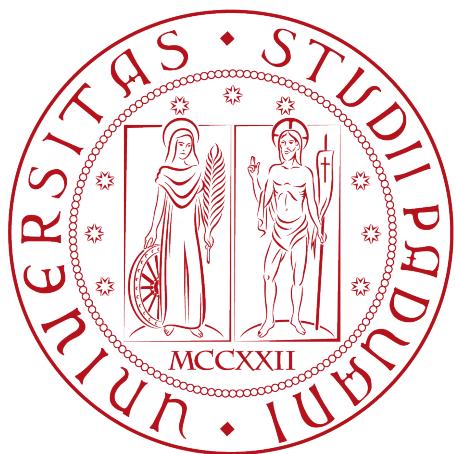


Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA

CORSO DI LAUREA IN INFORMATICA



## Tres - Tree recommendation system

*Tesi di laurea triennale*

*Relatore*

Prof. Tullio Vardanega

*Laureando*

Alberto Andeliero

---

ANNO ACCADEMICO 2015-2016

Alberto Andeliero : *Tres - Tree recommendation system*, Tesi di laurea triennale, ©  
Jan 2016.

# Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di circa trecento ore, dal laureando Alberto Andeliero presso l'azienda Nextep S.r.l. a Carmignano Di Brenta(PD).



# Indice

<b>1 L’azienda</b>	<b>1</b>
1.1 Presentazione azienda . . . . .	1
1.1.1 L’azienda . . . . .	1
1.1.2 Prodotti e servizi . . . . .	2
1.1.3 Tecnologie di riferimento . . . . .	3
1.2 Processi aziendali . . . . .	5
1.2.1 Metodologia agile . . . . .	5
1.3 Strumenti a supporto dei processi . . . . .	7
1.3.1 Gestione di progetto . . . . .	7
1.3.2 Documentazione . . . . .	8
1.3.3 Sistema di versionamento . . . . .	8
1.3.4 Ambiente di sviluppo . . . . .	8
1.3.5 Sistemi operativi . . . . .	9
1.4 Clientela tipo e propensione all’innovazione . . . . .	9
<b>2 Il progetto nella strategia aziendale</b>	<b>11</b>
2.1 L’azienda e gli stage . . . . .	11
2.2 Il progetto . . . . .	12
2.2.1 Il progetto . . . . .	12
2.2.2 Obiettivi . . . . .	12
2.3 Vincoli . . . . .	13
2.3.1 Vincoli temporali . . . . .	13
2.3.2 Vincoli metodologici . . . . .	13
2.3.3 Vincoli tecnologici . . . . .	15
2.4 Obiettivi personali . . . . .	17
<b>3 Il progetto di stage</b>	<b>19</b>
3.1 Organizzazione . . . . .	19
3.1.1 Pianificazione . . . . .	20
3.2 Analisi dei requisiti . . . . .	20
3.2.1 Requisiti . . . . .	20
3.2.2 Casi d’uso . . . . .	22
3.3 Progettazione architetturale . . . . .	25
3.3.1 Visione ad alto livello . . . . .	25
3.3.2 Architettura . . . . .	26
3.3.3 Database . . . . .	29
3.3.4 Design pattern . . . . .	30
3.4 Progettazione di dettaglio . . . . .	32

3.4.1	Controllo stato . . . . .	32
3.4.2	Inserimento behavior . . . . .	32
3.4.3	Raccomandazione . . . . .	33
3.5	Verifica e Validazione . . . . .	34
3.5.1	Resoconto risultati . . . . .	35
<b>4</b>	<b>Valutazione Retrospettiva</b>	<b>37</b>
4.1	Bilancio sui risultati . . . . .	37
4.1.1	Obiettivi conseguiti . . . . .	37
4.1.2	Obiettivi personali . . . . .	38
4.2	Bilancio formativo . . . . .	38
4.3	Distanza tra formazione universitaria e lavoro . . . . .	38
<b>A</b>	<b>Appendice A</b>	<b>39</b>
	<b>Bibliografia</b>	<b>41</b>

# Elenco delle figure

1.1	Logo di Nextep srl . . . . .	1
1.2	Sito Associazione Maestri Sci Italiani: realizzato da Nextep. Immagine tratta da <a href="#">Nextep</a> . . . . .	2
1.3	Classificazione delle tecnologie utilizzate . . . . .	3
1.4	Diagramma della metodologia di sviluppo <i>agile</i> . Immagine tratta da <a href="#">GlobalTeckz</a> . . . . .	5
1.5	Diagramma del modello <i>Scrum</i> . Immagine tratta da <a href="#">Wikipedia</a> . . . . .	7
1.6	Piattaforma di gestione di progetto Jira . . . . .	8
1.7	Classificazione degli strumenti utilizzati . . . . .	9
2.1	Esempio di un albero decisionale . . . . .	12
2.2	Pannello di controllo travis . . . . .	14
2.3	Tipica architettura di un'applicazione che utilizza Play Framework. Immagine tratta da <a href="#">Omer Haderi Blog</a> . . . . .	15
2.4	Esempio di modellazione in OrientDb. Immagine tratta da <a href="#">Scalac Blog</a>	17
3.1	Suddivisione delle attività . . . . .	19
3.2	Distribuzione requisiti in percentuale per tipologia . . . . .	21
3.3	Distribuzione requisiti in percentuale per importanza . . . . .	22
3.4	UC0: Tres . . . . .	23
3.5	UC1: Tres . . . . .	24
3.6	Visione generale dell'architettura . . . . .	25
3.7	Diagramma del <i>package tres</i> . . . . .	26
3.8	Diagramma del <i>package controllers</i> . . . . .	27
3.9	Diagramma del <i>package models</i> . . . . .	27
3.10	Diagramma del <i>package commons</i> . . . . .	28
3.11	Diagramma del <i>package storage</i> . . . . .	28
3.12	Diagramma del <i>package algorithm</i> . . . . .	29
3.13	Modello a grafo del database . . . . .	29
3.14	Contesto di utilizzo del <i>design pattern MVC</i> . . . . .	30
3.15	Contesto di utilizzo del <i>design pattern DAO</i> . . . . .	30
3.16	Contesto di utilizzo del <i>design pattern strategy</i> . . . . .	31
3.17	Contesto di utilizzo del <i>design pattern singleton</i> . . . . .	32
3.18	Diagramma di sequenza: controllo stato. . . . .	32
3.19	Diagramma di sequenza: inserimento comportamento. . . . .	33
3.20	Diagramma di sequenza: raccomandazione. . . . .	33
4.1	Riassunto requisiti . . . . .	37

## Elenco delle tabelle

3.4 Tabella dei <i>test</i> di unità . . . . .	36
3.5 Tabella dei <i>test</i> di integrazione . . . . .	36

# Capitolo 1

## L’azienda

*Questo capitolo tratta dettagliatamente della azienda ospitante, il suo business, l’organizzazione interna e l’innovazione.*

### 1.1 Presentazione azienda

Questa sezione descrive l’azienda che ha ospitato lo *stage*, elencando generalità e la struttura generale dell’azienda. Inoltre si illustrano alcuni prodotti e i servizi offerti da Nextep srl alla propria clientela.

#### 1.1.1 L’azienda

L’azienda con cui ho svolto lo stage formativo è Nextep srl<sup>1</sup>, la cui sede operativa si trova a Carmignano Di Brenta(PD). Nextep è una società fondata nel 2000 che opera nel settore dell’informatica e della comunicazione. Il *core business* dell’azienda è riconosciuto nelle attività di progettazione, realizzazione e gestione di infrastrutture di *Information and Digital Communication Technology*. L’organico di Nextep srl è formato da una ventina di persone, suddivise tra *web marketing* e tecnici(sviluppatori, grafici e sistemisti). L’ambiente di lavoro è condiviso con altre due aziende: Allos e Zero12. Allos si occupa della formazione del capitale umano di organizzazioni medio grandi mentre Zero12 si occupa di servizi *cloud* e applicazioni *mobile*. Il risultato è un ambiente di *co-working*, dove le competenze del singolo vengono messe a disposizione di tutti, in modo da facilitare la crescita professionale dei dipendenti delle varie aziende. Non è raro che le tre aziende collaborano tra di loro per la realizzazione di alcuni progetti.



**figura 1.1:** Logo di Nextep srl

---

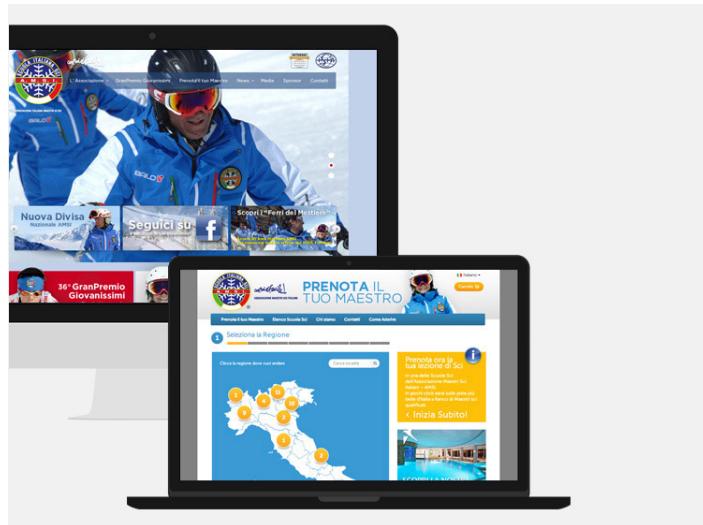
<sup>1</sup><http://www.nextep.it/>.

### 1.1.2 Prodotti e servizi

Il principale settore di attività dell’azienda è il *Web*. Nextep è specialista nella progettazione e sviluppo di siti *web*, portali, *social intranet* e soluzioni di *e-commerce*. L’azienda fornisce sistemi per il *knowledge management*, che sono dei sistemi per raccogliere, sviluppare, conservare e rendere accessibile la conoscenza delle persone che fanno parte di una organizzazione. Inoltre si occupa di analisi, progettazione e realizzazione di applicazioni mobile. L’azienda pone il suo focus nella realizzazione di sistemi con interfacce innovative e soprattutto accessibili all’utente grazie alle competenze acquisite nel ramo della *User Interface* e *User Experience*. Realizza sistemi per l’integrazione con tecnologie e servizi di *Cloud Computing*.

I principali servizi offerti da Nextep srl ai propri clienti sono:

- Fornitura di servizi a canone (*server virtuali*, applicazioni *web*, servizi di *monitoring* di infrastrutture *Information and Communications Technology (ICT)*, *backup-online*, canoni di assistenza e manutenzione dei sistemi, canoni di gestione dei sistemi);
- Elaborazione dati, analisi e supporto decisionale in ambito *web marketing*, *social marketing* e *social intranet*;
- Servizio Clienti, attività di supporto tecnico e di gestione di infrastrutture *ICT* e siti *web*.



**figura 1.2:** Sito Associazione Maestri Sci Italiani: realizzato da Nextep. Immagine tratta da [Nextep](#)

In particolare l’azienda fornisce servizi per migliorare l’efficacia delle strategie di comunicazione *web* dedicando particolare attenzione alla reputazione e all’identità digitale. Collabora, inoltre, assieme alle aziende nella gestione delle informazioni digitali, della sicurezza e della disponibilità dei dati e delle applicazioni. Nel ramo del *marketing*, Nextep realizza azioni di *web marketing* e fornisce consulenza *web marketing*, *social marketing* e *web analytics*.

### 1.1.3 Tecnologie di riferimento

Essendo il *web* un settore soggetto a continue evoluzioni, le tecnologie utilizzate cambiano col tempo. Nextep cerca di aggiornare le tecnologie utilizzate in modo da fornire ai propri clienti prodotti sempre migliori e innovativi.

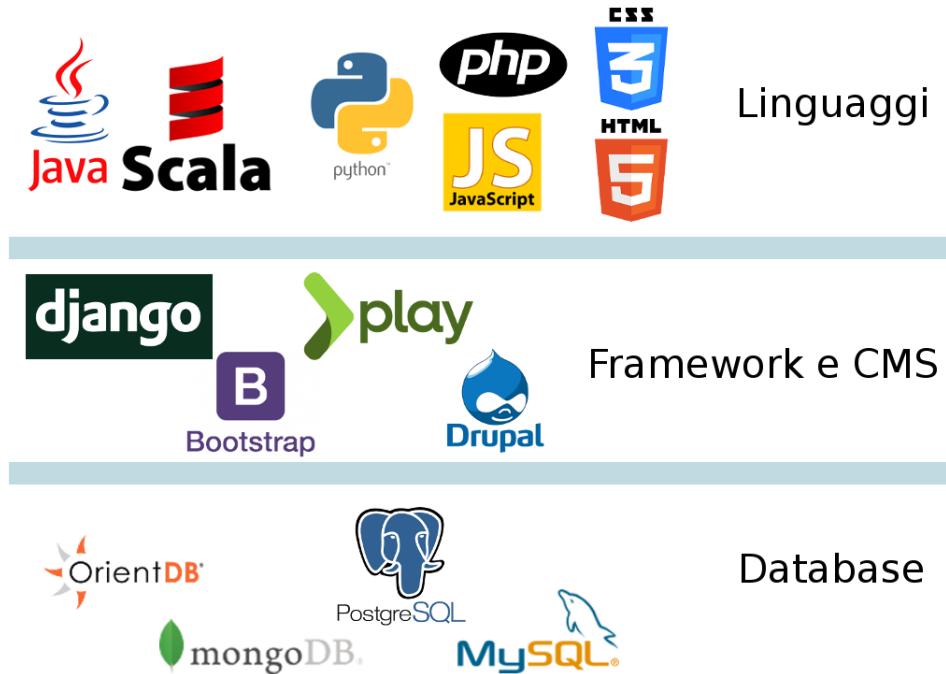


figura 1.3: Classificazione delle tecnologie utilizzate

#### Linguaggi di programmazione

L’azienda lavora principalmente in ambito *web*, quindi utilizza i linguaggi di programmazione più moderni per lo sviluppo dei prodotti. I linguaggi di programmazione più utilizzati in ordine di importanza sono Php, Python, Java e Scala. Per quanto riguarda le interfacce grafiche naturalmente utilizza tecnologie quali Html5, Css3 e Javascript.

#### Framework e CMS

**Django** è un *web framework open source* per lo sviluppo di applicazioni *web*, scritto in linguaggio Python, seguendo il *pattern model-view-controller*. Fornisce nativamente delle applicazioni per la gestione dei contenuti e degli accessi.

**Play Framework** è un *web framework open source* scritto in Java e Scala, che implementa il *pattern model-view-controller*. Il suo scopo è di migliorare la produttività degli sviluppatori usando il paradigma *Convention Over Configuration*.

**Bootstrap** è *framework* per la creazione di siti e applicazioni per il *web*. Essa contiene modelli di progettazione basati su Html e Css, sia per la tipografia, che per le varie componenti dell'interfaccia come moduli, button e navigazione, e altri componenti dell'interfaccia.

**Drupal** è una piattaforma *software* di *Content Management System (CMS)*, modulare, scritta in linguaggio Php e distribuita sotto licenza GNU GPL. Grazie alla sua architettura modulare permette il riutilizzo sistematico del codice scritto e quindi un vantaggio in termini di tempo e manutenzione.

### Database

Il *team* di sviluppo utilizza per la persistenza sia *database* relazionali sia non-relazionali. Fra i *database* relazionali impiega per la maggiore le tecnologie quali MySql, Sql Server e PostgreSql. Per quanto riguarda i *database* non-relazionali utilizza soluzioni come MongoDb oppure OrientDb. A seconda del contesto vengono studiati i pro e contro, in modo da poter trarre tutti i vantaggi possibili.

## 1.2 Processi aziendali

Questa sezione illustra l'organizzazione interna dell'azienda, descritta per processi.

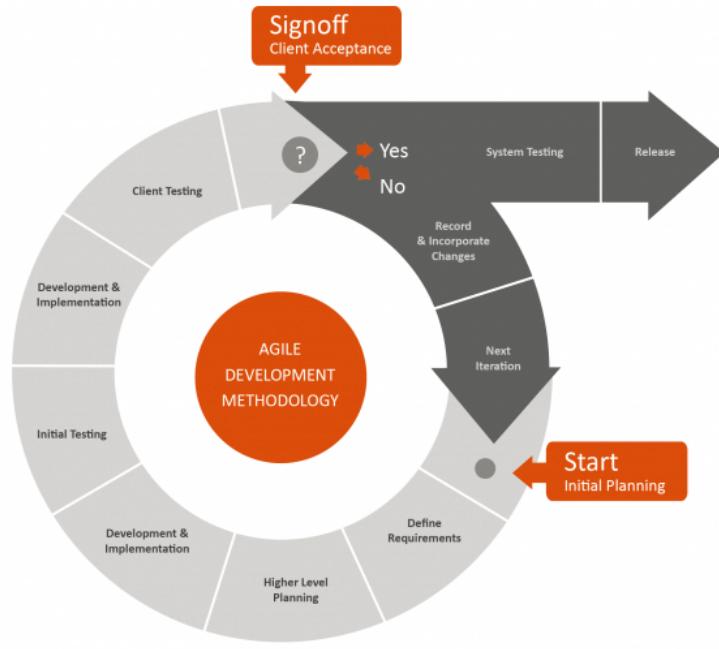


figura 1.4: Diagramma della metodologia di sviluppo *agile*. Immagine tratta da [GlobalTeckz](#).

### 1.2.1 Metodologia agile

L'azienda per la gestione dei progetti adotta la metodologia *agile*<sup>2</sup>. Questa metodologia si basa su 4 principi:

- Gli individui e le interazioni più che i processi e gli strumenti;
- Il *software* funzionante più che la documentazione esaustiva;
- La collaborazione col cliente più che la negoziazione dei contratti;
- Rispondere al cambiamento più che seguire un piano.

In particolare tra le metodologie *agile* l'azienda segue nello specifico il modello *Scrum* dove pianifica ciclicamente degli *sprint* (un periodo di 2-4 settimane).

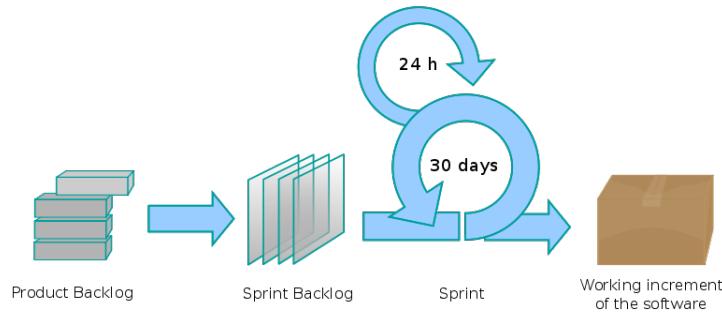
<sup>2</sup><http://www.agilemanifesto.org/>.

Gli elementi costruttivi principali in *Scrum* sono:

- **Product Backlog**, è un documento ad alto livello per l'intero progetto. Contiene una lista delle funzionalità desiderate con priorità assegnate in base al valore di *business* e i *Backlog Item* che sono descrizioni di tutte le caratteristiche richieste;
- **Sprint Backlog**, è un documento che contiene informazioni su come il *team* sta procedendo nell'implementare le funzionalità per lo *sprint* successivo. Le funzionalità vengono prelevate dalla pila del documento di *Product Backlog*, in una quantità fattibile per essere realizzata nello *sprint*;
- **Burn Down Chart**, è un grafico che rappresenta il lavoro nello *Sprint Backlog* che deve essere ancora completato. Aggiornato ogni giorno, dà una semplice visione di come procede lo *sprint*;
- **Incremento**, è la somma di tutti gli elementi del *Product Backlog* completati durante uno *sprint* e tutti gli *sprint* precedenti.

Le principali pratiche che Nextep svolge sono:

- **Daily scrum**, detto anche *stand up*, è una riunione che tiene quotidianamente e serve per verificare lo stato di avanzamento durante uno *sprint*;
- **Scrum of Scrums**, è una riunione che tiene meno frequentemente del *Daily Scrum* e consente ai *team* di discutere del loro lavoro, concentrandosi su aree di sovrapposizione e d'integrazione;
- **Sprint Planning Meeting**, è una riunione che l'azienda svolge all'inizio di ogni *sprint*. Vengono pianificati gli obiettivi con relative priorità e le tempistiche. In questa riunione partecipano *Product Owner*, *Scrum Master*, intero *Scrum Team* e tutti i *manager* del caso interessati o dai rappresentanti della clientela;
- **Sprint Review**, al termine dello *sprint* l'azienda svolge questa riunione per ispezionare l'incremento e adattare, se necessario, il *Product Backlog*. In conformità a questo e dei cambiamenti al *Product Backlog* fatti durante lo *sprint*, i partecipanti pianificano il prossimo incremento;
- **Sprint Retrospective Meeting**, si svolge dopo la *Sprint Review*, in questa occasione lo *Scrum Team* ispeziona se stesso e crea un piano di miglioramento da attuare durante il successivo *sprint*.



**figura 1.5:** Diagramma del modello *Scrum*. Immagine tratta da [Wikipedia](#).

Questa metodologia consente e promuove un maggiore coinvolgimento del cliente rispetto a quelle tradizionali, inoltre consente una maggiore flessibilità e reattività al cambio dei requisiti che possono avvenire in corso d'opera.

## 1.3 Strumenti a supporto dei processi

Questa sezione illustra gli strumenti utilizzati a supporto dei processi e le tecnologie utilizzate per lo sviluppo.

### 1.3.1 Gestione di progetto

Per fornire assistenza ai clienti l'azienda utilizza Zendesk<sup>3</sup>, che è uno strumento che permette di centralizzare tutti i canali di comunicazione tra i quali *social network*, *email* e telefono. Inoltre fornisce una panoramica generale della prestazione dell'assistenza e della soddisfazione del cliente.

Per la gestione di progetto invece utilizza lo strumento Jira<sup>4</sup>, che fornisce degli strumenti per la gestione di sviluppo agile con il modello *Scrum* utilizzato nell'azienda. Nello specifico, Jira permette:

- Assegnazione dei *ticket* tra i membri del *team* di sviluppo;
- Segnalazione di *issue*;
- Integrazione con il sistema di *repository*;
- Pianificazione dello *sprint*.

Nextep non utilizza la *Kanban Board* integrata in Jira, perché questo strumento di processo non è compatibile con la metodologia *agile* in uso, pertanto adopera una *Scrum Board* fisica presente in ufficio.

<sup>3</sup><https://www.zendesk.it/>.

<sup>4</sup><https://www.atlassian.com/software/jira>.

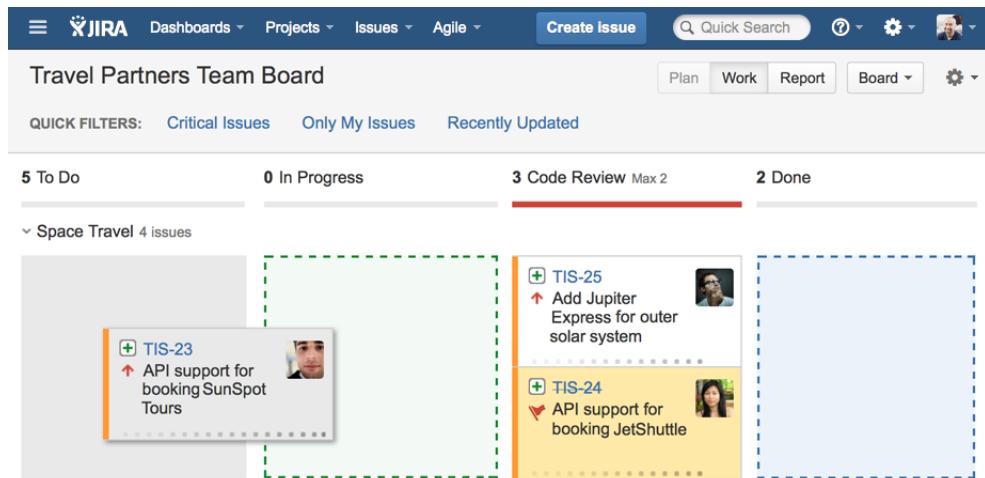


figura 1.6: Piattaforma di gestione di progetto Jira

### 1.3.2 Documentazione

Nextep per la produzione documentale utilizza una serie di strumenti di Google Docs, per una serie di fattori tra i quali:

- Compatibilità con tutti i sistemi operativi utilizzati;
- Permette di lavorare collaborativamente ai documenti;
- Disponibilità;
- Affidabilità.

### 1.3.3 Sistema di versionamento

Come sistema di versionamento il *team* utilizza lo strumento di Git<sup>5</sup>. I *repository* sono gestiti in un *server* locale ed occasionalmente in server esterni.

### 1.3.4 Ambiente di sviluppo

A seconda del contesto, il *team* utilizza l'ambiente di sviluppo che meglio si adatta alle tecnologie in uso nel progetto. In particolar modo adopera i seguenti *IDE*:

- Eclipse o IntelliJ: per lo sviluppo di applicazioni in Java e/o Scala;
- Smultron/PhpStorm: per lo sviluppo di applicazioni in Php;
- Sublime Text: per lo sviluppo in Html/Css, Javascript e Php.

---

<sup>5</sup><https://git-scm.com/>.

### 1.3.5 Sistemi operativi

L'azienda non impone nessun obbligo di utilizzo di un sistema operativo. I dipendenti utilizzano, a seconda delle preferenze, sistemi operativi quali Windows, Linux e MacOs. In particolar modo quest'ultimo, visto il recente investimento aziendale in terminali MacOs, per incentivare i dipendenti alla migrazione a questo sistema operativo.



**figura 1.7:** Classificazione degli strumenti utilizzati

## 1.4 Clientela tipo e propensione all'innovazione

La tipologia di clientela per cui l'azienda lavora copre una gamma che va dal singolo privato alle organizzazioni di grandi dimensioni.

Essendo gli *e-commerce* il *core business* dell'azienda, Nextep lavora principalmente per aziende dell'ambito manifatturiero e rivendite.

L'azienda lavora per clienti/aziende italiane e molto spesso per clienti della zona visto il riconoscimento guadagnato nel padovano.

Nextep riconosce questi valori guida da seguire:

- Attitudine all'innovazione per mantenere competitività;
- Condivisione del *know how* con le altre aziende in sede.

Questi principi rendono possibile un ambiente di *co-working* dove le 3 aziende della sede sperimentano e innovano grazie alle conoscenze messe in condivisione. Per alimentare questa dinamica l'azienda promuove periodicamente la formazione dei dipendenti, in modo che essi possano essere motivati e partecipi in questo ambiente. Visto l'importanza delle dinamiche personali, l'azienda organizza ogni volta che sia necessario degli incontri di *team building* per avere sempre dei dipendenti in grado di lavorare in gruppo.



## Capitolo 2

# Il progetto nella strategia aziendale

*Questo capitolo fornisce una descrizione dettagliata del progetto di stage, e dei motivi che hanno spinto l'azienda a proporlo oltre che i benefici derivati da tale progetto. Vengono inoltre discussi i vincoli tecnologici e metodologici imposti.*

### 2.1 L'azienda e gli stage

Questa sezione descrive il rapporto dell'azienda con gli stage e spiega quali sono i motivi che portano l'azienda a farli, in particolare per il progetto svolto.

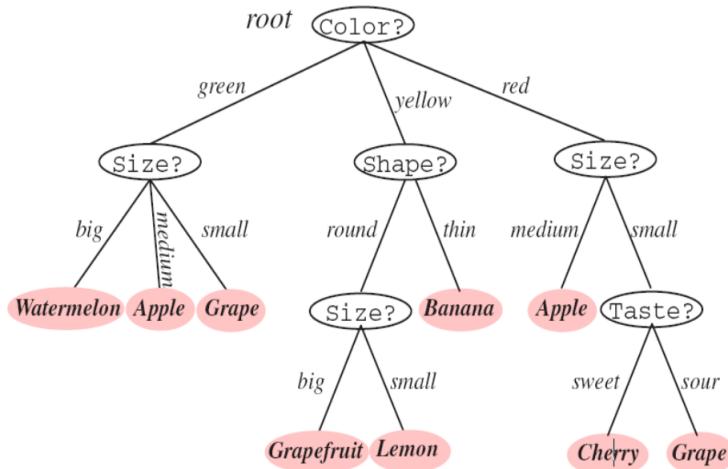
Data la forte evoluzione del settore *web*, l'azienda persegue l'obiettivo di esplorare i nuovi segmenti del campo per mantenere un ruolo di rilievo nel mercato. Per questo motivo c'è l'esigenza di sperimentare continuamente nuove soluzioni, per valutare possibili impieghi e vantaggi. Si vengono quindi a creare le condizioni per un'opportunità di *stage*, dove l'azienda richiede allo stagista una soluzione per le funzionalità richieste. Inoltre l'azienda sfrutta l'occasione degli *stage* per valutare una possibile collaborazione e successivamente una assunzione in azienda nel caso che il soggetto sia valutato positivamente. Nel precedente *stage*, visto l'interesse nel campo delle raccomandazioni, sono stati sviluppati dei prototipi che potessero soddisfare una raccomandazione utente in ambito *e-commerce*. I sistemi di raccomandazione in ambito commerciale sono molto utili, perché permettono un guadagno dal punto di vista economico, e dal punto di vista della *user experience* perché permettono di inserire pubblicità più pertinenti ai gusti dell'utente. In precedenza durante un recente *stage*, l'azienda aveva sviluppato il modulo di raccomandazione denominato DRE, un sistema basato sul *Collaborative filtering* che permette di fornire una raccomandazione in base alla somiglianza degli utenti.

## 2.2 Il progetto

Questa sezione descrive il progetto di *stage* e gli obiettivi che l'azienda ha fissato in relazione alle sue specifiche esigenze di mercato.

### 2.2.1 Il progetto

Il progetto di stage consiste nella realizzazione di un sistema di raccomandazione basato sulla classificazione del comportamento degli utenti all'interno del sito. Il sistema deve poter permettere l'inserimento delle azioni dell'utente ogni volta che il suo comportamento ha prodotto un risultato. Una volta collezionata una considerevole mole di dati relativi ai comportamenti, il sistema deve poter fornire una raccomandazione all'utente sulla base del suo comportamento attuale. Il comportamento di un utente è descritto tramite una serie di iterazioni e il risultato raggiunto. A sua volta una iterazione è descritta tramite l'azione e l'oggetto su cui viene svolta. Questo comportamento viene poi elaborato dall'albero di decisione generato dai dati collezionati precedentemente. La generazione dell'albero si basa sull'algoritmo di classificazione ID3 (*Iterative Dichotomiser 3*) che permette di selezionare l'attributo per discriminare il comportamento con il miglior guadagno informativo. Il sistema deve poter fornire una raccomandazione anche probabilistica nel caso di dati parziali sul suo comportamento.



**figura 2.1:** Esempio di un albero decisionale

### 2.2.2 Obiettivi

Prima dell'inizio dello stage, Nextep ha redatto un *Piano di Lavoro* contenente gli obiettivi da realizzare nelle settimane di *stage*. Gli obiettivi erano suddivisi in minimi e massimi, successivamente in seguito ad una variazione delle priorità aziendali sono stati effettuati dei cambiamenti. Inizialmente, ho effettuato il *porting* del modulo preesistente DRE, in modo che la persistenza fosse compatibile con OrientDb e che fosse compatibile con l'ultima versione di Play Framework. Visto il protrarsi delle attività di *porting*, l'azienda ha deciso di sospendere il lavoro per poter lavorare sul nuovo modulo *Tres*.

Qui di seguito presento gli obiettivi dello stage:

- Obiettivi formativi

- Formazione sui sistemi di raccomandazione e degli algoritmi di apprendimento;
  - Studio delle tecnologie utilizzate, Scala e OrientDb;
  - Studio degli strumenti utilizzati, in particolar modo dei *framework* e degli *IDE*.

- Obiettivi minimi

- Utilizzo del database OrientDb.

- Obiettivi massimi

- Implementazione di nuovi modelli di algoritmi di apprendimento per migliorare l'individuazione dei gusti dell'utente;
  - Migliorare la fase di apprendimento del modello stesso;
  - Implementazioni di algoritmi di *cluster* per il raggruppamento di elementi omogenei in un insieme di dati.

## 2.3 Vincoli

Questa sezione elenca i vincoli relativi al progetto di *stage*.

### 2.3.1 Vincoli temporali

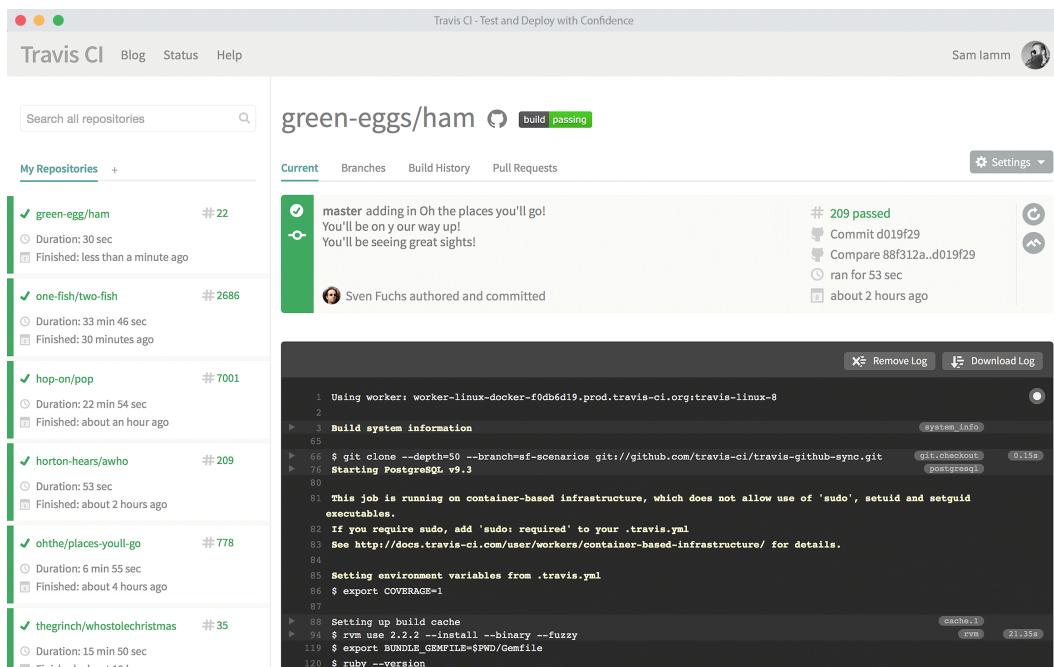
Per lo *stage* i vincoli temporali sono imposti direttamente dall'università, essa stabilisce una durata minima di 300 ore e una durata massima di 320 ore. Per questo motivo ho modulato il *Piano di Lavoro* in modo da distribuire le ore lavorative in 8 settimane. Con l'azienda ho concordato un impegno di 300 ore, divise in 37.5 ore settimanali. Le variazioni del piano sono state concordate con il *tutor* aziendale in maniera da rispettare i vincoli temporali imposti in seguito a ritardi dovuti al *porting* del modulo DRE.

### 2.3.2 Vincoli metodologici

Come strumento di versionamento ho scelto Git perché già utilizzato in precedenza e per il supporto che riceve da molti strumenti di sviluppo. Per la documentazione del codice ho scelto Scaladoc perché viene fornito insieme al compilatore e similmente a Javadoc permette di documentare direttamente nel codice sorgente. Per il *testing* nel progetto ho utilizzato TravisCI, esso fornisce un servizio di *continuous integration* facilmente integrabile con i *repository* di Github<sup>1</sup>.

---

<sup>1</sup><https://github.com/>.



**figura 2.2:** Pannello di controllo travis

Per lo sviluppo del codice ho scelto come *IDE* IntelliJ in quanto:

- Già precedentemente utilizzato;
- Ambiente di sviluppo compatibile con i maggiori linguaggi *JVM-based*;
- Integrazione con Git;
- Integrazione per lo sviluppo in Scala con Sbt.

Per la gestione progetti ho utilizzato il *build tool* Sbt, perché nell'ambiente di sviluppo di Scala è la scelta predominante ed è compatibile con il *framework* utilizzato *Play Framework*. Le caratteristiche distintive di questo *framework* sono elencate di seguito:

- Definizione del file di configurazione in Scala;
- Compatibilità con Scaladoc per documentazione;
- Supporto per progetti misti Java/Scala;
- Supporto per i maggiori *tool* di *testing*;
- Supporto per la gestione delle librerie.

Per la realizzazione dei diagrammi *UML* (*Unified Modeling Language*) ho utilizzato Astah Professional<sup>2</sup>, uno strumento già utilizzato in precedenza e perché compatibile con gli ultimi standard *UML 2.0*.

<sup>2</sup><http://astah.net/>.

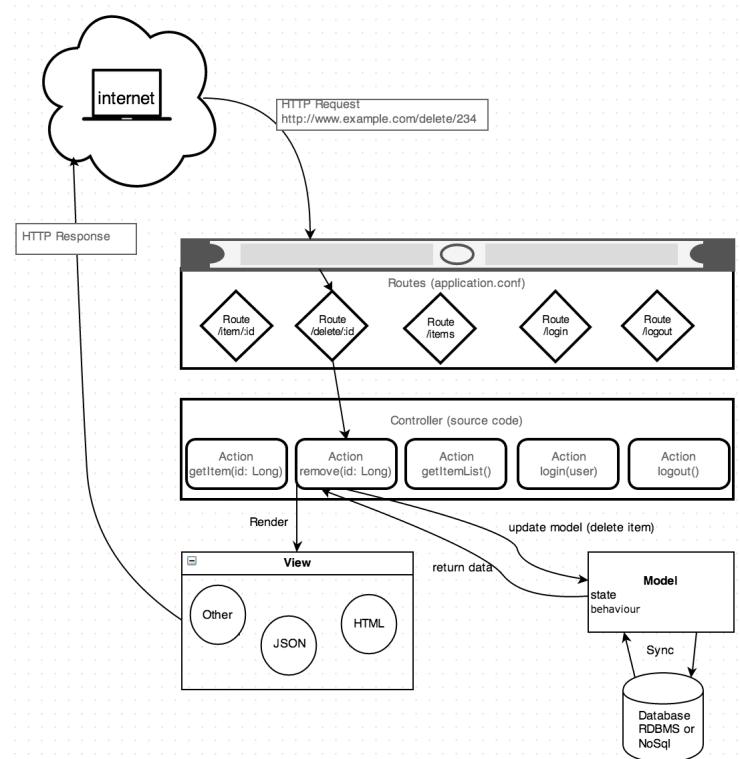
### 2.3.3 Vincoli tecnologici

#### Scala

Il principale linguaggio di programmazione adottato è Scala<sup>3</sup> (acronimo di *"Scalable Language"*). E' stato studiato per poter interoperare con *Java Runtime Environment (JRE)*, quindi è possibile scrivere applicazioni sia in Java e Scala che eseguono nella stessa *Java Virtual Machine (JVM)*. E' un linguaggio multi paradigma, che integra concetti provenienti dai linguaggi della programmazione ad oggetti e della programmazione funzionale. Scala è un linguaggio completamente *object oriented*, infatti ogni elemento del linguaggio è un oggetto, compresi i tipi primitivi e le funzioni. Inoltre è un linguaggio funzionale, include funzioni di prima classe e una libreria con strutture dati immutabili efficienti.

#### Play Framework

Play è un *framework open source*, scritto in Java e Scala, che permette l'implementazione di *web application* con il *pattern model-view-controller*. Questo *framework* trae ispirazione ed è simile a Ruby on Rails e Django.



**figura 2.3:** Tipica architettura di un'applicazione che utilizza Play Framework. Immagine tratta da [Omer Haderi Blog](#).

<sup>3</sup><http://www.scala-lang.org/>.

Le peculiarità che lo contraddistinguono dagli altri *framework* sono le seguenti:

- Privo di stato: non viene mantenuta alcuna sessione sul server relativa ai dati dell'utente corrente;
- Metodi statici: tutti i metodi dei *controller* che vengono invocati dal *framework* sono statici, o nel caso in cui si usi la versione Scala, sono funzioni di oggetti di Scala;
- Gestione asincrona dell'*input* e dell'*output*: grazie all'uso di Netty Server, Play può gestire le richieste lunghe in modo asincrono;
- Architettura modulare: come per Rails e Django, ci sono i moduli;
- Supporto nativo per Scala: non solo Play è fatto internamente in Scala, ma espone anche delle interfacce Scala. Le interfacce Java sono state messe appositamente in *package* diversi affinché possano seguire le convenzioni di Java.

Al suo interno permette una implementazione delle *views*, specificando l'interfaccia con un linguaggio di *markup* (e.g. Html, Xml, Json) in combinazione con del codice Scala. Vengono fornite molte classi di *utility* come ad esempio Play Json Library che permette di operare con dei dati in formato Json, oppure un adattamento della libreria Specs2 che permette un supporto per il *testing* di una applicazione in Play. Una tipica applicazione in Play funziona convogliando tutte le *HTTP Request* nella componente *Router* che invocherà a sua volta il *Controller* pre-selezionato. Il *Controller* interagisce con il *Model* che a sua volta restituisce un risultato. In genere questo risultato verrà poi *renderizzato* con la *View* e fatto ritornare tramite un *HTTP Response*.

## OrientDB

Per la persistenza dei dati ho utilizzato OrientDb<sup>4</sup>, un *Database Management System* (DBMS) multi-modello con una licenza *open source*. Supporta i seguenti modelli di persistenza:

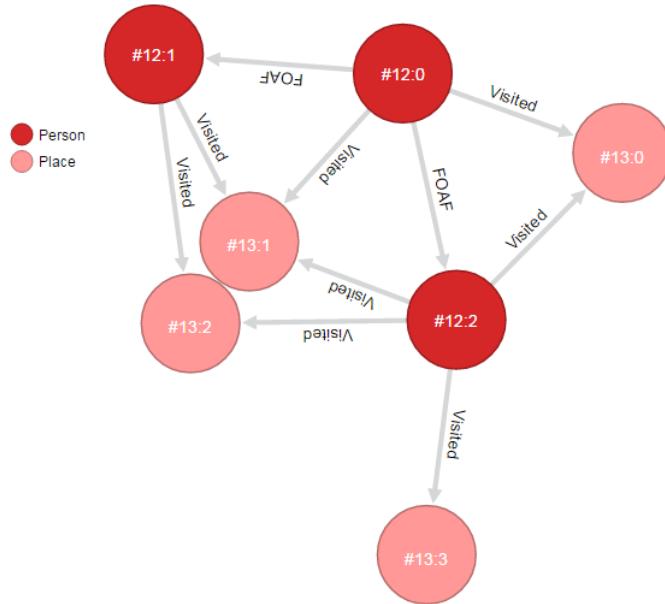
- *Graph*: modello a grafo;
- *Document*: modello a documenti come MongoDB;
- *Object*: modello ad oggetti che permette di memorizzare direttamente i *POJO*.

Inoltre supporta modelli di definizione dello schema come *schema-less*, *schema-full* e *schema-mixed*. Per motivi prestazionali non permette le *join*, le relazioni sono gestite come in un database a grafo, ovvero delle connessioni dirette tra i *record*. Questa caratteristica lo rende adatto nel contesto dei *Big Data*, infatti la complessità computazionale per gestire una creazione di un *link* è pari a O(1). Tra le caratteristiche più importanti di OrientDb:

- Scalabile: ovvero è possibile aggiungere al sistema un altro *server* per aumentarne le prestazioni;
- Transazionale: supporta transazioni completamente *ACID* garantendo che tutte le transazioni vengano processate in modo affidabile e nel caso di *crash* viene ripristinato lo stato precedente(*rollback*);
- Polyglot: supporta *query* in un dialetto *SQL-like*, oppure delle *API REST* che forniscono l'accesso ai dati.

---

<sup>4</sup><http://orientdb.com/>.



**figura 2.4:** Esempio di modellazione in OrientDb. Immagine tratta da [Scalac Blog](#)

## 2.4 Obiettivi personali

Questa sezione elenca gli obiettivi personali che hanno portato allo svolgimento dello *stage*.

Ho conosciuto l’azienda durante l’evento *Stage-IT*. Questo evento mi ha dato la possibilità di approfondire molte proposte di *stage*, da parte aziende provenienti da tutta la provincia di Padova e non. In seguito a tutte le proposte di *stage* disponibili ho accettato la proposta di Nextep S.r.l. per i seguenti motivi:

- Il progetto permetteva di lavorare con tecnologie innovative che mi avrebbero fornito un valore professionale aggiunto;
- Il progetto proposto era interessante perché toccava rami dell’intelligenza artificiale e dei sistemi di raccomandazione;
- L’utilizzo di Scala, in modo da poter padroneggiare un linguaggio con un paradigma funzionale in forte ascesa di popolarità.



# Capitolo 3

## Il progetto di stage

*Questo capitolo tratta dettagliatamente delle attività di pianificazione, studio, analisi, progettazione e sviluppo, svolte durante lo stage.*

### 3.1 Organizzazione

Per conseguire tutti gli obiettivi fissati ho previsto un ammontare di 300 ore. Insieme al *tutor* aziendale ho concordato la data di inizio stage il 5/10/2015 e la data di fine stage il 27/11/2015, quindi un carico lavorativo di 37.5 ore settimanali, ovvero 7.5 ore giornaliere.

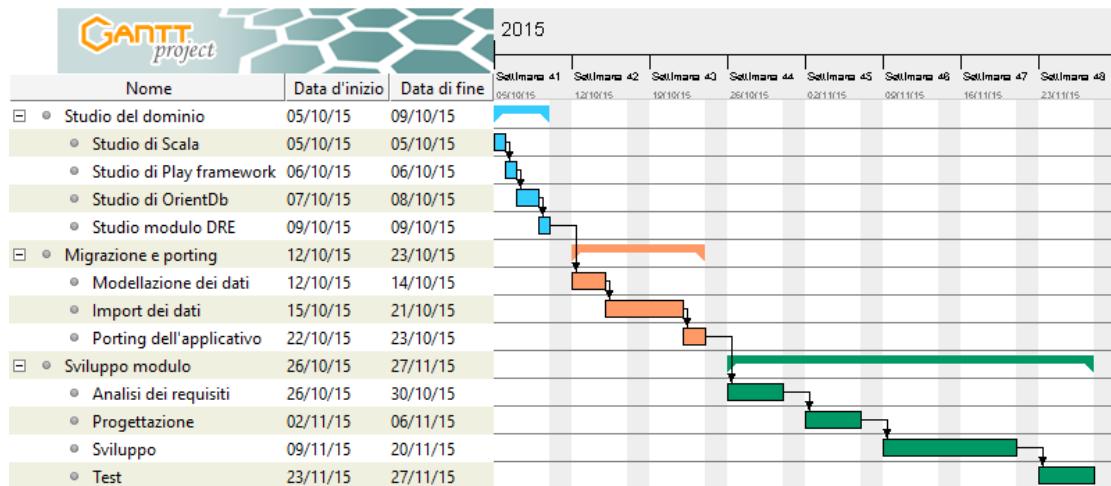


figura 3.1: Suddivisione delle attività

### 3.1.1 Pianificazione

In sede di pianificazione prima dell'inizio dello *stage* ho programmato 3 macro attività sequenzialmente. Ho assegnato nel primo periodo una settimana per l'attività di studio del dominio, in quanto questa attività è propedeutica alle successive attività. Nel periodo successivo di 2 settimane, ho pianificato l'attività di *porting* dell'applicativo DRE su richiesta del *tutor* aziendale per motivi di strategia aziendale. Nell'ultimo periodo ho programmato l'attività di sviluppo del modulo denominato Tres. Ho privilegiato quest'ultima attività assegnando un totale di 5 settimane lavorative. Qui di seguito illustro le sotto-attività programmate per ogni settimana.

- **Prima settimana:**
  - Studio del *database* OrientDB;
  - Studio del linguaggio di programmazione Scala;
  - Studio del *framework* Play;
  - Studio del progetto preesistente;
- **Seconda e terza settimana:** migrazione dalla tecnologia MongoDB a OrientDB e *porting* all'ultima versione di Play;
- **Quarta settimana:** Analisi dei requisiti del nuovo modulo;
- **Quinta settimana:** progettazione architetturale;
- **Sesta e settima settimana:** implementazione modulo;
- **Ottava settimana:** test.

La figura 3.1 presenta il diagramma di *gantt* utilizzato per la pianificazione.

## 3.2 Analisi dei requisiti

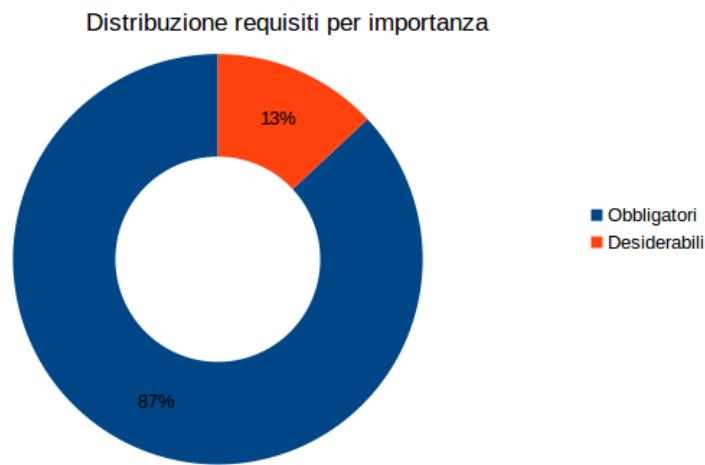
### 3.2.1 Requisiti

Per l'attività di analisi dei requisiti ho svolto molteplici incontri insieme al *tutor* aziendale per delineare quali funzionalità dovesse offrire il sistema. Il *tutor* mi ha fornito molti esempi ed illustrazioni per capire al meglio quali requisiti il sistema dovesse soddisfare. Tuttavia questa attività è stata molto ostica perché sia il *tutor* e sia il *CEO* esprimevano richieste diverse e spesso molto discordanti. Visto che il proponente non aveva le idee molto chiare sul sistema da implementare, ho cercato di assumere un atteggiamento più propositivo per aiutarlo a comprendere i suoi bisogni e portare a termine l'analisi dei requisiti. Una volta definito le funzionalità, ho trascritto i requisiti in un file testuale, in un formato organizzato tabellare. Visto le dimensioni ridotte del progetto ho prediletto il tracciamento dei requisiti con un file di testo anziché utilizzare un sistema automatizzato. Il file è stato poi sottoposto a versionamento all'interno del *repository* per permettere operazioni di consultazione e modifica all'evolversi dei requisiti. La notazione scelta per suddividere i requisiti è la seguente:

R[Importanza][Tipologia][Codice]

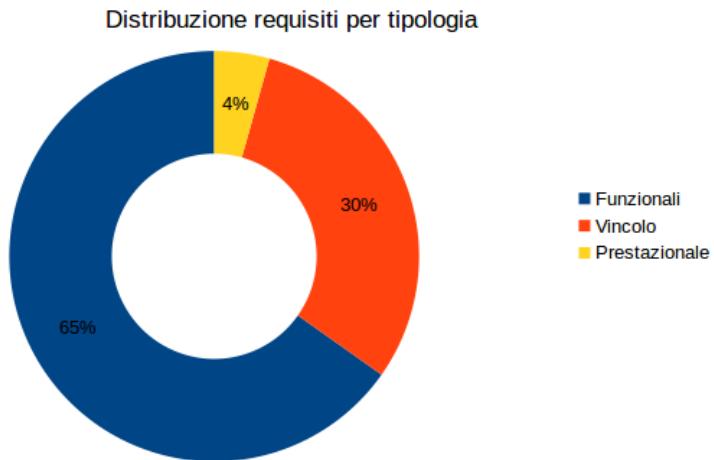
- Importanza può assumere i seguenti valori:

- **OBB:** requisito obbligatorio. Il soddisfacimento è necessario per il raggiungimento degli obiettivi dello *stage*;
- **DES:** requisito desiderabile. L'implementazione non è fondamentale, ma dà valore aggiunto al prodotto.
- Tipologia può assumere i seguenti valori:
  - **F:** requisiti funzionali. Specifica una funzionalità che il *software* deve avere;
  - **V:** requisiti di vincolo. Specifica il vincolo che il *software* deve avere;
  - **P:** requisiti di prestazione. Specifica un vincolo di *performance* che il *software* deve fornire.
- Codice è un *id* numerico univoco.



**figura 3.2:** Distribuzione requisiti in percentuale per tipologia

Durante l'analisi dei requisiti ho prima di trasporre i requisiti secondo la notazione precedentemente menzionata, ho rielaborato i requisiti in modo che non contenessero ambiguità verbali, se presenti ho corredato al documento di analisi un glossario dove specificare il termine ed evitare così interpretazioni errate del requisito. Inoltre per ogni requisito l'ho suddiviso in più requisiti fino a che ogni singolo requisito risultasse indivisibile, oppure i requisiti che risultavano ridondanti tra loro, li ho uniti perché ogni requisito risultasse indipendente. Infine ho verificato che ogni requisito fosse verificabile secondo una metrica in modo da certificarne il soddisfacimento. Al termine dell'analisi, i requisiti individuati sono 23 di cui, come raffigurato nell'immagine 3.2, 15 di natura funzionale, 1 prestazionale e i restanti 7 requisiti di vincolo. I requisiti sono stati classificati per priorità, determinando così 20 requisiti obbligatori e 3 desiderabili, questi ultimi riguardano nella maggior parte gli obiettivi di massima fissati dal piano di lavoro.



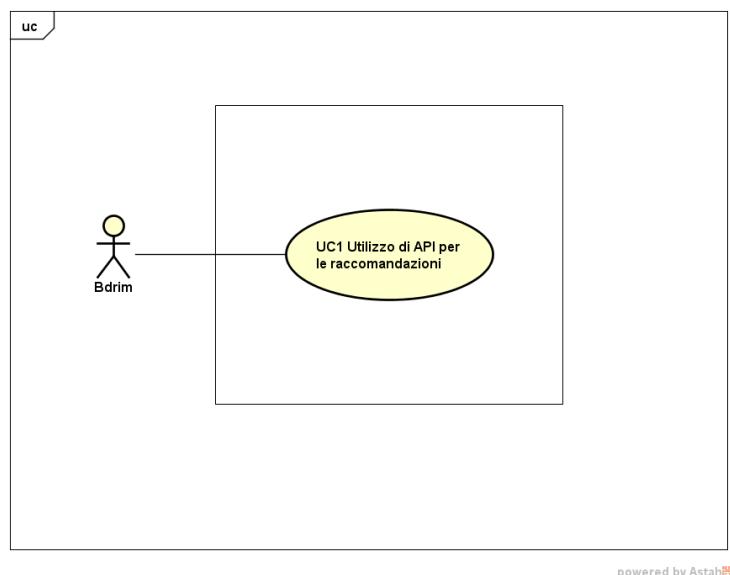
**figura 3.3:** Distribuzione requisiti in percentuale per importanza

### 3.2.2 Casi d'uso

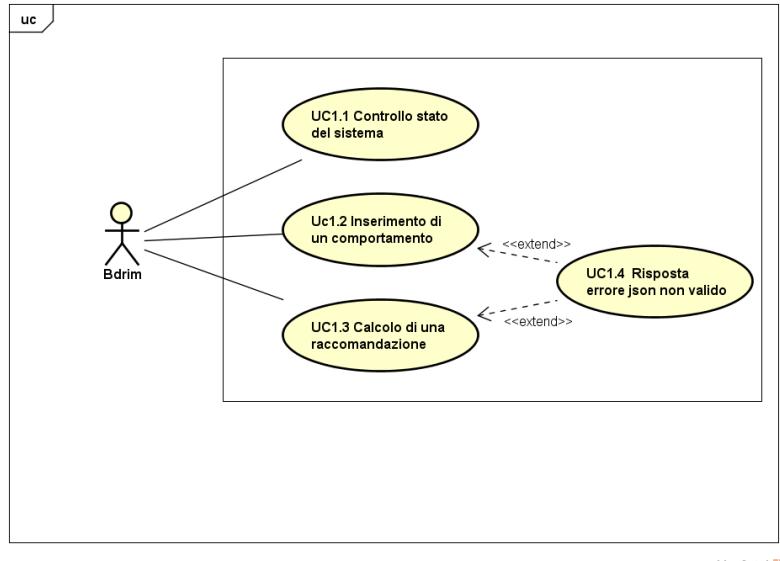
Durante l'analisi dei requisiti ho realizzato dei diagrammi *UML*, utilizzando il *software* Astah. Tres è concepito per operare con Bdrim, una piattaforma di raccolta dati sviluppato in collaborazione con Allos. Quindi l'unico attore coinvolto nell'utilizzo di Tres è Bdrim che espone un caso d'uso principale. In questa sezione riporto i casi d'uso principali che descrivono maggiormente il sistema nel suo complesso. Per ogni caso d'uso ho specificato:

- Gli attori coinvolti;
- Pre e post condizione;
- Scopo e descrizione;
- Flusso principale degli eventi.

Inoltre, se previsto, ho specificato estensione e lo scenario alternativo.

**UC0****figura 3.4:** UC0: Tres

- **Attori:** Bdrim
- **Scopo e descrizione:** Questo caso d'uso descrive le funzionalità messe a disposizione dal sistema. Principalmente Bdrim può usufruire delle *API* per le raccomandazioni fornite da Tres.
- **Pre-condizione:** Il *server* è stato avviato.
- **Flusso principale degli eventi:** Bdrim utilizza le *API* per ottenere una raccomandazione.
- **Post-condizione:** Il sistema risponde alle richieste HTTP effettuata da Bdrim.

**UC1**

powered by Astah

**figura 3.5:** UC1: Tres

- **Attori:** Bdrim
- **Scopo e descrizione:** Questo caso d'uso illustra le funzionalità offerte dal sistema a Bdrim. Bdrim ha la possibilità di effettuare delle richieste *REST* al sistema;
- **Pre-condizione:** Il *server* è stato avviato e *Tres* è stato attivato.
- **Flusso principale degli eventi:**
  - 1 Bdrim richiede lo stato del sistema [UC1.1];
  - 2 Nel caso il sistema sia pronto Bdrim richiede la raccomandazione[UC1.3];
  - 3 Bdrim può inserire un comportamento[UC1.2].
- **Scenario alternativo:** Bdrim effettua delle richieste *HTTP* inviando un oggetto *Json* malformato oppure invalido e ritorna una risposta *HTTP* che segnala l'errore in formato *Json*.
- **Estensione:** Bdrim riceve una risposta *HTTP* e all'interno il *Json* che segnala l'errore di dati non validi [UC1.4];
- **Post-condizione:** Il sistema risponde alle richieste *HTTP* ritornando una risposta *HTTP*, ritornando i dati in formato *Json*.

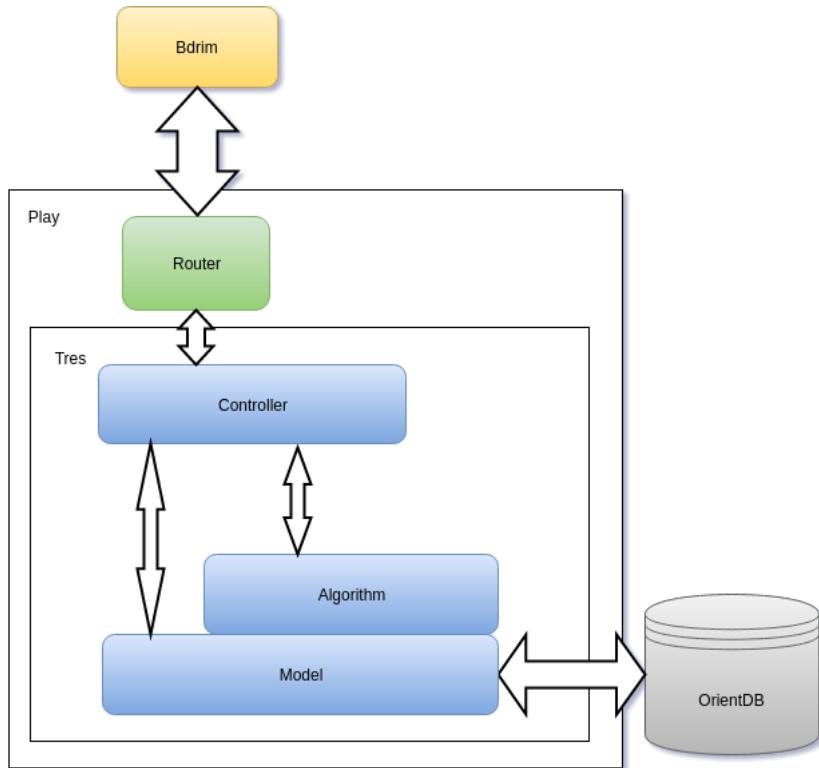
### 3.3 Progettazione architetturale

Questa sezione descrive l'attività di progettazione. Illustra l'architettura generale del sistema, le relazioni interne, la struttura del *database* e infine i *design pattern* utilizzati.

#### 3.3.1 Visione ad alto livello

Il contesto di utilizzo dell'applicativo è la piattaforma di Bdrim. Questa piattaforma permette di gestire nuovi moduli per introdurre nuove funzionalità. Ogni modulo deve essere assolutamente isolato per poter essere facilmente modificabile e interscambiabile con altri moduli che migliorino le sue funzionalità. Quindi la visione generale del sistema si riconduce ad un modello *client-server* dove i ruoli sono:

- **Server:** rappresentato da Tres, elabora le richieste *HTTP* ricevute da Bdrim.
- **Client:** rappresentato unicamente da Bdrim.

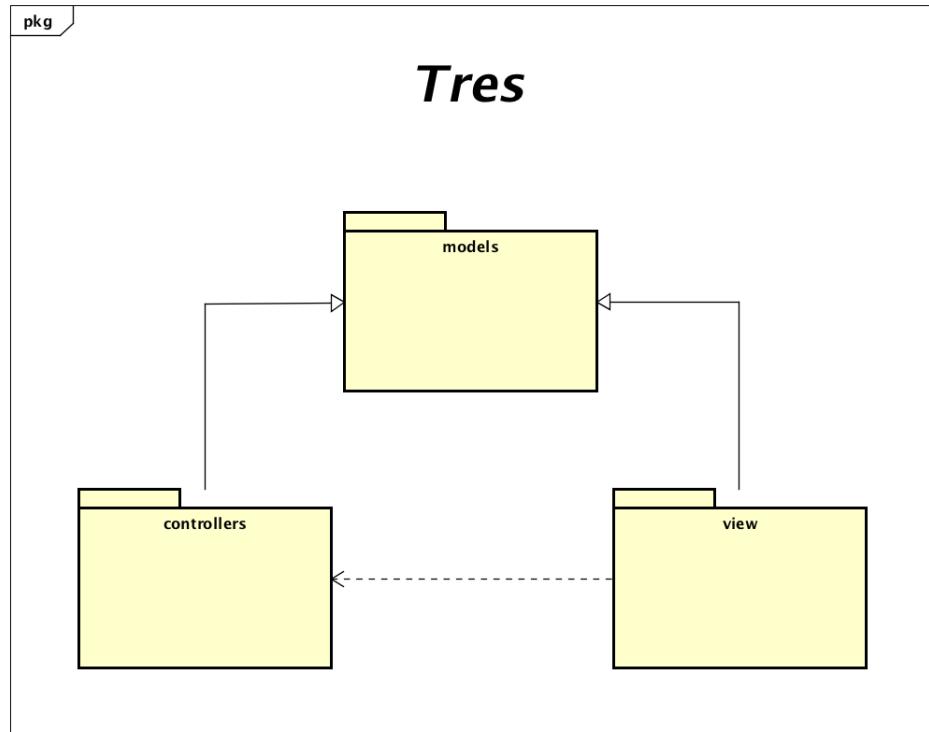


**figura 3.6:** Visione generale dell'architettura

Le due componenti sono indipendenti tra loro e comunicano utilizzando una architettura *REST*.

### 3.3.2 Architettura

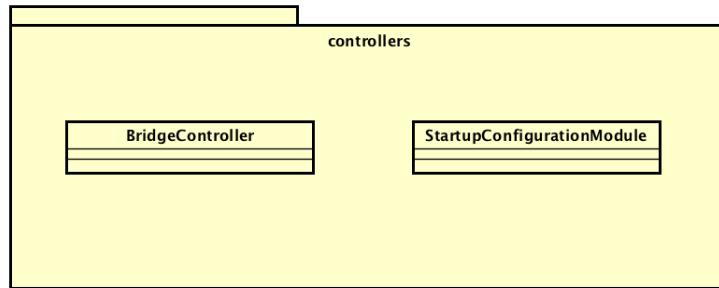
Per la realizzazione di Tres, ho seguito un *design* architetturale *MVC*. Quindi i *package* principali sono *model*, *view* e *controller*. Attualmente il *package* *views* non viene utilizzato, il suo utilizzo tornerà utile in eventuali sviluppi successivi per la creazione di una interfaccia di amministrazione per Bdrim o altre eventuali funzionalità. Qui di seguito illustro la visione generale e successivamente le componenti del sistema.



**figura 3.7:** Diagramma del *package tres*

### controllers

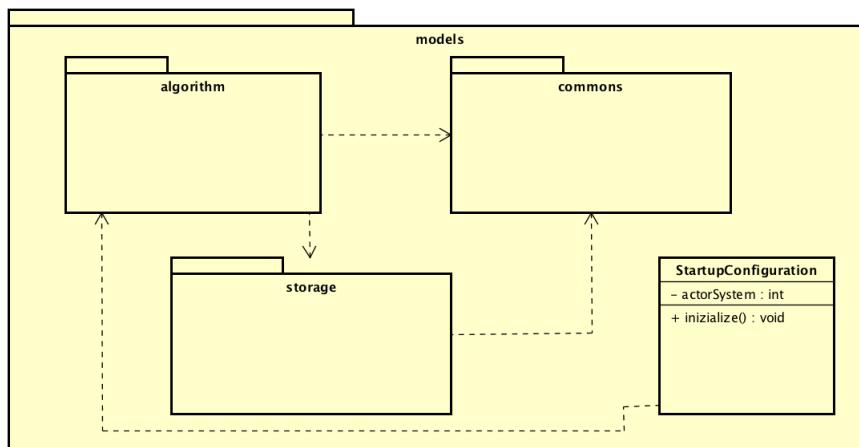
Il *package controllers* contiene le componenti responsabili della logica di controllo e nello specifico contiene due classi. *BridgeController* è la componente che gestisce le richieste *HTTP* instradate dalla componente *router* del *framework Play*. Il *package* interagisce con *model* utilizzando i servizi messi a disposizione e le rappresentazioni dei dati.



**figura 3.8:** Diagramma del *package controllers*

### models

Il *package models* contiene tutta la logica di *business* del sistema e di accesso ai dati. È responsabile di fornire una separazione dalla rappresentazione interna dei dati del database. Fornisce tutte le interfacce necessarie al *controller* necessarie per fornire i servizi.



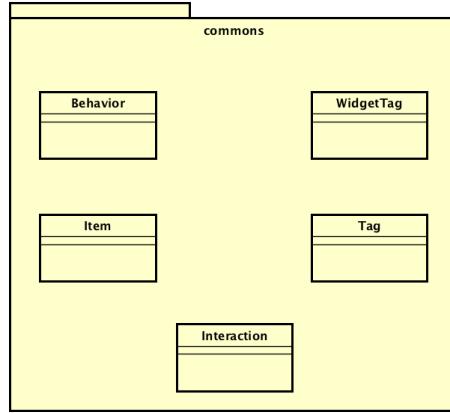
**figura 3.9:** Diagramma del *package models*

### Classi contenute

- **StartupConfiguration:** Questa classe è responsabile di schedulare la generazione dell'albero per le raccomandazioni. La generazione dell'albero avviene ogni 24 ore e solamente se sono disponibili almeno 100 comportamenti memorizzati.

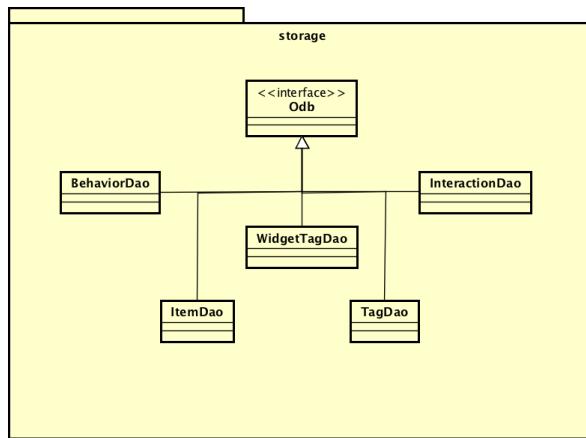
### Package contenuti

**commons** Il *package commons* contiene le componenti per la rappresentazione interna dei dati. All'interno sono presenti dei convertitori impliciti, che permettono la validazione e la conversione in formato Json, funzionalità utili per le componenti nel *controller*.



**figura 3.10:** Diagramma del *package commons*

**storage** Il *package storage* è responsabile dell'accesso ai dati. Fornisce una interfaccia che fornisce delle funzionalità *CRUD*. Il *package* dipende dalle componenti in *commons*, e per ognuna implementa l'interfaccia per l'interazione con OrientDb. Le componenti che gestiscono la persistenza in OrientDb utilizzano le API Blueprints<sup>1</sup> perché OrientDb aderisce a questo standard di default. TinkerPop Blueprints fornisce delle API per manipolare *database* a grafo ed è incluso nello *stack* dei progetti di TinkerPop<sup>2</sup>



**figura 3.11:** Diagramma del *package storage*

<sup>1</sup><https://github.com/tinkerpop/blueprints>.

<sup>2</sup><http://tinkerpop.incubator.apache.org/>.

**algorithm** Il package *algorithm* contiene le componenti *software* che realizzano l'algoritmica basata essenzialmente su *ID3* ed è fortemente dipendente dalle componenti presenti in *commons*. Fornisce la funzionalità per la costruzione dell'albero decisionale sulla base dei *behavior* in *input*. Offre la funzionalità per ricevere in *output* la raccomandazione sulla base di un *behavior* in *input*.

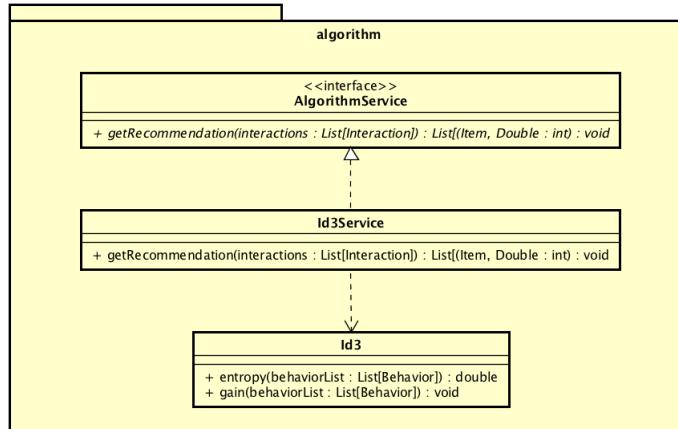


figura 3.12: Diagramma del package *algorithm*

### 3.3.3 Database

OrientDb è un *database* a grafo, quindi esistono due entità, i vertici e gli spigoli. Entrambi possono contenere degli attributi al loro interno, ma nel caso degli spigoli è fortemente sconsigliato per motivi prestazionali<sup>3</sup>. Per la persistenza dei dati ho scelto una definizione dei dati di tipo *schema-full* per evitare un controllo dei dati a *run-time* nelle componenti del package *storage*.

Il vertice principale è *Behavior* che rappresenta il comportamento di un utente. Da un *behavior* si dirama l'associazione verso un *Item* attraverso lo spigolo *Result* ed esprime che il comportamento dell'utente ha come risultato un *Item*. Un *Item* è descritto con una serie di *Tag*, quindi un vertice di tipo *Item* è associato a molteplici *Tag* tramite uno spigolo *HoldsTag*. Un *Behavior* descrive una serie di interazioni su dei *WidgetTag*, quindi un vertice di tipo *Behavior* è associato a molteplici *WidgetTag* tramite uno spigolo *Interaction*. *Interaction* contiene la tipologia svolta su di un determinato *WidgetTag*.

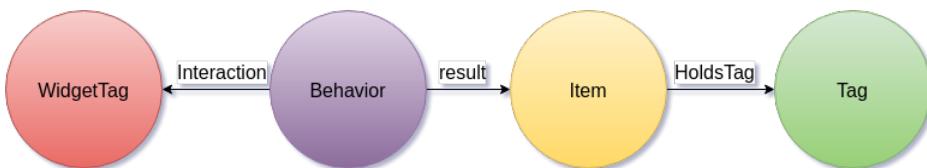


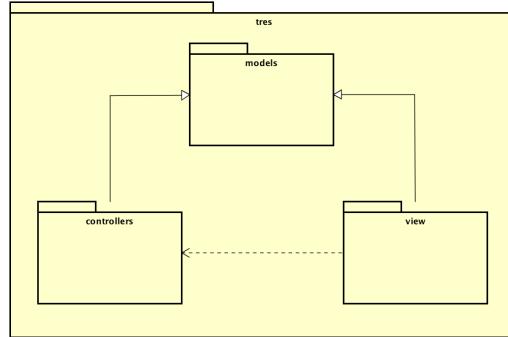
figura 3.13: Modello a grafo del database

<sup>3</sup><http://orientdb.com/docs/last/Performance-Tuning-Graph.html>

### 3.3.4 Design pattern

Qui di seguito elenco i *design pattern* utilizzati per la realizzazione di Tres.

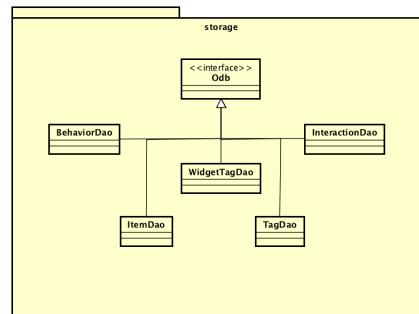
#### MVC



**figura 3.14:** Contesto di utilizzo del *design pattern* *MVC*

- **Scopo dell'utilizzo:** questo *design pattern* viene utilizzato per separare i compiti delle diverse componenti *software* dell'applicazione, separando *business logic*, *application logic* e viste, rappresentate rispettivamente da *models*, *controller* e *views*;
- **Contesto di utilizzo:** questo *pattern* viene implementato nativamente dal *framework* Play. In particolare implementa una tipologia di *MVC pull-model* che permette un salto tecnologico tra la *view* e il *controller*. Questo pattern architettonicale è importantissimo in azienda perché permette di lavorare parallelamente sulle componenti. In genere i programmatori lavorano sulle componenti del controller e del model, invece i designer lavorano sulle componenti di view. (RIVEDERE?)

#### Dao



**figura 3.15:** Contesto di utilizzo del *design pattern* *DAO*

- **Scopo dell'utilizzo:** questo *design pattern* è utilizzato per disaccoppiare la logica di *business* dalla logica di persistenza. Per rendere così le componenti indipendenti dal sistema di *database*;
- **Contesto di utilizzo:** viene utilizzato dalle componenti presenti nel *package storage* per interagire col *database* OrientDb. L'azienda utilizza diverse tecnologie di persistenza dei dati, e quindi per essere facilmente interscambiabile in scenari futuri è importante l'utilizzo di questo pattern che permette una facile transizione da una tecnologia all'altra.

### Strategy

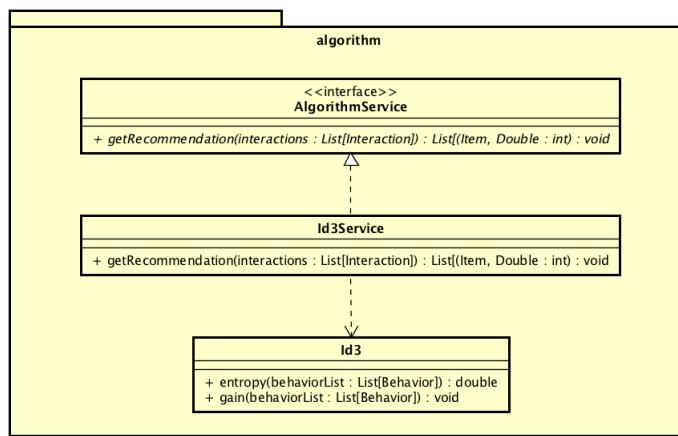


figura 3.16: Contesto di utilizzo del *design pattern strategy*

- **Scopo dell'utilizzo:** il *design pattern strategy* definisce una famiglia di algoritmi, incapsulati e resi intercambiabili. *Strategy* permette agli algoritmi di variare indipendentemente dai client che ne fanno uso;
- **Contesto di utilizzo:** viene utilizzato dalle componenti presenti nel *package algorithm*. E' presente una interfaccia per l'algoritmo che viene implementata da *Id3Service*, che realizza l'algoritmo *ID3*. Ho ritenuto importante applicare questo pattern nell'ottica di rendere l'algoritmo facilmente interscambiabile con altri algoritmi di raccomandazione, come ad esempio degli algoritmi basati sulla teoria dei giochi.

### Singleton

- **Scopo dell'utilizzo:** il *design pattern singleton* assicura che una certa classe abbia una sola istanza e fornisce un punto d'acceso globale a tale istanza;
- **Contesto di utilizzo:** viene utilizzato nelle componenti del *package controller* e viene implementato nativamente dal linguaggio Scala dichiarando la classe come *object*. Il suo utilizzo è importante perché previene da inutili sprechi di memoria che si avrebbero in un contesto *multi-thread* come questo applicativo *web*.

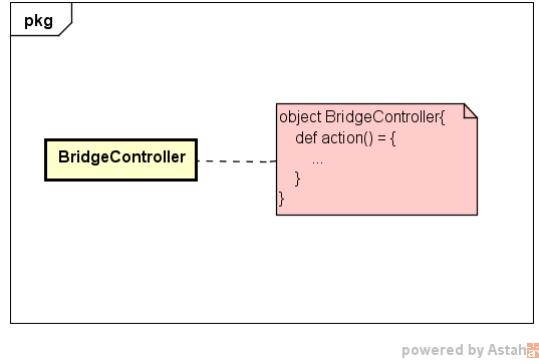


figura 3.17: Contesto di utilizzo del *design pattern singleton*

### 3.4 Progettazione di dettaglio

La progettazione di dettaglio con la metodologia *agile* avviene di pari passo durante la codifica, il che è rischioso perché introduce molti errori di codifica, tuttavia le ore risparmiate della progettazione di dettaglio sono state reimpiegate nel correggere eventuali errori. Riporto qui di seguito i diagrammi di sequenza UML più significativi:

#### 3.4.1 Controllo stato

In fase di *startup* il sistema non possiede nessun dato, e quindi non è in grado di fornire raccomandazioni. Un modo per verificare lo stato operativo del modulo è attraverso una chiamata mediante questa chiamata dell’interfaccia *REST*:

Tipo	Chiamata
GET	/tres/ready

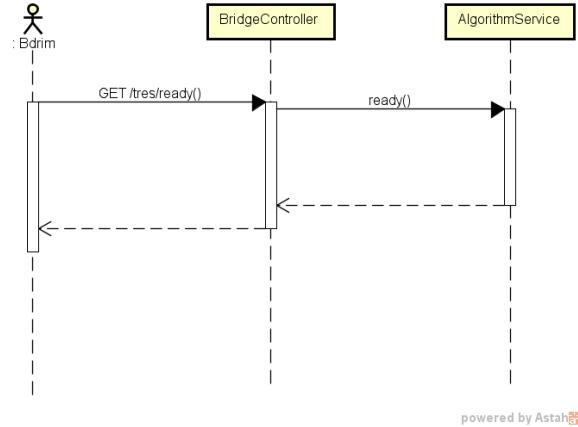


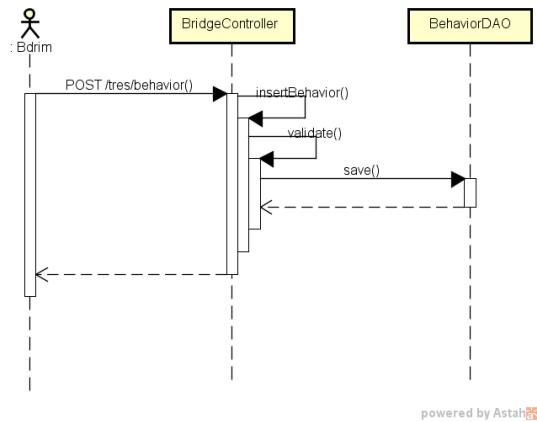
figura 3.18: Diagramma di sequenza: controllo stato.

Mediante il *router* la richiesta *HTTP* giunge fino ad al metodo *ready* della classe *BridgeController* che a sua volta controllerà la presenza dell’albero tramite *AlgorithmService*. Al termine ritorna in *output* una risposta *HTTP* contenente un oggetto *Json*.

#### 3.4.2 Inserimento behavior

Per istruire l’algoritmo di selezione è necessario fornire dei dati per l’apprendimento. Ho implementato una procedura di inserimento dei comportamenti mediante questa chiamata dell’interfaccia *REST*:

Tipo	Chiamata
------	----------



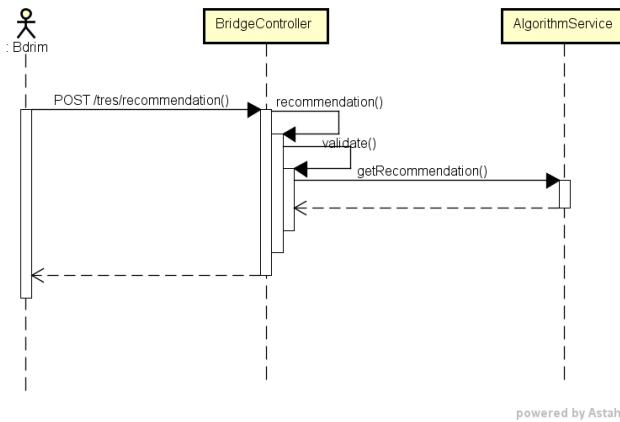
**figura 3.19:** Diagramma di sequenza: inserimento comportamento.

Da Bdrim arriva una richiesta *HTTP POST* che comprende un *Json* contenente un *Behavior*. Il *router* di Play accoglie la richiesta che la reindirizza al metodo della classe *BridgeController insertBehavior*. Il flusso continua con la validazione del *Json* e successivamente inserisce il comportamento invocando il metodo *save* della classe *BehaviorDAO*.

### 3.4.3 Raccomandazione

Una volta raccolti almeno 100 *Behavior* e che l'algoritmo ha generato l'albero, il sistema è in grado di fornire una raccomandazione. Ho implementato la procedura di raccomandazione mediante questa chiamata dell'interfaccia REST:

Tipo	Chiamata
POST	/tres/recommendation



**figura 3.20:** Diagramma di sequenza: raccomandazione.

Da Bdrim arriva una richiesta *HTTP POST* che comprende un *Json* contenente le interazioni dell'utente. Il *router* convoglia la richiesta al metodo *recommendation* di *BridgeController*. Il flusso procede con la validazione del *Json* ricevuto che estrapola una lista di interazioni dell'utente. Questa lista viene elaborata dall'albero tramite la chiamata al metodo *getRecommendation* di *AlgorithmService* che ritornerà le probabilità delle raccomandazioni.

### 3.5 Verifica e Validazione

Questa sezione descrive l'attività di verifica e validazione svolta durante la realizzazione di Tres, indicando gli strumenti utilizzati, le metriche utilizzate e infine un resoconto finale dei test. L'attività di verifica mi ha permesso di individuare prontamente errori di codifica e ridurre i tempi di sviluppo. Ho cercato di automatizzare il più possibile questa attività per focalizzarmi nello sviluppo. In primis ho effettuato dei test di unità per le componenti più complesse e successivamente ho verificato la loro integrazione. Per motivi di tempo non ho potuto realizzare tutti test di unità e di integrazione per tutte le componenti, mi sono focalizzato sulle componenti principali.

#### Strumenti

Durante lo sviluppo ho utilizzato il più possibile degli automatismi che mi permettessero di fare analisi statica e dinamica risparmiando più tempo possibile. Ho adottato i seguenti strumenti:

- **IntelliJ IDEA:** questo IDE fornisce funzionalità per identificare errori di programmazione come ad esempio sintassi errata, incompatibilità tra tipi e variabili non definite. Fornisce la percentuale di documentazione dell'intero progetto.
- **TravisCI:** è uno strumento che fornisce un automatismo per la compilazione e l'esecuzione dei *test* del progetto. Se la compilazione o almeno un test fallisce, invia una notifica via *email* con la possibilità di consultare il *log*.
- **Codecov:** questo strumento fornisce la percentuale delle linee di codice coperte durante i *test*.
- **Specs2:** è una libreria di *testing* fornita direttamente da Play Framework. Mi ha permesso di testare l'intera applicazione anche con le chiamate HTTP.

#### Metriche adottate

Per la realizzazione di Tres ho adottato le seguenti metriche:

- **Attributi per classe:** un numero troppo elevato di attributi potrebbe indicare la necessità di suddividere la classe in una gerarchia. Ho fissato un *range* ottimale i seguenti valori, un minimo 1 fino ad un massimo di 8;
- **Complessità ciclomatica:** è un indice utilizzato per misurare la complessità di funzioni, metodi, moduli e classi di un programma. Misura direttamente il numero di cammini linearmente indipendenti attraverso il grafo di controllo di flusso. I nodi del grafo corrispondono a gruppi indivisibili di istruzioni, mentre gli archi connettono due nodi se il secondo gruppo di istruzioni può essere eseguito immediatamente dopo il primo gruppo. Per questa metrica ho fissato un *range* ottimale da un minimo di 1 fino ad un massimo di 10;

- **Copertura dei commenti:** è indice della percentuale di classi e metodi corredati di commenti. Ho fissato un range ottimale da 80% a 100%;
- **Copertura dei test:** è una metrica utilizzata per misurare l'efficacia del collaudo. Si utilizza un indice che tiene traccia di quante volte è stata eseguita ogni istruzione durante il *testing*, le istruzioni eseguite almeno una volta sono dette "coperte". L'obiettivo è coprire il maggior numero possibile di istruzioni, in modo da avere una minore quantità di errore. Per questa metrica ho fissato un valore ottimale minimo di 70%.

### 3.5.1 Resoconto risultati

#### Metriche misurate

Qui di seguito sono riportate le metriche software misurate:

- **Complessità ciclomatica:** per il calcolo della complessità ciclomatica ho utilizzato Sbt, al suo interno è disponibile il *tool* styleCheck. Questo strumento segnala con un *warning* se la complessità supera il valore 10. Il *tool* ha restituito esito positivo e non ha segnalato alcun *warning*;
- **Attributi per classe:** ho rilevato per questa metrica come valore medio 2, mentre come valore massimo ho rilevato 4;
- **Copertura commenti:** il valore di copertura è pari al 100%;
- **Copertura dei test:** il valore di copertura dei *test* è del 73%.

#### Test di unità

Qui di seguito sono riportati tutti i *test* di unità eseguiti e il loro relativo esito:

Test	Descrizione	Stato
TU1	Viene verificato che il sistema salvi correttamente un comportamento all'interno del <i>database</i>	Esito positivo
TU2	Viene verificato il corretto caricamento di una lista di <i>widgetTag</i> distinti	Esito positivo
TU3	Viene verificato il corretto caricamento di una lista di <i>item</i> distinti	Esito positivo
TU4	Viene verificato il corretto caricamento di una lista di <i>behavior</i> con una <i>interaction</i> specifica	Esito positivo
TU5	Viene verificato il corretto caricamento di una lista di <i>behavior</i> con <i>tag</i> e <i>action</i> specifici	Esito positivo
TU6	Viene verificato il corretto caricamento di una lista di <i>action</i> distinti	Esito positivo

TU7	Viene verificato il corretto calcolo dell'entropia	Esito positivo
TU8	Viene verificato il corretto calcolo del guadagno di informazione	Esito positivo
TU9	Viene verificato che venga creata la raccomandazione corretta da fornire	Esito positivo
TU10	Viene verificato il corretto calcolo delle probabilità	Esito positivo

**tabella 3.4:** Tabella dei *test* di unità

### Test di integrazione

Qui di seguito sono riportati tutti i *test* di integrazione eseguiti e il loro relativo esito:

Test	Descrizione	Componente	Stato
TI1	Viene verificato che il sistema crei correttamente un albero di decisione	algorithm	Esito positivo
TI2	Viene verificato che il sistema aggiorni l'albero di decisione ogni 24 ore	algorithm	Esito positivo
TI3	Viene verificata la corretta creazione del <i>training set</i>	algorithm	Esito positivo

**tabella 3.5:** Tabella dei *test* di integrazione

### Test di sistema

Ho effettuato alcuni *test* di sistema inerenti alle funzionalità di inserimento dati, calcolo di una raccomandazione e infine il calcolo probabilità.

- **Inserimento dati:** viene verificato il corretto comportamento del sistema alla richiesta di inserimento dati;
- **Calcolo raccomandazioni:** viene verificato il calcolo della raccomandazione da fornire;
- **Calcolo probabilità:** viene verificato il calcolo delle probabilità associate ai vari *item*.

### Sommario funzionamento

TODO

# Capitolo 4

## Valutazione Retrospettiva

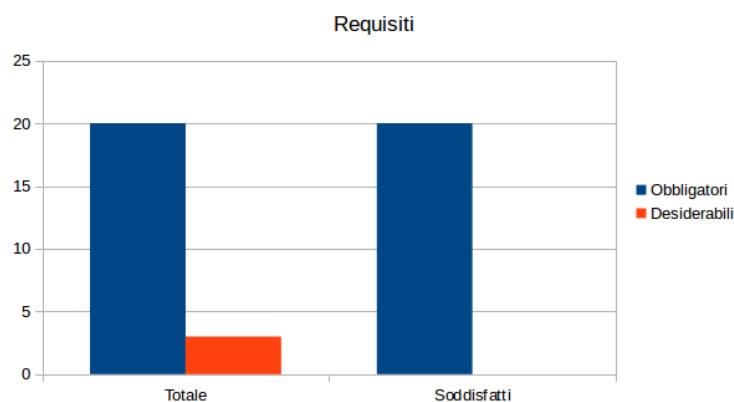
*Questo capitolo riporta un bilancio finale su quanto svolto durante lo stage.*

### 4.1 Bilancio sui risultati

In questa sezione riassumo gli obiettivi aziendali e gli obiettivi personali raggiunti durante lo stage.

#### 4.1.1 Obiettivi conseguiti

Gli obiettivi fissati all'inizio dello stage hanno subito delle modifiche perché l'azienda ha voluto privilegiare il porting di DRE. Il *porting* dell'applicativo è parzialmente completato, l'unica funzionalità non ancora implementata una procedura di *map-reduce*. Quest'ultima funzionalità non l'ho completata perché implementata con la tecnologia javascript da eseguire all'interno di MongoDB, le mie competenze per comprendere quel codice erano insufficienti ed avrebbero introdotto un ulteriore ritardo per lo sviluppo di Tres inoltre l'obiettivo di migliorare la fase di apprendimento di questo modulo non l'ho raggiunto. Durante lo sviluppo di Tres ho individuato 23 requisiti di cui 20 obbligatori e 3 desiderabili.



**figura 4.1:** Riassunto requisiti

I requisiti non soddisfatti riguardano direttamente gli obiettivi (e quindi non ho raggiunto) di implementazione di algoritmi di *clustering* e di implementazione di algoritmi per l'individuazione dei gusti dell'utente. Tuttavia quest'ultimo obiettivo l'ho centrato grazie all'implementazione di ID3 per fornire una raccomandazione sulla base del comportamento utente. Infine in questa attività ho soddisfatto l'obiettivo minimo riguardante l'utilizzo di OrientDb.

#### 4.1.2 Obiettivi personali

All'inizio dello stage il mio obiettivo era quello di imparare nuove tecnologie da aggiungere alle mie conoscenze perché ritenevo il mio bagaglio personale insufficiente per affrontare il mondo del lavoro. Questo obiettivo è stato certamente raggiunto, infatti lo stage mi ha permesso di padroneggiare ad un buon livello tecnologie quali OrientDb, web framework di concezione moderna e Scala. Quest'ultimo mi ha dato la possibilità di apprendere le nozioni per un corretto stile di programmazione funzionale, che era uno degli obiettivi prefissatomi. Ritengo l'obiettivo riguardante l'approfondimento di argomenti quali, intelligenza artificiale e i sistemi di raccomandazione, parzialmente soddisfatto. Durante lo stage non ho trovato il supporto necessario per approfondire queste competenze, soprattutto per quanto riguarda i sistemi di raccomandazione visto la natura del progetto di stage.

### 4.2 Bilancio formativo

In questa sezione viene riportato le competenze acquisite durante lo stage.

### 4.3 Distanza tra formazione universitaria e lavoro

In questa sezione viene esposto la valutazione personale della distanza tra la formazione ricevuta durante il corso di studi e lo stage formativo.

## Appendice A

### Appendice A

Citazione

---

Autore della citazione



# Bibliografia

Riferimenti bibliografici

Siti Web consultati