



SAPIENZA
UNIVERSITÀ DI ROMA

Let's Party: realizzazione e sviluppo di un innovativo provider di servizi per locali

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica

Corso di Laurea in Ingegneria Informatica e Automatica

Candidato

Gianluca Procopio

Matricola 1942103

Relatore

Prof. Leonardo Querzoni

Anno Accademico 2022/2023

Let's Party: realizzazione e sviluppo di un innovativo provider di servizi per locali

Tesi di Laurea. Sapienza – Università di Roma

© 2012 Gianluca Procopio. Tutti i diritti riservati

Questa tesi è stata composta con L^AT_EX e la classe Sapthesis.

Email dell'autore: procopio.1942103@studenti.uniroma1.it

Indice

1	Introduzione	1
1.1	Obiettivi della relazione	1
1.2	Descrizione generale dell'applicazione	2
1.3	Architettura del sistema	2
2	Raccolta e analisi dei requisiti	5
2.1	Realtà di interesse	5
2.2	User Experience	5
2.2.1	Creazione e gestione profilo personale	6
2.2.2	Recensioni	9
2.2.3	Mappa	10
2.2.4	Lista locali preferiti ed eventi	11
2.2.5	Social Network	12
3	Tecnologie e metodologie utilizzate	15
3.1	Metodologie utilizzate	15
3.2	Organizzazione del lavoro	16
3.3	Tecnologie utilizzate	17
3.3.1	API impiegate	17
3.3.2	Node.JS	17
3.3.3	Presentation tier	21
3.3.4	Data tier	22
4	Il data tier: progettazione concettuale	25
4.1	Analisi dei requisiti	25
4.2	Diagramma Entità-Relazioni	26
4.2.1	Vincoli	28
4.2.2	Tabelle	29
5	Testing e Validazione	33

5.1	Software Testing	33
5.1.1	Test Driven Development	34
5.1.2	Behavior Driven Development	34
6	Conclusioni	39
6.1	Sintesi dei punti chiave	39
6.2	Valutazione complessiva dell'applicazione	39
	Bibliografia	41

Elenco delle figure

1.1	Architettura a tre livelli	3
2.1	Schermata in cui l'utente può effettuare il log in	6
2.2	Schermata in cui l'utente può effettuare la registrazione	7
2.3	Sidebar con impostazioni	7
2.4	Schermata in cui l'utente può modificare il profilo	8
2.5	Schermata che l'utente può utilizzare per invitare un amico	8
2.6	Profilo pubblico	9
2.7	Schermata in cui l'utente può scrivere una recensione	9
2.8	Card review	10
2.9	Mappa dei locali	10
2.10	Pop up locale	11
2.11	Lista card locali	11
2.12	Vista del profilo	12
2.13	Feed	12
2.14	Scrittura post	13
2.15	Post con foto	13
2.16	Eliminazione post	13
3.1	Connextra template	16
3.2	Codice passport	18
3.3	Hashing password	19
3.4	Inserimento password criptata	19
3.5	Password nel database	19
3.6	Mittente nodemailer	20
3.7	Invio della mail	20
3.8	File .env	21
3.9	File EJS	21
3.10	File dbConfig.js	23
3.11	Esempio query GET	23

3.12 Esempio query POST	23
4.1 Diagramma ER	27
5.1 Esempio di scenario di test	35
5.2 Esempio di scenario per testare il login	35
5.3 Preparazione ambiente per test	36
5.4 Definizione dei vari step	36

Elenco delle tabelle

4.1	Tabella delle entità	29
4.2	Tabella delle relazioni	30
4.3	Tabella degli attributi	31

Capitolo 1

Introduzione

1.1 Obiettivi della relazione

L'obiettivo generale di questo progetto è la realizzazione di un'applicazione web che faciliti l'interazione tra utenti, locali e organizzatori di eventi. Questa relazione esplorerà a fondo Let's Party, mettendo in mostra le funzionalità e le peculiarità del nostro progetto, oltre ad un'approfondita analisi tecnica delle fasi di progetto e sviluppo, prestando particolare attenzione alle funzionalità da me implementate e le tecnologie da me utilizzate.

Durante la fase di ideazione del progetto, il nostro focus principale è stata l'esperienza dell'utente. L'idea del progetto è nata proprio da un problema che ci siamo trovati ad affrontare: "C'è qualcosa da fare stasera?". Crediamo che la nostra applicazione sia una risposta accessibile e intuitiva a chi si è trovato ad affrontare il medesimo problema. Dal momento che riteniamo indispensabile l'interazione tra gli utenti, abbiamo aggiunto al nostro progetto una componente social. Questo ci ha portato alla progettazione e realizzazione di funzionalità che consentono agli utenti di rendere pubbliche e visibili ad altri utenti le loro esperienze; in particolare, l'utente ha la possibilità di scrivere post testuali, aggiungere immagini e scrivere recensioni ai locali frequentati.

Il software si avvale dell'utilizzo di NodeJS, un moderno e avanzato runtime environment per consentire l'esecuzione di codice JavaScript lato server. La nostra scelta è ricaduta su NodeJS data la sua efficienza nella gestione delle comunicazioni client-server, garantendo all'utente un'esperienza di utilizzo reattiva e fluida.

1.2 Descrizione generale dell'applicazione

L'applicazione si presenta con una caratteristica interfaccia moderna e intuitiva. Nella nostra applicazione sono presenti due tipi di utenti: gli utenti semplici e gli organizzatori di eventi. I secondi hanno più funzionalità dei primi, tra cui la possibilità di organizzare nuovi eventi, i quali compariranno nella schermata "Mappa". Sulla barra di navigazione troviamo le principali sezioni che compongono la nostra applicazione: Mappa, Social Network, Future Projects, Contacts, Login, Profile e Partners. Una volta effettuata l'autenticazione si potrà avere accesso a tutte le funzionalità del software. Dalla sezione "Mappa" sarà possibile visualizzare la mappa sulla quale verrà mostrata la posizione dell'utente e i locali a lui vicini. Inoltre comparirà una lista di card, ognuna contenente le informazioni di ogni locale. Ogni utente avrà la possibilità di vedere i diversi locali presenti, aggiungerli ai preferiti o prenotare l'ingresso a uno di essi. Nella sezione "Social network" invece ogni utente ha la possibilità di interagire con altri utenti mediante post testuali, visibili a tutti, e ai quali ogni utente può mettere like. Inoltre ogni utente può scrivere recensioni sui locali che ha visitato, e ogni recensione è pubblica e visibile da ogni altro utente. Questa componente social ci aspettiamo coinvolga il più possibile gli utenti, spingendoli ad intrattenersi più tempo possibile nell'applicazione, ma anche che renda più completa e immersiva l'esperienza d'uso.

1.3 Architettura del sistema

Per progettare la nostra applicazione abbiamo sfruttato l'architettura a tre strati come guida di progettazione (Fig. 1.1). Questa scelta ci permette di suddividere l'applicazione in 3 componenti chiave: presentation tier, application tier e data tier. Ogni livello ha ruoli e responsabilità specifiche e sono interconnesse per garantire la realizzazione di un software completo e scalabile.

Il *presentation tier* è responsabile della gestione dell'interfaccia e delle interazioni dell'utente. Si concentra sulla presentazione dei dati all'utente e sulla raccolta degli input dello stesso. Include, ad esempio, le componenti dell'interfaccia utente, come pagine web e moduli.

L'*application tier* è il livello intermedio che funge da intermediario tra il presentation e il data tier. Esso garantisce che i dati in input vengano elaborati e manipolati correttamente, esegue operazioni basate sulla logica di business dell'applicazione ed effettua controlli per garantire il corretto funzionamento di quest'ultima.

Infine, il *data tier* è responsabile della gestione, del salvataggio e del recupero dei dati mediante interazioni con database, API esterne o qualsiasi altra fonte di dati.

Questo livello si occupa della conservazione dei dati, assicurandosi che essi siano memorizzati, recuperati e aggiornati in modo efficiente e protetto.

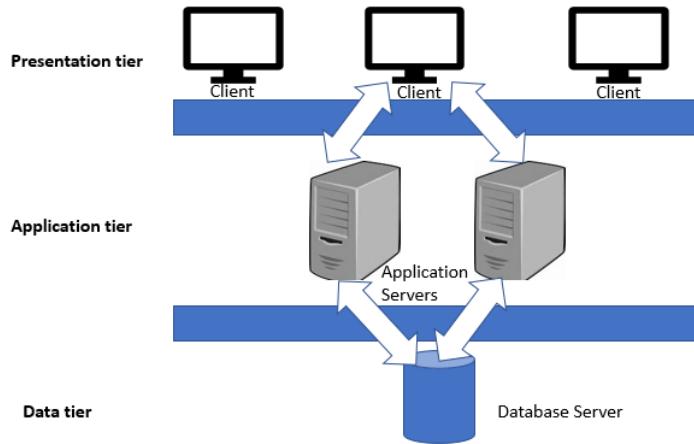


Figura 1.1. Architettura a tre livelli

La scelta dell'architettura a tre livelli è ottimale nello sviluppo di applicazioni software. Essa permette una netta separazione delle responsabilità dei diversi livelli, favorendo una maggiore organizzazione del codice e consentendo quindi una maggiore facilità di manutenzione e scalabilità. La modularità del codice e la divisione delle responsabilità dei livelli favoriscono la collaborazione tra gli sviluppatori, in quanto ognuno può concentrarsi su determinati aspetti dell'applicazione senza interferire con gli altri.

Essendo i livelli indipendenti tra loro, un vantaggio importante è la scalabilità che consiste nella possibilità di scalare ciascun livello separatamente per gestire carichi sempre maggiori (maggiore capacità di elaborazione, maggiore quantità di dati da gestire).

Inoltre questa architettura favorisce la riusabilità del codice: i componenti all'interno di ciascun livello possono essere utilizzati in più parti della medesima applicazione, o addirittura in altre applicazioni per accelerarne lo sviluppo e ridurre la duplicazione del codice.

Infine, questa architettura promuove anche una maggiore sicurezza delle applicazioni. La separazione tra livello di presentazione e livello dei dati evita l'accesso diretto a questi ultimi e permette una più efficiente gestione delle autorizzazioni, limitando o garantendo l'accesso ai dati.

Capitolo 2

Raccolta e analisi dei requisiti

2.1 Realtà di interesse

L'idea del progetto nasce da una problematica che ci siamo trovati ad affrontare in prima persona, ma che abbiamo riscontrato anche parlando con i nostri coetanei: voler trovare posti nuovi in cui uscire e dove poter conoscere nuove persone.

Da questo problema è nata la nostra ricerca di una soluzione tecnologica che semplifichi tale processo, ovvero la possibilità di ritrovare nella stessa piattaforma locali ed eventi nelle vicinanze e soprattutto la possibilità di poter prenotare direttamente sulla piattaforma, evitando così alle persone perdite di tempo all'ingresso del locale o per contattare gli organizzatori delle serate. Oltre a facilitare la vita dei clienti, Let's Party mette nelle mani dei proprietari di locali e organizzatori uno strumento potente per pubblicizzare i propri eventi e per arrivare ad una clientela vasta.

Questo progetto è pensato per giovani e adulti attivi che hanno voglia di provare nuove esperienze, ma anche per turisti e viaggiatori che si trovano in una nuova città o zona e desiderano scoprire locali ed eventi vicini a loro. L'applicazione pertanto ha come target di riferimento un'audience molto vasta e promuove così una comunità di utenti interessati a esplorare e provare nuove esperienze.

2.2 User Experience

In questa sezione mi concentrerò sull'esperienza che un utente può provare all'interno dell'applicazione. La nostra piattaforma prevede due tipi di utenti: gli utenti semplici e gli organizzatori di eventi¹.

¹La creazione e la gestione degli utenti organizzatori è stata discussa nella relazione di un mio collega

Gli utenti semplici, una volta registrati e loggati, avranno accesso ad un dato insieme di funzionalità, invece gli organizzatori (la registrazione come tale prevede uno specifico form di iscrizione) hanno funzionalità aggiuntive, ad esempio la creazione di eventi visibili a tutti gli utenti. Per il resto gli organizzatori potranno usare l'applicazione come se fossero utenti comuni.

Successivamente esporrò le principali funzionalità che un utente semplice può trovare sulla nostra piattaforma, avendo curato principalmente questo aspetto dell'applicazione.

2.2.1 Creazione e gestione profilo personale

Questa sezione mostrerà come gli utenti semplici possono creare il proprio profilo, modificare le informazioni personali, disconnettersi o eliminare il proprio account. La registrazione di un utente è indispensabile ai fini dell'utilizzo dell'applicazione; infatti, senza autenticazione le funzionalità disponibili sono quasi nulle.

Il primo passaggio necessario per utilizzare l'applicazione è l'accesso.

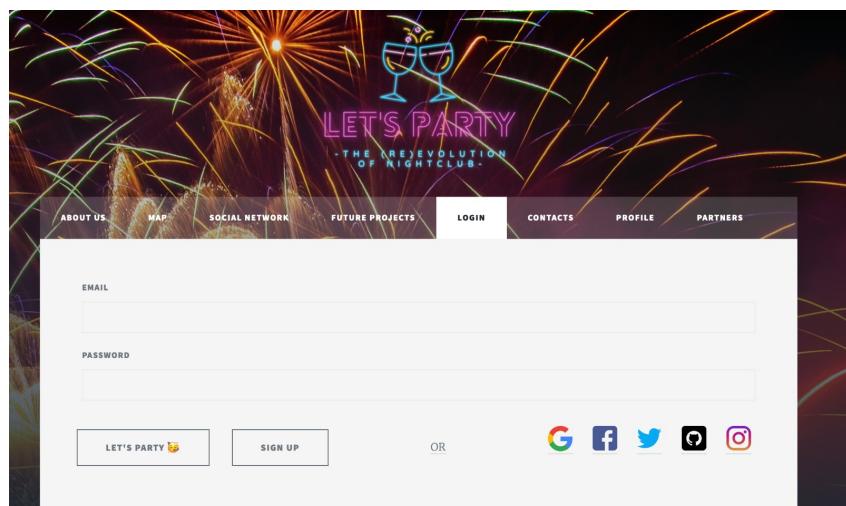


Figura 2.1. Schermata in cui l'utente può effettuare il log in

Oppure la registrazione se un utente non ha già creato un account.

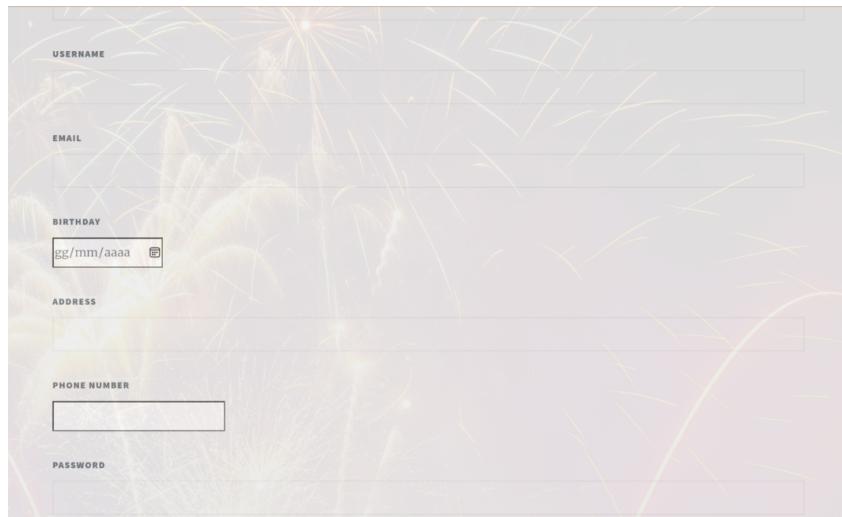


Figura 2.2. Schermata in cui l'utente può effettuare la registrazione

I campi necessari per la registrazione sono: *Nome*, *Cognome*, *Email*, *Username*, *Data di nascita*, *Indirizzo*, *Telefono*, *Password* e *Conferma*. Tutti i campi sono obbligatori e sono presenti controlli di sicurezza sulla password e di integrità e conformità dei dati su *Data di nascita*, *Email* e *Telefono*. Invece per l'accesso basteranno *email* e *password*. Ovviamente *email* e *username* sono unici (non possono esistere due utenti con la stessa mail o due utenti con lo stesso username).

Accesso e registrazione sono disponibili anche direttamente con google tramite l'API messa a disposizione da google stessa (google log-in API).

Cliccando su *Settings* sarà possibile accedere ad una sidebar da dove l'utente potrà scegliere se: Modificare il suo profilo, Disconnettersi dal suo profilo o Eliminare il profilo.

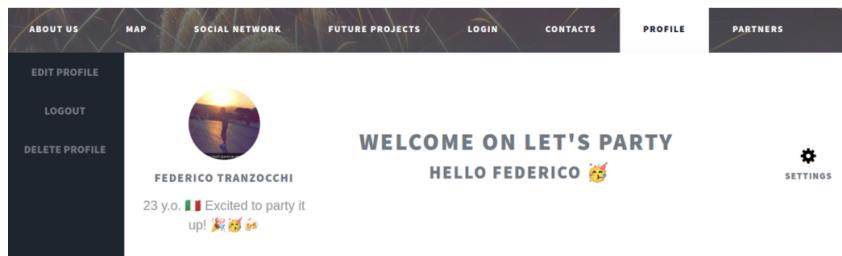


Figura 2.3. Sidebar con impostazioni

Selezionando la voce "Edit profile" sarà possibile accedere alla pagina di modifica del profilo, contenente un form con tutti i dati precedentemente inseriti in fase di

registrazione. Per motivi di sicurezza e integrità dei dati l'*username* non può essere cambiato, a differenza dei campi *Address*, *Email* e *Telefono*.

The screenshot shows a user interface for editing a profile. It includes fields for:

- FIRST NAME:** Federico
- SECOND NAME:** Tranzocchi
- USERNAME:** Federico.Tranzocchi
- EMAIL:** federico.tranzocchi@gmail.com
- ADDRESS:** via certo
- BIRTHDAY:** gg/mm/aaaa
- PHONE NUMBER:** 23
- BIO:** 23 y.o. 🇮🇹 Excited to party it up! 🎉🎉🎉

A **CONFIRM** button is located at the bottom right.

Figura 2.4. Schermata in cui l'utente può modificare il profilo

Inoltre da questa schermata sarà possibile procedere ad un'ulteriore personalizzazione del profilo modificando la propria foto profilo o la propria biografia, entrambi visibili ad ogni altro utente.

La schermata del profilo personale andrà a raccogliere le attività che l'utente svolgerà sulla piattaforma e che verranno successivamente illustrate nello specifico. Infatti nel proprio profilo personale compariranno i post e le recensioni pubblicati dall'utente, i suoi locali preferiti e i locali in cui ha prenotato.

In fondo l'utente può trovare un'ulteriore sezione "Invita amici" che gli permette di invitare altri amici ad iscriversi alla piattaforma dopo aver riempito un breve form dove sono richiesti: *Nome* e *Cognome* dell'invitato, *Email* e un messaggio di invito personalizzabile.

The screenshot shows a form titled "INVITE YOUR FRIENDS" with the following fields:

- YOUR FRIEND'S NAME:** (input field)
- EMAIL:** (input field)
- MESSAGE:** (text area)

A **SEND MESSAGE** button is located at the bottom left.

Figura 2.5. Schermata che l'utente può utilizzare per invitare un amico

Infine ogni profilo ha anche una versione pubblica e visibile a tutti dove compariranno Nome, Cognome, Foto profilo, Username e Bio personale. Essa contiene l'attività pubblica di un utente, come post, recensioni e locali preferiti.

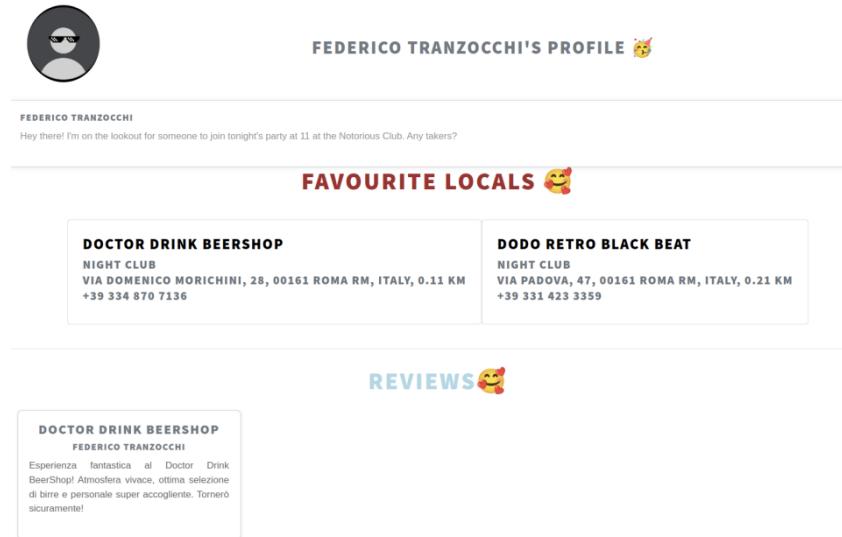


Figura 2.6. Profilo pubblico

2.2.2 Recensioni

Rimanendo nella schermata profilo personale ogni utente può trovare una sezione "Recensioni" con un bottone "Scrivi una recensione" che porterà ad un'apposita schermata.



Figura 2.7. Schermata in cui l'utente può scrivere una recensione

Ogni utente dovrà prima scegliere un locale e successivamente scriverà la recensione vera e propria. La scelta del locale da recensire è vincolata da due condizioni:

1. Ogni utente può recensire solo locali a cui ha messo like;

2. Ogni utente non potrà recensire lo stesso locale più di una volta.

Una volta scritta la recensione essa comparirà nel proprio profilo attraverso una card (contenente Locale, Username e contenuto recensione) e sarà visibile a tutti coloro che visualizzeranno il suo profilo. Ogni utente è anche libero di eliminare una recensione cliccando sul *cestino*.



Figura 2.8. Card review

2.2.3 Mappa

La funzionalità di punta del nostro progetto è nella sezione "Mappa", dove comparirà una mappa (possibile tramite la Google Maps API) che mostrerà la posizione in tempo reale dell'utente e una serie di marker, o cluster di marker, che indicheranno i locali nelle vicinanze.

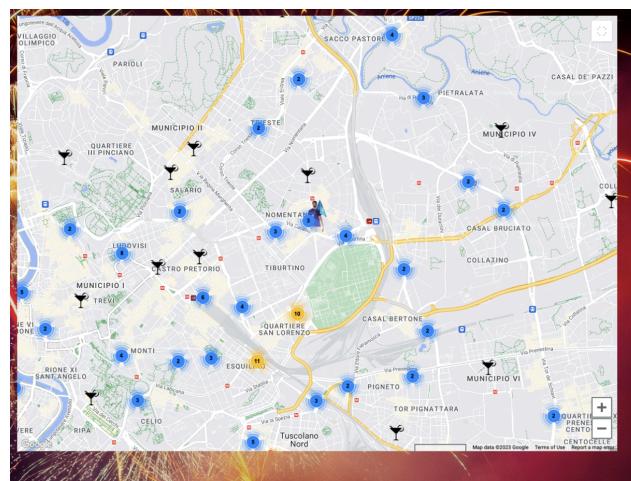


Figura 2.9. Mappa dei locali

La lista di locali è stata ottenuta effettuando scraping di dati su google maps. Infatti, impostando gli appositi parametri, mediante uno script² python viene generata una lista (salvata in un file .json) composta da tutti i locali che si trovano entro una

²Lo scraping è stato trattato in maniera approfondita nella relazione di un altro collaboratore del progetto

certa distanza da noi prestabilita, la quale è calcolata partendo dalle coordinate dell'utente.

Nel file *.json* sono raccolte tutte le informazioni principali di ogni locale: *Nome*, *Coordinate*, *Indirizzo*, *Telefono*, *Sito web* e altre ancora.

Passando col cursore su ogni icona verrà mostrato un pop-up contenente i dati del locale selezionato.

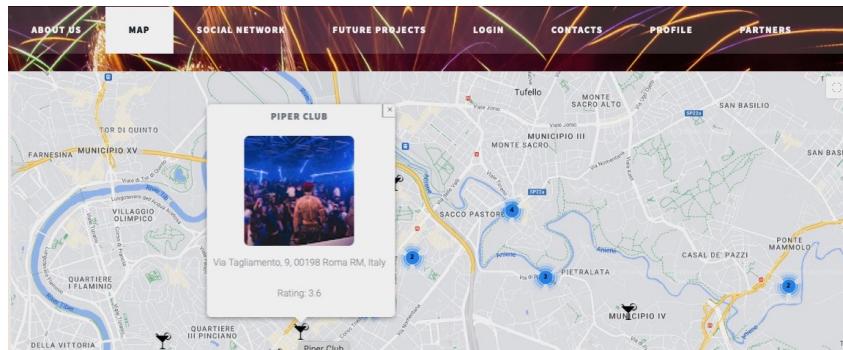


Figura 2.10. Pop up locale

2.2.4 Lista locali preferiti ed eventi

Immediatamente sotto la mappa vengono generate dinamicamente, da uno script JavaScript, una griglia di card. L'utente può visualizzare due tipi di card: eventi e locali. Le prime card visibili contengono informazioni su eventi organizzati da un *Organizzatore*, le successive invece rappresentano i locali individuati dallo scraping.

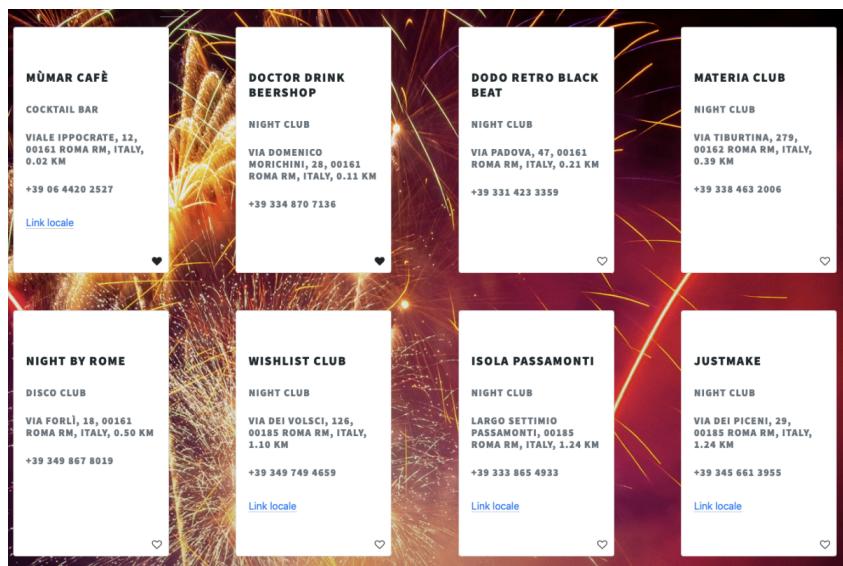


Figura 2.11. Lista card locali

Ogni card contiene tutte le informazioni principali del locale a cui si riferisce.

L'utente può interagire con le card in due modi: o cliccando sul *cuore*, o cliccando sul nome del locale. Cliccando sul cuore verrà aggiunto il locale ai preferiti e tale card comparirà nella pagina personale nella sezione "Locali preferiti". Invece, cliccando sul nome si aprirà una pagina che permette di prenotare un ingresso nel locale scelto e tale prenotazione verrà mostrata nella pagina personale sotto la sezione "Prenotazioni"³.

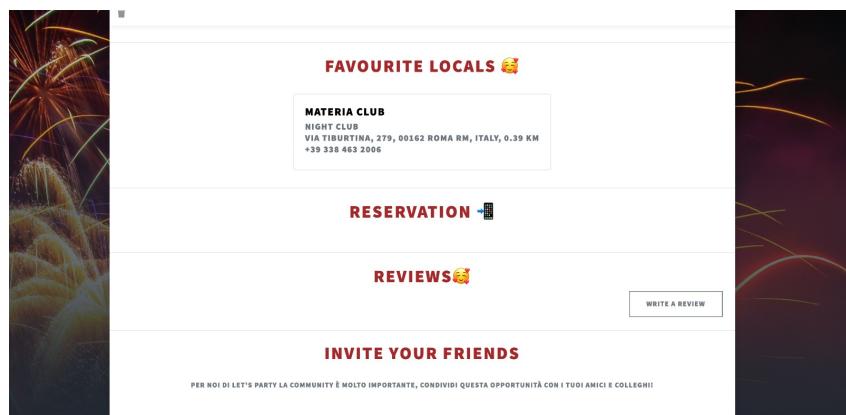


Figura 2.12. Vista del profilo

2.2.5 Social Network

La sezione Social Network è di fondamentale importanza per il nostro progetto. Accedendo a questa sezione l'utente si troverà una schermata contenente il bottone "Scrivi un post" e al di sotto una lista dei post pubblicati da tutti gli utenti, con la possibilità di mettere like a quelli che preferiscono.

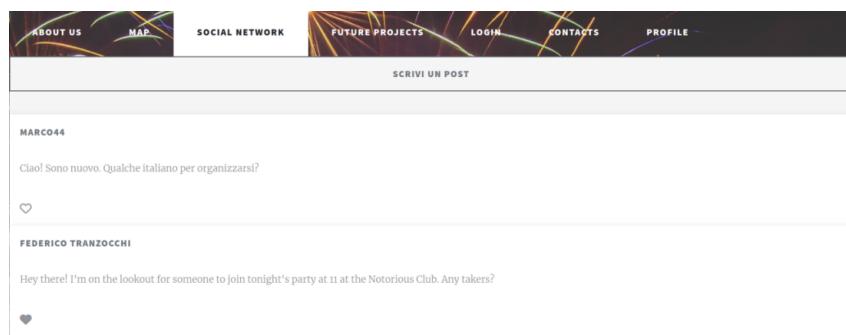


Figura 2.13. Feed

³Le prenotazioni sono state trattate in maniera approfondita nella relazione di un altro collaboratore del progetto

La schermata è molto intuitiva e di facile comprensione. Cliccando sul pulsante "Scrivi un post" l'utente potrà scrivere e pubblicare un post testuale.

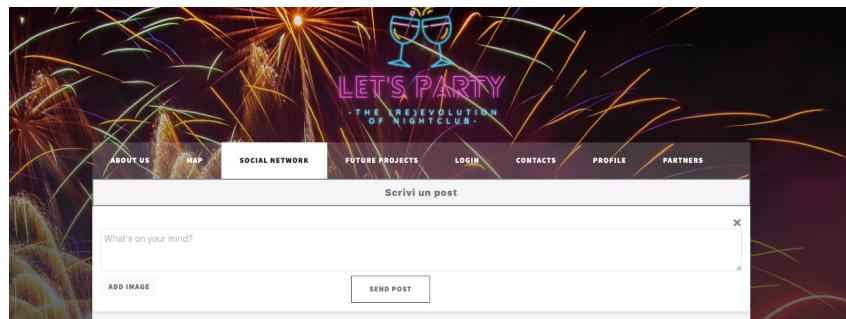


Figura 2.14. Scrittura post

Oltre ai semplici post testuali, ogni utente ha la possibilità di pubblicare fotografie per mostrare dove si trova, aggiungere foto di un locale o pubblicare foto con amici.



Figura 2.15. Post con foto

Come precedentemente detto, ogni post pubblicato da un utente sarà visibile a tutti e verrà mostrato anche nel proprio profilo, dove l'utente troverà l'icona *cestino* che gli permetterà di eliminare in maniera definitiva il post.

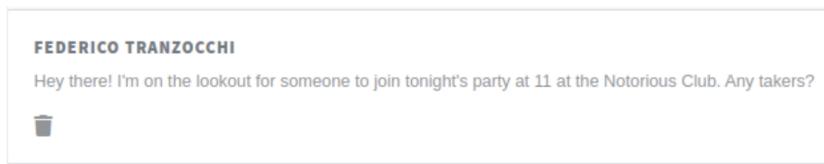


Figura 2.16. Eliminazione post

Questa componente social permette agli utenti di interagire tra loro per scambiarsi consigli oppure per sapere chi partecipa a determinati eventi, rendendola una delle feature più importanti della nostra piattaforma.

Capitolo 3

Tecnologie e metodologie utilizzate

In questo capitolo illustrerò brevemente gli strumenti utilizzati e le metodologie applicate per la realizzazione di questo progetto.

Nello sviluppo software ci sono diversi tipi di approcci alla realizzazione di un software, e in fase di ideazione abbiamo deciso di prediligere un approccio con *metodologia agile*.

3.1 Metodologie utilizzate

Il *modello a cascata* divide il processo di realizzazione di un software in cicli di vita, ognuno dei quali composto dalla successione dei medesimi passi; con il tempo questo modello è stato progressivamente abbandonato a favore di altre metodologie più efficienti, come ad esempio la *metodologia agile*. Alcune principali caratteristiche della metodologia agile sono:

1. Iterazione e incremento, processo di sviluppo diviso in iterazioni o cicli, ognuno con piccoli incrementi;
2. Adattamento ai cambiamenti, flessibile ed adatto ai cambiamenti nei requisiti e nelle priorità;
3. Test continui (*TDD*, Test Driven Development) per garantire la correttezza e la qualità del codice;
4. Valore per l'utente finale, ovvero le funzionalità vengono sviluppate in base all'importanza per l'utente.

La *programmazione estrema*, che abbiamo scelto di usare in questo progetto, è un esempio di metodologia agile. L'obiettivo principale di questo approccio consiste nel migliorare la qualità del codice e aumentare la capacità di adattamento ai cambiamenti nei requisiti del cliente. Questo metodo permette agli sviluppatori di concentrarsi principalmente sulla realizzazione del software cercando di soddisfare il più possibile i requisiti del cliente, portando anche a cicli di sviluppo brevi con pubblicazioni frequenti.

Le *storie utente* sono uno dei principali strumenti per spiegare agli sviluppatori, in linguaggio naturale e con esempi nel mondo reale, le funzionalità e i requisiti che il cliente richiede. Questa metodologia permette agli sviluppatori di concentrarsi, in fase di sviluppo, esclusivamente sulle funzionalità desiderate dal cliente (*BDD*, Behavior-Driven Development).

Ogni *storia utente* solitamente segue una struttura semplice, come ad esempio il *Connextra template*

**As a < type of user >,
So that < I can achieve some goal >,
I want to < do some task >**

Figura 3.1. Connextra template

3.2 Organizzazione del lavoro

Il nostro gruppo è composto da tre membri, in fase di ideazione ci siamo riuniti e abbiamo discusso su come organizzare il lavoro. Ogni membro del gruppo ha raccolto una serie di task da sviluppare per curare un determinato aspetto della piattaforma.

Il lavoro è stato svolto principalmente da remoto, pertanto si è reso necessario l'utilizzo di GitHub, un servizio di hosting per lo sviluppo software. Dopo la creazione di una repository ogni membro ha creato il proprio branch su cui caricare i propri aggiornamenti, rendendo quindi necessarie periodiche riunioni per effettuare la fusione delle diverse versioni garantendo l'integrità del codice e il funzionamento del software.

Dato il risultato raggiunto, ritengo che la scelta della *programmazione estrema* si sia rivelata vincente. Infatti, grazie a questa divisione dei lavori e delle funzionalità da sviluppare, ogni membro ha goduto della completa autonomia rispetto agli altri colleghi, ovvero ognuno ha avuto libera scelta riguardo a tecniche e stili di programmazione.

3.3 Tecnologie utilizzate

In questa sezione andrò a illustrare gli strumenti da noi impiegati in fase di sviluppo.

3.3.1 API impiegate

Successivamente elencherò le principali API da noi utilizzate.

Google maps

L'API di Google Maps ci ha permesso di inserire, nella sezione "Mappa", una mappa da noi personalizzata. Infatti abbiamo tolto tutte le icone da noi ritenute inutili, lasciando soltanto strade e trasporto pubblico, per poi aggiungere dinamicamente le nostre icone personalizzate che andranno a indicare la posizione di ogni locale.

In combinazione con questa API abbiamo utilizzato l'API *MarkerClusterer* che ci permette di raccogliere più icone in singoli cluster, rendendo quindi la mappa più pulita e ordinata.

L'utilizzo e la personalizzazione della mappa è stato possibile in seguito alla generazione di una nostra key personale dalla quale abbiamo generato e gestito la mappa stessa. Le due API sono state importate nell'header del file *map.ejs*.

Google auth

Tramite questa API è stato possibile implementare la funzionalità del login, o registrazione, utilizzando direttamente il proprio account google¹.

SerpAPI

Questa API è molto utile perchè permette di effettuare ricerche o scrape di dati. Nel nostro caso l'abbiamo utilizzato per fare scraping² dei locali da google maps e formattare i risultati in un file *.json*.

3.3.2 Node.JS

Node.JS è diventato sempre più popolare nello sviluppo di applicazioni web, per ciò la nostra scelta è ricaduta su di esso come ambiente di runtime principale. Esso permette di eseguire codice JavaScript anche dal lato server, non più solo dal lato client, permettendo di incentrare lo sviluppo di applicazioni web intorno a JavaScript

¹L'implementazione di questa funzionalità è stata curata da un collaboratore del progetto.

²Questa funzionalità è stata trattata in maniera più approfondita nella relazione di un mio collega.

stesso. Esso è ottimo per applicazioni web real-time perchè è *orientato agli eventi* (rende possibile la comunicazione asincrona).

I motivi che ci hanno portati a questa scelta sono molteplici:

1. **Efficienza e velocità di esecuzione;**
2. **Librerie**, è possibile usare una vasta quantità di libreria. Infatti Node ha un suo gestore di pacchetti chiamato *npm* (Node Package Manager);
3. **Scalabilità**, permette di gestire un traffico di utenti sempre più alto ed un carico di lavoro sempre più vasto;

Nel nostro progetto abbiamo usato Node.JS per creare il server vero e proprio. Successivamente elencherò una serie di librerie o framework basati su Node che ci sono tornati utili in fase di sviluppo.

Express

Express è un framework per web app basato su Node.JS. Noto per la sua facilità d'uso, è molto importante perchè consente di gestire le richieste HTTP, definire route, creare middleware personalizzati e gestire la generazione dinamica di pagine web.

Passport

Passport è un framework per l'autenticazione in Node.JS. Serve per gestire l'autenticazione degli utenti in applicazioni web, inoltre è molto versatile in quanto permette diverse strategie di autenticazione (username e password, token, OAuth ecc..). Passport ci ha permesso di poter implementare in maniera semplice i processi di accesso e registrazione al sito, ma anche di salvare la sessione attiva e mantenere l'utente autenticato.

```
const passport = require("passport");
const initializePassport = require("./passportConfig");
var result;
initializePassport(passport);
app.use(passport.initialize());
app.use(passport.session());
app.use(bodyParser.json());
app.use(session({ secret: 'secret', saveUninitialized: false,
  resave: false, cookie: { maxAge: 1000 } }));
app.use(flash());
```

Figura 3.2. Codice passport

Bcrypt

Bcrypt è una libreria di Node.JS utile per la crittografia delle password, basato sull'algoritmo di hashing *bcrypt* (robusto e sicuro). L'uso di questa libreria migliora la sicurezza dell'applicazione perchè rende difficile, in caso di un attacco informatico, decodificare o indovinare la password.

```
let hashedPassword = await bcrypt.hash	pw, 10);
```

Figura 3.3. Hashing password

Esso permette di effettuare l'hashing delle password prima di inserirle nel database, così da preservare la sicurezza delle password anche in caso di compromissione del database.

```
pool.query(
  `INSERT INTO users (name, surname, username, email, b_day, address, p_num, pw)
  VALUES ($1, $2, $3, $4, $5, $6, $7, $8 )
  RETURNING username,pw` , [name, surname, username, email, b_day, address, p_num, hashedPassword], (err, results) =>{
    if(err){
      throw err;
    }
  }
```

Figura 3.4. Inserimento password criptata

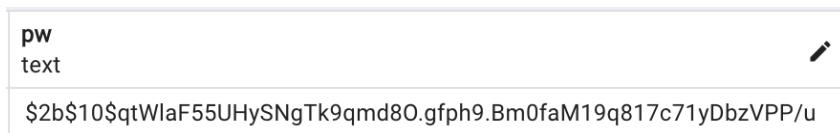


Figura 3.5. Password nel database

Nodemailer

Nodemailer è una libreria Node.JS che permette di mandare email tramite un'applicazione implementata mediante Node. Risulta molto utile perchè permette di implementare funzionalità come l'invio automatico di mail (ad esempio in fase di registrazione). Nel nostro software questa libreria è stata utilizzata per l'implementazione di due storie utente: il form per contattare gli amministratori del sito e il form presente nella pagina del profilo per invitare un amico.

Lato server, l'invio della mail è gestito mediante un metodo *POST*. Una volta installato e inizializzato il *nodemailer* nel server ho dovuto prima di tutto gestire il mittente:

```
var transporter = nodeMailer.createTransport({
  service: 'gmail',
  auth: {
    user: process.env.GMAIL_USER,
    pass: process.env.GMAIL_PASSWORD
  }
})
```

Figura 3.6. Mittente nodemailer

Gmail permette la creazione di una chiave di accesso personalizzata per consentire l'invio di una mail tramite questa libreria. Successivamente ho impostato i vari parametri della mail (contenuto e nome del mittente sono mandati dal form compilato dall'utente) e mediante il metodo `.sendMail(..)` avviene l'invio vero e proprio.

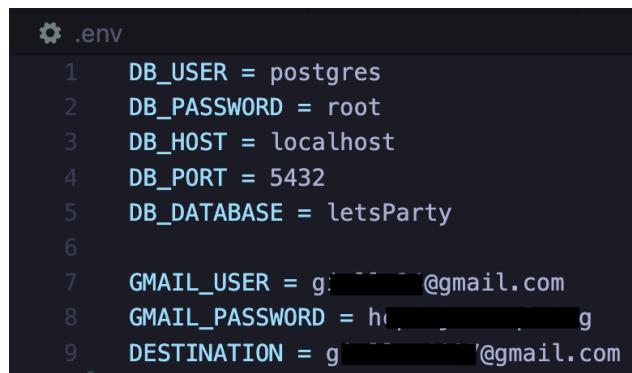
```
var mailOptions = {
  from: process.env.GMAIL_USER,
  to: process.env.DESTINATION,
  subject: `Message from ${_name}, ${_email}`,
  html: `${_msg}`
}
transporter.sendMail(mailOptions, function(error, info){
  if (error){
    throw error;
  }
  else {
    console.log("email sent");
  }
  res.redirect("/");
})
```

Figura 3.7. Invio della mail

Nel caso in cui l'utente voglia contattare gli amministratori le mail del mittente e del ricevente sono entrambe possedute da noi amministratori, mentre se l'utente vuole invitare un amico il mittente sarà sempre quello definito tramite il nodemailer mentre il destinatario sarà mandato al server dal form compilato.

I parametri evidenziati in giallo sono variabili d'ambiente da me definite nel file `.env`³, il quale contiene una serie di variabili di ambiente utili al corretto funzionamento del server.

³La censura nella Fig. 3.8 si è resa necessaria perché le mail da me utilizzate sono entrambe in funzione anche al di fuori del progetto.



```
⚙ .env
1 DB_USER = postgres
2 DB_PASSWORD = root
3 DB_HOST = localhost
4 DB_PORT = 5432
5 DB_DATABASE = letsParty
6
7 GMAIL_USER = g...@gmail.com
8 GMAIL_PASSWORD = h...g
9 DESTINATION = g...@gmail.com
```

Figura 3.8. File .env

Come si può vedere da questa immagine, in questo file sono contenute anche le variabili utili alla connessione con il database.

3.3.3 Presentation tier

In questa sezione mi concentrerò sugli aspetti tecnici e le scelte tecnologiche riguardanti l'User Interface (UI).

File HTML e EJS

Nello sviluppo dell'interfaccia grafica siamo partiti da semplici pagine html. In HTML però è possibile solo creare pagine statiche, quindi si è reso necessario convertire alcune pagine HTML in file EJS (*Embedded JavaScript*), i quali sono tutti raccolti nella cartella `"/views"`.

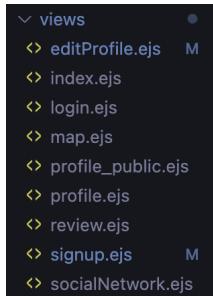


Figura 3.9. File EJS

Questo tipo di file ci permette di generare dinamicamente contenuti HTML e facilita l'incorporazione di codice JavaScript all'interno di un documento HTML. Pertanto questa conversione si è resa indispensabile per creare pagine web dinamiche e personalizzate, permettendoci ad esempio di inserire variabili dinamiche e visualizzare dati o informazioni provenienti da sorgenti esterne.

Bootstrap

Bootstrap è un framework per lo sviluppo del front-end di applicazioni e siti web. Esso fornisce una serie di strumenti per la realizzazione di interfacce moderne e accattivanti. Abbiamo utilizzato Bootstrap per creare lo stile unico e moderno che contraddistingue il nostro sito, ma me ne sono servito anche nella realizzazione della sezione "Profile". Esso viene anche utilizzato per creare interfacce responsive, andando a migliorare l'interattività del sito e di conseguenza rendendo l'esperienza dell'utente più fluida e piacevole.

jQuery

jQuery è una libreria JavaScript ampiamente utilizzata per aggiungere interattività ed effetti di animazione ad applicazioni web, semplificando lo sviluppo lato client. In particolare, nel nostro progetto, è stato utilizzato nel file *main.js* (*/public/javascript*) per aggiungere un *effetto parallax* all'interfaccia.

L'*effetto parallax* è un particolare effetto utilizzato nella programmazione lato client per dare una sensazione di profondità e dinamicità alla schermata, offrendo all'utente un'esperienza visiva fluida e immersiva. Questo effetto è ottenibile facendo scorrere gli elementi in primo piano a velocità maggiore di quelli sullo sfondo mentre l'utente naviga o interagisce con l'interfaccia.

3.3.4 Data tier

In questa sezione farò un'analisi riguardante l'implementazione del data tier.

PostgreSQL

PostgreSQL è un sistema di gestione di database relazionali (RDBMS). La scelta è ricaduta su questo RDBMS perchè, in seguito ad un confronto coi miei colleghi, riteniamo sia la scelta giusta per il nostro progetto. Infatti PostgreSQL è noto per la sua robustezza, affidabilità e garantisce l'integrità dei dati, supporta le transizioni ACID (*Atomicity, Consistency, Isolation e Durability*).

Inoltre utilizza un linguaggio SQL che offre diverse funzionalità avanzate come le *views* (viste), *JOIN* complessi e tante altre.

Lato server

Lato server la connessione al database avviene attraverso l'esecuzione dello script *dbConfig.js*. Le variabili evidenziate in giallo si riferiscono alle variabili di ambiente definite nel file *.env* di cui ho precedentemente parlato.

```
require('dotenv').config();
const { Pool } = require('pg');
const isProduction = process.env.NODE_ENV === "production";
const connectionString = `postgresql://${process.env.DB_USER}:${process.env.DB_PASSWORD}`
    `@${process.env.DB_HOST}:${process.env.DB_PORT}/${process.env.DB_DATABASE}`;
const pool = new Pool({
  connectionString: isProduction? process.env.DATABASE_URL : connectionString
});
module.exports = { pool };
```

Figura 3.10. File dbConfig.js

Una volta stabilita la connessione, le interazioni tra il server e il database sono possibili mediante delle query testuali. Client e server possono interagire principalmente mediante metodi *POST* e *GET*: i metodi *POST* servono a mandare dati al server mentre i metodi *GET* servono a ottenere dati dal server.

Pertanto in occasione di un metodo *GET* verrà eseguita una query che selezionerà i dati richiesti dall'utente e li restituirà mediante la funzione *res.send(..)*.

```
app.get("/profile/list/:name", (req,res) => {
  var name = req.params.name == '0' ? req.user.username : req.params.name;
  pool.query(`SELECT *
    FROM fav
    WHERE utente = $1`, [name], (err,results) => {
      if (err) throw err;
      res.send(JSON.stringify(results.rows));
    })
})
```

Figura 3.11. Esempio query GET

Invece in caso di un metodo *POST* sarà possibile mandare i dati al server che mediante una query provvederà ad aggiornare il database (inserire un nuovo elemento o aggiornare un elemento già esistente).

```
app.post('/users/map/update',(req, res) => {
  let {title, type, address, dist, phone, website} = req.body.card;
  //req.user è definito
  pool.query(`
    INSERT INTO fav
    VALUES ($1,$2,$3,$4,$5,$6,$7)`, [req.user.username,title,type,address,
                                         dist,phone,website], (err, res) => {
      if (err) throw err;
    })
})
```

Figura 3.12. Esempio query POST

Capitolo 4

Il data tier: progettazione concettuale

In questo capitolo tratterò la progettazione concettuale della base di dati. L'obiettivo di quest'ultima consiste nel fornire una rappresentazione corretta e affidabile della realtà di interesse partendo dai requisiti e dalle specifiche informali.

Presenterò la progettazione della base di dati partendo dall'analisi dei requisiti, indispensabile per la progettazione perchè ci permette di avere una base di dati coerente con i requisiti che l'applicazione deve rispettare. Successivamente presenterò il diagramma entità-relazione che darà una rappresentazione visiva delle relazioni tra le varie entità e dei vari requisiti.

4.1 Analisi dei requisiti

L'analisi dei requisiti è una fase fondamentale nella progettazione di una base di dati, è il processo completo di identificazione delle problematiche che l'applicazione deve affrontare e delle funzionalità che l'applicazione deve includere. Inizialmente, i requisiti sono raccolti attraverso specifiche che vengono generalmente espresse in linguaggio naturale, il che può renderle ambigue e prive di organizzazione.

Successivamente elencherò le specifiche da noi identificate che tale basi di dati deve rispettare per garantire il corretto funzionamento dell'applicazione.

- **Users**, ci sono due tipi di utenti: quelli semplici e quelli *Company*. Di ogni utente ci interessa Nome, Cognome, Username, Data di nascita, Telefono, Indirizzo, Password ed Email. A questi si aggiungono Immagine del profilo e Bio che sono modificabili in un secondo momento. Ogni utente può mettere

like ad un post al massimo una volta e può scrivere più post. Ogni utente può mettere like ad i locali, ma non può mettere like allo stesso locale più volte. Ogni utente può recensire solo locali che ha messo tra i preferiti e non può recensire lo stesso locale più di una volta. Ogni utente può prenotare in un locale, effettuando anche più prenotazioni per lo stesso locale. Queste funzionalità sono condivise sia dagli utenti semplici che dagli utenti Company. Questi ultimi avranno dei campi in più (CoRole, CoName e VAT number) e potranno anche organizzare eventi.

- **Post.** Ogni post è identificato da un ID e ha un testo e il numero di like. Ogni post ha un unico autore e può ricevere like da tutti gli utenti, anche dall'autore stesso.
- **Fav,** è identificato delle coppie Utente-locale perchè ogni utente può mettere like ad un locale al massimo una volta. Ogni utente può mettere like a più locali (anche zero).
- **Review.** Ogni review è identificata dall'utente che l'ha scritta e dal locale a cui si riferisce, oltre al testo della recensione. Ogni recensione scritta da un utente su un determinato locale è unica, ovvero non ci possono essere due recensioni dello stesso utente sullo stesso locale. Affinchè un utente possa recensire un locale è necessario che l'utente abbia messo like a quel locale.
- **Locale.** Ogni locale è identificato dal suo nome. Dei locali conosciamo: Tipo, Coordinate, Indirizzo, Telefono e Sito web* (se presente). Ogni locale può essere messo tra i preferiti da ogni utente ma non più di una volta da ognuno. Ogni locale può essere recensito da tutti gli utenti che l'hanno messo tra i preferiti ma non può essere recensito più di una volta dallo stesso utente.
- **Book,** di ogni prenotazione interessa se include o meno il tavolo o se è un ingresso normale. Ogni utente può effettuare più prenotazioni per lo stesso locale.
- **Evento,** identificato da utente organizzatore e dalla data. Ogni evento in una certa data è creato da un solo organizzatore, mentre un utente organizzatore può organizzare più eventi.

4.2 Diagramma Entità-Relazioni

Il diagramma ER è uno schema utilizzato nella progettazione concettuale di una base di dati per dare una rappresentazione visiva dei dati di interesse, della loro struttura e delle relazioni tra le entità che vanno a comporre il sistema.

L'utilizzo di questo diagramma in fase di progettazione è molto importante perché è slegato dall'effettiva implementazione della base di dati ma, se ben fatto, ci dà una descrizione ben precisa della realtà di interesse, andando a facilitare l'implementazione della base di dati finale.

Le componenti chiave di un diagramma ER sono:

- **Entità**, rappresentano oggetti che sono di interesse per la base di dati e sono indicate con un rettangolo;
- **Relazioni**, indicano come sono tra loro collegate le entità e sono rappresentate mediante dei rombi;
- **Attributi**, ogni entità o relazione può avere degli attributi, i quali sono caratteristiche o proprietà particolare dell'elemento a cui si riferiscono;
- **Identifieri**, servono a identificare un'entità tramite degli attributi unici, ovvero non esisteranno due entità con lo stesso identifieri (ad esempio due post con lo stesso ID nel nostro caso);
- **Chiavi esterne**, sono relazioni tra entità diverse all'interno di una base di dati relazionale. Servono per esprimere le relazioni tra le entità.

Per quanto riguarda la nostra applicazione, questo è il diagramma che descrive più fedelmente la nostra realtà di interesse.

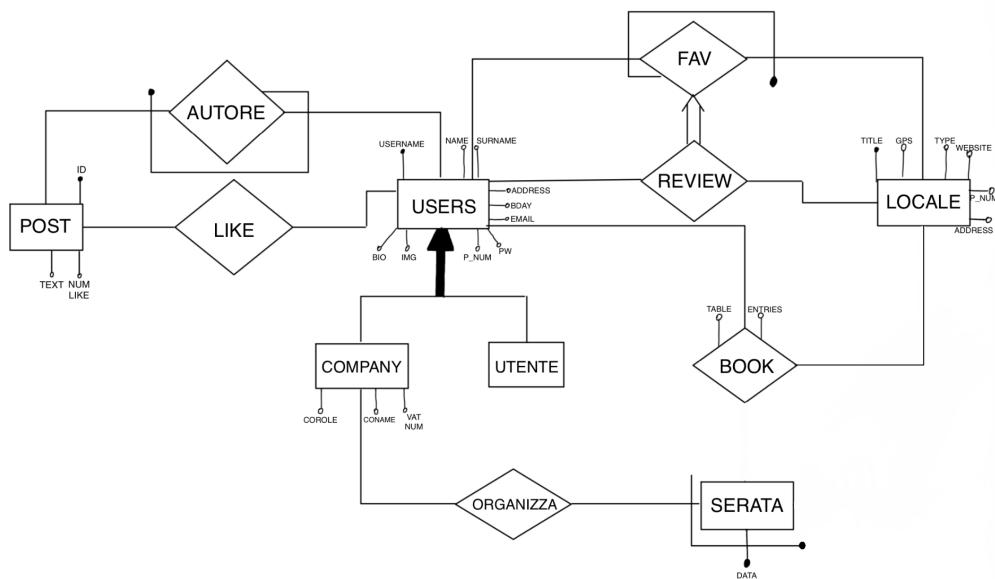


Figura 4.1. Diagramma ER

Per dare una descrizione più accurata del diagramma ho provveduto a specificare i vincoli esterni, i vincoli di chiave esterna e a realizzare tre tabelle che analizzassero le componenti principali del diagramma.

La Tabella 4.1 rappresenta la tabella delle entità. Essa contiene per ogni entità una breve descrizione, gli attributi e gli identificatori.

La Tabella 4.2 è la tabella delle relazioni. Di ogni relazione ho fatto una breve descrizione e ho illustrato le entità che la compongono, gli attributi che contiene e i relativi identificatori.

La Tabella 4.3 analizza gli attributi di ogni entità/relazione.

4.2.1 Vincoli

Qui elenco i vincoli di chiave esterna:

- Organizza[Organizzatore] è contenuto in Company[Username];
- Utente[Username] è contenuto in Users[Username];
- Company[Username] è contenuto in Users[Username];
- Autore[Username] è contenuto in Users[Username];
- Autore[PostID] è contenuto in Post[ID];
- Like[Username] è contenuto in Users[Username];
- Like[PostID] è contenuto in Post[ID];
- Fav[Username] è contenuto in Users[Username];
- Fav[NomeLocale] è contenuto in Locale[Nome];
- Review[Username] è contenuto in Users[Username];
- Review[NomeLocale] è contenuto in Locale[Nome];
- Review[Username, NomeLocale] è contenuto in Fav[Username, NomeLocale];
- Book[Username] è contenuto in Users[Username];
- Book[NomeLocale] è contenuto in Locale[Nome];

Infine, elenco i vincoli esterni presenti nel diagramma ER:

- Company[Username] unione Utente[Username] == Users[Username];
- Post[ID] >= 0;
- Post[Numero Like] >= 0;

- Serata[Data] è una data futura;

4.2.2 Tabelle

Entità	Descrizione	Attributi	Identifieri
Users	Indica il generico utente che naviga nell'applicazione, contiene due sottinsiemi di utenti (Company o semplici)	Nome, Cognome, Username, Telefono, Indirizzo, Email. , Password, Bio, IMG	Username
Utente	Utente semplice		Username
Company	Utente che può organizzare eventi	VAT number, Company Name, Company Role	Username
Post	Contenuto testuale o visivo che qualsiasi tipo di utente può pubblicare	ID, Text, Numero Like	ID
Locale	Locale ottenuto mediante lo scraping	Nome, Indirizzo, Telefono, Indirizzo, Coordinate, Tipo	Nome
Serata	Serata organizzata da un utente Company	Data, Titolo	Organizzatore, Data

Tabella 4.1. Tabella delle entità

Relazione	Descrizione	Componenti	Attributi	Identificatori
Autore	Lega ogni post al suo unico autore	Post, Users	PostID, Username	PostID, Username
Like	Associa ogni post ad un utente che gli ha messo like	Post, Users	PostID, Username	PostID, Username
Fav	Contiene le coppie utente-locale, ovvero per ogni utente contiene i locali a cui ha messo like	Users, Locale	Username, NomeLocale	Username, Nome
Review	Contiene un sottinsieme delle coppie utente-locale di Fav (sottorelazione). Contiene le recensioni che un utente fa ai locali	Users, Locale	Username, Nome, Contenuto	Username, NomeLocale
Book	Describe le prenotazioni che un utente fa ad un locale	Users, Locale	Username, NomeLocale, Table, Entry	Username, NomeLocale
Organizza	Contiene gli eventi organizzati da un utente Company	Company, Serata	Data, Nome-Serata, Organizzatore	Data, Organizzatore

Tabella 4.2. Tabella delle relazioni

Attributo	Entità/Relazione	Dominio	Descrizione
ID	Post	SERIAL INT	Identificatore post
Text	Post	STRING	Contenuto del post
Numero Like	Post	INT	Numero dei like ricevuti
Username	Users	STRING	Identificatore dell'utente
Nome	Users	STRING	Nome dell'utente
Cognome	Users	STRING	Cognome dell'utente
Indirizzo	Users	STRING	Indirizzo dell'utente
Data di nascita	Users	DATE	Data di nascita dell'utente
Email	Users	STRING	Email dell'utente
Password	Users	STRING	Password dell'utente
Telefono	Users	NUMERIC	Numero di telefono
Img	Users	BIN	Immagine bio
Bio	Users	STRING	Biografia dell'utente
VAT Number	Company	NUMERIC	Numero VAT
Company Name	Company	STRING	Nome dell'azienda per cui lavora
Company Role	Company	STRING	Ruolo dell'utente nell'azienda
Nome	Locale	STRING	Nome del locale e identificatore
GPS	Locale	STRING	Coordinate gps del locale
Tipo	Locale	STRING	Tipo di locale
Website	Locale	STRING	Sito web del locale
Telefono	Locale	NUMERIC	Numero di telefono del locale
Indirizzo	Locale	STRING	Indirizzo del locale
Data	Serata	DATE	Data serata
Nome	Serata	STRING	Titolo serata
Testo	Review	STRING	Testo della review
Table	Book	STRING	Ingresso con tavolo
Entry	Book	STRING	Ingresso semplice

Tabella 4.3. Tabella degli attributi

Capitolo 5

Testing e Validazione

Il *testing* è un processo fondamentale nello sviluppo di un software. Consente di individuare difetti, errori e di valutarne la coerenza coi requisiti specificati e consiste nell'eseguire il software, o parte di esso, andando a ricercare malfunzionamenti. Pertanto le principali finalità del testing sono:

- Aumentare l'affidabilità;
- Migliorare la qualità del software;
- Rilevare errori;
- Ricerca vulnerabilità e verifica della sicurezza.

Invece la fase di *validazione* si concentra sulla verifica che il software soddisfi i requisiti previsti e le aspettative degli utenti e che sia adatto all'uso previsto. Piuttosto che test tecnici, la *validazione* comprende l'esecuzione di test reali con l'utente finale, o con il cliente che ha commissionato tale software, per verificare che il prodotto sia coerente con le esigenze degli utenti e con le specifiche di progetto.

5.1 Software Testing

Inizialmente la fase di sviluppo e la fase di testing erano separate e gli autori di queste due parti erano distinti. Invece con lo sviluppo di nuove tecniche più moderne, come nel caso dei *metodi agili*, ogni iterazione agile include una fase di testing e saranno gli sviluppatori stessi a testare il loro codice. Inoltre, al giorno d'oggi, gli strumenti per effettuare il testing sono molto avanzati e anche automatizzati.

Questa ulteriore differenza tra l'approccio agile (moderno) e l'approccio tradizionale va a rimarcare l'inefficienza e lo spreco di tempo e risorse legati a quest'ultimo;

infatti, nei metodi agili è di fondamentale importanza ridurre i costi generali di sviluppo e rispondere rapidamente ad ogni esigenza (ad esempio cambi nei requisiti o fix di bug ed errori).

A differenza degli approcci tradizionali al testing i metodi agili introducono delle novità, come ad esempio:

- Automazione dei test;
- Test collaborativi;
- Test singole Storie Utente;
- Testing continuo in ogni fase di sviluppo;

Due esempi di approcci allo sviluppo di un software sono il *Behavior Driven Development* (BDD) e il *Test Driven Development* (TDD). Tali approcci sono caratterizzati dalla scrittura di test prima o in parallelo con lo sviluppo.

5.1.1 Test Driven Development

Impiegando questo approccio i test vengono preparati prima di scrivere il codice, dando vita ad un ciclo chiamato "*Red-Green-Refactor*":

- **Red**, inizialmente si scrive un test, fallimentare, per una nuova funzionalità non ancora implementata. Rappresenta il punto di partenza del ciclo;
- **Green**, si scrive il codice minimo necessario a passare il test;
- **Refactor**, una volta passato il test col codice minimo, si ottimizza e migliora quest'ultimo senza cambiare la funzionalità implementata.

Questa tecnica risulta molto efficiente perché garantisce un'alta qualità del codice e assicura che l'inserimento di modifiche nel codice non generi errori o alteri il comportamento del software.

5.1.2 Behavior Driven Development

Esso si basa sullo stesso concetto del TDD ma lo estende analizzando il comportamento del software da un punto di vista più ampio.

Gli scenari di test sono scritti in linguaggio naturale, utilizzando ad esempio uno stile chiamato *Gherkin*. *Gherkin* è un linguaggio, detto "di dominio specifico", usato nel testing di software per scrivere scenari di test a tutti comprensibili. Tali scenari, infatti, sono scritti in linguaggio naturale e si basano sull'utilizzo di parole chiave (*Feature, Scenario, Given, When, Then, And*).

```

Feature: User can manually add movie | 1 Feature

Scenario: Add a movie | ≥1 Scenarios / Feature
Given I am on the RottenPotatoes home page
When I follow "Add new movie"
Then I should be on the Create New Movie page
When I fill in "Title" with "Men In Black"
And I select "PG-13" from "Rating"
And I press "Save Changes"
Then I should be on the RottenPotatoes home page
And I should see "Men In Black"

3 to 8 Steps / Scenario

```

Figura 5.1. Esempio di scenario di test

Tali scenari possono essere automatizzati attraverso dei tools (es. *Cucumber*).

Cucumber è un framework di testing automatizzato famoso per la capacità di impiegare test scritti in linguaggio naturale. Viene utilizzato nel contesto del Behavior Driven Development e del Test Driven Development ed è molto popolare per la sua integrazione con diversi linguaggi di programmazione (Java, Ruby, JavaScript).

In seguito mostro un esempio di test da me realizzato con Cucumber e Puppeteer per testare l'accesso a un account di un utente semplice. Puppeteer è una libreria di Node utilizzata per controllare e automatizzare il browser web Chrome ed è ampiamente impiegato nello scraping di dati e nel testing automatizzato di pagine web. Il suo utilizzo si è reso necessario per simulare l'apertura di un browser e l'inserimento dei dati per effettuare il login.

Inizialmente ho creato un file *accesso_account.feature* contenente la feature da testare con lo scenario e i relativi passi.

```

feature > accesso_account.feature
1   Feature: Accesso all'account per utente semplice
2
3     Scenario: Accesso con credenziali valide
4       Given L'utente è sulla pagina di accesso
5       When L'utente inserisce le credenziali gip@gmail.com e ciaociao
6       Then L'utente viene autenticato con successo

```

Figura 5.2. Esempio di scenario per testare il login

In seguito ho creato uno script JavaScript contenente la definizione dei passi definiti nel file *.feature*. All'inizio di questo file ho importato le tre librerie necessarie: *Cucum-*

ber, Puppeteer e Assert. L'utilizzo delle prime due è stato spiegato precedentemente, invece *Assert* è un modulo di Node utilizzato per verificare se le asserzioni siano vere o false. In questo caso l'ho impiegato per verificare se l'URL ottenuto fosse uguale a quello atteso.

Successivamente ho definito le variabili necessarie all'esecuzione dello script come *browser* e *page* utilizzate per la navigazione web con Puppeteer e gli URL di riferimento *URL_login* e *URL_profile*.

```

1  const { Given, When, Then } = require('cucumber');
2  const puppeteer = require('puppeteer');
3  const assert = require('assert');
4
5  let browser;
6  let page;
7  var URL_login = 'http://localhost:4060/users/login';
8  var URL_profile = 'http://localhost:4060/users/profile';

```

Figura 5.3. Preparazione ambiente per test

Infine ho definito i vari passi:

```

10 Given('L'utente è sulla pagina di accesso', async function () {
11   browser = await puppeteer.launch();
12   page = await browser.newPage();
13   await page.goto(URL_login);
14 });
15 When('L'utente inserisce le credenziali {string} e {string}', {
16   async function (email, password) {
17     await page.type('#email', email);
18     await page.type('#password', password);
19     await page.click('#login-button');
20   });
21 Then('L'utente viene autenticato con successo', async function () {
22   const currentURL = page.url();
23   assert.strictEqual(currentURL, URL_profile,
24                     'L'utente non è stato autenticato correttamente');
25   await browser.close();
26 });

```

Figura 5.4. Definizione dei vari step

- Nel passo *Given* ho creato una nuova finestra di un browser grazie a Puppeteer e mi sono collegato all'URL da me definito nell'intestazione (*URL_login*);
- Nel passo *When* mediante il metodo asincrono *type(..)* di Puppeteer scrivo le credenziali da me definite nello scenario nei campi input e mediante il metodo

asincrono `click(..)` confermo l'invio dei dati per tentare il login.

- Nell'ultimo passo *Then* verifico tramite `assert` che l'URL ottenuto sia uguale a quello atteso, il quale è stato da me definito precedentemente (`URL_profile`).

Per brevità riporto solamente questo esempio per mostrare come ho effettuato il test di una semplice feature, ma ovviamente il testing vero e proprio è esteso a tutti gli aspetti dell'applicazione.

Capitolo 6

Conclusioni

6.1 Sintesi dei punti chiave

In questa relazione abbiamo esplorato l'intero processo di ideazione, progettazione e realizzazione del nostro software chiamato "Let's Party". Questo innovativo provider di servizi per utenti, locali e organizzatori è progettato per semplificare la vita degli utenti e migliorare l'esperienza di chi cerca luoghi nelle vicinanze per divertirsi, fare nuove conoscenze con il vantaggio di poter prenotare tali locali direttamente sulla piattaforma.

Durante la fase di ideazione, ci siamo concentrati principalmente sul miglioramento dell'esperienza utente sotto tutti i punti di vista. È stato imperativo realizzare un'interfaccia grafica unica, moderna e accattivante in grado di attirare l'attenzione dell'utenza. Questo, combinato con la facilità d'uso e la creazione di un'interazione sociale all'interno dell'applicazione, ha creato la formula perfetta per un "neo-nato" social network.

6.2 Valutazione complessiva dell'applicazione

"Lets Party" rappresenta un progetto ambizioso e con un enorme potenziale. La combinazione di molteplici funzionalità semplici e utili, un'interfaccia grafica intuitiva e accattivante, con l'obiettivo di puntare sulla creazione di una community dedicata agli amanti del divertimento e delle feste ne fanno un'opzione allettante per gli utenti.

La nostra scelta di tecnologie avanzate e API all'avanguardia consentono all'utente un uso sicuro, affidabile e performante, mentre danno a noi sviluppatori una base solida per il futuro sviluppo e l'espansione dell'applicazione, fino ad arrivare ad una pubblicazione.

In conclusione, il progetto "Lets Party" ha il potenziale per avere un notevole successo e per soddisfare al meglio ogni tipo di utente. Ritengo che con un adeguato sviluppo e un'efficace strategia di marketing, questo progetto potrebbe raggiungere risultati significativi e diventare un punto di riferimento per giovani, turisti e amanti dell'intrattenimento.

Bibliografia

- [1] D. Calvanese, G. De Giacomo, M. Lenzerini, *Dispense del corso "Basi di dati"*.
- [2] L. Querzoni, D. C. D'Elia, *Dispense del corso "Laboratorio di Applicazioni Software e Sicurezza Informatica"*.
- [3] Documentazione ufficiale Node, <https://nodejs.org/it/docs>
- [4] Documentazione ufficiale *npm* (Node Package Manager), <https://www.npmjs.com>
- [5] Documentazione ufficiale Google Developers, <https://developers.google.com/docs>
- [6] La figura 1.1 è stata presa da <https://subscription.packtpub.com/book/programming/9781787287495/2/ch02lvl1sec19/three-tier-client-server-architecture>

Ringraziamenti

Ringrazio prima di tutto la mia famiglia. Mamma e papà per la possibilità che mi hanno dato e che continuano a darmi, sono sempre stati presenti e mi hanno supportato in ogni momento di questo percorso.

Ringrazio Alessandra, perché è sempre stata disponibile e pronta ad aiutarmi e a darmi consigli per affrontare questo percorso a me del tutto nuovo, ma soprattutto ha sopportato io che me la mia piangevo per ogni cosa.

Grazie a nonna e il suo “auguri e in bocca al lupo” mi ha sempre accompagnato in ogni esame di questi anni, con cui teneva a farmi sapere la sua vicinanza e il suo credere in me.

Grazie a zia Sandra che è sempre stata presente e pronta ad aiutarmi facendomi sentire la vicinanza della famiglia anche vivendo da solo, e soprattutto perché non ha mai perso occasione per chiedermi “ma fisica quando la fai?”.

Grazie a zia Patrizia e a zio Fernando che non hanno mai fatto mancare - da lontano e da vicino - la loro vicinanza, il loro affetto e i loro preziosi consigli.

Grazie a Elena, con lei abbiamo sempre la possibilità di confrontarci delle nostre esperienze, oltre al tempo che insieme condividiamo d'estate, anche se ormai il tempo per vedersi è sempre meno.

Grazie a zia Giuliana, che con le nostre chiamate domenicali mi è sempre stata vicina e ha sempre manifestato il suo supporto e il suo credere in me.

Passando agli amici;

Il primo ringraziamento va a tutto PIP. Matteo, Luca e Stefano ormai da anni siete una costante nella mia vita, con voi ho condiviso tutto quello che mi è successo negli ultimi anni e mi siete sempre stati vicini, nei momenti belli e nei momenti brutti, credendo sempre in me anche quando ero scoraggiato.

Ringrazio Matteo per quello che è stato per me fin da quando ci siamo conosciuti. Al di là dell'università sei stata una delle persone più importanti della mia vita e se sono arrivato a questo punto è anche merito tuo.

Grazie a Luca che negli ultimi anni sei stata una persona sempre più presente nella mia vita e con cui ho condiviso parecchi momenti, felici e non.

Infine grazie a Stefano che, anche se non ci sentiamo tanto, che non ha mai fatto mancare il suo supporto e la sua vicinanza.

E poi alle M*rde studiose. Francesco e Mario sono state le persone più importanti in questi due anni in cui mi sono trasferito a Roma e rendendo migliore questa mia esperienza.

Ringrazio Francesco che fin dal primo momento in cui ci siamo conosciuti - dal primo "solo una birra dai" - ho capito di aver trovato una persona veramente importante, ormai come un fratello.

Grazie a Mario che con il suo altruismo e la sua gentilezza è stata una presenza importantissima in questi due anni, sia dentro che fuori l'università. In lui ho trovato una persona con cui non mi annoio mai e con cui riesco a farmi le migliori risate - soprattutto facendo impazzire Ciccio.

Grazie a Zioreus, che ormai dopo parecchi anni è come un fratello, con cui ho condiviso un sacco di momenti importanti nonostante la distanza.

Grazie a Camilla che in questi anni ha ricoperto un ruolo sempre più importante. Dai pomeriggi del lontano 2021 in isola C a studiare fino al mare a Gioiosa, è ormai una costante presenza nella mia vita. Andare al Marco Polo era un po' meno spiacevole quando passavamo i pomeriggi insieme - e con le lufe ovviamente.

Ringrazio tutti gli Ignegneri. Le gemelle, Francesco, Mario e Filippo, che nonostante sia arrivato tardi, mi hanno fatto sentire subito a mio agio, aiutandomi molto in questi anni. Hanno reso più godibile ogni aspetto dell'università, dallo studio agli aperitivi e le cene.

Un ringraziamento speciale all'Ordine di Pat. Un gruppo pieno di persone con cui ho condiviso molto in questi anni e senza cui questa esperienza sarebbe stata senza dubbio peggiore, abbiamo condiviso infinite sessioni di studio e di lezione, ma anche ore di svago e divertimento: soprattutto devo ricordare i 3 giorni in Umbria, hanno reso indimenticabile il mio compleanno.

Di loro una menzione speciale a Simone. Piano piano, sessione dopo sessione, abbiamo iniziato a frequentarci e conoscerci sempre di più. È stata una persona per me importantissima, sempre pronte ad aiutarmi quando mi trovavo in difficoltà, ma

anche con cui fare conversazione importanti o uscire per svagarci - peccato che non stiamo vicinissimi.

Un ringraziamento gigante a Mario Pintore: ci siamo conosciuti quasi subito, ma il vero punto di svolta è stato Fisica. Non smetterò mai di ringraziarti per tutto il tempo speso ad aiutarmi, soprattutto quando ero a zero e demoralizzato. Abbiamo passato quasi tutta un'estate in videochiama a studiare e a farci forza per andare avanti e riuscire a superare questo ultimo ostacolo, con i ghiaccioli di mezzogiorno e le battute sulla Formula Uno. Ma ce l'abbiamo fatta, e pure con lo stesso voto!