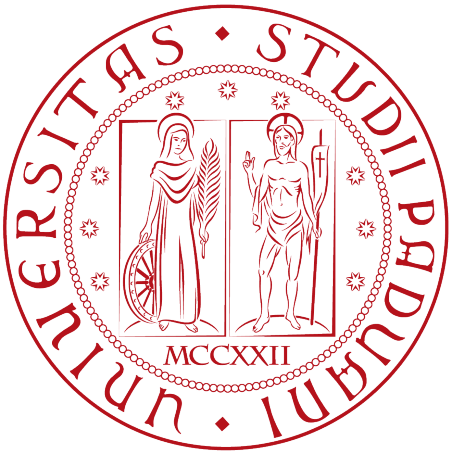
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica



Progettazione e sviluppo dell'applicazione per la gestione amministrativa di VIC SRL

Tesi di laurea triennale

Relatore
Dott. Silvia Crafa

Laureando
Luca Allegro

Anno Accademico 2017-2018

Be pitiful, for every man is fighting a hard battle. .

– Ian Maclaren , *The British Weekly*, 1897

Dedicato ad Alessio e nonno Lorenzo

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage dal laureando Luca Allegro presso l'azienda VIC s.r.l.
Il tirocinio si è tenuto presso il Dipartimento di Ricerca e Sviluppo (R&D) durante i mesi di luglio e agosto 2018 e ha avuto la durata di 310 ore.

L'obiettivo dello stage è stato lo sviluppo di un'applicazione web per la gestione amministrativa dell'azienda che comprende il tenere traccia dei movimenti bancari e la gestione delle fatture attive e delle fatture passive.

L'elaborato ha lo scopo di illustrare:

- il contesto aziendale dove è stato svolto lo stage (Capitolo 1);
- la descrizione del progetto sviluppato (Capitolo 2);
- le attività svolte durante esso (Capitolo 3);
- una valutazione finale sull'esperienza e le competenze acquisite(Capitolo 4).

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine alla Professoressa Silvia Crafa, relatrice della mia tesi, per la l'attenzione e l'aiuto fornitomi durante la stesura del lavoro.

Un sincero grazie a Lorenzo e Luca per avermi accompagnato in modo impeccabile nell'inserimento in azienda e aver reso lo stage un'esperienza di crescita.

Ringrazio i compagni di studio per aver reso indimenticabile questo percorso condividendo giornate (di studio e non), difficoltà e successi come una famiglia.

Ringrazio l'Associazione Italiana Arbitri per tutte le persone che mi ha permesso di conoscere e per le infinite occasioni di crescita e confronto che mi offre.

Ho desiderio di ringraziare poi i miei amici per tutti i bellissimi anni passati insieme e le mille avventure vissute.

Desidero ringraziare con affetto la mia famiglia per essermi stata vicina e per avermi dato fiducia durante tutti gli anni di studio.

Ringrazio miei genitori per gli insegnamenti di vita che ogni giorno mi danno, il miglior esempio da seguire.

Ringrazio tutte le altre persone a me care che conosco e ho conosciuto e che hanno contribuito, anche solo per un momento, a farmi diventare la persona che sono oggi.

Padova, Dicembre 2018 Luca Allegro

Indice

Elenco delle figure

Elenco delle tabelle

1 L'Azienda

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

1.1 Chi è VIC

VIC s.r.l.[1] offre servizi di ispezione delle merci in tutto il mondo per individuare eventuali non conformità delle merci al fine di tutelare i propri i clienti e ridurre al minimo le perdite.

VIC nasce a Venezia nel 2006 ed ora è una delle più grandi compagnie del mercato globale in quanto è presente con 22 diverse aziende e più di 50 laboratori in tutti i più grandi snodi di scambio del mondo.

La chiave del successo di VIC s.r.l. è l'attenzione che pone alla riduzione del tempo tra ispezione e reporting al cliente. Questo è stato possibile grazie all'automazione e alla semplificazione di molte attività svolte dai dipendenti mediante moderni sistemi informatici e allo sviluppo di diversi servizi web ed applicazioni mobile con l'obiettivo di fornire più informazioni possibili (inclusi foto e video) ai clienti.

Figura 1.1: Il logo di VIC

1.2 Il Dipartimento di Ricerca e Sviluppo

Il Dipartimento di Ricerca e Sviluppo (abbreviato: R&D) ha sede a Padova e si occupa di realizzare e manutenere tutti gli strumenti che l'azienda offre, ad esempio:

- il portale *VIC Online* per i clienti, mediante cui è possibile controllare lo stato dei propri ordini, ottenere informazioni utili, immagini e video riguardanti la merce ordinata;
- le applicazioni rivolte agli ispettori per raccogliere i dati sulla merce e realizzare dei resoconti sulle analisi effettuate e lo stato della merce.

Il dipartimento R&D é alla continua ricerca di metodi di innovazione per velocizzare, automatizzare e standardizzare le attività svolte dall'azienda nell'ambito del controllo qualità e quantità delle merci.

Tutti i servizi sviluppati dal dipartimento R&D sono composti dalle componenti front-end (client) e back-end (server). La comunicazione tra queste due componenti avviene utilizzando le quattro operazioni Create Read Update Delete (CRUD) di Hypertext Transfer Protocol (HTTP), in modo da essere conformi ai principi REST.

1.3 Opportunità di Stage

VIC s.r.l. è favorevole ad ospitare laureandi per lo stage perché crede che questa attività porti molteplici benefici all'azienda, nello specifico:

- il confronto con persone nuove e la loro esperienza può offrire spunti di miglioramento per i sistemi e le metodologie;
- per lo studio di nuove tecnologie e sul loro possibile utilizzo in progetti aziendali;
- la conoscenza di possibili candidati per l'assunzione.

2 Il Progetto

Descrizione del progetto di stage, motivazioni, obiettivi.

2.1 Aspettative personali

L'attività di stage è obbligatoria e prevista dal piano di studi del corso di laurea. Non avendo ancora nessuna esperienza lavorativa, ero molto emozionato all'idea di fare questo stage, per avere la possibilità di vedere una nuova realtà esterna al mondo universitario.
Al fine di effetturare la scelta dell'azienda in cui effetture lo stage in modo oculato ho deciso di partecipare all'iniziativa denominata Stage-IT, organizzato da Confindustria Padova in collaborazione con l'Università di Padova e Venezia.

Figura 2.1: Il logo dell'evento Stage-IT

Prima di andare all'evento, ho stilato una lista di possibili aziende a cui presentarmi, facendo soprattutto attenzione ai progetti proposti. Durante l'evento ho quindi colto l'occasione di discutere dell'attività di stage con molteplici aziende. Molte di tra queste erano più interessate a uno stage per inserimento lavorativo e non è stato possibile trovare un punto di incontro in quanto ero convinto di continuare gli studi dopo l'ottenimento della Laurea Triennale.

I miei obiettivi iniziali per lo svolgimento dello stage erano i seguenti:

- avere l'occasione di imparare una nuove tecnologie;
- integrarmi in un'azienda, apprendere le sue metodologie e capire eventuali difficoltà che si possono incontare in questo processo;

- vedere e capire il funzionamento di una realtà aziendale di medie dimensioni;
- entrare in contatto e confrontarmi con personale esperto e qualificato;
- valutare il mio livello di preparazione, dovuto in gran parte allo studio presso l'Università, rispetto al livello richiesto a ben figurare nel mondo del lavoro.

Considerando i miei obiettivi e la lista di aziende con le quali ho avuto un contatto a Stage-IT ho deciso di svolgere lo stage presso VIC s.r.l.

2.2 L'applicativo

L'azienda utilizza un software gestionale ricco di funzionalità, ma molto datato. Esso infatti è stato sviluppato interamente in PHP[2] ed è basato sul framework *QCubed*[3] che sta per concludere il suo ciclo di vita in quanto non è supportato più supportato e non è compatibile con le nuove versioni di PHP.

Lo scopo dello stage è stato quello di sviluppare un nuovo gestionale che si interfacci con lo stesso database del precedente e con le stesse funzionalità, ma basato su tecnologie moderne, prestando cura anche al rinnovamento dell'interfaccia grafica.

Nello specifico il software da realizzare rappresenta il Modulo Amministrativo: esso è uno dei moduli principali in cui è diviso il software di gestione in uso attualmente presso l'azienda ed si occupa di gestire tutte le fatture e tenere sotto controllo la gestione contabile.

Esso si divide in tre sottomoduli: Banche, Fatture Attive e Fatture Passive.

2.2.1 Banche

Il modulo banche si occupa di tenere sotto controllo la situazione contabile dei vari conti correnti aziendali.
Dovrà offrire le seguenti funzionalità:

- visualizzazione della lista delle banche associata alla sede corrente mostrandone per ognuna:
  - il saldo reale (inserito manualmente dagli operatori) di ogni banca;
  - la data di inserimento del saldo reale;
  - il saldo calcolato dal sistema in base ai movimenti bancari inseriti (che dovrebbe coincidere con il saldo reale, se aggiornato);
  - il fido di cassa, cioè la somma che la banca mette a disposizione dell' azienda;
  - il saldo disponibile, ottenuto sommando il saldo reale al, se presente, il fido di cassa;
- visione di tutti i movimenti di un conto corrente;
- aggiornamento del saldo di un conto corrente;
- caricamento e visione di estratti conti giornalieri e mensili;
- inserimento, modifica e rimozione di movimenti bancari;
- generazione di file nei formati *PDF* e *xls* della lista dei movimenti bancari.

2.2.2 Fatture attive

Il modulo delle fatture attive si occupa di tenere traccia di tutti gli ordini completati e in attesa di fatturazione, della creazione della fattura stessa con la possibilità di specificare i parametri richiesti (data fattura, scadenza, tipo fattura...), della possibilità di inviare le fatture al cliente e la gestione dello scadenziario delle fatture scadute e non ancora incassate.
Dovrà offrire le seguenti funzionalità:

- visualizzazione degli ordini completati e non ancora fatturati di cui è possibile creare la fattura;
- ricerca di fatture attraverso il numero di ordine o il nome della nave;
- creazione, modifica e rimozione di una fattura;
- visualizzazione della lista di fatture non ancora inviate al cliente;
- generazione di file in formato *PDF* corrispondenti alle fatture;
- invio di fatture al cliente tramite email;
- scadenziario fatture emesse da incassare
  - già scadute;
  - in scadenza.

2.2.3 Fatture passive

Il modulo delle fatture passive si occupa di tenere traccia di tutte le passività relative all'attività. Si dividono in due gruppi: dirette e indirette. Le fatture passive dirette rappresentano delle passività direttamente riconducibile ad un preciso ordine (fornitori, agenzie, laboratori, spedizionieri), mentre le indirette rappresentano dei costi generici di gestione (elettricità, bollette telefoniche, affitti,...). Le fatture passive indirette si suddividono a propria volta in due categorie: una tantum e cicliche.
Dovrà offrire le seguenti funzionalità:

- visualizzazione lista passive dirette non pagate raggruppate per fornitore;
- visualizzazione lista passive indirette (divise tra cicliche e una tantum);
- visualizzazione lista fatture per ogni fornitore;
- visione dettaglio singola fattura;
- modifica fattura;
- eliminazione fattura;
- generazione del movimento bancario associato (con riempimento automatico dei campi).

2.3 Obiettivi

Sulla base della durata massima di 320 ore prevista per lo stage, il tutor e lo stagista hanno stilato un piano di lavoro e coerentemente hanno concordato gli obiettivi minimi e massimi che si aspetta di veder raggiunti al termine del rapporto lavorativo.

Gli obiettivi minimi concordati sono:

1. comprensione del sotware e del database esistenti;
2. studio delle tecnologie;
3. studio degli strumenti di sviluppo;
4. definire, progettare, codificare e verificare le funzionalità che riguardano i moduli:
   - Banche;
   - Fatture Attive.

Gli obiettivi massimi concordati sono:

1. definire, progettare, codificare e verificare le funzionalità che riguardano il modulo:
   - Fatture Passive.

2.4 Vincoli

L'applicativo deve integrarsi con i restanti moduli del software gestionale quindi l'azienda ha imposto i seguenti vincoli:

- l'applicativo deve essere conforme ai principi REST;
- l'applicativo deve utilizzare il formato JavaScript Object Notation (JSON)[4] per lo scambio dei dati con il server;
- l'applicativo deve utilizzare JSON Web Tokens (JWT)[5] per l'autenticazione e gestione della sessione;
- la componente back-end dell'applicativo deve essere sviluppata in linguaggio Python[6] utilizzando il framework Django[7];
- la componente front-end dell'applicativo deve essere sviluppata utilizzando il framework Angular[8].

3 Svolgimento

Il presente capitolo descrive le varie fasi dello sviluppo dell'applicativo.

3.1 Pianificazione

3.1.1 Modello di sviluppo

Come modello per la gestione del progetto ho adottato la metodologia Agile Scrum. Tale modello mi ha permesso di poter rispondere con velocità e flessibilità alle esigenze e ai feedback degli stakeholder, in questo caso il tutor aziendale e il Dipartimento di Amministrazione.

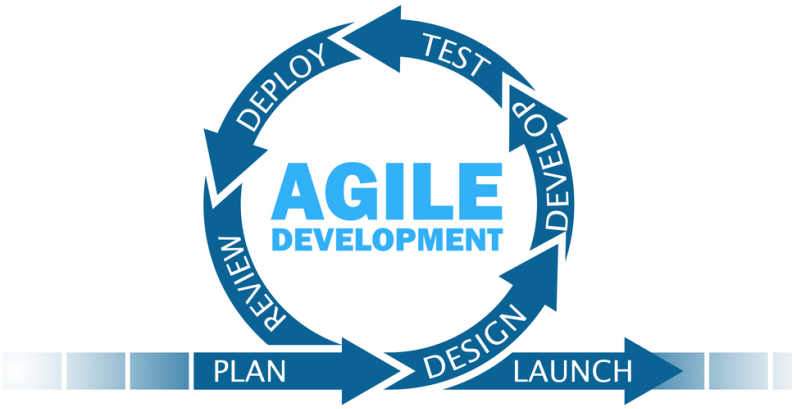


Figura 3.1: Metologia Agile

Il modello *Agile* prevede iterazioni continue, della durata massima di due settimane, entro le quali si susseguono attività di:

1. analisi dei requisiti che emergono dall'interazione con gli stakeholder;

2. pianificazione delle funzionalità da includere nello sprint e loro suddivisione in task;

3. progettazione delle funzionalità da implementare nell'iterazione corrente;

4. codifica e sviluppo delle funzionalità previste;

5. verifica;

6. rilascio;

7. monitoraggio continuo e verifica dello stato dell'applicazione.

Il punto di partenza di ogni iterazione è il risultato raggiunto con il precedente e il feedback ricevuto tramite la presentazione della soluzione raggiunta fin'ora agli stakeholder.

Al seguente indirizzo è possibile visionare i principi del modello di ciclo di sviluppo Agile:

http://agilemanifesto.org/iso/it/principles.html

3.1.2 Pianificazione settimanale

La duranta dello stage è stata di 8 settimane quindi ho pianificato 4 sprint della durata di due settimane ognuno in cui ho eseguito le seguenti operazioni:

1. studio delle tecnologie, degli strumenti di sviluppo e del sotware gestionale esistente;

2. progettazione, codifica e testing del modulo Banche;

3. progettazione, codifica e testing del modulo Fatture Attive;

4. progettazione, codifica e testing del modulo Fatture Passive.

3.2 Strumenti e Tecnologie Utilizzate

Per lo sviluppo del progetto ho utilizzato le seguenti tecnologie:

- Django e Django REST Framework;
- Angular;
- HTML 5, CSS3 e Bootstrap;
- Git;
- gli IDE Jetbrains.

3.2.1 Django e Django REST Framework

Django è un framework di alto livello, gratuito e open source per lo sviluppo in modo veloce e pulito di applicazioni web in Python.

Il framework è composto da:

- un Object Relational Mapping (ORM) che consente di mappare lo schema di un database relazionele su cui vengono eseguite le operazioni in modelli definiti in Django; inoltre consente la generazione dei modelli (classi in linguaggio Python che rappresentano le tabelle) a partire dallo schema del database e viceversa;
- un dispatcher URL basato sulle espressioni regolari in grado di effettuare il parsing degli indirizzi che vengono inseriti secondo delle specifiche anche complesse, cioè comprendendo parametri o espressioni regolari;
- un sistema di view per l'elaborazione delle richieste del client;
- un sistema di template per la visualizzazione dei contenuti.

Django REST Framework è un insieme di librerie che semplifica la creazione di API offrendo in particolare strumenti per la conversione di dati complessi, ad esempio i risutati delle interrogazioni al database, in tipi di dato nativi di Python che possono essere manipolati e facilmente renderizzati nel formato desiderato.

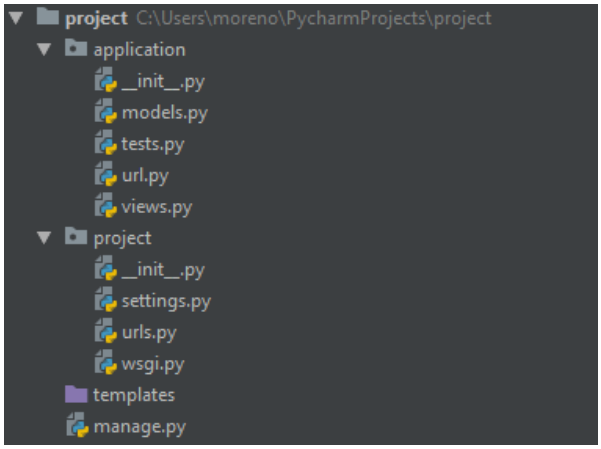


Figura 3.2: I file di un'applicazione Django

Come si può vedere in figura 3.2 il framework favorisce la sviluppo di applicazioni strutturate in quanto è raccomandato che ogni applicazione sia divisa nei seguenti moduli:

- **models:** il modulo che contiene le strutture dati che rappresentano le tabelle del database e metodi che rappresentano comportamenti essenziali dei dati che rappresentano.
- **serializers:** il modulo in cui si definiscono le classi che si occupano della conversione dei risultati delle interrogazioni al database in dizionari o liste pronte per la trasmissione e, viceversa, di creare le istanze da inserire nel database a partire da dizionari.

- **views:** il modulo che contiene le definizioni dei servizi del web-service, cioè funzioni che accettano come parametro una richiesta HTTP ed eventuali argomenti e ritornano una risposta HTTP.

- **urls:** il modulo in cui si definiscono le associazioni tra stringhe oppure espressioni regolari e i metodi contenuti in *views*.

- **tests:** il modulo che contiene la definizione dei test e ne consente l'esecuzione in modo automatico.

3.2.2 Angular

Angular è una piattaforma e un framework open-source per lo sviluppo di applicazioni client in HTML e TypeScript. Il framework si basa sui seguenti principi:

- **Mobile first**: Uno degli obiettivi del nuovo Angular è di proporsi per lo sviluppo di applicazioni Web sia per l'ambiente desktop che per il mobile. Infatti, Angular 2 supporta di default eventi touch e gesture;

- **Modularità**: l'architettura di Angular è altamente modulare e favorisce la scrittura di applicazioni modulari;

- **Prestazioni**: un'attenzione particolare è stata posta sulle prestazioni del framework, dalla riduzione dei tempi di caricamento e boostrapping dell'applicazione, al rilevamento efficiente delle modifiche nel modello dei dati ed alla velocità dei tempi di rendering.

Un'applicazione Angular è composta dai seguenti tipi di elementi:

- **Template**: file che contiene la sruttura di base della pagina HTML da mostrare;

- **Component**: classe gestore del template associato e di cui permette il cambiamento dinamicamente;

- **Services**: classi che forniscono funzionalità condivise da più componenti, ad esempio i metodi di interazione con il back-end, per favorire la riusabilità del codice ed evitare duplicazione;

- **Module**: classi che permettono il raggruppamento di componenti correlati logicamente fra loro; a differenza dei servizi ogni componente può far parte di uno e un solo modulo.

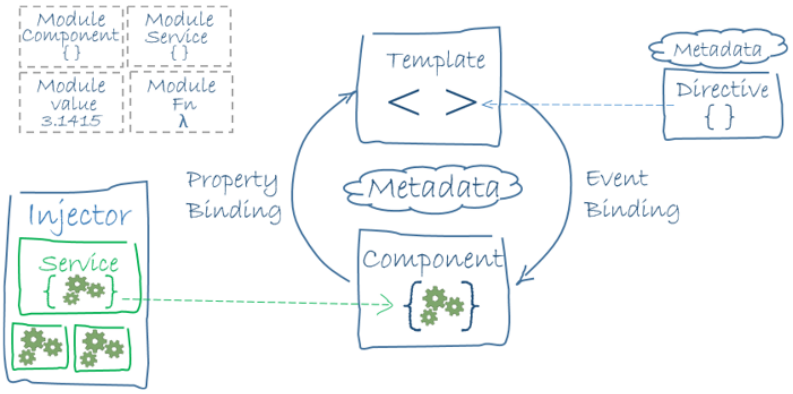


Figura 3.3: Architettura di Angular

3.2.3 HTML 5, CSS3 e Bootstrap

Per la realizzazione delle pagine web viene utilizzato il linguaggio di markup HyperText Markup Language (HTML)5 il quale è l'attuale standard di riferimento. Per facilitare la realizzazione del layout grafico delle pagine e la gestione di alcuni eventi ed animazioni viene utilizzato Cascading Style Sheets (CSS)3, abbinato alla vasta libreria offerta da Bootstrap[9].

3.2.4 Git

I progetto è soggetto a versionamento attraverso il software Git, un sistema di controllo di versione distribuito, creato da Linus Torvalds nel 2005.
Git supporta lo sviluppo non lineare con diramazioni e fusioni rapide e continue e comprende strumenti specifici per visualizzare e navigare una cronologia di sviluppo. Permette ad ogni sviluppatore una copia locale dell'intera cronologia di sviluppo e le modifiche vengono importate da un repository ad un altro.
Il relativo repository è ospitato su Bitbucket, un servizio di hosting web-based per progetti che usano i sistemi di controllo versione Mercurial o Git.

3.2.5 IntelliJ e derivati

Per l'attività di codifica dell'applicativo ho scelto di utilizzare PyCharm e WebStorm, due prodotti sviluppati da JetBrains come derivazioni di IntelliJ ottimizzate rispettivamento per progetti in Linguaggio Python e progetti Web.

Ho scelto di utilizzare questi IDE perchè offrono i seguenti vantaggi:

- permettono la configurazione del sistema di building in modo semplice;
- integrazione con il sistema di versionamento Git;
- contengono un sistema di versionamento integrato che tiene traccia di ogni modifica ad ogni singolo file e da cui è possibile ripristinare lo stato precedente;

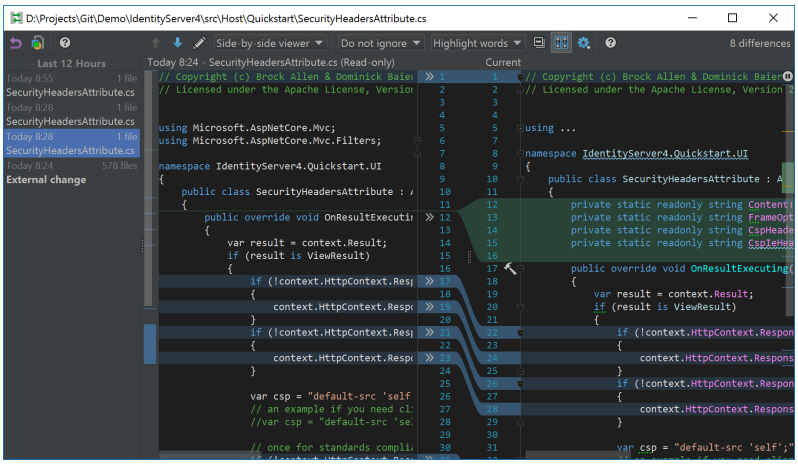


Figura 3.4: Il sistema di versionamento integrato dei prodotti JetBrains

- contengono un client ad ingerfaccia grafica per database;

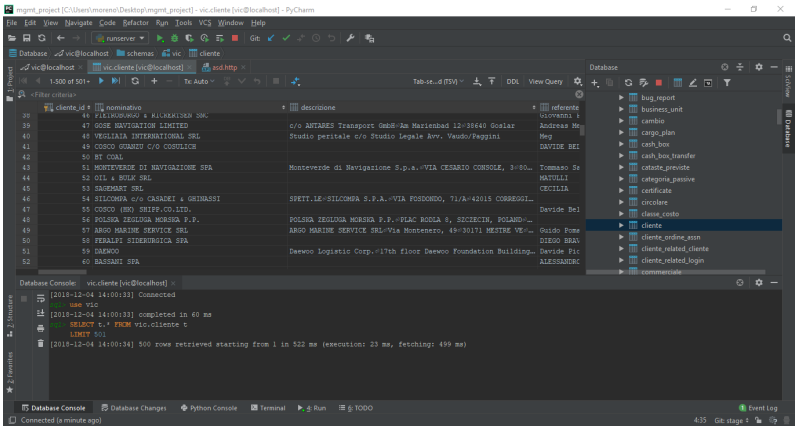


Figura 3.5: Il sistema di navigazione di database

- contengono un client per API REST con possibilità di effetturare test inserendo asserzioni sul risultato;

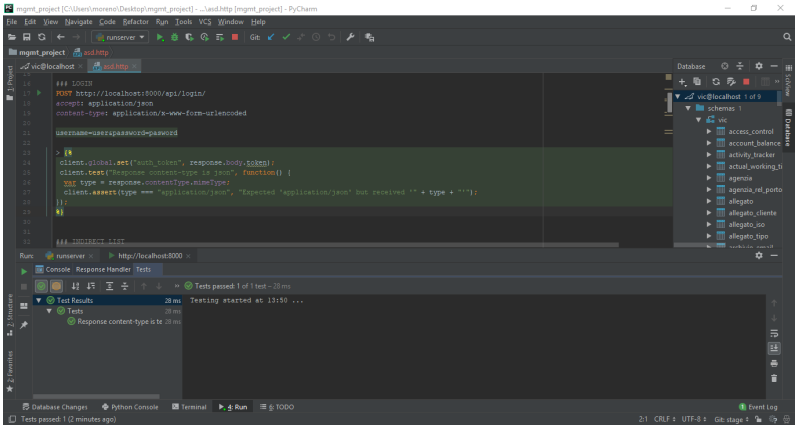


Figura 3.6: Il client per API REST

- integra uno strumento di analisi statica del codice con segnalazione degli errori.

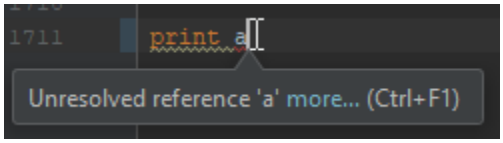


Figura 3.7: Esempio di errore segnalato dall'IDE

3.3 Progettazione

3.3.1 Progettazione Architetturale

Ho seguito il principio di progettazione *Separation of Concerns* (traduzione: separazione delle responsabilità) al fine di dividere l'applicativo in più parti logicamente separate e quindi più semplici.

Il sitema risultante, ad alto livello, è diviso in due componenti principali:

- **Front-end**: si occupa di gestire l'interazione con l'utente ed è responsabile dell'acquisizione dei dati di ingresso e della loro elaborazione in modo tale da renderli utilizzabili dal back end;

- **Back-end**: si occupa interagire con il database, elaborare i dati e mettere a disposizione i servizi al client (il front-end).

Le due componenti sono indipendenti tra loro e comunicano tramite API REST nelle quali vengono scambiati dati in formato JSON.

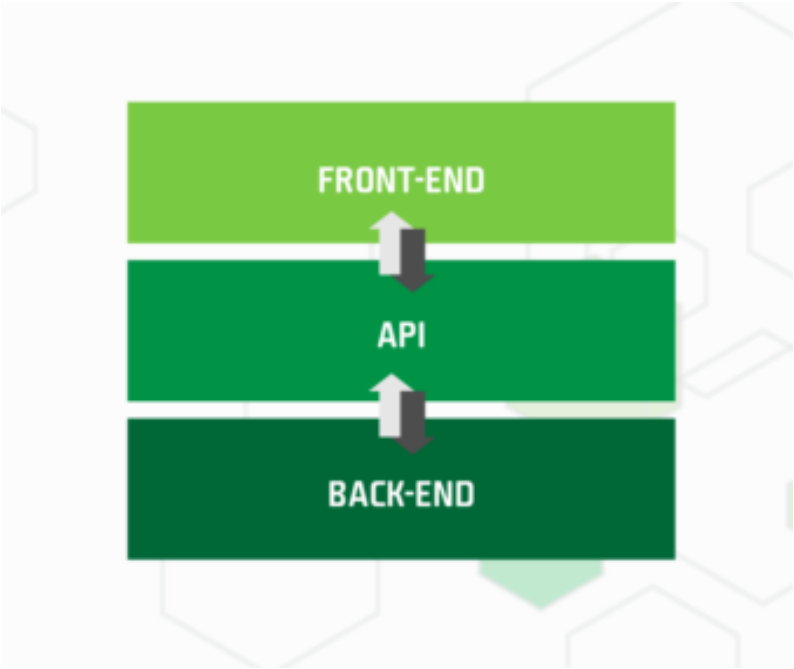


Figura 3.8: Architettura di alto livello

3.3.2 Back-end

Il back-end dell'applicativo è stato realizzato utilizzando i framework Django e Django REST Framework[10].
Esso è suddiviso in tre moduli che rispecchiano le tre sezioni dell'applicativo da realizzare:

- **banche**;
- **fatture_attive**;
- **fatture_passive**.

Ognuno di questi moduli contiene al suo interno tutti i sottomoduli descritti approfonditamente nella sezione 3.2.1 in modo da tenere separati oggetti non correlati e quindi facilitarne la manutenibilità e il riuso:

- models;
- serializers;
- views;
- urls;
- tests.

3.3.3 Front-end

L'utilizzo del framework Angular mi hanno da subito imposto una visione predefinita dell'architettura dell'applicazione.
Seguendo le direttive dettate dall'utilizzo del framework Angular, spiegate alla sezione 3.2.2, ho diviso l'applicativo in due *package*:

- **services** che si occupa di contenere tutti i servizi, cioè le classi che forniscono funzionalità condivise da più componenti, nello specifico i metodi di interazione con il back-end;
- **views** che contiene tutti i moduli, i componenti e i template ad essi associati per la per la realizzazione delle pagine.

Services

Questa parte contiene le classe che rappresentano i servizi che permettono di accedere alle API offerte dal corrispondente modulo del back-end:

- **BancheService**, che permette di accedere alle API offerte dal corrispondente modulo *banche*;
- **FattureAttiveService**, che permette di accedere alle API offerte dal corrispondente modulo *fatture_attive*;
- **FatturePassiveService**, che permette di accedere alle API offerte dal corrispondente modulo *fatture_passive*;

Views

Questa parte contiene la gerarchia dei moduli e dei componenti in essi contenuti.
Nello specifico, procedendo in ordine top-down, sono contenuti i seguenti moduli:

- **BancheModule**, che dipende dal servizio *BancheService* e contiene tutti i componenti necessari ad offrire all'utente le funzionalità facenti parte del Modulo Banche;
- **FattureAttiveModule**, che dipende dal servizio *FattureAttiveService* e contiene tutti i componenti necessari ad offrire all'utente le funzionalità facenti parte del Modulo Fatture Attive;
- **FatturePassiveModule**, che dipende dal servizio *FatturePassiveService* e contiene tutti i componenti necessari ad offrire all'utente le funzionalità facenti parte del Modulo Fatture Passive.

Al fine di evitare inutili dipendenze e mantenere una struttura del progetto semplice e coerente, è stato scelto che i template associati ai componenti:

- risiedano nella stessa cartella del loro componente associato;
- abbiano lo stesso nome del componente associato.

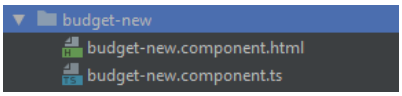


Figura 3.9: Esempio di cartella contenente un componente

3.3.4 Comunicazione tra front-end e back-end

La comunicazione tra il back-end e il front-end avviene tramite un insieme di API che il primo mette a disposizione del secondo seguendo i principi REST. Lo stile architetturale REST prescrive che lo stato in un'applicazione le sue funzionalità siano interpretati come risorse web univoche e accessibili tramite un Uniform Resource Locator (URL) attraverso il protocollo HTTP o il protocollo Simple Object Access Protocol (SOAP).

L'approccio architetturale REST è definito dai seguenti vincoli applicati ad un'architettura:

- **Client-server**: un insieme di interfacce uniformi separa il client dal server in modo da ridurre l'accoppiamento tra le componenti del sistema;
- **Stateless**: la comunicazione client–server è vincolata in modo che nessun contesto client venga memorizzato sul server tra le richieste. Ogni richiesta da ogni client contiene tutte le informazioni necessarie per richiedere il servizio, e lo stato della sessione è contenuto sul client;
- **Cacheable**: i client possono fare caching delle risposte;
- **Layered system**: i client non sono tenuti a conoscere quale livello del sistema server stanno interrogando.

Tale architettura è molto semplice, facile da implementare e riduce notevolmente l'accoppiamento tra client e server; per questi motivi è diffusa nelle applicazioni web.

3.4 Verifica e Validazione

3.4.1 Verifica

Durante l'intera attività di sviluppo ho verificato che quanto svolto fosse conforme alle attese del mio tutor e dei membri del Dipartimento di Amministrazione. Per far ciò ho attuato molte prove pratiche sulle funzionalità implementate, per assicurarmi che funzionassero in modo adeguato.

L'azienda ha preferito utilizzare le prove manuali piuttosto di dedicare tempo alla progettazione e codifica di un sitema automatizzato di verifica, in modo che potessi concentrarmi sull'adeguare il prodotto secondo le loro esigenze.
Grazie al lavoro svolto in fase di configurazione dell'ambiente di lavoro, ho potuto testare velocemente l'applicazione in locale con la base di dati contenente i dati reali.

Analisi Statica

Per l'attività di verifica durante tutto l'arco di sviluppo del progetto ho utilizzato strumenti di analisi integrati negli IDE in modo da controllare che il codice fosse uniforme, scritto secondo dei buoni canoni e che rispettasse opportuni standard di codifica in base al linguaggio, altrimenti segnalati con *warning*.

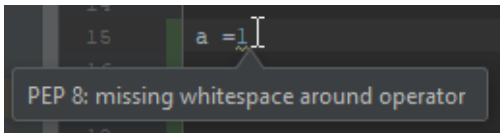


Figura 3.10: Esempio di warning dovuto al non rispetto delle norme di sintassi

In particolare tutto il codice scritto in linguaggio Python rispetta le convenzioni *PEP8*, reperibili al link:

https://www.python.org/dev/peps/pep-0008/

Invece il codice in linguaggio TypeScript è stato analizzando utilizzando il tool opensource TSLint, reperibile al link:

https://palantir.github.io/tslint/

3.4.2 Validazione

Il processo di validazione ha lo scopo di accertare che il prodotto finale corrisponda alle attese soddisfando tutti i requisiti prefissati inizialmente e va a dimostrare che il prodotto dello stage si comporti come specificato. Occorre provare che fornisce tutte le funzionalità, le prestazioni, l'affidabilità richieste e che non fallisca.

Test di accettazione

Segue un elenco dei test di accettazione effettuati durante il mio stage.

Tabella 3.1: Tabella dei test di Accettazione

Test	Descrizione	Esito
TA-1	Visualizzazione della lista delle banche associata alla sede	Superato
TA-2	Visione di tutti i movimenti di un conto corrente	Superato
TA-3	Aggiornamento del saldo di un conto corrente	Superato
TA-4	Caricamento di un estratto conto	Superato
TA-5	Visione di di un estratto conto	Superato
TA-6	Inserimento di un movimento bancario	Superato
TA-7	Modifica di movimento bancario	Superato
TA-8	Rimozione di movimento bancario	Superato
TA-9	Generazione del file *Pdf* della lista dei movimenti bancari	Superato
TA-10	Generazione del file *xls* della lista dei movimenti bancari	Superato
TA-11	Visualizzazione della lista degli ordini completati e non ancora fatturati	Superato
TA-12	Creazione di una fattura a partire dagli ordini completati	Superato
TA-13	Ricerca di fatture attraverso il numero di ordine	Superato
TA-14	Ricerca di fatture attraverso il nome della nave	Superato
TA-15	Creazione di una fattura attiva	Superato
TA-16	Modifica di una fattura attiva	Superato
TA-17	Rimozione di una fattura attiva	Superato
TA-18	Visualizzazione della lista di fatture non ancora inviate al cliente;	Superato
TA-19	Generazione di file in formato *Pdf* corrispondenti alle fatture;	Superato
TA-20	Invio di fatture al cliente tramite email	Superato
TA-21	Visualizzazione dello scadenziario delle fatture emesse da incassare	Superato
TA-22	Visualizzazione della lista delle fatture passive dirette non pagate raggruppate per fornitore	Superato
TA-23	Visualizzazione della lista delle passive indirette	Superato
TA-24	Visualizzazione del dettaglio singola fattura passiva	Superato
TA-25	Modifica di una singola fattura passiva	Superato
TA-26	Eliminazione di una fattura passiva	Superato
TA-27	Pagamento e generazione del movimento bancario associato a una fattura passiva	Superato

Validazione esterna

Al termine dello stage in presenza del tutor è stata effettuata una presentazione del lavoro svolto durante il periodo di stage e sono stati eseguiti tutti i test sopra elencati.

Tutti i test in sede di validazione esterna sono stati superati con successo e la presentazione è terminata con demo che dimostrava l'effettiva copertura di tutti i requisiti pianificati ed il soddisfacimento di tutte le funzionalità richieste.

4 Analisi retrospettiva

Considerazioni sulle conoscenze acquisite e sull'esperienza maturata durante lo stage

4.1 Competenze Acquisite

Dal punto di vista formativo l'attività di stage è stata estremamente positiva e ha rispettato le mie aspettative iniziali.

Ha arricchito il mio bagaglio personale di conscenze professionali importanti. Le competenze che posso dire di aver aggiunto o ampliato rispetto al bagaglio formativo in mio possesso prima dell'inizio dello stage sono molteplici e possono essere riassunte nei seguenti aspetti:

- esperienza lavorativa;
- tecnologie;
- lavoro individuale.

4.1.1 Esperienza Lavorativa

Non avendo esperienze lavorative, visto che in questi tre anni ho preferito concentrami sugli studi, credo sia questo sia l'aspetto in cui io mi sia maggiormete arricchito.

Questo ambito comprende:

- integrarmi in una nuova realtà aziendale e l'apprendere le sue metodologie;
- la collaborazione e il confronto con i colleghi;
- il rapporto con il datore di lavoro e con gli stakeholder, nel mio caso il tutor e i membri del Dipartimento di Amministrazione;
- il rispetto degli orari e delle scadenze.

Durante questa esperienza ho affrontato alcune dificoltà e capito l'impatto che esse hanno, tra cui:

- la volatilità dei requisiti;

- l'assenza di linee guida scritte e procedure a livello aziendale per la progettazione e la scrittura del codice;
- la non conoscenza del dominio dell'applicativo, nello specifico l'ambito amministrativo e riguardante la fatturazione;
- la carenza di documentazione e la conseguente perdita di tempo nello studiare il codice prodotto da terzi.

4.1.2 Tecnologie

Nonostante lo stage in questione abbia richiesto una considerevole mole di studio ed autoformazione, gli argomenti trattati e di ricerca sono risultati piacevoli, interessanti e utili.
Durante lo stage ho avuto modo di studiare in modo approfondito alcune tecnologie e di usarle in ambiente lavorativo permettendomi di comprenderle meglio, capirne pregi e difetti e imparare a farne un utilizzo più consapevole.

Nello specifico ho migliorato le mie conoscenze riguardanti i seguenti argomenti:

- **la metodologia Agile**: di cui ho apprezzato la flessibilità e leggerezza, ma che ha creato difficoltà in alcuni momenti a causa dell'insufficienza di documentazione;
- **lo stile architetturale REST**: grazie alla sua semplicità e alla sua versatilita l'architettura con back-end, front-end e la comunicazione tramite API REST viene utilizzata nella maggior parte delle applicazioni;
- il linguaggio **Python**: l'ho trovato molto intuitivo, semplice, conciso e molto ricco di librerie; per contro è un linguaggio a tipizzazione dinamica per cui è necessaria l'esecuzione del codice per rivelare la maggior parte degli errori provocando ingenti perdite di tempo per effettuare debugging e testing. A causa questo difetto credo che lungo periodo e in progetti di grandi dimensioni sia più conveniente utilizzare altri linguaggi, ad esempio Java;
- **Django**: l'ho trovato un ottimo framework in grado di rendere lo sviluppo di complesse applicazioni semplice e veloce;
- **Angular**: secondo la mia esperienza è ottimo per applicazioni complesse grazie all'architettura che impone, ma di difficile apprendimento e inutilmente prolisso quindi non adatto a sviluppatori poco esperti o progetti di piccole dimensioni;
- l'utilizzo di **IntelliJ**(e le sue derivate PyCharm e WebStorm): grazie all'elevato numero di strumenti integrati e alla possibilità di configurarlo secondo le proprie esigenze l'ho trovato estremante utile e il suo uso mi ha permesso di risparmiare molto tempo in sede di codifica e debug.

4.1.3 Lavoro individuale

Durante tutto lo stage ho dovuto lavorare da solo, visto che il progetto era stato appositamente calibrato in questa modalità. Questo mi ha permesso di raggiungere un grado di autonomia che fin'ora non possedevo.
Ho comunque avuto la possibilità di confrontarmi con personale qualificato in ogni

momento di difficoltà. Solitamente quando sono bloccato su un problema, preferisco cercare di risolverlo da solo, piuttosto che chiedere aiuto a qualcuno. Durante lo stage ho appreso che questa mia metodologia non è del tutto corretta: ho imparato a confrontarmi nei momenti di necessità per raggiungere in tempi più rapidi la soluzione. Questo insegnamento mi sarà sicuramente utile se in futuro lavorerò in un team di sviluppo.

4.2 Valutazione Personale

Nel complesso mi sento soddisfatto di questa esperienza che mi ha dato l'opportunità di entrare nel mondo del lavoro e confrontarmi con esso costituendo un'importante opportunità di crescita personale.
L'unico rammarico consiste nel fatto che la politica dell'azienda improntata principalmente al risultato non permetta di mettere molta attenzione alla qualità del software e alla documentazione, che sarebbe possibile attraverso l'investimento di più tempo nella formazione del personale e nell'adozione di linee guida e regole aziendali che normino tali attività.

Ritengo che l'università sia il luogo migliore per poter apprendere le basi di ciò che poi si dovrà applicare alla vita lavorativa.
Poter usufruire di un servizio universitario, che garantisce un'esperienza lavorativa all'interno del percorso di studi, permette a tutti gli studenti che entrano per la prima volta nel mondo del lavoro di conoscere già le dinamiche generali aziendali così da poter fin da subito adattarvisi.

Ritengo dunque il bilancio formativo davvero positivo e posso affermare che lo stage rappresenta una delle più importanti attività svolte nel mio personale percorso universitario.
Consiglio pertanto l'attività dello stage anche a tutti quelli studenti di altri corsi di studi che non la prevedono come obbligatoria, in quanto solo tramite il suo svolgimento è possibile dare il giusto valore alle nozioni apprese durante il corso di studi.

Glossario

Agile Insieme di principi per lo sviluppo software il cui obiettivo è soddisfare il cliente, rilasciando software in maniera continua e in periodi brevi, in modo che possa osservare l'andamento dello sviluppo e i risultati ottenuti. Di conseguenza è anche lecito accogliere eventuali cambiamenti dei requisiti durante l'intero ciclo di sviluppo . 7, 23

Api Insieme di servizi (procedure, funzioni, strutture dati) che un sistema espone in qualche modo ai propri utilizzatori. 9, 12, 13, 15, 16, 20, 23

Bootstrap Bootstrap è una libreria open source per lo sviluppo di pagine web responsive con HTML, CSS e JavaScript utile per velocizzare lo sviluppo della parte grafica delle pagine web . 11, 23

CRUD CRUD indica le quattro operazioni fondamentali effettuabili su dispositivi di memoriazzazione: creazione (Create), lettura (Read), aggiornamento (Update) e cancellazione (Delete) . 25

JSON Formato adatto all'interscambio di dati fra applicazioni client-server. . 25

JWT JWT è uno standard aperto basato su JSON per la creazione di token di accesso. . 25

PHP Linguaggio di scripting interpretato, con licenza open source e libera, originariamente concepito per la realizzazione di pagine web dinamiche. 4

REST Si tratta di un tipo di architettura software per i sistemi di ipertesto distribuiti come il World Wide Web. Un concetto importante in REST è l'esistenza di risorse (fonti di informazioni), a cui si può accedere tramite un identificatore globale (un URI). Per utilizzare le risorse, le componenti di una rete (componenti client e server) comunicano attraverso una interfaccia standard (ad es. HTTP) e si scambiano rappresentazioni di queste risorse. 2, 6, 12, 13, 16, 20

URL URL (Uniform Resource Locator) è una sequenza di caratteri che individua univocamente una risorsa web collegata alla rete internet . 25

Acronimi

CRUD Create Read Update Delete. 2

CSS Cascading Style Sheets. 11

HTML HyperText Markup Language. 11

HTTP Hypertext Transfer Protocol. 2, 10, 16

JSON JavaScript Object Notation. 6, 13

JWT JSON Web Tokens. 6

ORM Object Relational Mapping. 9

PHP PHP: Hypertext Preprocessor. 23

REST REpresentational State Transfer. 23

SOAP Simple Object Access Protocol. 16

URL Uniform Resource Locator. 16

Bibliografia

Siti web consultati

[1] *VIC s.r.l.* URL: http://vicworldwide.com/ (cit. a p. 1).

[2] *PHP*. URL: http://www.php.net/ (cit. a p. 4).

[3] *QCubed*. URL: http://qcubed.github.io/ (cit. a p. 4).

[4] *JSON*. URL: https://www.json.org/json-it.html (cit. a p. 6).

[5] *JSON Web Tokens*. URL: https://jwt.io/ (cit. a p. 6).

[6] *Python*. URL: https://www.python.org/ (cit. a p. 6).

[7] *Django*. URL: https://www.djangoproject.com/ (cit. a p. 6).

[8] *Angular*. URL: https://angular.io/ (cit. a p. 6).

[9] *Bootpstrap*. URL: https://getbootstrap.com/ (cit. a p. 11).

[10] *Django REST Framework*. URL: http://www.django-rest-framework.org/ (cit. a p. 14).