Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica

Advertising management system: Creazione e gestione di contenuti pubblicitari

Tesi di laurea

Relatore

Prof. Claudio Enrico Palazzi

Laureando

Alessandro Discalzi

Anno Accademico 2019-2020

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage dal laureando Alessandro Discalzi presso l'azienda SCAI ITEC. Lo stage è stato svolto al termine del percorso di studi della laurea triennale in informatica e la sua durata è stata di 312 ore. L'obbiettivo dello stage è stato di implementare un applicativo per la creazione e per la gestione di contenuti pubblicitari. Il presente documento vuole illustrare il contesto aziendale dove si è svolto lo stage, le attività svolte e una valutazione sul lavoro effettuato e su quanto appreso.

"Nessuno ha mai ottenuto nulla con le lacrime."

— Brucaliffo

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Prof. Claudio Enrico Palazzi, relatore della mia tesi, per l'aiuto e il sostegno che mi ha fornito durante la stesura del lavoro.

Desidero ringraziare con affetto la mia famiglia, e in particolare i miei genitori per il sostegno e per il grande aiuto che mi hanno dato durante gli anni di studio.

Desidero inoltre ringraziare il mio tutor aziendale, Dott. Bledar Gogaj, e il suo collega, Dott. Marco Lionello per l'enorme aiuto che mi hanno dato durante il periodo di stage.

Un ringraziamento infine, ai miei amici per tutti i bei momenti passati insieme e per avermi sopportato tutti questi anni.

Padova, Settembre 2020 Alessandro Discalzi

Indice

[1] **REST: acronimo di Representational State Transfer**.

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

1.1 L'azienda

SCAI ITEC[1] è un'azienda italiana appartenente al gruppo SCAI. ITEC si occupa di consulenza, System Integration ed Application management in ambito ICT. L'azienda opera principalmente in settore bancario, assicurativo, industriale e di pubblica amministrazione e servizi.
Gli elementi chiave del successo e della crescita di SCAI ITEC sono:

- vasta e profonda conoscenza delle tecnologie;
- grande attenzione per la soddisfazione del cliente;
- molta esperienza, maturata nel corso del tempo.

ITEC è oggi una delle maggiori realtà nel nord-est del paese in ambito ICT e si propone come partner per qualsiasi tipo di applicazione, progetto e servizio modellato sulle specifiche necessità del cliente.
Grazie alla consolidata esperienza nel ruolo di System Integrator[g] ed alle soluzioni leader di mercato proposte, ITEC è in grado di garantire ai propri clienti risposte rapide, concrete e qualificate in base alle specifiche esigenze di tipo gestionale e applicativo.
Ultimo, ma non meno importante, dei motivi per cui l'azienda è all'avanguardia rispetto le nuove tecnologie è il grande investimento di ITEC in ricerca, sviluppo e formazione del personale.

Figura 1.1: Logo SCAI ITEC

[1] **SCAI ITEC abbrev: ITEC**.

1.2 Scopo dello stage

L'obiettivo principale dello stage è stato quello di inserire lo studente all'interno di una nuova progettualità, con un particolare focus sulle tematiche legate alle tecnologie multimediali e alla loro distribuzione. Lo studente, affiancato da un IT Architect, ha avuto la possibilità di partecipare al disegno, alla progettazione e realizzazione di una nuova progettualità. L'obbiettivo è stato apprendere le tecnologie e le best practice utilizzate in azienda nel ciclo di vita di un applicativo. La progettualità vista è volta a creare un software per la gestione di contenuti informativi[g], per la loro creazione, modifica e distribuzione nei vari canali di vendita. Grazie a questa nuova applicazione si potrà:

- creare, modificare eliminare e clonare dei contenuti informativi;
- raggruppare i contenuti informativi su segmenti di mercato e distribuirli;
- pianificare l'esecuzione e l'aggiornamento dei contenuti informativi sui vari canali di distribuzione;
- seguire un processo di Authoring (paradigma Editore, Redattore, Supervisore) nella fase di creazione e distribuzione.

In quanto l'intero progetto è di dimensione non indifferente l'obbiettivo dello stage è stato di sviluppare le funzionalità relative al ruolo di Editore, più nello specifico:

- creazione di un contenuto;
- modifica di un contenuto;
- eliminazione di un contenuto;
- preparazione di un contenuto per la distribuzione;
- auditing delle azioni effettuate dagli utenti;
- documentazione delle funzionalità implementate.

1.3 Tecnologie utilizzate

1.3.1 JHipster

JHipster è una piattaforma di sviluppo, con uno stack tecnologico ben definito, utilizzata per generare, sviluppare e rilasciare, applicazioni e web services all'avanguardia. Supporta molteplici tecnologie per il frontend, tra le quali Angular, React e Vue. Fornisce inoltre supporto per le applicazioni per dispositivi mobili utilizzando Ionic e React Native. Per quanto riguarda il backend, JHipster supporta spring Boot (con l'ausilio di Java o Kotlin), Micronaut, Quarkus, NodeJS e .NET. Per il rilascio sono adottati i principi di Cloud nativo[g]. Il rilascio è inoltre supportato su AWS, Azure, Cloud Foundry, Google Cloud Platform, Heroku ed OpenShift.
L'obbiettivo di JHipster è generare applicazioni web o microservizi all'avanguardia, unendo:

- uno stack lato server robusto, ad alte prestazioni e coperto da test;

- un interfaccia utente accattivante, moderna e mobile-first usando Angular, React o Vue e Bootstrap per il CSS;
- un workflow ben definito per fare la build dell'applicazione con Maven o Gradle;
- un architettura a microservizi resiliente, utilizzando i principi di Cloud nativo;
- infrastruttura definita come codice, in modod da rendere la distribuzione su cloud veloce.

Nel corso dello stage JHipster è stato utilizzato per generare l'applicazione di base, utilizzata come punto di partenza per lo sviluppo.

Figura 1.2: Logo JHipster

1.3.2 Java Enterprise

Java Enterprise, conosciuto anche come Java EE è un insieme di specifiche che mirano ad estendere Java 8, aggiungendo funzionalità enterprise come elaborazione distribuita e servizi web. Le applicazioni Java EE possono essere eseguite sia come Microservizi[g] che su Application server[g]. In entrambi i casi vengono gestite: transazionalità, scalabilità, sicurezza e concorrenza. Nel corso dello stage Java EE è stato utilizzato per la programmazione lato backend.

Figura 1.3: Logo Java EE

1.3.3 Spring

Spring è un framework applicativo open source e un container per l'inversione di controllo[g] utilizzato dalla piattaforma Java. Le funzionalità di base possono essere

usate da una qualsiasi applicazione Java, mentre quelle più avanzate sono disponibili solamente per Java Enterprise. Il framework Spring ha a se associati vari moduli, nel corso del progetto sono stati utilizzati principalmente:

- Spring Boot: utilizzato per creare applicazioni basate su spring eseguibili senza la necessità di configurare un web server;
- Spring Data: il cui obbiettivo è di facilitare la gestione e l'interazione di applicazioni Java con un database.

Figura 1.4: Logo Spring

1.3.4 Hibernate

Hibernate è un framework per lo sviluppo di applicazioni in Java, utilizzato per gestire e mantenere su un database relazionale un insieme di oggetti Java.
Le sue funzionalità principali sono:

- mappare oggetti Java come tabelle su database;
- convertire i campi dati Java a quelli del DBMS[g] utilizzato;
- generare chiamate SQL e convertire la risposta ottenuta in un oggetto Java.

Tali funzionalità sono state largamente utilizzate nel corso dello stage.

Figura 1.5: Logo Hibernate

1.3.5 REST[2]

REST è un modello architetturale per i sistemi distribuiti. I sistemi rest si basano su HTTP e prevedono una struttura degli URL ben definita, che identifichi univocamente le risorse secondo la convenzione del modello stesso[3]. In REST per il trasferimento di dati vengono utilizzati i metodi HTTP, più nello specifico:

- GET: per il recupero di informazioni;
- POST, PUT, PATCH: per l'inserimento di informazioni;
- DELETE: per l'eliminazione di informazioni.

[2] **REST: acronimo di Representational State Transfer**.
[3] **Resource Naming: https://restfulapi.net/resource-naming/**.

I principi guida di REST sono:

- client-server: separazione dei problemi della UI da quelli di storage dei dati;
- statelessness: ogni richiesta deve avere tutte le informazioni necessarie per il suo processamento;
- cacheable: i client possono memorizzare in cache le risposte, queste devono essere definite esplicitamente o implicitamente cacheable, in modo da evitare il riutilizzo di dati errati;
- uniform interface: utilizzo di un'interfaccia di comunicazione omogenea tra client e server in modo da disaccoppiare e semplificare l'architettura per poterla modificare a blocchi;
- layered system:la struttura del sistema può essere composta da strati gerarchici. In questo caso ogni componente non può "vedere" oltre lo strato con cui sta interagendo;
- code on demand(opzionale): il codice lato client può essere esteso scaricando ed eseguendo applet o script.

Nella definizione di API se queste rispettano tutti i vincoli imposti dall'architettura REST allora possono essere definite RESTful.

Figura 1.6: Logo REST

1.3.6 Oracle

Oracle database è un DBMS di tipo relazionale prodotto da Oracle corporation.
I database oracle sono noti per offrire performance, scalabilità, affidabilità e sicurezza oltre a poter essere utilizzati sia on premise che nel cloud.

Figura 1.7: Logo Oracle

1.3.7 Liquibase

Liquibase è una libreria open source indipendente dal DBMS utilizzato. Durante il periodo di stage è stata utilizzata per tracciare, gestire e applicare le modifiche allo schema del database.

Figura 1.8: Logo Liquibase

1.3.8 Angular

Angular è un framework open source per la progettazione e lo sviluppo di applicazioni web. Le applicazioni angular vengono eseguite interamente a lato client ma grazie alla moltitudine di moduli presenti è possibile integrare un sistema di backend più complesso eseguito lato server. Nel corso dello stage, il frontend, è stato scritto interamente in Angular.

Figura 1.9: Logo Angular

1.4 Strumenti di sviluppo

1.4.1 Eclipse

L'IDE utilizzato per lo sviluppo, previo consiglio del tutor aziendale, è stato Eclipse. Eclipse è un ambiente di sviluppo integrato multipiattaforma e rientra nella categoria di software libero, distribuito secondo i termini della Eclipse Public License.

Figura 1.10: Logo Eclipse

1.4.2 Maven

Maven è uno strumento di gestione di progetti software, è basato su un Project Object Model (POM) e può gestire la build, il reporting e la documentazione di un progetto. Nel corso dello stage Maven è stato utilizzato principalmente per automatizzare la build del progetto, sia in sviluppo che in produzione.

Figura 1.11: Logo Maven

1.4.3 Git

Git è un sistema di controllo di versione distribuito, gratuito ed open source, progettato per gestire progetti di qualsiasi tipo. Nel corso dello stage l'utilizzo di Git è stato affiancato a quello di GitLab, una piattaforma web per la gestione di repository Git.

Figura 1.12: Logo Git

1.4.4 SQL Developer

SQL Developer è un ambiente di sviluppo integrato per lavorare con SQL nei database Oracle. Nel corso dello stage è stato utilizzato per gestire e testare il database Oracle usato in produzione.

Figura 1.13: Logo SQL Developer

1.5 Organizzazione del testo

Il secondo capitolo descrive gli obbietti dello stage, la pianificazione del lavoro effettuata a monte e le aspettative personali riguardanti lo stage;

Il terzo capitolo approfondisce la metodologia di lavoro e i ruoli adottati dal team di sviluppo;

Il quarto capitolo descrive l'analisi, la progettazione iniziale, e le soluzioni addottate durante la codifica;

Il quinto capitolo descrive le funzionalità del prodotto finale, la documentazione e i test eseguiti;

Il sesto capitolo contiene le considerazioni finali riguardanti lo stage e una valutazione personale sul lavoro svolto;

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

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

Capitolo 2

Obbiettivi e pianificazione

In questo capitolo sono descritti gli obbiettivi dello stage, la pianificazione del lavoro e le aspettative personali.

2.1 Obbiettivi

L'obbiettivo di questo stage è la realizzazione di una Proof of Concept dell'applicazione, in cui vengono rese disponibili le funzionalità descritte in §1.2. Per aumentare la produttività e permettere allo studente di conoscere nuove tecnologie viene utilizzato JHipster. Lo studente verrà inserito in un gruppo di lavoro composto da 4 persone in modo da favorire, oltre alla comprensione delle nuove tecnologie, la capacità di lavorare in un team. Al termine del periodo di stage verrà effettuata una presentazione del prodotto alla direzione dell'azienda.
Nel piano di lavoro, documento la cui stesura è avvenuta prima dell'inizio dello stage, sono stati individuati i seguenti obbiettivi suddivisi in obbligatori, desiderabili e opzionali.

2.1.1 Obbiettivi obbligatori

- Ob1: Conoscenza del framework Spring e in particolare Spring MVC REST;
- Ob2: Interazione e gestione database Oracle;
- Ob3: Realizzazione delle funzionalità backend del progetto.

2.1.2 Obbiettivi desiderabili

- D1: Realizzazione delle funzionalità frontend del progetto;
- D2: Conoscenza base sviluppo applicazioni frontend Angular;
- D3: Grado di autonomia nel processo di analisi/sviluppo.

2.1.3 Obbiettivi opzionali

- Op1: Conoscenza base dei strumenti per CI/CD.

2.2 Pianificazione

Lo stage prevede una durata di 312 ore complessive corrispondenti a 8 ore di lavoro giornalierio per un periodo di circa 8 settimane. L'orario di lavoro è dal Lunedì al Venerdì, dalle ore 9:00 alle 13:00 e dalle ore 14:00 alle 18:00. L'ora tra le 13:00 e le 14:00 è dedicata alla pausa pranzo.
La pianificazione redatta per il periodo di stage, divisa per obbiettivi settimanali, è la seguente:

- **prima settimana:**
  - studio strumenti di sviluppo (Eclipse, Maven, Git);
  - analisi dei requisiti.
- **seconda settimana:**
  - creazione struttura del database;
  - gestione database Oracle.
- **terza settimana:**
  - studio e utilizzo del framework Spring;
  - studio e utilizzo di Hibernate;
  - realizzazione dell'object-relational mapping.
- **quarta settimana:**
  - utilizzo spring MVC e Jackson;
  - realizzazione dei servizi REST.
- **quinta settimana:**
  - studio e utilizzo di elementi avanzati di Oracle;
  - gestione changeset Liquibase;
  - sviluppo di altri servizi REST.
- **sesta settimana:**
  - studio di Angular;
  - sviluppo frontend;
- **settima settimana:**
  - continuos integration e continuos delivery;
  - utilizzo di Sonarqube;
  - controllo qualità del codice.
- **ottava settimana:**
  - gestione cache dell'applicazione;
  - ottimizzazione;

2.3 Aspettative personali

Le mie aspettative per quanto riguarda lo stage erano molteplici.
Prima tra tutte la possibilità di lavorare in un'azienda che produce software in modo da poter capire, almeno in parte, come funziona la vita in azienda. In secondo luogo il mio obbiettivo era quello di imparare nuove tecnologie e le best practice adottate dall'azienda ospitante, anche grazie alla stretta collaborazione con il mio tutor. Ultima cosa ma non meno importante è la possibilità di farsi conoscere da un azienda leader nel settore, in modo da poter pensare ad eventuali collaborazioni future.

Capitolo 3

Metodologia di sviluppo

In questo capitolo viene descritta la metodologia di lavoro e i ruoli adottati dal team di sviluppo.

3.1 Scrum

Scrum è un framework agile per lo sviluppo, consegna e manutenzione di prodotti software e non. Scrum è progettato per l'utilizzo in team di dimensione ridotta. Di seguito vengono descritti ruoli, fasi e artefatti del framework.

3.1.1 Ruoli

Scrum master

Lo Scrum Master aiuta il team di sviluppo ad apprendere e applicare Scrum per conseguire valore di business. Lo Scrum Master fa tutto ciò che è in suo potere per aiutare il Team, il Product Owner e l'organizzazione ad avere successo. Lo ScrumMaster non è il manager dei membri del Team, né è un project manager, team leader, o rappresentante del team. Lo scopo dello Scrum Master è:

- aiutare a rimuovere gli ostacoli durante lo sviluppo;
- evitare interferenze esterne;
- aiutare il Team ad adottare al meglio le pratiche di sviluppo agile;
- fare in modo che tutti applichino Scrum nel miglior modo possibile.

Product Owner

Il Product Owner ha la responsabilità di massimizzare il ritorno sugli investimenti (ROI), di identificare le caratteristiche del prodotto, traducendole in una lista di priorità, di decidere cosa dovrebbe andare in cima alla lista per il prossimo Sprint, e di riassegnare le priorità, aggiornandole con continuità. Il Product owner detiene la responsabilità di profitto del prodotto, se questo è commerciale. In Agile il Product owner rappresenta il cliente e nell'applicazione di Scrum può e deve:

- definire il Product Backlog, le user stories e gli acceptance criteria;

- definire le priorità nel Product Backlog e la data di rilascio del prodotto;
- accettare o rifiutare quanto sviluppato;
- cancellare lo sprint se risulta fallimentare o poco utile.

Team di sviluppo

Il team di sviluppo è composto da un insieme di persone, in genere meno di 10, e si occupa di sviluppare quanto definito dal product owner. Il team Scrum deve essere "cross-funzionale", ovvero includere tutte le competenze necessarie allo sviluppo del prodotto. I membri del team devono essere proattivi e aperti allo studio di tecnologie che vanno oltre le loro competenze. Il team di sviluppo:

- costruisce il prodotto definito dal Product owner;
- possiede tutte le conoscenze per ottenere un prodotto potenzialmente rilasciabile alla fine di ogni sprint;
- è auto organizzato, con un alto grado di autonomia e responsabilità;
- decide quanti e quali elementi del Product backlog sviluppare;
- ha la responsabilità di sviluppo, test e rilascio del prodotto;
- non possiede un team leader, in quanto in Scrum nel team di sviluppo sono considerati tutti di pari livello.

Nel corso dello stage i ruoli erano così suddivisi:

- Scrum master: Bledar Gogaj
- Team di sviluppo: Alessandro Discalzi, Tania Parolin
- Product owner: Marco Lionello

3.1.2 Artefatti

Product backlog

Il Product backlog è un elenco di funzionalità, centrate sul cliente e ordinato per priorità; esiste e si evolve per tutta la durata del prodotto. Il Product backlog definisce quindi tutto ciò che deve essere fatto ed include una moltitudine di voci, più nello specifico:

- nuove funzionalità da implementare;
- obbiettivi di miglioramento;
- lavori di ricerca;
- difetti da risolvere, se in numero contenuto;

Un buon Product backlog è **DEEP**:

- **dettagliato**, con maggior attenzione alle voci di priorità più alta;

- **stimato**: per ogni voce deve esserci una stima per il completamento, la stima viene definita dal team di sviluppo;
- **emergente**: il Product backlog viene aggiornato in base alla variabilità del progetto;
- **prioritizzato**: le voci sono ordinate per priorità, quelle con priorità più alta forniscono maggior valore al prodotto.

Nel corso del progetto di stage per gestire il Product Backlog è stato utilizzato Taiga, un tool per la gestione di progetti agile.

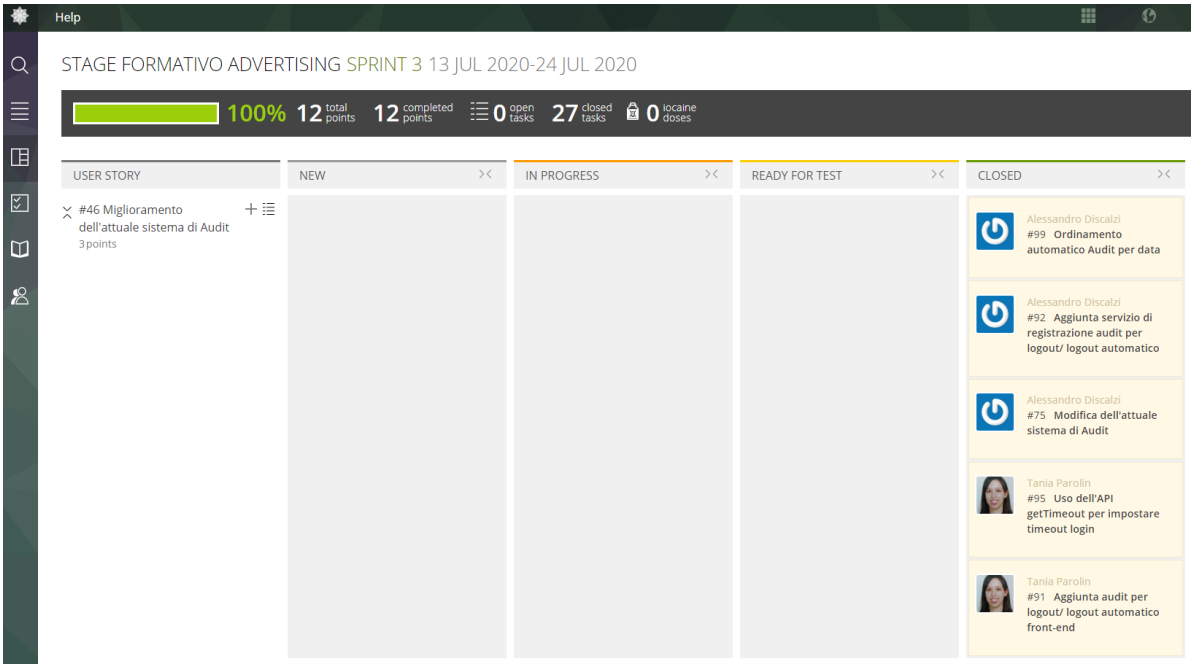

Figura 3.1: Screenshot del backlog presente su Taiga rispetto uno degli sprint terminati

Nel Product backlog di AMS la lista delle attività eseguite, in ordine di priorità, è la seguente:

- design e realizzazione della base di dati;
- design e realizzazione del template di base per i contenuti pubblicitari;
- realizzazione della funzione di preview di un contenuto;
- gestione autenticazione e autorizzazione in base ai ruoli;
- realizzazione dei servizi di back-end;
- design e implementazione della UI[g];
- integrazione front-end e back-end;
- implementazione della creazione di nuovi contenuti;

- implementazione della modifica di un contenuto;
- miglioramento UX[g] e UI;
- inizio stesura documentazione;
- implementazione sistema di auditing;
- miglioramento dei contenuti di esempio;
- predisposizione di oracle;
- implementazione breadcrumbs;
- aggiunta controlli di validazione;
- completamento della documentazione;
- controlli sull'accessibilità del software;
- miglioramento pipeline di CI[g];
- predisposizione demo.

Definition of done

Durante ogni sprint ciò che viene fatto costituisce un prodotto potenzialmente rilasciabile, questo deve essere approvato dal Product owner, dallo Scrum master e dal Team di sviluppo prima dell'inizio dello sprint successivo. La definition of done è un insieme di regole che definiscono quando ciò che viene fatto può essere definito rilasciabile. La definition of done adottata durante lo stage è la seguente:

- AMS (Advertising Management System) deve essere stato compilato senza warning;
- il software deve essere stato deployato nell'ambiente locale senza l'introduzione di nuovi errori e/o warning;
- devono essere stati eseguiti i test sulle nuove funzionalità;
- il codice deve essere stato revisionato dallo Scrum Master/Product Owner;
- la feature implementata deve essere accettata dal Product Owner.

Nei casi in cui questa definizione non fosse applicabile (ad esempio nel caso di modifiche che richiedono più voci/user stories/sprint) avrebbe dovuto essere evidenziata la violazione del Definition of Done allo Scrum Master ed al Product Owner ritardando la chiusura del Done a quando fosse stato effettivamente possibile.

3.1.3 Fasi

Sprint

Uno sprint è un periodo di tempo ben definito, di solito due settimane o un mese, durante il quale il team di sviluppo completa una parte di lavoro in base a quanto definito nel Product backlog. Gli sprint hanno durata fissa che non può essere estesa, tuttavia se lo sprint risulta fallimentare o obsoleto può essere cancellato prima del termine dal Product owner.
Nel corso del progetto didattico la durata degli sprint è stata fissata a due settimane, mentre la durata e gli argomenti discussi durante le riunioni descritte successivamente hanno seguito le regole di Scrum.

Sprint planning

Lo sprint planning è un incontro che viene effettuato prima di ogni sprint, la cui durata è limitata a due ore per ogni settimana di sprint. L'obbiettivo dello sprint planning è definire cosa rilasciare al termine del prossimo sprint e come farlo. Il lavoro da fare viene selezionato dal product backlog e inserito nello sprint anche in base alle stime di completamento definite dal team di sviluppo.

Daily scrum

Il Daily Scrum è una delle pratiche chiave di Scrum. Si tratta di un meeting giornaliero, della durata massima di 15 minuti, a cui partecipa il team di sviluppo, lo Scrum master e ,se richiesto, il Product owner. Il Daily scrum serve a sincronizzare il team e durante questo ogni membro deve rispondere a tre domande:

- cosa è stato fatto dall'ultima riunione?
- cosa sarà fatto prima della prossima riunione?
- quali difficoltà si sono incontrate?

Nel caso alcuni ostacoli necessitino di discussioni approfondite queste possono essere fatte al termine del Daily scrum.

Sprint review

La sprint review si tiene alla fine di uno sprint, in modo da ispezzionare gli incrementi e aggiornare, se necessario, il Product backlog di conseguenza. Durante la sprint review il team di lavoro e gli stakeholders collaborano per vedere, in base a quanto fatto durante lo sprint, quali sono le prossime cose che si potrebbero fare per aumentare il valore del prodotto. La sprint review dura massimo un'ora per ogni settimana di sprint, durante la riunione si svolgono le seguenti attività:

- il Product owner spiega quali attività del backlog sono state fatte e quali no;
- il team di sviluppo discute di cosa è andato bene, di cosa è andato storto e di come si sono affrontati i problemi durante lo sprint;
- il Team di sviluppo mostra il lavoro fatto e risponde ad eventuali domande riguardo l'incremento;
- se necessario il Product owner decide le date di rilascio in base a quanto fatto;

- il gruppo di lavoro collabora per decidere cosa fare prossimamente, questo serve anche come input allo sprint planning;
- review del potenziale di mercato del prodotto, se il prodotto è commerciale.

Sprint retrospective

La sprint retrospective ha luogo dopo la Sprint review e prima dello Sprint planning e la sua durata è limitata a 45 minuti per ogni settimana di sprint. Durante la retrospettiva, che viene vista come una possibilità di miglioramento per il team, si discute di cosa è andato bene, di cosa è andato storto e di come si può migliorare per il prossimo sprint. Questo favorisce il team in quanto si possono migliorare le metodologie adottate nello sprint precedente in base a ciò che è andato storto.

Capitolo 4

Analisi e progettazione

Il seguente capitolo descrive l'analisi, la progettazione iniziale, e le soluzioni addottate durante la codifica

4.1 Analisi e progettazione iniziale

Le funzionalità da implementare e gli obbiettivi da raggiungere erano già ben definiti all'inizio del progetto, in quanto l'analisi funzionale è stata fornita dal Product owner insieme ad una prima definizione del product backlog. Per questo motivo la fase di analisi dei requisiti da parte del team di sviluppo è stata molto breve e si è concentrata principalmente su come le funzionalità del prodotto dovessero essere fruibili all'utente finale. La progettazione del software non è avvenuta solamente all'inizio, ma si è protratta durante tutto il periodo di stage. Questo ha garantito maggior flessibilità nell'implementazione delle funzionalità e ha favorito l'utilizzo di Scrum. Tale scelta tuttavia ha anche introdotto il rischio che una debole progettazzione iniziale potesse causare problemi nelle settimane a venire. Questo rischio è stato subito mitigato grazie ad un attenta progettazione della base di dati pensando, nel miglior modo possibile, alle entità, a come esse fossero in relazione tra di loro e alla logica necessaria per implementare le funzionalità richieste. Inoltre, in quanto JHipster utilizza uno stack tecnologico ben definito, e impone dei vincoli architetturali da seguire, tale rischio è ulteriormente ridotto poiché non è necessario progettare l'architettura del prodotto. Nel corso di questo capitolo vengono descritte le principali attività di progettazione svolte durante il corso dello stage.

4.2 Progettazione del database

La progettazione del database è stata una delle attività più importanti e deteneva un alto grado di priorità anche nel Product backlog. La progettazione del database è stata divisa in due parti: la prima mirata a individuare le entità partendo dall'analisi dei requisiti fatta dal Product owner, mentre la seconda mettendo in relazione tali entità aggiungendo tabelle se necessario. Per quanto riguarda la vera e propria implementazione del database è stato sfruttato un tool fornito da JHipster ovvero JDLStudio, che verrà descritto in modo dettagliato successivamente nel corso di questa sezione.

4.2.1 Entità

Durante la progettazione della base di dati sono state definite le entità descritte di seguito. Per ciascuna di esse viene riportata una breve descrizione e la lista dei suoi campi dati corredata da tipo di dato, nella lista seguente non vengono riportate chiavi primarie ne chiavi esterne poiché l'inserimento di tali campi dati è automatizzato e verrà descritto dettagliatamente nella sezione relativa alle relazioni tra entità. Mentre si stava effettuando la progettazione del database è sorto il problema di come si volessero salvare i contenuti informativi e i template. Si è optato di suddividere i contenuti in pagine, ognuna delle quali può contenere uno o più elementi multimediali e/o testuali. Inoltre in uno stesso contenuto si può selezionare un template diverso per ogni pagina. Per quanto riguarda i template, i quali sono definiti a monte dal team di sviluppo, si è scelto di strutturarli definendo un template al cui interno sono contenuti vari item che possono essere sia testuali che multimediali. Questo rende tutto il più modulare possibile, in quanto non pone limiti né in fase di creazione di un contenuto, né nella definizione dei template e nel loro utilizzo.

Content

Definisce un contenuto informativo, i suoi campi dati sono:

- name (String): Nome del contenuto;
- state (State): Enumerazione che definisce i possibili stati di un contenuto;
- description (String): descrizione del contenuto informativo;
- slideTime (Integer): durata delle slide di un contenuto;

ContentPage

Definisce una pagina inserita in un contenuto. Ogni contenuto può avere più pagine che scorreranno come slide durante la visualizzazione dello stesso; ogni pagina contiene degli item multimediali o testuali. I suoi campi dati sono:

- pageNumber (Integer): numero della pagina all'interno di un contenuto.

MultimediaItem

Definisce un item multimediale (immagine o video) contenuto in una pagina. I suoi campi dati sono:

- path (String): percorso del file multimediale caricato sul server, nel caso il caricamento delle immagini avvenga su NAS;
- resolution (Resolution): enumerazione che definisce la risoluzione dell'immagine caricata;
- multimediaFile (Blob): file multimediale caricato su database, nel caso il caricamento non avvenga su NAS;

TextItem

Definisce un testo contenuto in una pagina. I suoi campi dati sono:

- text (Blob): contenuto testuale inserito in una pagina.

Template

Definisce un template che verrà utilizzato dalle pagine di un contenuto. Ogni pagina può utilizzare template definiti a monte che determinano come immagini, video e contenuti testuali vengono disposti nella pagina al momento della presentazione. I suoi campi dati sono:

- previewImage (Blob): immagine di preview del template, utilizzata per aiutare l'utente a capire come questo è definito;

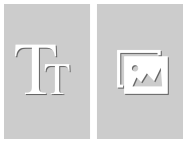

Figura 4.1: Immagine di preview di un template contenente un testo e un'immagine

- templateHTML (Blob): contiene l'html del template;
- name (String): Nome del template.

TemplateItem

Definisce un item all'interno di un template. Per item si intende un singolo testo, immagine o video. Questa entità viene utilizzata in fase di creazione e di visualizzazione di un contenuto ed è utilizzata per sapere cosa va inserito in una pagina che utilizza un determinato template. I suoi campi dati sono:

- description (String): descrizione dell'item del template (e.g. video a schermo intero);
- type (itemType): enumerazione che definisce il contenuto che viene inserito in quell'area del template;
- TemplateArea (Integer): area del template in cui un certo oggetto va inserito, utilizzato in fase di preview per inserire immagini, video e testi al posto giusto all'interno del template.

CSS

Definisce il CSS utilizzato da uno o più template. I suoi campi dati sono:

- templateCSS (Blob): contiene il CSS utilizzato da uno o più template.

CustomAudit

Entità definita per implementare il sistema di Audit. I suoi campi dati sono:

- action (Action): enumerazione che definisce il tipo di azione registrata negli audit;
- description (String): descrizione dettagliata dell'azione effettuata;
- timestamp (Instant): data e ora in cui l'azione registrata è stata eseguita.

User

Entità predefinita in JHipster contenente tutte le informazioni necessarie per registrare un utente. Per utilizzarla è sufficiente definire relazioni ad essa.

4.2.2 Enumerazioni

Durante la progettazione della base di dati, in alcuni casi, si è ritenuto necessario definire alcuni campi dati come enumerazione al posto di utilizzare una semplice stringa di testo. Questo garantisce maggior controllo e impone vincoli che evitano il verificarsi di errori relativi all'utilizzo di nomi diversi per esprimere lo stesso concetto. Le enumerazioni definite durante la progettazione del database sono le seguenti.

State

Lista dei possibili stati di un contenuto informativo, ovvero: *CREATED, TO_ASSOCIATE, ASSOCIATED, TO_AUTHORIZE, AUTHORIZED, PUBLISHED.*

Resolution

Lista delle possibili risoluzioni di un elemento multimediale, ovvero: *HIGH, MEDIUM, LOW.*

ItemType

Lista delle diverse tipologie di template item, ovvero: *IMAGE, VIDEO, TEXT.*

Action

Lista delle azioni da registrare nel sistema di auditing che possono essere effettuate da un utente, ovvero: *CREATE, DELETE, UPDATE, AUTHENTICATION_SUCCESS, AUTHENTICATION_FAILURE, LOGOUT, TIMEOUT.*

4.2.3 Relazioni

Di seguito vengono evidenziate le relazioni tra le entità descritte in precedenza e ne viene data una breve descrizione.

Content-ContentPage

Ogni Content può avere una o più ContentPage, ogni ContentPage è inserita in un solo contenuto.

Template-ContentPage

Ogni Template può essere utilizzato da una o più ContentPage, ogni ContentPage utilizza un solo Template.

Content-page-MultimediaItem

Ogni ContentPage può contenere uno o più MultimediaItem, ogni MultimediaItem è contenuto in una sola pagina.

ContentPage-TextItem

Ogni ContentPage può contenere uno o più TextItem, ogni TextItem è contenuto in una sola pagina.

Template-TemplateItem

Ogni Template ha al suo interno uno o più TemplateItem, ogni TemplateItem fa riferimento a un solo Template.

MultimediaItem-TemplateItem

Ogni MultimediaItem può riferirsi ad un solo TemplateItem, ad ogni TemplateItem possono essere riferiti uno o più MultimediaItem.

TextItem-TemplateItem

Ogni TextItem può riferirsi ad un solo TemplateItem, ad ogni TemplateItem possono essere riferiti uno o più TextItem.

Content-User

Ogni Content è creato da un solo User, ogni User può creare uno o più content.

CSS-Template

Ogni Template può avere uno o più CSS, ogni CSS può essere utilizzato da uno o più Template.

CustomAudit-User

Ogni Audit si riferisce ad un solo User, ad ogni User possono essere registrati uno o più Audit.

4.2.4 Implementazione

JDL Studio

JDL Studio è un tool fornito da JHipster che permette di definire lo schema entità-relazioni di un database tramite un apposito linguaggio di scripting chiamato, appunto, JDL. Una delle funzionalità più utili di JDLStudio è la possibilità di visualizzare il modello ER, che viene automaticamente aggiornato ad ogni modifica.

Definizione di un'entità in JDLStudio

Un'entità viene definita in JDL specificando il suo nome e i suoi campi dati. Un esempio di definizione di un'entità è il seguente:

Listing 4.1: Definizione entità Content

```
entity Content {
  name String unique,
  state State,
  description String,
  slideTime Integer
}
```

Definizione di un'enumerazione in JDLStudio

Un'enumerazione viene definita in JDL specificandone il nome e i suoi possibili valori dati. Un esempio di definizione di un'entità è il seguente:

Listing 4.2: Definizione enumerazione State

```
enum State {
  CREATED,
  TO_ASSOCIATE,
  ASSOCIATED,
  TO_AUTHORIZE,
  AUTHORIZED,
  PUBLISHED
}
```

Definizione di una relazione in JDLStudio

Per definire una relazione in JDL bisogna innanzitutto specificare il tipo di relazione (uno a uno, uno a molti, molti a molti). Fatto questo è sufficiente specificare l'entità di partenza della relazione e quella di arrivo. Un esempio di definizione di relazioni in JDL è il seguente:

Listing 4.3: Definizione relazioni uno a molti

```
relationship OneToMany {
  Content{page} to ContentPage{content},
  Template{page} to ContentPage{template},
  ContentPage{multimediaItem} to MultimediaItem{page},
  TemplateItem{multimediaItem} to MultimediaItem{templateItem},
  Template{templateItem} to TemplateItem{template},
  ContentPage{textItem} to TextItem{page},
  TemplateItem{textItem} to TextItem{templateItem}
}
```

Modello ER

Il modello ER autogenerato presenta imperfezioni nella visualizzazione delle relazioni, nonostante ciò e stato ritenuto sufficiente per gli scopi del progetto. Una volta creato il

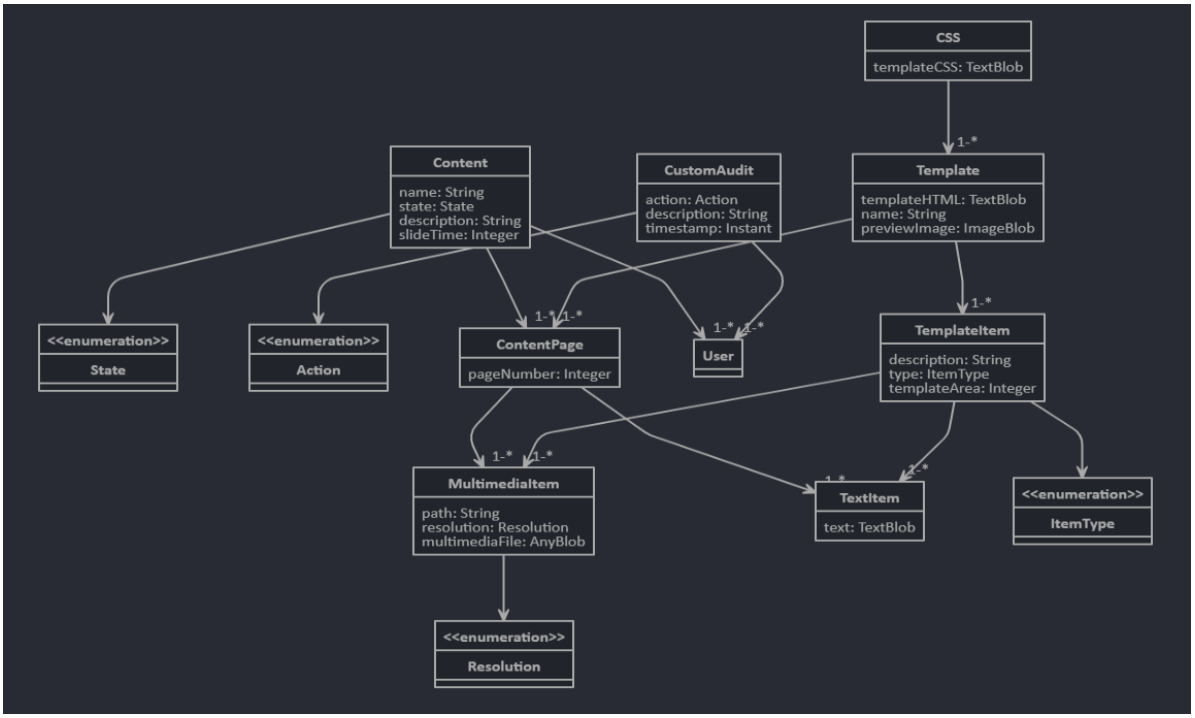

Figura 4.2: Modello ER generato da JDLStudio

modello del database tramite JDL Studio è possibile importarlo direttamente all'interno dell'applicativo. Tale procedura oltre a generare il database creerà anche una struttura di base dell'applicazione completa di back-end e front-end. Seppur questa non risulta utilizzabile evita la scrittura di numerose linee di codice al programmatore riducendo l'introduzione di errori umani.

4.3 Definizione dei template per i contenuti

In quanto nella progettazione del database è stato scelto di modellare i template in modo che fossero più modulari ed estendibili possibile, è stata necessaria un accurata definizione degli stessi. Si è optato per definire un template di base utilizzato da ogni contenuto informativo, e una serie di template personalizzati utilizzati dalle pagine inserite in un contenuto.

4.3.1 Template di base

Il template di base definito contiene poche informazioni. Definisce infatti la struttura base della pagina html e la funzione per visualizzare le slide. Contiene inoltre vari

placeholder utilizzati per inserire dinamicamente le informazioni di un determinato contenuto informativo. I placeholder definiti sono:

- ${CSS}: Utilizzato per inserire il css dei template utilizzati;
- ${CONTENTID}: Utilizzato per inserire l'id del contenuto come id nel div più esterno;
- ${PAGES}: Utilizzato per inserire le pagine presenti in un contenuto;
- ${SLIDETIME}: Utilizzato per impostare lo slidetime scelto in un contenuto;

4.3.2 Template di pagina

I template di pagina sono quelli scelti dall'utente durantela creazione di un contenuto. Anche in questo caso i template, scritti in html, avranno al loro interno opportuni placeholder per inserire i dati al momento dell'utilizzo. Poiché non si può sapere a priori come saranno definiti i template futuri sono state definite delle regole da seguire durante la creazione di un nuovo template. La lista delle regole definite è la seguente:

- definire un nome del template univoco;
- anteporre il nome del template ad ogni classe o id utilizzati per il CSS;
- non definire css per gli elementi html, utilizzare solamente classi e id;
- definire un tag * ... * differente per ogni item del template;
- nel caso si volessero inserire video o immagini all'interno di un nuovo template seguire gli esempi forniti insieme alla documentazione;
- definire i placeholder per l'inserimento di video, immagini o elementi testuali nel formato *${ItemTypeTemplateArea}*;

4.4 Definizione dei servizi di backend

Durante l'importazione del modello del database JHipster genera automaticamente alcuni servizi RESTful per ciascuna entità. Più nello specifico:

- creazione di un record;
- eliminazione di un record;
- modifica di un record;
- ricerca di un record per ID;
- ricerca di tutti i record presenti;

I servizi autogenerati tuttavia non erano sufficienti, alcuni di essi hanno avuto la necessità di essere modificati mentre per altre funzionalità ne sono stati implementati di nuovi. I servizi con la necessità di essere adattati erano, in genere, quelli di modifica. Questi sono stati adattati in modo che gestissero le relazioni correttamente. Sono stati inoltre modificati i servizi di eliminazione, in modo che rispettassero i vincoli di integrità referenziale I serviti definiti ad hoc sono invece i seguenti:

- servizio di update dello stato di un contenuto;
- servizio di creazione della preview di un contenuto;
- servizio di upload di un file multimediale;
- servizio di ricerca di un file multimediale per ID e che ritorna il file in Base64;

Capitolo 5

Prodotto

Introduzione

Capitolo 6

Considerazioni finali

Introduzione

Bibliografia