legati a sei diversi tipi di utente utilizzatore del sistema di fedeltà Unipiazza. Le sei *personas* delineate sono rispettivamente: maschio giovane, adulto e senior, femmina giovane, adulta e senior. Per ognuno di questi profili è stato fatto un lavoro di analisi, in modo da poter calcolare per ciascuno la distanza media percorsa tra le attività visitate, le prime tre categorie di locale maggiormente frequentate e altri dati utili alla profilazione.
In una seconda fase è stato creato un *algoritmo* in grado di analizzare i dati di un singolo utente come genere, età, distanza media percorsa e categoria di locali più frequentati, in modo da assegnare un punteggio ai diversi profili di persona per ogni caratteristica similare tra l'utente analizzato e le *personas* definite in precedenza.
Se ad esempio l'utente passato in *input*[g] all'*algoritmo* è un maschio, viene sommato un determinato valore (grande o piccolo in base all'importanza della caratteristica analizzata) al punteggio totale dei profili di maschio giovane, adulto e senior. Il punteggio temporaneo delle *personas* si sommerà ad altri valori in base alle successive specifiche dell'utente valutate. Dopo aver analizzato tutti i dati dell'utente l'*algoritmo*, selezionando il profilo di *persona* che ha totalizzato il maggior numero di punti, suggerirà all'utente la categoria di locale che ritiene per esso più pertinente.
Precedentemente all'analisi di dati e alla creazione dell'*algoritmo* sono stati svolti diversi lavori utili a prendere coscienza dei progetti di Unipiazza e migliorarne l'utilizzo da parte degli utenti. Tuttavia, non essendo questi di rilevante importanza per quanto concerne lo sviluppo dell'*algoritmo* (fine ultimo del progetto di stage), saranno citati in maniera più marginale.

2.2 Obiettivi

2.2.1 Notazione

Si farà riferimento ai requisiti secondo le seguenti notazioni:

* O per i requisiti obbligatori, vincolanti in quanto obiettivo primario richiesto dal committente;
* D per i requisiti desiderabili, non vincolanti o strettamente necessari, ma dal riconoscibile valore aggiunto;
* F per i requisiti facoltativi, rappresentanti valore aggiunto non strettamente competitivo.

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

2.2.2 Obiettivi fissati

Il progetto prevedeva lo svolgimento dei seguenti obiettivi:

* **Obbligatori**
  - O01: Capacità di organizzazione e lavoro in gruppo, destrezza nell'utilizzo degli strumenti già utilizzati dal team (*Github*[g] , *Trello*[g] , *Slack*[g] ..);
  - O02: Presa coscienza della complessità di un progetto tecnologicamente variegato e in produzione da ormai 6 anni;
  - O03: Raggiungere una buona capacità a integrarsi con codice già scritto da altri;
  - O04: Scrittura di codice utile ad aumentare le funzionalità che Unipiazza offre ai suoi clienti;
  - O05: Trovare soluzione per la funzionalità Unipiazza Bite.
* **Desiderabili**
  - D01: Ricerca di soluzioni e *algoritmi* per poter sfruttare in maniera efficace i dati disponibili da Unipiazza.
* **Facoltativi**
  - F01: Studio di un nuovo ambiente di lavoro (*Android*[g]).

2.3 Pianificazione del lavoro

2.3.1 Pianificazione settimanale

* **Prima settimana**
  - Introduzione a Unipiazza, il servizio fedeltà, il suo funzionamento e la struttura organizzativa interna;
  - Presa visione degli strumenti utilizzati per lo *smart working*[g] ;
  - Panoramica dei prodotti tecnologici creati dal team Unipiazza;
  - Presa visione e formazione sulle tecnologie coinvolte;
  - Configurazione dell'ambiente di lavoro.
* **Seconda settimana**
  - Configurazione e prima esecuzione progetti web in *AngularJs*[g] , *React*[g] e *back-end*[g] in *Nodejs*[g] ;
  - Studio e presa visione del sistema di rilascio e sviluppo dell'ambiente production e staging;
  - Esercizio di riconoscimento ed eliminazione di utenti "fake" attualmente nel *database*. Scelta di come si possano eliminare (task *nodejs* o a mano dal *front-end*[g]).
* **Terza settimana**
  - Revisione e breve report dell'attuale implementazione del *server*[g] di *cache*[g] *Redis*[g] ;
  - "Caccia" alle chiamate *http*[g] troppo lente sui gestionali e applicazioni. Lo studente dovrà proporre se raffinare la *query*[g] sul *back-end* oppure aumentare l'utilizzo di *Redis*.
* **Quarta settimana**
  - Test funzionalità "Promozioni" su chiosco Unipiazza e applicazione *Android*;
  - Implementare la modalità "multishop" sulla funzionalità promozioni con interventi al *back-end*.
* **Quinta, sesta e settima settimana**
  - Analisi dati sulla grande quantità di dati disponibili nei *database* di Unipiazza, proposte per il loro utilizzo.
* **Ottava, nona e decima settimana**
  - Studio e test di una possibile prossima funzionalità di Unipiazza, gli Unipiazza bite;
  - Creazione *algoritmo* Unipiazza bite che studia i comportamenti degli utenti e promuove possibili attività commerciali a loro affini.
* **Undicesima settimana**
  - Simulazione dell'*algoritmo* con i dati di production e studio dei risultati.

- **Dodicesima settimana - Conclusione**
  - Analisi dei risultati dell'*algoritmo* delle settimane precedenti con tutto il team Unipiazza;
  - Inserimento di un proprio *easter egg*[g] nascosto in una delle varie applicazioni Unipiazza.

2.3.2 Pianificazione oraria

La pianificazione, in termini di quantità di ore di lavoro, è stata distribuita nella maniera seguente:

Tabella 2.1: Tabella pianificazione oraria

Durata in ore	**Descrizione dell'attività**
10	**Formazione sulle tecnologie**
10	**Visione architettura di riferimento e relativa documentazione**
240	**Definizione ed esecuzione del lavoro**
40	*Analisi dei requisiti delle singole task*
50	*Progettazione e discussione delle soluzioni*
35	*Analisi Dati*
90	*Esecuzione*
10	*Stesura documentazione relativa ad analisi e progettazione*
15	*Integrazione con il codice esistente*
40	**Collaudo Finale**
30	*Collaudo e Test*
10	*Stesura documentazione finale*
Totale ore	**300**

Capitolo 3

Tecnologie utilizzate

In questo capitolo vengono descritti gli strumenti e le tecnologie che sono stati adoperati durante la progettazione e il conseguente sviluppo del progetto.

3.1 Ambiente di sviluppo

L'intero progetto è stato sviluppato sul *sistema operativo*[g] *Ubuntu 20.04.5 LTS*.

3.1.1 VSCodium

Figura 3.1: Logo VSCodium

VSCodium è una distribuzione guidata dalla comunità e con licenza gratuita dell'editor *Visual Studio Code* di Microsoft.
Visual Studio Code è un editor di codice semplificato con supporto per operazioni di sviluppo come il *debug*[g] , l'esecuzione di attività e il controllo di versione. Mira a fornire solo gli strumenti di cui uno sviluppatore ha bisogno per un ciclo rapido di compilazione e *debug* del codice e lascia flussi di lavoro più complessi a *IDE*[g] con funzionalità più complete, come *Visual Studio IDE*.

3.2 Linguaggi e framework

3.2.1 JavaScript

Figura 3.2: Logo JavaScript

JavaScript è un linguaggio di programmazione orientato agli eventi, comunemente utilizzato nella programmazione Web lato *client* ma esteso in seguito anche al lato *server*. Questo linguaggio è necessario alla creazione di effetti dinamici interattivi, innescati sull'applicazione web dall'utente tramite l'invocazione di funzioni di *script* (ad esempio attraverso l'uso di mouse o tastiera). All'interno dei progetti Unipiazza è adoperato per lo sviluppo del *front-end* tramite l'utilizzo del *framework*[g] *React* e nel *back-end* con *Node.js*.

3.2.2 Node.js

Figura 3.3: Logo Node.js

Node.js è un *runtime environment*[g] *JavaScript* costruito sul motore JavaScript V8 di *Google Chrome*, progettato per creare applicazioni di rete scalabili. Consente quindi di utilizzare *JavaScript* per scrivere codice da eseguire lato *server*, ad esempio per la creazione di pagine dinamiche. *Node.js* si basa sul concetto di eventi, permettendo di gestire le operazioni in modo asincrono, non bloccante. Questo approccio porta enormi benefici ai tempi di esecuzione, permettendo di gestire un'enorme quantità di richieste al secondo e ottimizzando le risorse a disposizione. I progetti di Unipiazza adoperano questo strumento per lo sviluppo del codice adibito al *back-end*.

3.2.3 React

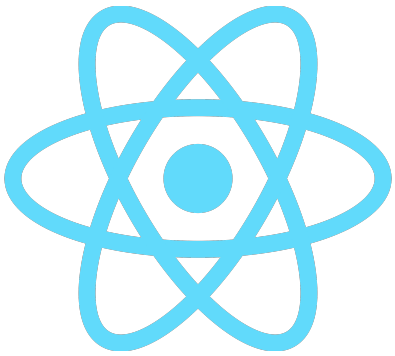


Figura 3.4: Logo React

React è una libreria *JavaScript* per la creazione di interfacce utente che consente di sviluppare applicazioni dinamiche, le quali non necessitano di ricaricare la pagina per visualizzare i dati modificati. Nelle applicazioni sviluppate con *React* inoltre, le modifiche effettuate sul codice si possono visualizzare in tempo reale; ciò permette uno sviluppo rapido, efficiente e flessibile delle applicazioni web. Le piattaforme di Unipiazza utilizzate rispettivamente da partner e utenti adoperano questa tecnologia per lo sviluppo della propria interfaccia utente.

3.2.4 Angular

Figura 3.5: Logo Angular

Angular è un *framework open source* atto allo sviluppo del *front-end* di applicazioni web, il linguaggio principalmente usato per lo sviluppo con questo *framework* è *TypeScript*[g] . Questa tecnologia permette di implementare progetti con immediati vantaggi in termini di robustezza del codice, testabilità e manutenibilità, creando applicazioni veloci e performanti. *Angular* mette a disposizione un ambiente per la creazione veloce di app, moduli e componenti; inoltre il supporto degli *IDE* e i numerosi *plugin* facilitano la stesura del codice. La piattaforma di Unipiazza utilizzata internamente

all'azienda per gestire i negozi dei propri partner adopera questo *framework* per lo sviluppo dell'interfaccia utente.

3.3 Database

3.3.1 MongoDB

Figura 3.6: Logo MongoDB

MongoDB è un *DBMS*[g] non relazionale orientato ai documenti. Classificato come *database* di tipo *NoSQL*[g] , *MongoDB* si allontana dalla struttura tradizionale basata su tabelle dei *database* relazionali in favore di documenti in stile *JSON*[g] con schema dinamico (*MongoDB* chiama il formato *BSON*), rendendo l'integrazione di dati di alcuni tipi di applicazioni più facile e veloce. I *database* dei progetti Unipiazza vengono gestiti con questa tecnologia, di conseguenza è stata adoperata per l'esecuzione delle *query* necessarie all'analisi dei dati durante il progetto.

3.4 Strumenti per il debug

3.4.1 Postman

Figura 3.7: Logo Postman

Postman è una piattaforma per la creazione e l'utilizzo di *API*[g] . Questo strumento ha permesso di testare le *API* in maniera più rapida ed efficiente durante lo sviluppo di determinate funzionalità per le piattaforme web di Unipiazza.

3.5 Strumenti per il versionamento

3.5.1 Git

Figura 3.8: Logo Git

Git è un *VCS*[g] *open source*[g] progettato per gestire il versionamento di codice con velocità ed efficienza. I progetti di Unipiazza adoperano questo strumento ed è stato utilizzato anche per il versionamento del codice durante il progetto di stage.

3.5.2 GitHub

Figura 3.9: Logo GitHub

GitHub è uno strumento web di *version control Git*. Usando questo *tool* i programmatori possono lavorare in modo coordinato sulla stessa base di codice, pur sviluppando in maniera indipendente. *GitHub* offre funzionalità di *hosting*[g] e revisione del codice, commenti e *feedback*, collaborazione e gestione del team. I programmatori vengono aggiornati in tempo reale sull'evoluzione del progetto, inoltre è possibile ripercorrere l'intera storia del codice e ripristinarne una versione precedente grazie al salvataggio di ogni modifica, *branch* e *fork* effettuati.

3.6 Strumenti per la comunicazione

3.6.1 Slack

Figura 3.10: Logo Slack

Slack è un'applicazione web di messaggistica per le aziende. Grazie alla possibilità di scrivere alla singola persona o creare canali riguardanti specifici argomenti, ha permesso di velocizzare il processo di comunicazione con il tutor aziendale e gli altri membri dell'azienda, anche in caso di necessità per l'invio di *file*[g] .

3.6.2 Gather

Figura 3.11: Logo Gather

Gather è una piattaforma online che permette di interagire in uno spazio virtuale con altri utenti. Grazie a questo strumento, l'organizzazione di riunioni e l'interazione con i membri del team risultano rapide e semplificate. Avendo lavorato in *smart working* per l'intero periodo dello stage la piattaforma, utilizzata per la creazione di un ufficio virtuale, si è rivelata fondamentale per un'interazione immediata col tutor aziendale.

3.7 Strumenti per la pianificazione delle attività

3.7.1 Trello

Figura 3.12: Logo Trello

Trello è uno strumento visivo che consente la gestione di qualsiasi tipo di progetto, flusso di lavoro e monitoraggio delle attività. Grazie alla creazione di una bacheca strutturata per spostare le attività visivamente attraverso diverse fasi di sviluppo, permette di organizzare al meglio il proprio flusso di lavoro. All'interno del progetto di stage, questo strumento è stato impiegato per pianificare le attività in maniera ordinata ed efficiente.

Capitolo 4

Analisi dei dati

Questo capitolo espone approfonditamente il lavoro di analisi dei dati eseguito sul database per la profilazione degli utenti tipo, denominati personas. I dati raccolti in questa fase del progetto sono stati in seguito adoperati per la progettazione e lo sviluppo di un algoritmo, adibito alla promozione di attività commerciali.

4.1 Introduzione

Al fine di progettare un *algoritmo* capace di promuovere attività commerciali basandosi sull'analisi dei dati di un singolo utente, si è resa necessaria la creazione di un bacino di utenti tipo da cui esso potesse attingere, andando ad analizzare le somiglianze tra questi profili e lo specifico utente oggetto di studio. Per questo motivo è stato svolto un lavoro di identificazione e catalogazione degli utenti tipo, denominati *Personas*.

4.2 Personas

Per delineare i sei profili di *personas* ottenuti al termine di questo processo, è stato svolto un lavoro di analisi dei dati degli utenti di Unipiazza tramite l'esecuzione di *queries* sul *database*.
Tutte le interrogazioni al *database* sono state fatte basandosi sui dati degli utenti attivi nella provincia di Padova. Inizialmente erano state eseguite *queries* filtrando gli utenti di tutta Italia; vedendo tuttavia che i dati erano potenzialmente sfalsati poiché Padova era la città con la maggior varietà di categorie dei negozi (al contrario di altre città italiane) e visto l'alto costo computazionale, si è optato per analizzare i dati dei soli utenti padovani attivi negli ultimi due anni (per un totale di circa tredicimila utenti).
In primis si è svolto uno studio al fine di ottenere il genere e l'età media dei maggiori usufruitori del servizio fedeltà. Ciò che è emerso da queste prime *queries* è stato il fatto che, nonostante le femmine superassero i maschi per numero, i valori non fossero marcatamente differenti, di conseguenza si è optato per la creazione di profili di ambo i generi (maschile e femminile). Da una seconda analisi sull'età degli utenti si è osservato che, nonostante giovani (sotto i 34 anni) e adulti (dai 35 ai 54 anni) formassero la parte più rilevante degli utenti totali di Unipiazza, anche la popolazione senior (oltre i 55 anni) costituiva un'importante fetta di mercato.
Per questo motivo si è deciso di creare sei diversi profili di *personas*: maschio giovane,

maschio adulto, maschio senior, e i corrispettivi femminili, femmina giovane, femmina adulta e femmina senior.

4.3 Queries

Questa sezione descrive le ulteriori interrogazioni che sono state eseguite sul *database* al fine di approfondire il livello di dettaglio dei diversi profili di *personas*, migliorando di conseguenza l'efficienza di profilazione del successivo *algoritmo* che è stato sviluppato. Tutte le *queries* elencate in seguito sono state eseguite singolarmente per ogni profilo di *persona*, al fine di raccogliere i dati necessari.

4.3.1 Analisi su categorie di attività maggiormente frequentate

Una delle prime *queries* sviluppate è servita a conoscere in maniera più precisa quali fossero le prime tre categorie di negozi maggiormente visitate per *persona*. Questa analisi si è resa utile per permettere all'*algoritmo* sviluppato successivamente di controllare se vi fosse un'affinità tra le attività frequentate dall'utente oggetto di studio e i profili di *personas* identificati. Per creare la *query* sono stati selezionati tutti gli utenti nella provincia di Padova con almeno una ricevuta generata entro gli ultimi due anni, in modo tale da selezionare solamente gli utenti attivi in tempi recenti. Dopo questa selezione, sono stati estratti tutti i negozi in cui gli utenti avevano effettuato almeno una transazione. Analizzando la categoria delle attività commerciali filtrate (ad esempio bar, ristorante, copisteria, panificio) e sommandola ad un contatore si è potuto verificare quali fossero le categorie di locali maggiormente visitate per fascia di età e genere. Questo dato si è rivelato fondamentale in quanto era presente una marcata differenza tra i risultati dei diversi profili di *personas*, è stato quindi utilizzato successivamente durante la progettazione e lo sviluppo dell'*algoritmo*.

4.3.2 Analisi sulla distanza media percorsa

I dati estratti da questa *query* si sono rivelati utili per calcolare la distanza media percorsa dalla *persona* in base alle diverse attività visitate. Per questa interrogazione sono state estratte tutte le persone della provincia di Padova con almeno una ricevuta generata negli ultimi due anni. Successivamente sono stati estratti per ogni utente i diversi locali visitati e, grazie alle precise coordinate di latitudine e longitudine di ogni attività commerciale, è stato fatto un calcolo per analizzare il raggio medio di distanza presente tra i diversi locali visitati dall'utente. Questo dato si è rivelato utile in quanto era presente una differenza tra i risultati dei profili di *personas*, è stato quindi adoperato successivamente durante la progettazione e lo sviluppo dell'*algoritmo*.

4.3.3 Analisi periodo settimanale con maggior affluenza

Questa *query* è stata sviluppata per verificare se vi fossero differenze tra i profili di *personas* riguardo al periodo settimanale di maggior frequentazione delle attività commerciali, in settimana o durante il weekend. Per ottenere i dati di questa interrogazione sono stati estratti tutti gli utenti della provincia di Padova con almeno una ricevuta negli ultimi due anni, in seguito per ogni ricevuta generata dagli utenti è stato verificato il giorno di creazione della stessa ed è stato incrementato un contatore. Nel caso in cui la ricevuta fosse stata generata in un giorno compreso tra il lunedì e il venerdì

(quest'ultimo prima delle ore 18:00) veniva incrementata la variabile *week*, in caso contrario (quindi dal venerdì dalle ore 18:00 in poi fino alla domenica a mezzanotte) veniva incrementata la variabile *weekend*. I risultati ottenuti tramite questa interrogazione si sono rivelati di minore importanza rispetto alle due *queries* precedenti in quanto tutti i profili di *personas* frequentavano maggiormente i locali in settimana rispetto al weekend, per tale motivo i dati risultanti non sono stati utilizzati per la successiva progettazione dell'*algoritmo*.

4.3.4 Analisi scontrino medio per persona

Questa interrogazione è stata eseguita per controllare se vi fossero differenze tra i diversi profili di *personas* per quanto riguarda la spesa media per scontrino. Per generare la *query* sono state estratte tutte le persone nella provincia di Padova aventi almeno una ricevuta negli ultimi due anni. Successivamente per ogni utente filtrato sono state estratte tutte le ricevute, è stato sommato il prezzo totale delle ricevute e il numero totale delle ricevute, infine è stato diviso il prezzo totale per il numero totale delle ricevute. Non essendoci una differenza sostanziale tra i risultati dei diversi profili di *personas*, i dati estratti non sono stati utilizzati per la successiva progettazione e sviluppo dell'*algoritmo*.

4.4 Conclusione analisi dati

Dopo aver raccolto dati a sufficienza al fine di identificare in maniera più accurata i sei diversi profili di *personas*, si è proceduto con la progettazione e lo sviluppo dell'*algoritmo* atto alla promozione di attività commerciali.

Capitolo 5

Progettazione e funzionamento algoritmo

Questo capitolo tratta la progettazione e il funzionamento di un algoritmo adibito alla promozione di attività commerciali. L'algoritmo è stato ideato sulla base dei dati raccolti in precedenza e opera tramite la profilazione di un singolo utente.

5.1 Introduzione

Come anticipato in precedenza, al fine di poter analizzare i dati del singolo utente e paragonarli ad uno specifico profilo di *persona*, è stato necessario in primis svolgere un lavoro di identificazione dei sei diversi utenti tipo. In seguito all'analisi dei dati si è reso possibile il lavoro di profilazione dello specifico utente, passando in *input* ad un *algoritmo* l'*idUtente* (il codice identificativo univoco dell'utente) per restituire in *output*[g] un'attività commerciale compatibile con l'utente analizzato. A causa del poco tempo disponibile per lo sviluppo dell'*algoritmo*, non è stato valutato un modello basato su *machine learning*. Questo poiché parte del tempo del tirocinio formativo è stato impiegato per conoscere le piattaforme di Unipiazza, prendere confidenza col codice sviluppando funzionalità utili alle piattaforme ed eseguire l'analisi dei dati. Per questo motivo, essendo disponibile poco tempo per imparare ulteriormente nuove tecnologie sia da parte dello studente che del tutor aziendale, si è deciso di sviluppare l'*algoritmo* concentrandosi in modo più approfondito sull'analisi di dati e la profilazione delle *personas*, gettando le basi per poter ampliare e adattare l'*algoritmo* in futuro.

5.2 Tecnologie adoperate per lo sviluppo

5.2.1 NodeJs

L'*algoritmo* è stato sviluppato interamente per il suo utilizzo lato *back-end*, è stato quindi implementato sul *runtime environment Node.js*. la stesura del codice è avvenuta nel progetto *back-end unipiazza-tasks*, nel quale vengono sviluppate nuove funzionalità collegate al *database* e risulta semplice l'esecuzione del codice da testare. Nel progetto *unipiazza-tasks*, i diversi task sviluppati vengono eseguiti tramite il metodo *async.series*, importato nel *file* di esecuzione dei task tramite il codice *async = require("async");*. questo metodo permette di eseguire in fila i diversi task aventi codice asincrono. Inoltre, se un'operazione passa un errore tramite la propria *callback*, viene interrotta l'esecuzione delle funzioni.

5.2.2 MongoDB

Per le elaborazioni necessarie ad estrarre i dati dell'utente (utili alla profilazione dello stesso) e le sue *balance* (le interazioni utente/negozio, fondamentali all'*algoritmo* per estrarre i dati dei negozi visitati dall'utente oggetto di studio), sono state eseguite *queries* asincrone sul *database MongoDB*, contenente i dati degli utenti usufruitori del servizio Unipiazza e delle attività commerciali registrate sulla piattaforma Unipiazza partner. Per interfacciarsi col *database* è stata utilizzata la *libreria*[g] *Mongoose*, importata nei *file* del progetto tramite la riga di codice *var mongoose = require("mongoose");*. *Mongoose* è una libreria *ODM*[g] per *MongoDB* che permette di modellare lo schema di dati degli oggetti, ricavati dal *database* tramite le *queries*. Ad esempio, il modello dell'utente che viene restituito dalla query nella fase iniziale di estrazione dei dati da parte dell'*algoritmo*, è presente nel *file user.js* situato dentro la cartella *models* di *unipiazza-tasks*. In questo *file*, dopo aver istanziato la variabile *var UserSchema = new Schema* (*Schema* è una classe inclusa nella *libreria Mongoose*), vengono definiti gli attributi di *User* e i suoi metodi. Il modello *User* viene infine esportato tramite la riga di codice *module.exports = mongoose.model("User", UserSchema);*. La seguente immagine mostra la *query* eseguita nell'*algoritmo* per estrarre l'oggetto *user* coi suoi relativi dati:

```
User.findOne({ _id: user_id }).exec(function (err, user) {
```

Figura 5.1: Query user algoritmo

In questa *query* viene utilizzato il metodo *findOne* per ritornare un solo oggetto come risultato dell'interrogazione al *database*. L'*id* dello *user* da cercare sarà quello passato alla funzione, quindi il campo *_id* presente nella collezione *user* del *database* dovrà combaciare con lo *user_id* oggetto della ricerca. Se la *query* viene eseguita correttamente sarà restituito l'oggetto *user*, in caso contrario verrà segnalato un errore. Grazie alla modellazione di *Mongoose*, gli attributi dell'oggetto ritornato potranno essere ricavati tramite il simbolo "." seguito dal nome dell'attributo. Per ottenere ad esempio il genere dell'utente oggetto di studio, basterà scrivere il codice *user.gender*.

La seguente *query* presente all'interno dell'*algoritmo* viene adoperata per estrarre i negozi visitati dall'utente tramite l'utilizzo del modello *UserBalance*:

```
UserBalance.find({
  user_id: user._id,
  receipts_counter: { $gt: 0 },
})
  .populate("shop_id")
  .exec(function (err, userShops) {
```

Figura 5.2: Query user balance algoritmo

Il modello *UserBalance*, contenente i dati delle interazioni tra un utente e un singolo negozio, viene adoperato per estrarre i dati delle attività commerciali frequentate dall'utente tramite lo *shop_ id*. Le *balance* ottenute vengono filtrate per *user_ id*, in aggiunta saranno selezionate solamente quelle in cui il numero di ricevute generate nel negozio sia maggiore di zero. Grazie al metodo *populate* la *query* permette di ottenere come oggetto di ritorno direttamente *userShops*, contenente i negozi frequentati dall'utente, senza dover eseguire un'ulteriore interrogazione al *database*.

5.2.3 Trello

Per organizzare i diversi step necessari allo sviluppo dell'*algoritmo* è stata aggiunta una scheda dedicata su *Trello*, la quale veniva aggiornata spostandola al termine del completamento delle varie fasi di sviluppo del lavoro suddivise in *DOING*, *TO TEST* e *DONE*, e aggiornata tramite l'aggiunta e il completamento di attività inserite in una *checklist*. Su questa piattaforma sono stati inoltre caricati i *file* contenenti i dati utili alla creazione dell'*algoritmo* ricavati in fase di analisi, per poterli reperire facilmente in caso di un'ipotetica consultazione futura. La piattaforma si è rivelata utile anche per l'aggiunta di commenti da parte del tutor aziendale riguardanti le attività svolte, in modo da velocizzare l'interazione con esso e la ricezione di *feed-back* riguardanti lo sviluppo dell'*algoritmo*.

5.2.4 GitHub

Durante la stesura del codice è stata tenuta traccia del lavoro svolto sulla piattaforma *Github*, eseguendo *commit* periodici. Questo ha permesso di verificare eventuali differenze e migliorie apportate al codice nel tempo e conoscere le fasi di avanzamento del progetto, tramite i relativi commenti aggiunti ai *commit*.

5.3 Studio del singolo utente

Durante lo studio di uno specifico utente, l'*algoritmo* verifica la sua compatibilità coi diversi profili di *personas* basandosi su quattro caratteristiche fondamentali:

* Genere;
* Età;
* Categoria delle attività commerciali maggiormente frequentate;
* Distanza media percorsa tra le diverse attività commerciali frequentate.

Per prima cosa viene associato un contatore ad ognuno dei sei diversi profili di *persona*. Analizzando una per volta le peculiarità dell'utente, l'*algoritmo* identifica i profili di *personas* con caratteristiche similari e ne incrementa il contatore di un determinato valore, il quale varia in base all'importanza della proprietà analizzata. Al termine dei calcoli l'*algoritmo* ordina le *personas* partendo dai contatori con punteggio maggiore, ottenendo una classifica degli utenti tipo maggiormente compatibili con l'utente oggetto di studio. Nella lista delle proprietà da esaminare non sono stati inseriti il periodo settimanale di maggiore attività dell'utente (in settimana o durante il weekend) e la spesa media per *persona*. Queste caratteristiche sono state studiate durante il lavoro di analisi ma non si sono rivelate utili allo sviluppo dell'*algoritmo*, in quanto restituivano una scarsa differenza di risultati tra i diversi profili di *personas* analizzati.

5.4 Funzionamento dell'algoritmo

Inizialmente l'*algoritmo* riceve in *input* tramite passaggio di parametri alla funzione l'*id* dell'utente da analizzare, di cui vengono estratti dal *database* i dati. Viene dichiarata una variabile *Counter* contenente due oggetti: l'oggetto *personas* contiene i contatori utili a tenere i punteggi per la classifica degli utenti tipo, *shopCategory* contiene i contatori necessari a tenere i punteggi delle categorie di attività commerciali maggiormente frequentate dall'utente analizzato. Tutti i contatori vengono inizializzati assegnandovi il valore 0.

```
let Counter = {
  personas: {
    young_male: 0,
    adult_male: 0,
    senior_male: 0,
    young_female: 0,
    adult_female: 0,
    senior_female: 0,
  },
  shopCategory: {
    bar: 0,
    pub: 0,
    drink: 0,
    ciboAsporto: 0,
    divertimento: 0,
    hamburger: 0,
    fastFood: 0,
    ristorante: 0,
    dolci: 0,
    panificio: 0,
    shopping: 0,
    saluteBenessere: 0,
    copisteria: 0,
    gelateria: 0,
    serviziConsulenze: 0,
  },
};
```

Figura 5.3: Counter algoritmo

Per prima cosa viene controllato il genere dell'utente, se maschio vengono incrementati di un punto i contatori dei profili di *personas* relativi al genere maschile: *young_male*, *adult_male* e *senior_male*. Se femmina vengono incrementati di un punto i contatori delle *personas* di genere femminile: *young_female*, *adult_female* e *senior_female*.

Algorithm 1: Pseudo-codice incremento genere

```
//controllo che genere utente sia valorizzato
if User.gender then
    //controllo genere dell'utente
    if User.gender = "male" then
        Counter.personas.young_male++;
        Counter.personas.adult_male++;
        Counter.personas.senior_male++;
    else if User.gender = "female" then
        Counter.personas.young_female++;
        Counter.personas.adult_female++;
        Counter.personas.senior_female++;
    end
end
```

Successivamente, tramite la data di nascita inserita dall'utente in fase di registrazione, viene analizzata l'età: se l'utente è minore di 34 anni viene incrementato di un punto il contatore dei profili di *personas* giovani: *young_male* e *young_female*. Se l'utente è di anni compresi tra i 35 e i 54 viene incrementato di un punto il contatore dei profili di *personas* adulte: *adult_male* e *adult_female*. Infine, se l'utente ha un'età superiore ai 54 anni viene incrementato di un punto il contatore dei profili di *personas* senior: *senior_male* e *senior_female*.

Algorithm 2: Pseudo-codice incremento età

```
//controllo che data nascita utente sia valorizzata
if User.birthday then
    //controllo età utente calcolata tramite data di nascita
    if age < 34_years then
        Counter.personas.young_male++;
        Counter.personas.young_female++;
    else if age ≥ 35_years and age ≤ 54_years then
        Counter.personas.adult_male++;
        Counter.personas.adult_female++;
    else if age > 54_years then
        Counter.personas.senior_male++;
        Counter.personas.senior_female++;
    end
end
```

Genere ed età valgono entrambi un punto di incremento, sono le caratteristiche con maggiore importanza e relativo valore di incremento dei contatori. In seguito vengono estratte tramite *query* sul *database* tutte le *balance* dell'utente presenti: questi dati servono a raccogliere tutte le interazioni fatte tra l'utente e un singolo negozio. Tramite l'analisi dei dati delle *balance* è possibile ottenere la lista delle attività commerciali frequentate dall'utente. Scorrendo i negozi visitati dall'utente viene estratta per ognuno la categoria di negozio corrispondente, per ciascuna di queste viene quindi incrementato il rispettivo contatore dell'oggetto *shopCategory*.

Algorithm 3: Pseudo-codice incremento contatore shopCategory

```
//controllo che categoria negozio sia valorizzata
if Shop.cat then
    cat ← Shop.cat;
    //eseguo controlli in base alla categoria
    if cat = cat_bar then
        Counter.shopCategory.bar + +;
    else if cat = cat_pub then
        Counter.shopCategory.pub + +;
    else if cat = cat_drink then
        Counter.shopCategory.drink + +;
    else if cat = cat_ciboAsporto then
        Counter.shopCategory.ciboAsporto + +;
    //...
    //eseguo il controllo per tutte le categorie ed eventualmente
      incremento
    end
end
```

Al termine dell'incremento dei contatori per le categorie di locali, viene utilizzata la funzione *findBig3* sull'oggetto *Counter.shopCategory* per ottenere le tre categorie di negozi maggiormente visitate dall'utente.

Algorithm 4: Pseudo-codice funzione findBig3

```
//Ordino in maniera decrescente la lista di valori dell'oggetto
  obj passato alla funzione
ret ← obj.sort();
//Prendo i primi tre valori
ret.slice(0, 3);
//Ritorno i primi tre valori dell'oggetto ordinato
return ret;
```

Per ciascuna di queste viene verificato se è presente in una delle tre migliori categorie frequentate dai profili di *personas*, in caso positivo viene incrementato di 0.33 punti il contatore dell'utente tipo corrispondente. Se ad esempio l'utente analizzato frequenta maggiormente locali di categoria shopping, verrà incrementato di 0.33 punti il contatore di femmina adulta, femmina senior e maschio senior: *adult_female*,*senior_female* e *senior_male*.

Algorithm 5: Pseudo-codice incremento categoria preferita utente

```
//Controllo che sia presente una categoria preferita
if favCatUser then
    //verifico che categoria preferita di utente che sto
      analizzando sia inclusa in una delle prime tre più
      frequentate per personas, in caso positivo sommo 0.33 punti
    if favCatUser ∈ bestShopsYM then
        Counter.personas.young_male+ = 0.33;
    end
    if favCatUser ∈ bestShopsAM then
        Counter.personas.adult_male+ = 0.33;
    end
    if favCatUser ∈ bestShopsSM then
        Counter.personas.senior_male+ = 0.33;
    end
    if favCatUser ∈ bestShopsYF then
        Counter.personas.young_female+ = 0.33;
    end
    //...
    //procedo in questa maniera per tutti i profili di personas
end
```

Infine viene calcolato il raggio medio di distanza percorsa dall'utente tra i locali frequentati, questo è possibile grazie ai dati di latitudine e longitudine delle attività commerciali presenti sul *database* di Unipiazza. Se sono stati frequentati più di due locali differenti dall'utente (se si può quindi stimare in maniera abbastanza precisa la distanza media ipoteticamente percorsa dall'utente stesso), viene fatto un controllo tra il raggio calcolato e il range di valori presenti per i diversi profili di *personas*. Nel caso in cui il valore sia compreso in uno specifico range, verranno sommati 0.5 punti al contatore delle *personas* identificate.

Algorithm 6: Pseudo-codice incremento categoria preferita utente

```
//se utente ha visitato più di due negozi differenti
if differentShopsNumber > 2 then
    //faccio controllo in base a range di distanze percorse da
      personas, nel caso in cui raggio sia compreso sommo 0.5
      punti
    if AvgRadius > 0 and AvgRadius < 3000 then
        Counter.personas.senior_female+ = 0.5;
    end
    if AvgRadius ≥ 3000 and AvgRadius ≤ 4500 then
        Counter.personas.senior_male+ = 0.5;
        Counter.personas.young_female+ = 0.5;
    end
    if AvgRadius ≥ 3500 and AvgRadius < 5000 then
        Counter.personas.adult_male+ = 0.5;
        Counter.personas.adult_female+ = 0.5;
    end
    if AvgRadius ≥ 5000 then
        Counter.personas.young_male+ = 0.5;
    end
end
```

Nella pagina seguente è presente lo pseudo-codice completo dell'*algoritmo*, comprensivo degli incrementi e delle funzioni precedentemente descritti.

Algorithm 7: Pseudo-codice algoritmo completo

```
//Trovo nel db utente tramite id
User ← user;
if User then
    //Inizializzo Counter con personas e shopCategory
    Counter ← [...];
    if User.gender then
        //Incremento personas per genere
    end
    if User.birthday then
        //Incremento personas per età
    end
    //estraggo da db negozi visitati da utente
    UserShops ← usershops;
    if UserShops then
        foreach Shop ∈ UserShops do
            if Shop.cat then
                //incremento contatori categorie negozio visitate
            end
            //inserisco latitudine/longitudine negozio per calcolo
              raggio medio distanza
            Coords ← latitudine, longitudine;
            //incremento contatore numero negozi visitati
            differentShopsNumber + +;
        end
        //trovo migliori tre categorie negozi frequentati da utente
        bestThreeCatUser ← findBig3(Counter.shopCategory);
        favCatUser_0 ← bestThreeCatUser[0];
        //...
        if favCatUser_0 then
            //incremento personas per prima categoria preferita da
              utente
        end
        //eseguo stessa operazione per seconda e terza categoria
          preferita utente
        //calcolo raggio medio percorso da utente
        AvgRadius ← avgradius;
        if differentShopsNumber > 2 then
            //incremento personas per raggio medio distanza utente
        end
        //trovo migliori tre profili personas correlati ad utente
        bestThreeMatchPersonas ← findBig3(Counter.personas);
        MatchPersonas_0 ← bestThreeMatchPersonas[0];
        //...
        //ritorno a utente attività commerciale con categoria
          compresa tra migliori categorie di persona vincitrice del
          confronto
    end
end
```

5.5 Risultato dell'algoritmo

Al termine dell'analisi delle caratteristiche dell'utente, viene stilata una classifica dei tre migliori profili di *personas* ordinandone i contatori in maniera decrescente grazie nuovamente all'utilizzo della funzione *findBig3* sull'oggetto *Counter.personas*. Di questi viene infine selezionato il profilo col punteggio maggiore. All'utente verrà quindi suggerita un'attività commerciale avente la stessa categoria di una delle tre migliori categorie frequentate dalla *persona* vincitrice del confronto.

5.6 Vincoli sul risultato

Sono presenti dei vincoli per poter suggerire un'attività commerciale all'utente: il negozio proposto non deve essere già stato visitato in precedenza e non dev'essere concorrente di un altro locale visitato precedentemente dall'utente analizzato. Per questo motivo l'*algoritmo* proporrà se possibile un'attività commerciale avente la stessa categoria della prima maggiormente visitata per *persona* vincitrice del confronto, in caso contrario la seconda ed infine la terza.

Capitolo 6

Verifica e validazione

Questo capitolo descrive le attività di verifica, analisi e validazione riguardanti il lavoro svolto durante lo stage.

6.1 Verifica

La verifica del codice prodotto, secondo quanto stabilito nel piano di lavoro, non ha richiesto forme di analisi dinamica, come la scrittura di test automatici di unità e di integrazione. Durante il tirocinio formativo sono stati privilegiati lo sviluppo e il *debug* di tutti i task svolti. L'attività di verifica è stata limitata pertanto alla sola analisi statica del codice tramite l'utilizzo dell'*IDE*.

6.1.1 Analisi statica

L'ambiente di sviluppo utilizzato per la stesura del codice si è rivelato utile nell'individuare eventuali errori di sintassi, porzioni di codice non raggiungibile o inutilizzato e mostrare possibili proposte per il completamento del codice. Questo ha permesso di evitare perdite di tempo nell'esecuzione di codice problematico e di minimizzare gli eventuali errori presenti.

6.2 Validazione

La validazione interna del codice è stata eseguita al termine dello sviluppo di ogni task, effettuando un lavoro di *debug* e testandone il corretto funzionamento. Superata questa fase, prima di rilasciare il codice prodotto è stata effettuata la validazione esterna, con una serie di test effettuati da parte del tutor aziendale. Al termine dello sviluppo dell'*algoritmo*, l'ultima settimana di stage è stata interamente dedicata al suo collaudo e relativa validazione. Queste attività di validazione e *debug* del codice hanno portato grande beneficio alle funzionalità implementate, mostrando possibili problematiche o errori logici che non erano stati individuati inizialmente e migliorando in alcuni casi l'efficienza computazionale del codice stesso. L'esperienza del tutor aziendale e la sua conoscenza delle piattaforme Unipiazza hanno permesso inoltre di evidenziare sistemazioni necessarie e di evitarne ulteriori altre.

6.2.1 Postman

Durante la fase di validazione e il relativo *debug*, per testare le *API* è stato possibile utilizzare *Postman*. Generando delle chiamate *API GET* e *POST* fittizie per riprodurre le interazioni in un negozio da parte di un utente, questa piattaforma ha permesso di simulare l'utilizzo di una carta Unipiazza in un negozio fisico, velocizzare le operazioni di test sulle funzioni modificate e controllare che i valori ritornati dalle funzioni fossero coerenti con i risultati attesi. Per configurare le operazioni di un utente col tablet di Unipiazza presente in un negozio fisico, è stata aggiunta nella schermata *Params* delle *API* configurate una chiave *pass* con un codice identificativo rappresentante una carta Unipiazza valida. Nella schermata *Headers* delle *API* configurate è stata aggiunta la chiave *Accept*, contenente l' *id* univoco del negozio in cui si voleva simulare l'interazione. Infine, è stato inserito l'*URL*[g] contenente l'indirizzo della chiamata *API* da effettuare.

6.3 Conclusioni validazione

La validazione interna del codice ha permesso di sistemare eventuali comportamenti anomali del codice nel caso in cui non venissero restituiti i valori attesi. La validazione esterna da parte del tutor aziendale è stata utile soprattutto al miglioramento dell'efficienza delle *queries* prodotte, permettendomi di ragionare in maniera differente e migliore nello sviluppo delle stesse per *database* di tipo *NoSQL* come nel caso di *MongoDB*. In conclusione, la validazione esterna riguardante il codice prodotto si è svolta con successo e ha confermato la coerenza del lavoro con le aspettative iniziali, rispettando tutti gli obiettivi concordati.

Capitolo 7

Conclusioni

Questo capitolo contiene il consuntivo finale del progetto di stage. Descrive inoltre il raggiungimento degli obiettivi preposti, le conoscenze acquisite e una valutazione personale del lavoro svolto presso l'azienda ospitante.

7.1 Consuntivo finale

Lo stage si è svolto dal giorno 4 luglio 2022 al 23 settembre 2022 compresi, richiedendo un impegno di 5 ore al giorno per un totale di 300 ore effettive. La seguente tabella mostra le differenze tra quanto preventivato nel piano di lavoro e il numero di ore effettivamente svolte:

Tabella 7.1: Tabella pianificazione oraria consuntivo

Durata in ore	**Descrizione dell'attività**
20 (+10)	**Formazione sulle tecnologie**
10	**Visione architettura di riferimento e relativa documentazione**
230 (-10)	**Definizione ed esecuzione del lavoro**
30 (-10)	*Analisi dei requisiti delle singole task*
50	*Progettazione e discussione delle soluzioni*
55 (+20)	*Analisi Dati*
75 (-15)	*Esecuzione*
10	*Stesura documentazione relativa ad analisi e progettazione*
10 (-5)	*Integrazione con il codice esistente*
40	**Collaudo Finale**
30	*Collaudo e Test*
10	*Stesura documentazione finale*
ore preventivate	**300**
ore effettive	**300**
differenza	**+0**

La poca conoscenza iniziale delle tecnologie adoperate nei progetti di Unipiazza ha reso necessario un maggior numero di ore riguardanti la formazione sulle tecnologie. Avendo maturato maggiore esperienza durante lo sviluppo delle diverse attività, si è reso necessario un minor numero di ore atte all'analisi dei requisiti delle singole task, questo grazie anche ad una frequente interazione col tutor aziendale che si è dimostrato sempre disponibile per eventuali chiarimenti. Avendo deciso in accordo col tutor aziendale di dare maggior importanza al lavoro di analisi dei dati rispetto a quanto preventivato, il numero di ore riguardanti l'analisi è cresciuto in confronto a quanto ipotizzato inizialmente. Il numero di ore di sviluppo del codice effettivo è diminuito, poiché gran parte del lavoro per ideare l'algoritmo è stato svolto in precedenza attraverso l'analisi dei dati e i tempi di scrittura del codice si sono ridotti. Infine, il lavoro di integrazione col codice preesistente si è rivelato meno complicato del previsto, questo ha portato ad un risparmio di ore durante lo sviluppo. In totale le ore preventivate erano 300, lo stesso numero di ore effettivamente trascorse durante lo sviluppo del progetto. Non si è quindi reso necessario un allungamento dei tempi per il completamento degli obiettivi preposti.

7.2 Raggiungimento degli obiettivi

Tutti gli obiettivi obbligatori e desiderabili fissati nel piano di lavoro sono stati raggiunti. Il requisito facoltativo riguardante lo studio di un ambiente di lavoro *android* non è stato svolto poiché in accordo col tutor aziendale è stata data maggior importanza all'analisi dei dati, allo sviluppo dell'*algoritmo* e alla scrittura di codice utile ad aumentare le funzionalità delle piattaforme web di Unipiazza.

7.3 Conoscenze acquisite

Durante il tirocinio formativo presso Unipiazza ho potuto apprendere diverse tecnologie che non avevo mai avuto modo di approfondire in precedenza. l'utilizzo di *MongoDB* (un *database* di tipo *NoSQL*) e l'esecuzione di *queries* in maniera asincrona, sono stati per me una sfida ma anche motivo di interesse, inoltre ho potuto impratichirmi con l'utilizzo del *framework Node.js*. Nel corso dello stage sono stati inoltre eseguiti task nei quali sono stati adoperati (anche se in maniera più marginale) *React* e *Angular*, permettendomi di conoscere queste due tecnologie. Infine, il lavoro eseguito in modalità telematica mi ha permesso di conoscere e utilizzare piattaforme web come *Slack*, *Gather* e *Trello*, che si sono rivelate utili all'organizzazione aziendale e a semplificare l'interazione tra le persone.

7.4 Valutazione personale

Nell'azienda Unipiazza ho trovato un clima accogliente e propositivo. Il tutor aziendale mi ha seguito nel corso degli sviluppi dimostrandosi sempre disponibile per spiegazioni e chiarimenti e ho potuto apprendere tecnologie moderne adoperate in larga scala. Mi sento più preparato e confidente riguardo l'utilizzo di *database* di tipo *NoSQL* e ho potuto mettermi in gioco nello sviluppo di un'*algoritmo*, a partire dalla progettazione fino alla sua realizzazione concreta. Il tirocinio mi ha permesso inoltre di mettere in pratica molti dei concetti appresi durante il percorso di laurea e di ampliare ulteriormente le mie conoscenze. Mi ritengo quindi soddisfatto del lavoro svolto, nella speranza che il progetto da me sviluppato venga mantenuto e ampliato in futuro.

Acronimi e abbreviazioni

API Application Programming Interface. 10

DBMS Data Base Management System. 10

HTTP Hypertext Transfer Protocol. 5

IDE Integrated Development Environment. 7

JSON JavaScript Object Notation. 10

ODM Object Data Modeling. 20

PMI Piccole e Medie Imprese. 2

URL Uniform Resource Locator. 32

VCS Version Control System. 11

Glossario

Algoritmo Procedimento di calcolo esplicito e descrivibile con un numero finito di regole che conduce al risultato dopo un numero finito di operazioni. 2

Android *Android* è un *sistema operativo* per dispositivi mobili sviluppato da Google, progettato principalmente per sistemi *embedded* quali smartphone e tablet. 4

AngularJs *AngularJS* è un *framework* per applicazioni web *open source*, sviluppato nel 2010 da Google e dalla comunità di sviluppatori individuali, al fine di affrontare le difficoltà che si incontrano nello sviluppo di applicazioni su singola pagina. 5

Application Programming Interface In un programma informatico, con *application programming interface*, si indica un insieme di procedure (in genere raggruppate per strumenti specifici) atte a risolvere uno specifico problema di comunicazione tra diversi computer, *software* o tra componenti di *software*. 37

Back-end La parte di un sistema *software* non visibile all'utente, atto all'elaborazione e modifica dei dati. 5

Cache Con il termine *cache*, in informatica, si indica un'area di memoria estremamente veloce ma solitamente con una bassa capacità di memorizzazione dei dati in termini di quantità. Il suo scopo è velocizzare l'esecuzione dei programmi. 5

Data Base Management System il *data base management system* è un sistema di gestione di basi di dati. Le principali funzioni dei *DBMS* sono quelle di garantire il mantenimento della corretta strutturazione dei dati nei diversi *database* gestiti e di facilitare l'accesso delle applicazioni ai dati, tramite opportune istruzioni impartite al *sistema operativo*. A queste funzionalità di base si aggiungono quelle di interrogazione e modifica del *database* e la possibilità di produzione di documentazione di sintesi per il controllo delle operazioni. 37

Database Un insieme di informazioni (o dati) strutturate, archiviate elettronicamente in un sistema informatico. 2

Debug Il *debugging* (o semplicemente *debug*), in informatica, nell'ambito dello sviluppo *software*, indica l'attività che consiste nell'individuazione e correzione da parte del programmatore di uno o più errori (*bug*) rilevati nel *software*, direttamente in fase di programmazione oppure a seguito della fase di *testing* o dell'utilizzo finale del programma stesso. 7

Easter egg Un *Easter egg* in informatica è un contenuto, di solito di natura bizzarra e innocuo, che i progettisti o gli sviluppatori di un prodotto (specialmente *software*) nascondono nel prodotto stesso. 6

Feed-back Riscontro, positivo o negativo, riguardante un determinato argomento. 2

File Il termine *file*, in informatica, indica un contenitore di informazioni/dati in formato digitale, tipicamente presenti su un supporto digitale di memorizzazione opportunamente formattato in un determinato *file system*. Le informazioni scritte/codificate al suo interno sono leggibili solo tramite uno specifico *software* in grado di effettuare l'operazione. 12

Framework In informatica, un *framework* è un sistema che consente di estendere le funzionalità del linguaggio di programmazione su cui è basato, fornendo allo sviluppatore una struttura coerente ed efficace al fine di effettuare azioni e comandi in modo semplice e veloce. 8

Front-end Nel campo della progettazione *software* il *front-end* è la parte di un sistema che gestisce l'interazione con l'utente o con sistemi esterni che producono dati di ingresso. 5

Github *GitHub* è uno strumento web di *version control Git*. Usando questo *tool* i programmatori possono lavorare in modo coordinato sulla stessa base di codice, pur sviluppando in modo indipendente. 4

Hosting L'*hosting* è un servizio online che consente di pubblicare il proprio sito web o applicazione web su *internet*. Quando ti iscrivi a un servizio di *web hosting*, prendi "in affitto" dello spazio *web hosting* su un *server* fisico dove puoi archiviare tutti i *file* e i dati necessari per il corretto funzionamento del tuo sito internet o applicazione web. 11

Hypertext Transfer Protocol In telecomunicazioni e informatica l'*HyperText Transfer Protocol* (*HTTP*) è un protocollo a livello applicativo usato come principale sistema per la trasmissione d'informazioni sul web. 37

Input *Input* è un termine inglese con significato di «immettere» che in campo informatico definisce una sequenza di dati o informazioni immessi per mezzo di una periferica (o passati come valori ad una funzione), e successivamente elaborati. 3

Integrated Development Environment Un ambiente di sviluppo integrato (in inglese *integrated development environment* ovvero *IDE*) è un *software* che, in fase di programmazione, supporta i programmatori nello sviluppo e *debugging* del codice sorgente di un programma. 37

JavaScript Object Notation *JSON* (*JavaScript Object Notation*) è un semplice formato per lo scambio di dati. Per le persone è semplice da leggere e scrivere, mentre per le macchine risulta facile da generare e analizzarne la sintassi. Si basa su un sottoinsieme del Linguaggio di Programmazione *JavaScript*, Standard ECMA-262 Terza Edizione - Dicembre 1999. *JSON* è un formato di testo completamente indipendente dal linguaggio di programmazione, ma utilizza convenzioni conosciute dai programmatori di linguaggi della famiglia del C. Questa caratteristica fa di *JSON* un linguaggio ideale per lo scambio di dati. 37

Libreria In informatica una *libreria* è un insieme di funzioni o strutture dati predefinite e predisposte per essere collegate ad un programma software attraverso un opportuno collegamento. Il collegamento può essere statico o dinamico. 20

Node.Js Node.js è un *runtime environment JavaScript* costruito sul motore *JavaScript V8* di *Google Chrome*, progettato per creare applicazioni di rete scalabili. Consente quindi di utilizzare *JavaScript* anche per scrivere codice da eseguire lato *server*, ad esempio per la creazione di pagine dinamiche. 5

NoSQL I *database NoSQL* (noti anche come "non solo SQL") non sono tabulari e memorizzano i dati in modo diverso rispetto alle tabelle relazionali. I *database NoSQL* sono disponibili in una varietà di tipologie in base al loro modello di dati. I tipi principali sono di documenti e forniscono schemi flessibili e facilmente scalabili con grandi quantità di dati e carichi elevati degli utenti. 10

Object Data Model Un modello di dati a oggetti (*Object Data Model*) è un modello di dati basato sulla programmazione orientata agli oggetti che associa metodi a oggetti, i quali possono trarre vantaggio dalle gerarchie di classi. 37

Open source *Software* di cui l'utente finale, che può liberamente accedere al *file* sorgente, è in grado di modificare a suo piacimento il funzionamento, correggere eventuali errori, ridistribuire a sua volta la versione da lui elaborata. L'esempio più noto è il *sistema operativo Linux*. La distribuzione di un *software* in formato *open source* presuppone la rinuncia da parte dei programmatori al diritto di proprietà intellettuale. 11

Output dati che il programma trasmette in uscita verso un soggetto terzo. 19

Query In informatica, il termine *query* indica l'interrogazione di una base di dati da parte di un utente. 5

React *React* è una libreria *JavaScript* per la creazione di interfacce utente (UI, *User Interface*) che consente di sviluppare applicazioni dinamiche le quali non necessitano di ricaricare la pagina per visualizzare i dati modificati. 5

Redis *Redis* è un archivio *open source* di strutture dati utilizzato come *database*, *cache*, *broker* di messaggi e *streaming engine*. 5

Runtime environment Gli ambienti di *runtime* (in breve *RTE*) agiscono come piccoli sistemi operativi e forniscono tutte le funzionalità necessarie per l'esecuzione di un programma. Ciò include interfacce per parti fisiche dell'*hardware*, interazioni utente e componenti *software*. Un ambiente di *runtime* carica le applicazioni e le fa eseguire su una piattaforma. Tutte le risorse necessarie per funzionare indipendentemente dal *sistema operativo* sono disponibili su questa piattaforma. 8

Server Un *server* in informatica e telecomunicazioni è un dispositivo fisico o sistema informatico di elaborazione e gestione del traffico di informazioni. Un *server* fornisce, a livello logico e fisico, un qualunque tipo di servizio ad altre componenti (tipicamente chiamate *client*) che ne fanno richiesta attraverso una rete di *computer* o all'interno di un sistema informatico. 5

Sistema operativo Un *sistema operativo*, in informatica, è un *software* di base che gestisce le risorse *hardware* e *software* del computer o più in generale di una macchina. 7

Slack *Slack* è un'applicazione web di messaggistica per le aziende. 4

Smart working Il lavoro agile (o *smart working*) è una modalità di esecuzione del rapporto di lavoro subordinato caratterizzato dall'assenza di vincoli orari o spaziali e un'organizzazione per fasi, cicli e obiettivi, stabilita mediante accordo tra dipendente e datore di lavoro. La definizione di *smart working*, contenuta nella legge n. 81/2017, pone l'accento sulla flessibilità organizzativa, sulla volontarietà delle parti che sottoscrivono l'accordo individuale e sull'utilizzo di strumentazioni che consentano di lavorare da remoto (come, ad esempio, pc portatili, tablet e smartphone). 5

Startup Impresa giovane, che ha al centro del proprio modello di business l'innovazione. 2

Trello *Trello* è uno strumento visivo che consente la gestione di qualsiasi tipo di progetto, flusso di lavoro o monitoraggio delle attività tramite la creazione di una bacheca virtuale per la gestione dei lavori. 4

Typescript *TypeScript* è un linguaggio di programmazione *open source* sviluppato e mantenuto da Microsoft. È un rigoroso superset sintattico di *JavaScript* e aggiunge la tipizzazione statica al linguaggio. È progettato per lo sviluppo di applicazioni di grandi dimensioni. Poiché è un *superset* di *JavaScript*, anche i programmi *JavaScript* esistenti sono compatibili con *TypeScript*. 9

Uniform Resource Locator Lo *Uniform Resource Locator* è una sequenza di caratteri che identifica univocamente l'indirizzo di una risorsa in Internet, come un documento o un'immagine. È l'elemento che ci permette di trovare un sito web, cioè l'indirizzo che noi digitiamo nel browser quando cerchiamo una pagina o un *file*. 37

Version Control System Con sistema di controllo della versione o *VCS* (*Version Control System*), noto anche come sistema di controllo della revisione o della sorgente, si intende un'*utility software* che è in grado di monitorare e gestire le modifiche apportate a un *filesystem*, nonché di offrire *utility* collaborative che consentono di condividere e integrare tali modifiche con altri utenti di *VCS*. Un *VCS* tiene traccia delle operazioni di aggiunta, eliminazione e modifica applicate a *file* e *directory*. I sistemi di controllo della versione più richiesti sono *Git*, *Mercurial*, *SVN* e *Perforce*. 37

Bibliografia

Siti web consultati

Algoritmo. URL: https://www.treccani.it/vocabolario/algoritmo/.

Android. URL: https://it.wikipedia.org/wiki/Android.

Angular. URL: https://www.elbuild.it/tecnologie/angular.html.

Application programming interface. URL: https://it.wikipedia.org/wiki/Application_programming_interface#cite_note-1.

Backend. URL: https://it.wikipedia.org/wiki/Front-end_e_back-end.

Cache. URL: https://it.wikipedia.org/wiki/Cache.

Database. URL: https://www.oracle.com/it/database/what-is-database/.

DBMS. URL: https://www.treccani.it/enciclopedia/dbms.

Debug. URL: https://it.wikipedia.org/wiki/Debugging.

Easter egg. URL: https://it.wikipedia.org/wiki/Easter_egg.

File. URL: https://it.wikipedia.org/wiki/File.

Framework. URL: https://reteinformaticalavoro.it/blog/framework-cosa-sono-e-quali-dominano-le-classifiche/.

Frontend. URL: https://it.wikipedia.org/wiki/Front-end_e_back-end.

Gather. URL: https://www.gather.town/.

GitHub. URL: https://www.geekandjob.com/wiki/github.

Hosting. URL: https://www.creativemotions.it/hosting-definizione/.

Hypertext Transfer Protocol. URL: https://it.wikipedia.org/wiki/Hypertext_Transfer_Protocol.

IDE. URL: https://it.wikipedia.org/wiki/Ambiente_di_sviluppo_integrato.

Input. URL: https://it.wikipedia.org/wiki/Input#Entry.

JavaScript. URL: https://it.wikipedia.org/wiki/JavaScript.

JSON. URL: https://www.json.org/json-it.html.

Libreria. URL: https://it.wikipedia.org/wiki/Libreria_(software).

Node.js. URL: https://en.wikipedia.org/wiki/Node.js.

NoSQL. URL: https://www.mongodb.com/it-it/nosql-explained.

Object Data Model. URL: https://www.gartner.com/en/information-technology/glossary/object-data-model.

Open source. URL: https://www.treccani.it/enciclopedia/open-source/.

Output. URL: https://it.wikipedia.org/wiki/Input/output.

React. URL: https://www.geekandjob.com/wiki/react.

Redis. URL: https://redis.io/docs/about/.

Runtime environment. URL: https://www.ionos.com/digitalguide/websites/web-development/what-is-a-runtime-environment/.

Server. URL: https://it.wikipedia.org/wiki/Server.

Sistema operativo. URL: https://it.wikipedia.org/wiki/Sistema_operativo.

Slack. URL: https://slack.com/intl/it-it/help/articles/115004071768-Che-cos%E2%80%99%C3%A8-Slack-.

Smart working. URL: https://miur.gov.it/lavoro-agile.

Typescript. URL: https://en.wikipedia.org/wiki/TypeScript.

Uniform Resource Locator. URL: https://www.federprivacy.org/informazione/sapresti-rispondere/cosa-e-un-url.

Version Control System. URL: https://bitbucket.org/product/it/version-control-software.

Visual studio code. URL: https://code.visualstudio.com/learn.

VSCodium. URL: https://vscodium.com/.