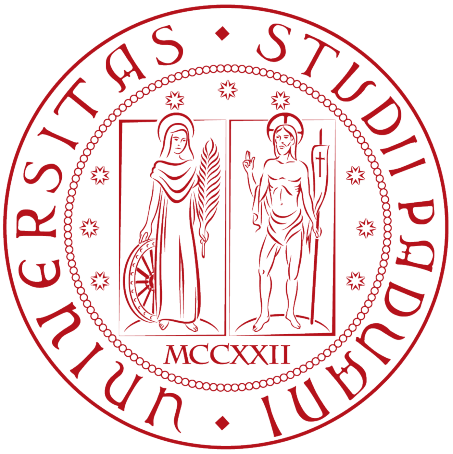
Università degli Studi di Padova

Dipartimento di Matematica

Corso di Laurea in Informatica



Sviluppo di applicazioni native per iOS in JavaScript

Tesi di laurea triennale

Relatore

Prof. Claudio Enrico Palazzi

Laureando

Giacomo Manzoli

Anno Accademico 2014-2015

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di circa trecento ore, dal laureando Giacomo Manzolipresso l'azienda WARDA S.r.l.. L'obiettivo di tale attività di stage è l'analisi dei vari framework disponibili per lo sviluppo di applicazioni native utilizzando il linguaggio JavaScript, al fine di creare un'applicazione nativa per iOS simili alla gallery sviluppata dall'azienda.

"Fuck it, i did it"

— Confucius

Ringraziamenti

Bla bla bla

Bla bla bla

Bla bla bla

Even more bla bla bla

...

Padova, Dec 2015 Giacomo Manzoli

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Il contesto aziendale

1.1 L'azienda CoffeeStrap inc.

CoffeeStrap inc[1]. è un'azienda start-up in ambito web/mobile nata nei primi mesi del 2013 dalla collaborazione di Mahesh Casiraghi ed Alessandro Maccagnan (il quale nell'ambito dello stage assume il ruolo di *tutor aziendale*), che attualmente ricoprono al suo interno rispettivamente i ruoli di CEO e CTO. Dal mese di Aprile del 2014 inoltre si è aggiunto al team un ulteriore componente, Milo Ertola, che attualmente ricopre il ruolo di CPO e si occupa dello sviluppo web.

Maccagnan ed Ertola, inoltre, hanno frequentato il corso di laurea in informatica presso l'Università degli studi di Padova, iscrivendosi nel 2001 e terminando il loro percorso nel 2004.

figura 1.1: Logo di CoffeeStrap inc.

La start-up nasce con sede operativa a San Francisco, in cui ha trascorso il suo primo anno di sviluppo, per poi spostarsi all'inizio del 2014 nel cuore Amsterdam, partecipando al programma dell'acceleratore Rockstart[2], che fornisce attualmente supporto diretto all'azienda, mettendo a disposizione, oltre alle postazioni di lavoro (è infatti anche incubatore), un ambiente fresco e innovativo, circondato da esperti del settore e a stretto e costante contatto con il mondo esterno e con la rete degli investitori. Rockstart possiede al suo interno inoltre un gruppo di **mentori**, che mettono a disposizione la loro esperienza e le loro capacità, fornendo assistenza diretta in diversi ambiti. Questo tipo di figura è molto importanti in un ambiente di questo tipo, in quanto le aziende, essendo alle origini, hanno bisogno di essere indirizzate nella giusta direzione per poter sviluppare il proprio prodotto in modo efficace e ricevere dunque finanziamenti. La sede operativa contestualmente al mio stage è stata appunto quest'ultima, nella quale ho potuto conoscere il mondo delle start-up ed entrare in contatto con una comunità di sviluppatori, esperti di business e mentori che hanno contribuito alla mia crescita sia professionale che personale.

[1] http://www.coffeestrap.com/
[2] http://www.rockstart.com/accelerator/

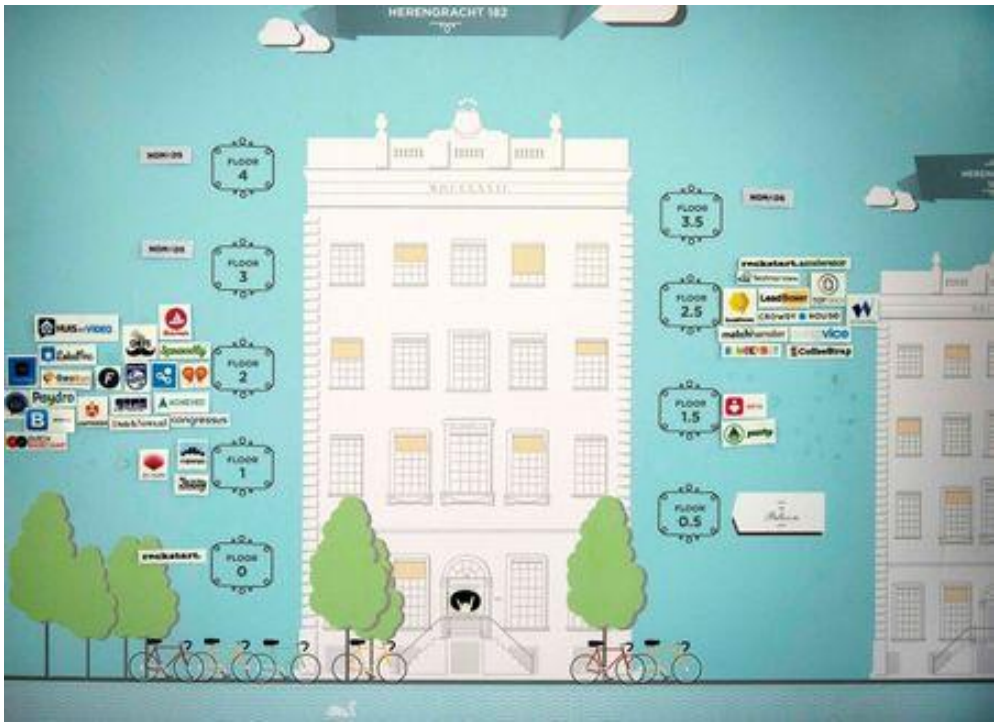


figura 1.2: Organizzazione dell'edificio di Rockstart

Rockstart è suddiviso in quattro sezioni principali, ciascuno con il proprio scopo e modalità operative:

- **Rockstart accelerator**, che fornisce un programma di accelerazione di 150 giorni, offrendo inoltre un finanziamento iniziale in cambio di una percentuale di quota societaria. Esso si suddivide a sua volta in due sotto-branchie:
  - **Web e mobile accelerator program**, focalizzato sulle start up in ambito web/mobile. È esattamente in questa branchia in cui l'azienda CoffeeStrap colloca la propria posizione all'interno di Rockstart;
  - **Smart energy**, che fornisce supporto nelle start-up che mettono in correlazione l'*information technology* con il problema del risparmio energetico. Il loro programma differisce leggermente rispetto al precedente.
- **Rockstart spaces**, che si occupa della gestione degli uffici e degli eventi all'interno di Rockstart. Quest'ultimo ospita infatti al suo interno numerosi eventi nella spaziosa **ballroom**, molto spesso organizzati da entità al di fuori di Rockstart;
- **Rockstart answers**, che organizza eventi mirati a creare una sorta di comunità all'interno di Rockstart nella quale tutti possono confrontarsi tra di loro ed incontrarsi ad eventi organizzati *ad-hoc* che trattano diversi argomenti o danno la possibilità alle start-up di effettuare presentazioni;
- **Rockstart impact**, che fornisce supporto alle start-up nei paesi in via di sviluppo, in particolare in Nepal.

figura 1.3: La ballroom a Rockstart

1.1.1 Il contesto della start-up

L'azienda ospitante è una particolare realtà aziendale, l'impresa è nella sua fase iniziale e, come la maggior parte delle start-up, parte con un'idea di riferimento, cercando di rendere profittevole quest'ultima tramite veloci iterazioni sul prodotto. L'incubatore all'interno del quale lavoravo incapsula questo mondo in maniera molto significativa. Sono entrato in un ambiente molto personalizzante, nel quale si è a stretto contatto con il team (essendo quest'ultimo inoltre di piccole dimensioni) e le altre aziende, con le quali si è creata un'implicita rete collaborativa, dalla quale è possibile imparare molto e confrontarsi quotidianamente. Tale collaborazione fornisce valore al prodotto e all'azienda, in quanto permette, oltre ad un importante feedback sulle iterazioni, anche una quotidiana attività di brainstorming, che permette di visualizzare i problemi sotto punti di vista differenti e ad arrivare così ad una migliore soluzione.

1.2 Il dominio applicativo

L'idea di CoffeeStrap è quella di fornire un sistema peer-to-peer per l'apprendimento di una lingua straniera tramite interazioni con altre persone che parlano quella determinata lingua. L'intento è quello di creare una comunità di utenti che vogliano imparare una lingua e siano allo stesso tempo disposti a fornire supporto ad altri utenti con le lingue cui sono già fluenti. L'interazione può avvenire tramite **comunicazione testuale**, in cui viene messa a disposizione una vera e propria *chat*, oppure tramite **comunicazione audio/video**, in cui gli utenti parlano tra di loro nella lingua concordata. Gli utenti hanno la possibilità di iscriversi tramite email o tramite facebook. Il profilo utente è caratterizzato dai seguenti campi:

- Nome;
- Cognome;
- Età (opzionale);
- Immagine profilo;
- Città di provenienza;
- Lingue conosciute;
- Lingue da imparare.

Una volta completato il profilo l'utente inizia a parlare una lingua passando attraverso un flusso:

- L'utente X seleziona la lingua Y che vuole praticare;
- Il sistema cerca all'interno della comunità un sottoinsieme Z di persone che parlano la lingua Y (come lingua madre o seconda lingua);
- Il sistema notifica quest'ultimo, informando gli utenti che l'utente X vuole praticare la lingua Y;
- Gli utenti del sottoinsieme Z decidono se accettare o meno la richiesta dell'utente X di praticare la lingua Y;
- Se l'utente accetta la richiesta allora viene creato un **match** tra quest'ultimo e l'utente X, ed essi iniziano un'interazione.

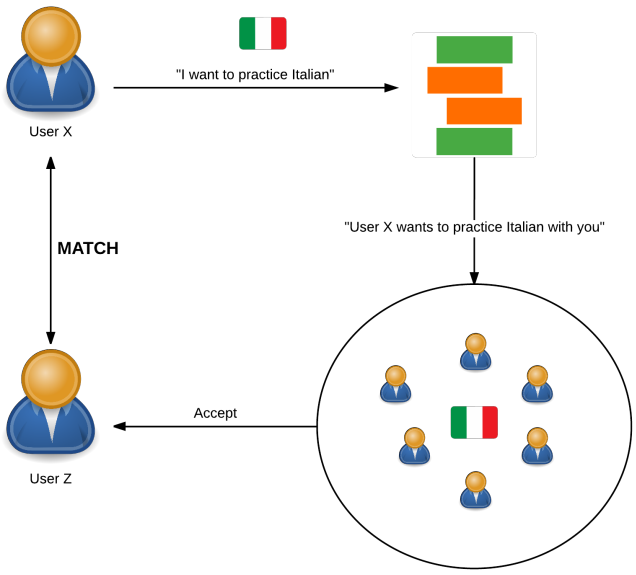


figura 1.4: Rappresentazione del flusso utente di CoffeeStrap

Gli utenti vengono notificati tramite **notifica email** e, in seguito allo sviluppo dell'applicazione, **notifica mobile**.

1.3 Metodologie e strumenti di sviluppo

CoffeeStrap adotta la metodologia di sviluppo **agile**[3]. Il particolare contesto entro il quale si trova fa sì che i requisiti cambiano molto spesso e porta dunque quest'ultima a dover iterare molto velocemente sul proprio prodotto, che si trova ancora nella sua fase primordiale e necessita di modifiche e miglioramenti costanti. Proprio a fronte di queste necessità è importante mantenere la massima flessibilità possibile nello sviluppo, motivo per cui questo modello di ciclo di vita è in questo momento il più conveniente in termini di efficienza ed efficacia. In particolare la metodologia adottata è quella **agile scrum**.

Il tipo di iterazione utilizzata è su base settimanale. Una volta a settimana, normalmente il Mercoledì, avviene lo **scrum meeting**, una riunione della durata di circa due ore nella quale si fissano **story** da risolvere nello **sprint** corrente. Una story è un concetto molto simile a quello di **ticket**, ed è composta dai seguenti campi:

- **Titolo**;
- **Tipologia**, che può assumere i seguenti valori:
  - **Feature**, quando c'è da aggiungere un "pezzo" al sistema;
  - **Bug**, quando c'è da risolvere una problematica all'interno del sistema;
  - **Chore**, quando c'è da eseguire *refactoring* e manutenzione del codice;
  - **Release**, sono marcatori di *milestone* che tracciano il progresso del team rispetto agli obiettivi.
- **Descrizione**, che fornisce una dettagliata sintesi della story.
- Una o più **etichette**, che forniscono un filtro;
- **Complessità**, che è un indice soggettivo di quante risorse sono necessarie per il completamento della story. Questo valore segue la sequenza di Fibonacci e può assumere i valori 1, 2, 5, 8, 13. Se una story ha una complessità superiore a 13 essa dev'essere necessariamente spezzata in due o più sotto-story. Solamente alle features può essere assegnata complessità, in quanto sono le uniche story che realmente forniscono un valore al business;
- **Richiedente**, che dev'essere unico;
- Uno o più **responsabili** (*owners*), che possono coincidere con il richiedente e sono coloro che devono occuparsi direttamente della story;

Opzionalmente inoltre è possibile aggiungere alla story dei tasks, in modo da dividere ulteriormente il lavoro. Ciononostante su di essi non vi è alcun controllo, in quanto il loro utilizzo è a discrezione dei responsabili.

Ciascuna story è inserita all'interno di una **board**. Ci sono tre tipologie di board:

- **Current**, contenente le story attive nello sprint corrente;

[3] http://agilemanifesto.org/

Titolo	Login/registrazione con Facebook
Tipologia	Feature
Descrizione	È necessario interfacciarsi con le API di facebook per Android per poter implementare il sistema di login/registrazione tramite facebook. Una volta ottenuto il token di accesso l'applicazione deve effettuare una chiamata API a CoffeeStrap fornendo quest'ultimo e memorizzando il cookie restituito dalla chiamata.
Etichette	`Mobile`
Complessità	8
Richiedente	Alessandro Maccagnan
Responsabili	Luca De Franceschi

tabella 1.1: Un esempio di story

- **Backlog**, contenente la "coda" delle story da attivare;
- **Icebox**, contente story in attesa di approvazione e attivazione. Le story possono rimanere su questa board per un tempo indeterminato, finché vengono spostate sul backlog o rimosse definitivamente.

A ciascun membro del team vengono assegnate una o più story, la cui scadenza è lo scrum meeting successivo. A quest'ultimo viene richiesto inoltre di assegnare la complessità a quella story, la quale dovrà essere calcolata secondo criteri derivati dalla propria esperienza di sviluppo. Ogni story dev'essere analizzata, progettata, sviluppata e infine testata. La chiusura della story viene effettuata solamente al seguito di una verifica e validazione da parte del responsabile e del richiedente o, se il richiedente coincide con il responsabile, da un altro membro del team. In generale i responsabili non possono approvare le proprie story.

Una volta creata la story deve passare attraverso diverse fasi, per questo si parla di **story workflow**:

1. La story viene **creata** e **definita** (*define and design*);
2. I responsabili assegnano loro una **stima di complessità**, che va generalmente discussa all'interno del team (*estimate*);
3. La story viene **attivata**: da questo momento i responsabili possono procedere allo sviluppo (*start*);
4. La story una volta implementata viene **testata** (*test*);
5. Superata la fase di testing la story viene **ultimata** (*finish*);
6. A questo punto i responsabili "pushano" il codice sull'ambiente di **staging** (*deliver*);
7. La story viene poi **accettata** dal richiedente o comunque da una persona esterna al gruppo di responsabili e viene effettuato il pushing sull'ambiente di **production** (*accept*). Se la story non viene accettata si itera dal punto 4;
8. Infine, se la story è stata accettata avviene il suo **rilascio**, che coincide con la terminazione del suo ciclo di vita (*release*).

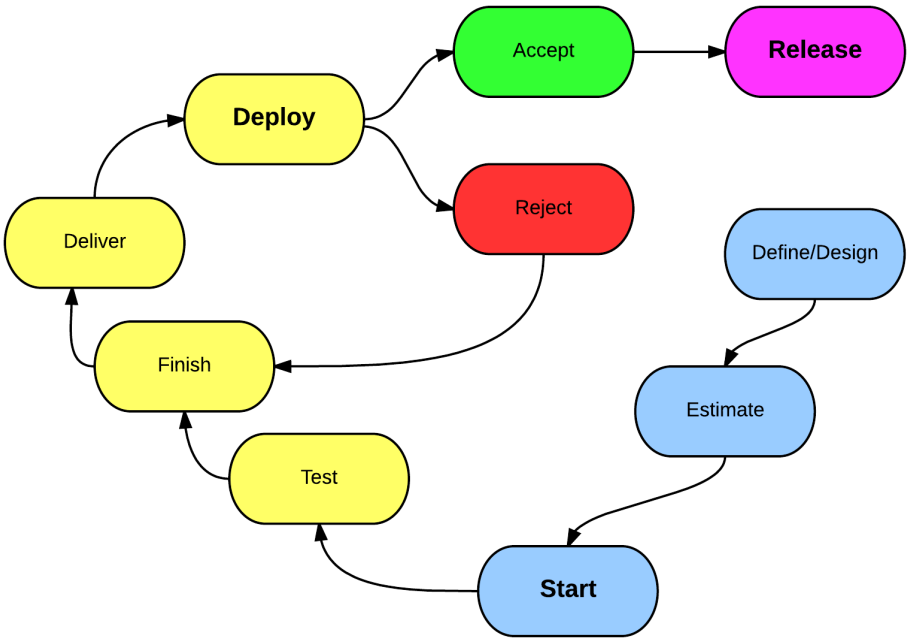


figura 1.5: Il flusso di una story

Alla fine dello sprint viene tracciata la somma delle complessità di tutte le story assegnate a tutti i membri e che sono state chiuse nello sprint corrente. Tale somma è detta **velocity**, ed è un indice di efficienza di sviluppo dello sprint, che va confrontato con gli sprint precedenti.

Oltre allo scrum meeting viene instanziato quotidianamente lo **stand-up meeting**, detto anche **daily scrum**, nel quale prima dell'inizio della giornata lavorativa si effettua una breve riunione con il team, in cui ciascuno espone quanto fatto nel giorno precedente e ciò che ha in programma di fare nel giorno corrente. Quest'attività è di fondamentale importanza per **monitorare** il progresso dello sprint e sollevare eventuali difficoltà riscontrate nello svolgimento della story.

1.4 Tecnologie

1.4.1 Back-end

Il back-end è composto da un server scritto tramite il framework Node.js[4]. Questa scelta è dovuta alla necessità di ottimizzare i tempi di produzione e manutenzione del software e la grande potenza e scalabilità di quest'ultimo viene incontro a questo bisogno. La **programmazione asincrona** e il motore *JavascriptV8* garantiscono una velocità di trasmissione molto elevata e le applicazioni web scritte con *express.js*[5] sono in grado di reggere un alto carico di lavoro, sostenendo centinaia se non migliaia di richieste simultanee su macchine che non hanno a disposizione molte risorse di memoria. Inoltre

[4] http://nodejs.org/
[5] http://expressjs.com/

Node.js è un linguaggio di programmazione all'avanguardia, in costante aggiornamento, la cui comunità è in forte espansione. Tramite **npm**[6], gestore di pacchetti per Node.js, è possibile attingere ad una vasta quantità di librerie utili per ogni evenienza.

Il server fornisce tutta una serie completa di **API RESTful** accessibili dall'esterno, alcune delle quali richiedono autenticazione. Ciò permette a tutti i client che si interfacciano con esso, in particolare l'applicazione web e l'applicazione mobile, di poter comunicare con esso in modo semplice e lineare, dovendo riferirsi solamente alle specifiche di queste ultime ed utilizzando una libreria per la gestione delle chiamate API.

Parallelamente al server è stato instanziato un servizio di **pushing** fornito da **firebase**[7], un **Cloud Service Provider** utilizzato per il sistema di notifiche e per fornire un punto d'accesso centralizzato dal quale i diversi componenti (intesi come componenti software) possono comunicare. Firebase fornisce infatti un **database in tempo reale**, con il quale è possibile interagire e "*mettersi in ascolto*" degli eventi su di esso. Esso fornisce un'API pubblica per diversi stack tecnologici e piattaforme.

1.4.2 Front-end

Il front-end è composto da un'applicazione web scritta con **Angular.js**[8] e **HTML5**, utilizzando il framework CSS **Bootstrap**[9]. Tramite Angular.js è possibile interfacciarsi alle API fornite dal back-end in maniera molto efficiente e popolare dunque le pagine in base alle risposte provenienti da esso. Inoltre, analogamente ad *npm*, per lo sviluppo di applicazioni con Angular.js è disponibile **bower**[10], gestore di pacchetti per quest'ultimo.

1.4.3 Database

CoffeeStrap utilizza **MongoDB**[11] come database documentale di riferimento. Questa scelta è dovuta alla grande flessibilità offerta dai database di questo tipo, oltre al fatto che negli ultimi mesi è divenuto molto popolare nella comunità di sviluppatori e presenta prestazioni notevoli.

Dal momento che i requisiti cambiano molto rapidamente è necessario avvalersi di strumenti che siano il più flessibili possibili ai cambiamenti, e MongoDB risponde perfettamente a questo tipo di esigenza. Inoltre MongoDB presenta un facile approccio al suo utilizzo sia per l'utilizzo da parte dello sviluppatore che in termini di scalabilità.

1.4.4 Ambienti di sviluppo

All'interno dell'azienda si delineano principalmente due ambienti di sviluppo:

- **Production**, all'interno del quale risiedono le versioni ufficiali del codice esposte pubblicamente agli utenti;
- **Staging**, un ambiente riservato esclusivamente al team e che rappresenta una sorta di "cantiere", nel quale risiede una versione del codice ancora da verificare, che può quindi essere testata liberamente e non crea danni verso l'esterno.

[6] https://www.npmjs.com/
[7] https://www.firebase.com/
[8] https://angularjs.org/
[9] http://getbootstrap.com/
[10] http://bower.io/
[11] http://www.mongodb.org/

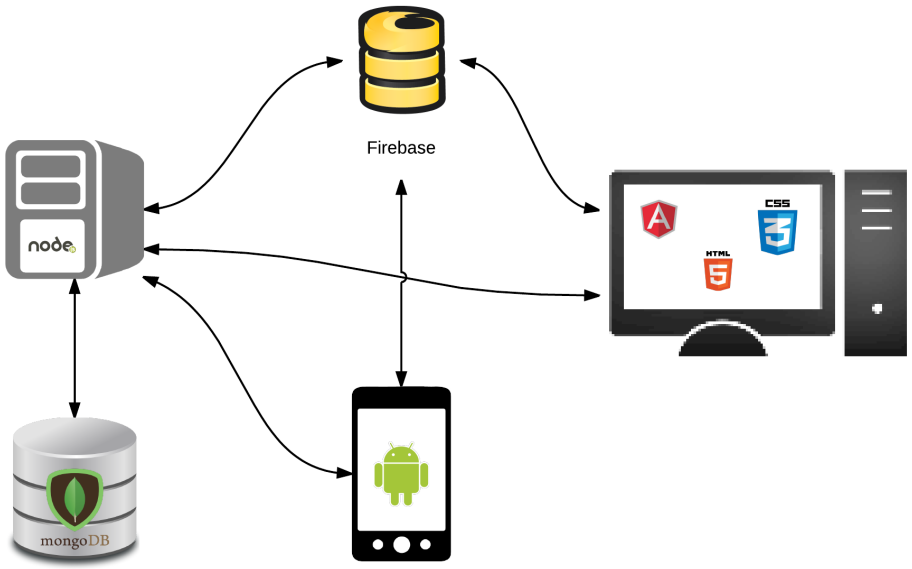


figura 1.6: Composizione della tecnologia adottata da CoffeeStrap

Ciascun ambiente è caratterizzato dalla propria configurazione, possiede il proprio server e il proprio database ed è agnostico rispetto all'altro.

Occasionalmente è inoltre uso comune instanziare un ambiente ausiliario di sviluppo, per circostanze particolari che necessitano di una configurazione diversa. Solitamente sono ambienti "*usa e getta*", che nascono da esigenze particolare e vengono rimossi alla fine del loro utilizzo.

1.4.5 Versionamento

Come strumento di controllo versione del codice sorgente l'azienda utilizza **Git**[12], utilizzando **Bitbucket**[13] come servizio di hosting. I motivi che hanno spinto all'utilizzo di Git in luogo di altri sistemi di versionamento sono i seguenti:

- **Disponibilità**: ciascun sviluppatore ha una copia locale dell'intero repository ed effettua *commit* in locale, potendo dunque lavorare anche in assenza di connessione;
- **Semplicità**: oltre ad essere veloce Git permette con facilità la creazione di *branch* e successivi *merge*. L'utilizzo di questi ultimi è fondamentale per poter soddisfare il punto seguente:
- **Utilizzo di Gitflow**: questo **workflow** definisce uno stretto modello di *branching* che ruota intorno al progetto. Ciò non aggiunge nuovi concetti o strumenti a Git, ma piuttosto assegna ruoli specifici ai diversi rami. Un approfondimento di questa metodologia è raccolta nell'appendice A.

[12] http://git-scm.com/
[13] https://bitbucket.org/

1.5 Clientela di riferimento

CoffeeStrap non ha ancora acquisito finanziamenti da parte di investitori e non presenta associazioni con altre aziende o entità, motivo per cui essi devono rispondere unicamente agli **utenti finali** del prodotto. Questi ultimi sono persone reali provenienti da ogni parte del mondo, che mettono a disposizione le proprie competenze linguistiche e vogliono allo stesso tempo imparare una lingua straniera entrando in contatto con altri utenti. L'intera struttura di CoffeeStrap si basa su questa **rete sociale**, composta da un insieme eterogeneo di persone che espongono il proprio profilo all'esterno. È possibile immaginare CoffeeStrap come un vero e proprio *social network* il cui scopo è l'apprendimento linguistico.

figura 1.7: Distribuzione degli utenti di CoffeeStrap nel mondo aggiornata a Dicembre 2014

Capitolo 2

Il progetto all'interno dell'azienda

2.1 Presentazione del progetto

Il progetto svolto da me durante l'attività di stage consiste nello sviluppo di un'applicazione mobile, sviluppata nativamente in Android, che replica il meccanismo e il flusso utente già esistente nel web integrandolo all'interno di questo tipo di piattaforma.

2.1.1 Motivazioni

Come già presentato nel capitolo 1, CoffeeStrap ha come primo e unico scopo lo sviluppo di un sistema peer-to-peer per l'apprendimento di una lingua tramite interazioni tra utenti. Lo scopo principale in questa prima fase di sviluppo consiste nels raccogliere un bacino utenti considerevole, in modo da poter capire su quale linea portare avanti l'idea e modellare quest'ultima in base al *feedback* e al tipo di iterazione dell'utente medio. Prima dell'inizio del mio stage CoffeeStrap disponeva solamente dell'applicazione web, la quale possedeva circa 2000 iscritti e mediamente 15 utenti attivi al giorno.

Il progetto di stage che ho svolto nasce dunque dalla necessità di affiancare all'applicazione web l'applicazione mobile, in modo da poter fornire all'utente più possibilità di utilizzo del prodotto ed arrivare inoltre ad un considerevole incremento del bacino utenti. La considerazione principale a monte di ciò è il fatto che oggigiorno l'utente medio spende la maggior parte del proprio tempo sul proprio smartphone piuttosto che sul proprio PC, per tutti i vantaggi legati alla tecnologia mobile. È di fatto una necessità imprescindibile per una start-up in ambito *web/mobile* estendersi su più piattaforme se si vuole competere in un certo tipo di mercato ed ottenere dei finanziamenti. In aggiunta CoffeeStrap può essere categorizzato di fatto come *social network*, ulteriore motivo per il quale sviluppare il prodotto su mobile è di importanza fondamentale.

L'avere a disposizione l'applicazione a qualsiasi ora del giorno e in qualsiasi luogo permette all'utente di fruire del prodotto senza limitazioni e nel momento esatto in cui ne sente la necessità. Da questo ne deriva un aumento del numero di iterazioni e della quantità di utenti attivi giornalieri, dati di fondamentale importanza per gli investitori.

Possiamo sintetizzare in cinque punti i vantaggi per cui un'azienda dovrebbe avere un'applicazione mobile:

1. **Visibilità**: i dati rilevati indicano che le ricerche all'interno degli *App stores* siano molto elevate, motivo per cui comparire tra i risultati di ricerca rappresenta un'ottima **pubblicità** per l'azienda;
2. **Aumento del valore del brand**: avere la propria applicazione è indice di un'azienda *dinamica*, sempre attenta all'evoluzione tecnologica e al passo coi tempi. Questi valori aumentano di molto il valore dell'azienda;
3. **Copertura mediatica**: con la tecnologia mobile è possibile coprire anche zone con scarsa connettività ad internet, coprendo una zona molto maggiore ed aprendo nuove possibilità per il business;
4. **Crescita del mercato mobile**: il mondo mobile è in grossa crescita e prospera più velocemente rispetto al web. Lo smartphone è al giorno d'oggi uno strumento essenziale per qualsiasi persona, sia per il tempo libero che per il lavoro.
5. **Capacità di differenziarsi dai concorrenti**: le aziende concorrenti che non dispongono di un'applicazione mobile passano in secondo piano rispetto ad un'azienda che ne dispone.

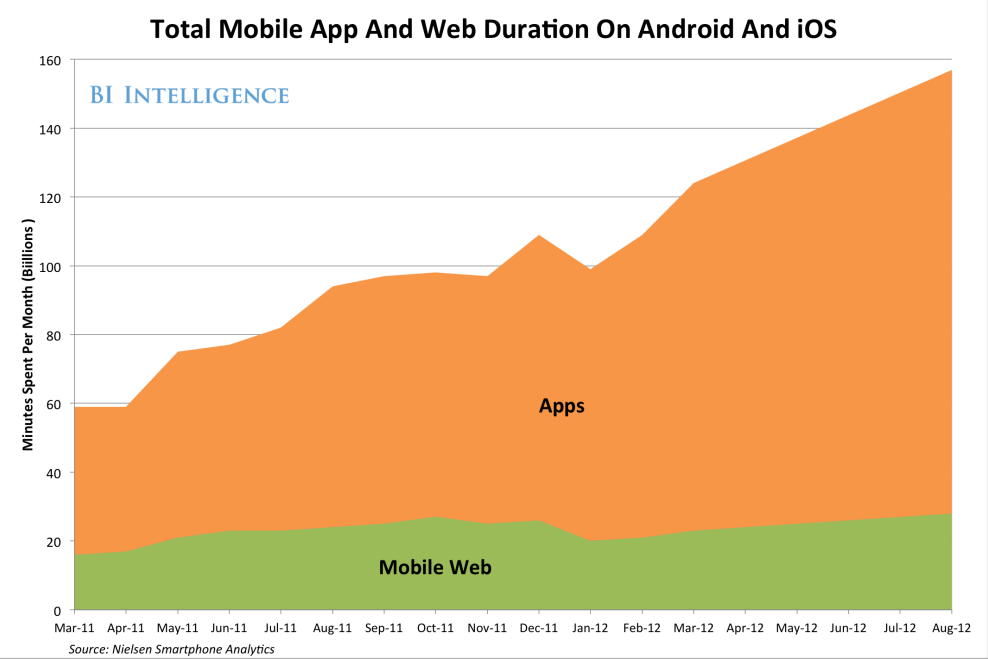


figura 2.1: Raffronto di traffico web mobile e mobile app tra il 2011 e il 2012.

Fonte: http://waracle.net/exploring-the-future-of-mobile-app-development-html5-vs-native/

2.1.2 Aspettative

La mia preparazione e formazione in ambito di sviluppo mobile poteva ritenersi pari a zero all'inizio dell'attività di stage. Ciononostante l'azienda era consapevole di quali fossero le mie capacità, avendo già collaborato con me e il mio *team* per il

progetto di ingegneria del software dell'anno accademico 2013/2014. Pochi mesi prima, in collaborazione con il team **steakholders**, avevo infatti sviluppato un framework per l'amministrazione di dati provenienti da MongoDB. CoffeeStrap ha assunto il ruolo di **committente** all'interno di questo progetto, quindi vi erano state diverse interazioni tra me e l'azienda, la quale tuttora utilizza quotidianamente come strumento di supporto. Il progetto in questione si chiama **MaaP**, rilasciato sotto licenza MIT, disponibile come pacchetto all'interno di npm[1]. Inoltre su github[2] è disponibile il codice sorgente.

A fronte della realizzazione di questo progetto l'azienda poneva in me delle buone aspettative circa le tempistiche di apprendimento della tecnologia e di integrazione all'interno del team, per cui il **rischio** di ospitare un'attività di stage era mitigato da questi fattori. Inoltre i membri di CoffeeStrap, avendo frequentato anch'essi il mio stesso corso di studi, erano al corrente di quale tipo di formazione l'università di Padova offrisse agli studenti e in particolare in che modo e secondo quali prassi il corso di *Ingegneria del Software* preparasse questi ultimi ad essere inseriti velocemente nel mondo del lavoro, avendo alle spalle un'esperienza corposa ed istruttiva. Queste motivazioni hanno fatto sì che l'azienda decidesse di ospitare al loro interno un'attività di stage che potesse soddisfare gli obiettivi preposti.

Considerando la durata dello stage e le tempistiche legate all'apprendimento della tecnologia, l'aspettativa da parte dell'azienda era quella di arrivare alla fine del periodo di stage con in mano un **prototipo funzionante**, pronto per essere inserito in una fase successiva di test più approfondito e di raffinamento dei componenti, comunque non ancora sufficientemente maturo per essere inserito pubblicamente nel mercato. L'aspettativa era quella di una versione dell'applicazione che in ambito informatico viene generalmente indicata come *beta*.

Gli obiettivi minimi delineati sono i seguenti:

- *Signup* e *login* all'applicazione da parte dell'utente;
- Visualizzazione e modifica delle preferenze da parte dell'utente;
- Comunicazione audio;
- Sistema di notifica, quest'ultimo punto fondamentale all'interno del progetto;

Oltre agli obiettivi minimi inoltre sono stati posti degli obiettivi massimi:

- Comunicazione testuale;
- Comunicazione video;

Durante il corso dell'attività di stage ciononostante i requisiti sono aumentati, in base alla necessità di realizzare appieno il flusso previsto, il quale è stato riprogettato nel medesimo periodo. Gli obiettivi sono dunque rimasti invariati ma sono stati aggiunti i seguenti:

- Richiesta di un **match** in base a una lingua, dove per *match* si intende un *collegamento* tra due utenti mantenibile nel tempo;
- Notifica di invito da parte di un utente di parlare una lingua;

[1] https://www.npmjs.com/package/maap
[2] https://github.com/steakholders/maap

- Creazione e mantenimento della lista di match;

Questi ultimi sono stati necessari per arrivare ad un'applicazione con un grado di usabilità accettabile.

2.1.3 Obiettivi di formazione

L'azienda, essendo di piccole dimensioni e in fase primordiale, ha come prima necessità, oltre allo sviluppo e diffusione del prodotto, la formazione di un team qualificato con all'interno elementi di natura tecnica. Nelle aziende di questo tipo la prima necessità e risorsa sono le **persone**, in quanto il valore dell'azienda deriva dal lavoro dei suoi dipendenti. A fronte di questa necessità ospitare un'attività di stage permette all'azienda di poter integrare personale giovane, con una buona formazione e dotati di forte motivazione e stimoli, oltre che una buona elasticità mentale.

Tra gli obiettivi formativi si delineano i seguenti:

- Apprendimento della tecnologia mobile, in particolare introduzione all'utilizzo della piattaforma Android;
- Apprendimento del protocollo REST e comunicazione lato server tramite API;
- Formazione sullo sviluppo di servizi di terze parti utilizzate dall'azienda;
- Introduzione alla metodologia agile scrum;
- Introduzione al contesto della start-up e alla sua particolare realtà.
- Comprensione dei limiti e delle potenzialità delle tecnologie utilizzate.

2.2 Vincoli

2.2.1 Vincoli tecnologici

All'inizio dello stage è stata fatta una valutazione riguardo lo *stack tecnologico* da adottare per la realizzazione dell'applicazione. Sono state delineate tre possibilità:

- **Titanium**[3], un framework di sviluppo mobile basato su *javascript* e *HTML5*, che permette lo sviluppo su più piattaforme;
- **Ionic**[4], altro framework ibrido molto simile al precedente;
- **Nativo Android**, in cui viene messo a disposizione un vero e proprio SDK ufficiale[5] per lo sviluppo, basato su Java.

Il vantaggio dei primi due è quello legato all'utilizzo di questo genere di framework, ovvero la possibilità di effettuare un *delivery* rapido e multi piattaforma di un'applicazione. Il codice, infatti, si scrive velocemente con un meta linguaggio ed attraverso librerie esterne precompilate esso viene adattato per i diversi sistemi operativi in fase di compilazione.

[3] http://www.appcelerator.com/titanium/
[4] http://ionicframework.com/
[5] http://developer.android.com/sdk/index.html

D'altra parte una **criticità** di questi framework consiste nella loro rigidità in fase progettuale: il loro utilizzo, infatti, limita in parte la possibilità di sfruttare le caratteristiche native del *device*, delle linee guida dell'interfaccia e del sistema operativo su cui avverrà il *deployement* dell'applicazione. Spesso dunque si corre il rischio di non arrivare ad un prodotto conforme alle attese con un'interfaccia utente poco ergonomica e intuitiva. Inoltre non è garantito un supporto su lungo periodo alle medesime condizioni. Alcuni framework tendono infatti a non essere più supportati dopo un lungo periodo e il rischio è quello di dover riprogettare tutto daccapo.

Sviluppare un'applicazione nativa garantisce quindi maggiore efficienza in fase di sviluppo e progettazione e porta ad un risultato migliore in termini di usabilità. Chiaramente lo svantaggio è quello di poter supportare un solo sistema operativo, motivo per cui in un futuro CoffeeStrap dovrà ad esempio instanziare un nuovo progetto per l'applicazione *iOS*, molto richiesta dal mercato.

Alla fine la scelta è ricaduta proprio sullo sviluppo nativo, quindi il vincolo tecnologico principale è stato lo sviluppo su piattaforma Android, utilizzando l'SDK ufficiale, ovvero il vero e proprio pacchetto di strumenti che permette lo sviluppo su questa piattaforma. Esso viene normalmente scaricato all'interno di ambienti di sviluppo integrati (IDE), in particolare *Eclipse*[6] e *Android Studio*. Non è stato posto un vincolo circa la scelta di quest'ultimo, che è stata totalmente a carico mio. Questi ambienti mettono a disposizione, oltre ad un ambiente di programmazione, anche strumenti per il design delle diverse schermate, che viene realizzato utilizzando il linguaggio *XML*.

È stata inoltre richiesta l'integrazione all'interno dell'applicazione di tutti i servizi utilizzati dall'azienda per i diversi requisiti:

- Utilizzo dell'**SDK di facebook**[7] per il login e signup all'applicazione da parte dell'utente;
- Utilizzo della libreria di **firebase** per il sistema di *pushing*;
- Utilizzo di una libreria per la gestione delle chiamate API al server, la quale è stata scelta e discussa durante la fase di progettazione;
- Utilizzo della libreria **bugsnag**, per il servizio di *bug tracking*;
- Utilizzo della libreria **opentok**[8] per il servizio WebRTC di comunicazione audio/video;
- Utilizzo della libreria di **mixpanel**[9] per la piattaforma di *web/mobile analysis*.

2.2.2 Vincoli metodologici

CoffeeStrap ha ritenuto importante il mio inserimento all'interno del team e l'apprendimento da parte mia delle loro metodologie di lavoro, pertanto fin dalla prima settimana sono stato inserito all'interno dei meccanismi di sviluppo agile, ho partecipato attivamente agli **scrum meeting** e agli **stand-up meeting** e ho creato e svolto le mie story con gli stessi procedimenti attuati dagli altri membri.

Gli scrum meeting, decisamente i più importanti, si svolgevano normalmente il mercoledì mattina dalle 10:00 alle 13:00 e la loro frequenza era da ritenersi *obbligatoria*

[6] https://eclipse.org/
[7] https://developers.facebook.com/docs/android/getting-started?locale=it_IT
[8] https://tokbox.com/opentok/
[9] https://mixpanel.com/

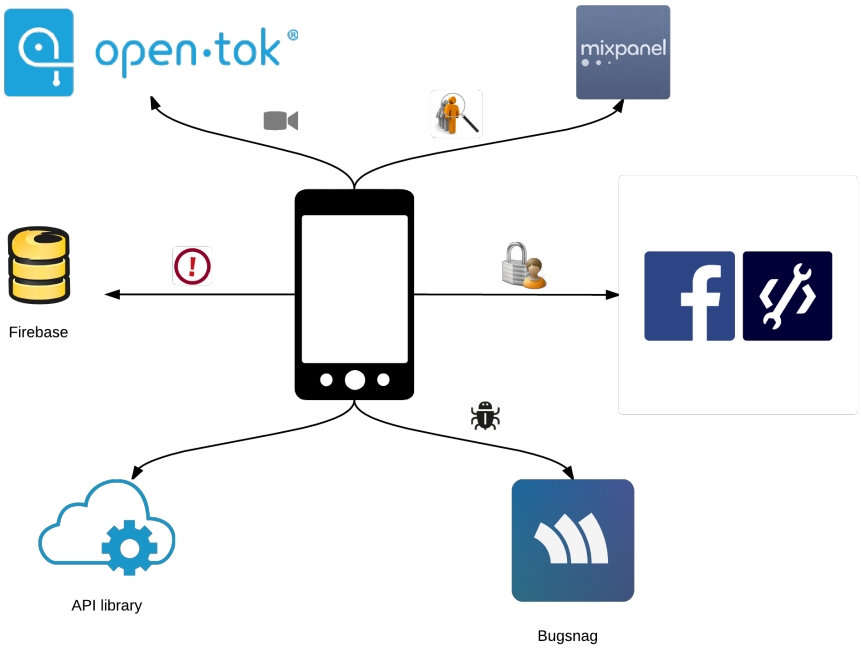


figura 2.2: Stack tecnologico imposto

per tutti i membri del team. Si svolgevano usualmente in una sala riunioni appositamente riservata, in cui vi erano a disposizione un tavolo, una lavagna e un proiettore. Tutti i membri si sedevano attorno al tavolo. La prima parte della riunione consisteva nell'effettuare un consuntivo dello sprint passato, analizzando quanto fatto, le problematiche incontrate e le nuove idee emerse. Nella seconda parte si instanziavano le *story* e ciascuno a turno procedeva nell'assegnare alle proprie tutti i parametri. L'output di questa attività era un "muccchio" di story da svolgere nello *sprint* successivo.

Gli stand-up meeting avevano un peso decisamente inferiore ma erano comunque molto utili per avere un costante monitoraggio dello sprint. Normalmente si svolgevano in una stanza diversa da quella in cui risiedevano le postazioni di lavoro, generalmente di fronte alla macchina del caffè e, come suggerisce il termine, tutti i membri partecipanti stavano in piedi. A turno ciascuno riassumeva le attività svolte nel giorno precedente, evidenziando le difficoltà riscontrate o avanzando proposte di modifica. In seguito dunque ciascuno presentava le attività che avrebbe svolto quel giorno.

Oltre a questo c'è stato un monitoraggio costante da parte del *tutor* aziendale sui processi e le metodologie da me instanziati per l'apprendimento e lo sviluppo della tecnologia, che hanno fatto sì che prendessi gradualmente confidenza con il *modus operandi* adottato dall'azienda.

Per quanto riguarda il periodo iniziale di autoformazione non è stato imposto alcun vincolo metodologico e mi è stata lasciata completa libertà di procedere nel modo che ritenessi più opportuno.

figura 2.3: Esempio di scrum meeting

figura 2.4: Esempio di stand-up meeting

2.2.3 Vincoli temporali

Lo stage viene distribuito in 8 settimane di calendario, per una durata complessiva di 300-320 ore lavorative. Ogni settimana sono previste 40 ore, considerando una giornata lavorativa composta di 8 ore. Lo stage può dunque considerarsi un'attività a tempo pieno.

Nonostante questo vincolo temporale imposto le modalità operative non prevedevano un conteggio delle ore lavorative, ma erano basate sullo svolgimento delle *story*, descritte nel capitolo 1. A fronte di ciò posso affermare che quanto realizzato a livello di ore lavorative differisce in eccesso rispetto a quanto pianificato, in quanto la completa libertà di gestire il proprio tempo ha fatto sì che molto spesso le mie giornate lavorative si prolungassero oltre le 8 ore e che il sabato e la domenica (giorni non considerati in un calendario lavorativo) venissero frequentemente spesi nello sviluppo dell'applicazione.

In nessun modo CoffeeStrap ha imposto vincoli temporali superiori a quanto pattuito. Queste scelte da me effettuate sono state puramente personali, derivate dalla forte motivazione e sollecitazione date dal progetto, di per se molto stimolante, e dalla crescente voglia di imparare.

Capitolo 3

Il progetto di stage

3.1 Pianificazione del lavoro

Le otto settimane lavorative sono state distribuite in quattro periodi:

- **Periodo di apprendimento ed analisi**, nel quale sono entrato in contatto con lo *stack tecnologico* da utilizzare, comprese le tecnologie da integrare, ed ho analizzato nel dettaglio quali fossero i requisiti richiesti dall'applicazione;
- **Periodo di progettazione**, nel quale per ogni requisito ho trovato la soluzione adeguata, andando a delineare l'architettura dell'applicazione. In questo periodo affiancato a questa attività ho inoltre sviluppato un prototipo, in modo da poter verificare l'adeguatezza e funzionalità delle soluzioni progettate;
- **Periodo di sviluppo e codifica**, nel quale ho realizzato la maggior parte del codice dell'applicazione e implementato tutte le *features*;
- **Periodo di rifinitura e testing**, nel quale ho testato l'applicazione con altri utenti e in base al feedback ho raffinato ulteriormente il prodotto, andando a risolvere i numerosi *bug* emersi;

Di seguito viene riportato il diagramma di Gantt relativo alla pianificazione del lavoro realizzata prima dell'inizio dell'attività di stage:

3.2 Norme, procedure e strumenti

Strumenti generali

Per la gestione delle *story* è stato scelto lo strumento **pivotal tracker**[1], tramite il quale è possibile tener traccia di tutte le story ed avere una panoramica sulle *board*. In particolare è possibile creare progetti condivisi tra più utenti e quindi aver sempre a disposizione il monitoraggio del proprio lavoro e di quello degli altri membri.

Per la comunicazione interna al *team* è stato scelto l'utilizzo di **hipchat**[2], il quale fornisce un sistema di messaggistica istantanea e di video conferenze, in modo analogo a quanto offerto da servizi come *skype* o *google hangout*. Hipchat permette inoltre di

[1] https://www.pivotaltracker.com/signin
[2] https://www.hipchat.com/

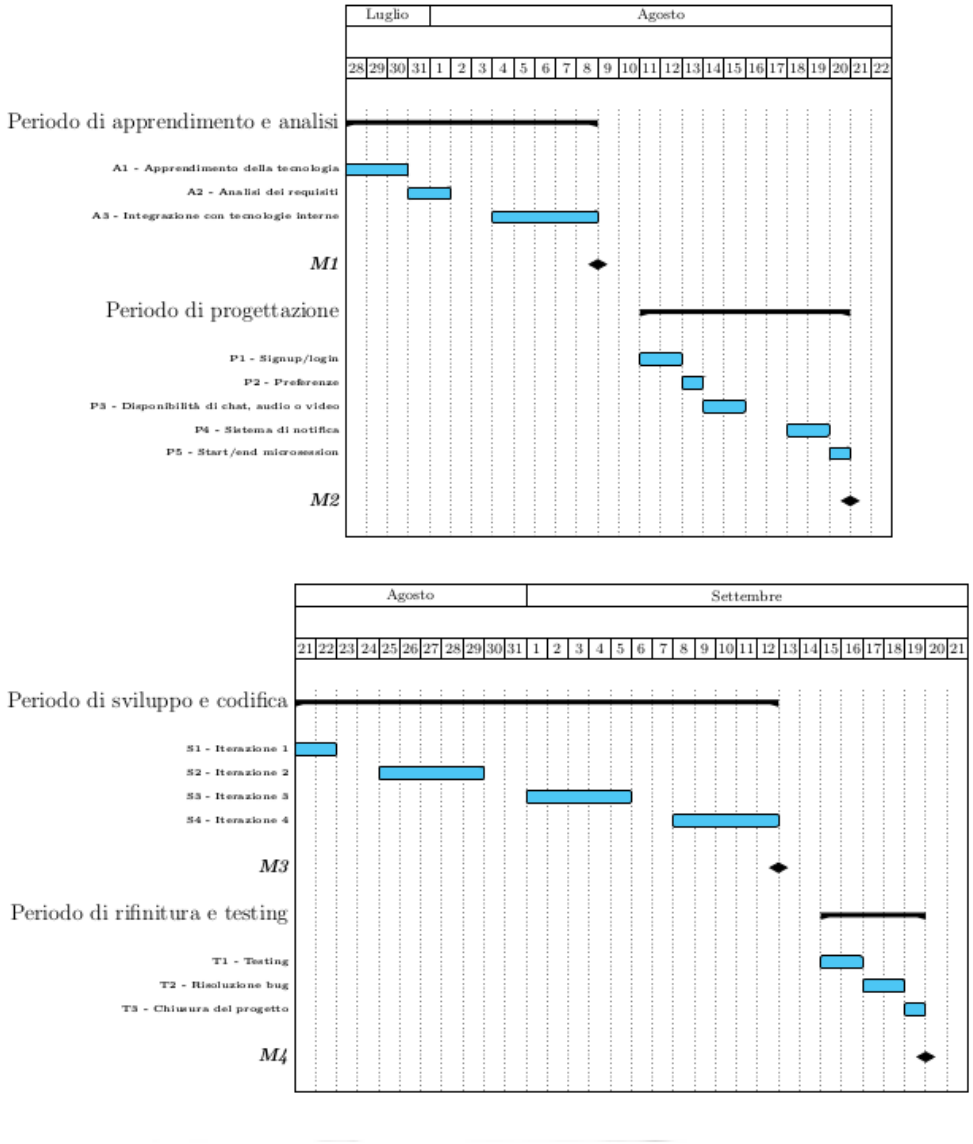


figura 3.1: Diagramma di Gantt relativo alla pianificazione del lavoro

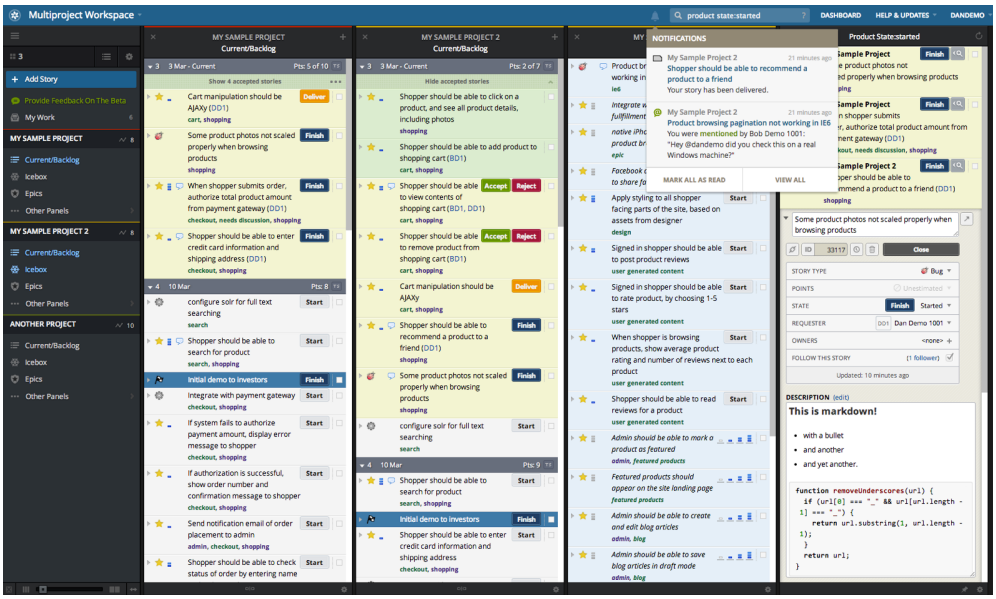


figura 3.2: Schermata di pivotaltracker

creare delle "stanze" con più utenti, in modo da poter discutere di un preciso argomento o tema. È altresì possibile marcare i messaggi con degli hashtag, in modo da poter gestire al meglio le tematiche. Questo strumento è di fatto fondamentale per poter lavorare da remoto.

Per quanto riguarda la produzione di documenti viene utilizzato lo strumento **google documents**[3]. Questa scelta è dovuta alla estrema facilità di utilizzo e familiarità di tale strumento oltre alla possibilità di condividere documenti in tempo reale ed avere essi disponibili in ogni momento e con ogni piattaforma. Inoltre con google documents è possibile inserire in ogni documento dei *suggerimenti*, che in seguito l'autore o gli autori prenderanno in considerazione.

Formazione e analisi

All'inizio dell'attività di stage il mio bagaglio tecnico era in linea con l'insegnamento accademico del mio corso di laurea, quindi sostanzialmente avevo buone conoscenze riguardo al linguaggio **C++**, **Java**, **Javascript** e **PhP**. Inoltre possedevo discrete conoscenze di gestione di database relazionali e documentali, oltre ad una buona preparazione circa lo sviluppo web.

Ciononostante non avevo mai prima di allora sviluppato un'applicazione mobile. Le mie conoscenze in ambito Android erano piuttosto limitate, avevo frequentato alcuni seminari presso l'università e mi ero cimentato per un breve periodo.

A fronte di questa mancanza di conoscenza è stato dunque necessario instanziare un periodo di autoformazione, in modo da entrare in contatto con la tecnologia e poter dunque realizzare l'applicazione. Quest'attività è stata svolta principalmente tramite la lettura e lo studio delle guide Android, messe a disposizione nel sito ufficiale.[4]. Oltre a questo ho consultato le altrettante guide per ciascuna delle tecnologie da

[3] http://www.google.com/docs/about/
[4] http://developer.android.com/training/index.html

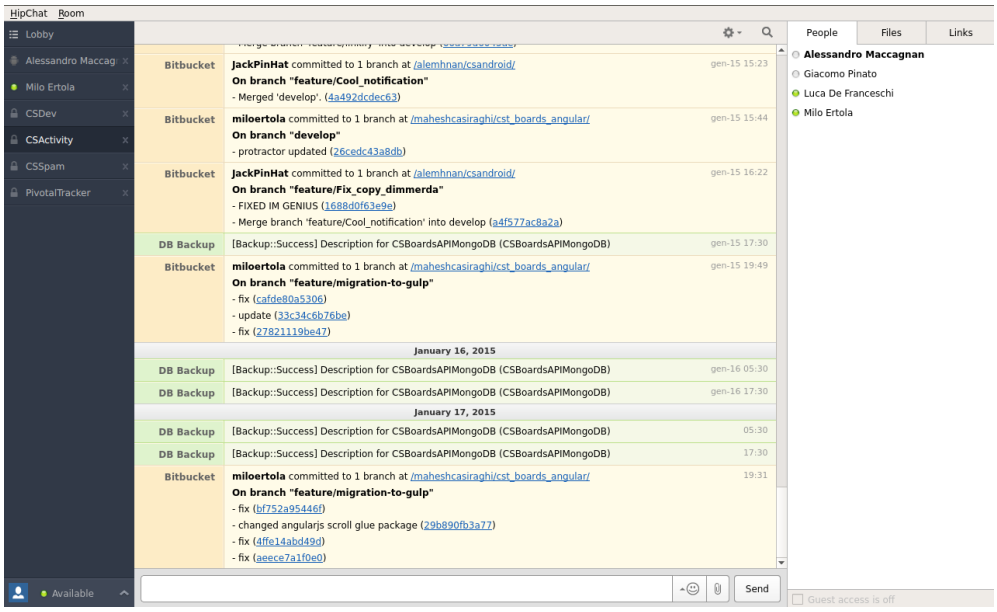


figura 3.3: Schermata di HipChat

integrare all'interno dell'applicazione. Tutti i siti web consultati possono essere trovati in bibliografia. Non ho ritenuto essenziale fornirmi di un libro di testo in quanto le guide e i numerosi esempi disponibili nella rete erano di per sè più che sufficienti.

Per quanto riguarda l'analisi dei requisiti non vi è stata una norma o attività instanziata *ad hoc*, in quanto essa avrebbe rallentato il processo di sviluppo. Non vi è stato inoltre una documentazione e un tracciamento dei requisiti in quanto, essendo che in essi cambiano rapidamente, quest'ultime attività avrebbero portato ad un dispendio di tempo notevole e ad un conseguente rallentamento dello sviluppo. La principale metodologia di supporto all'analisi è stata il semplice *brainstorming*.

Progettazione

In questo periodo ho iniziato ad ideare l'architettura software dell'applicazione e concorrentemente a sviluppare un *prototipo* secondo i requisiti emersi. Per questo ho installato e configurato l'SDK ufficiale di Android, utilizzando in primo luogo l'IDE Eclipse[5], il quale mette a disposizione un ottimo plugin per la sua integrazione ed è tuttora supportato ed ampiamente usufruito dalla comunità degli sviluppatori. La sua scelta è dovuta inoltre alla familiarità da parte mia con questo strumento, il quale era stato da me utilizzato per progetti precedenti.

Per la progettazione del layout dell'applicazione ho utilizzato **lucidchart**[6] come strumento per la realizzazione di mockup. Quest'ultimo stumento è stato inoltre utilizzato per la realizzazione dei diagrammi di attività e classi in UML. Lucidchart si è rivelato uno strumento molto utile e di facile utilizzo, che permette una gestione sia *online* che *offline* dei diagrammi e permette ad altri utenti di condividere un documento e modificarlo in tempo reale. Quest'ultima caratteristica si è rivelata particolarmente utile nel momento in cui vi era necessità di discutere sulla progettazione.

[5] https://eclipse.org/home/index.php
[6] https://www.lucidchart.com/?utm_expid=39895073-63.LohXHUI7T6yhe2x7Iyjc7Q.0

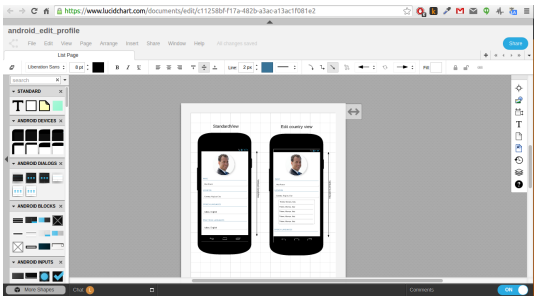


figura 3.4: Lucidchart per la realizzazione di mockup

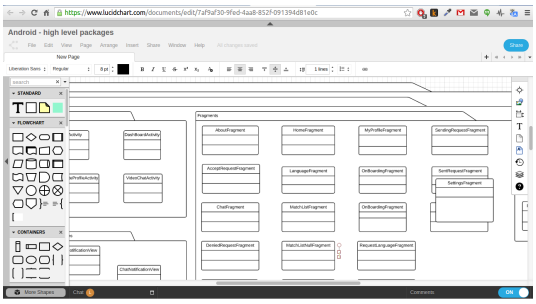


figura 3.5: Lucidchart per la realizzazione di diagrammi UML

Codifica

In seguito al periodo di progettazione ho deciso, per questioni di efficienza, di abbandonare Eclipse per passare dunque ad **Android Studio**[7], il quale è sviluppato dall'azienda stessa e può quindi supportare meglio gli ultimi aggiornamenti della piattaforma. Le principali differenze tra i due IDE sono le seguenti:

- Android Studio utilizza un sistema di compilazione basato su **gradle**[8], a differenza di Eclipse che utilizza **ant**[9]. Gradle incorpora al suo interno i concetti e la logica di *ant* ed **apache maven**[10], introducendo inoltre un DSL basato su **Groovy**[11], che permette la facile codifica di script per l'automatizzazione di processi quali compilazione, verifica o caricamento di file `.apk` all'interno dello store. Ant, al contrario, si basa su una sintassi in formato XML ed è molto più "anziano" rispetto a gradle ma permette anch'esso la codifica di script per l'automatizzazione di processi. Avere a disposizione una serie di processi automatizzati ha come conseguenza un notevole risparmio di tempo, che si traduce in efficienza, oltre che a considerevoli vantaggi in termini di estensibilità e manutenzione del software. La scelta di gradle è dovuta in primo luogo al supporto da parte di Android di questo strumento, il quale viene indicato come *best practice* per il presente e il futuro. Inoltre gradle è risultato per me di più facile utilizzo per la gestione delle dipendenze e per la compilazione, nonchè per la produzione ed esportazione dei file `.apk`;

- Android Studio offre uno strumento di refactoring migliore rispetto ad Eclipse, inoltre il sistema di auto-completamento del codice a mio parere è altresì migliore;

- La gestione dell'interfaccia grafica è a mio parere migliore in Android Studio rispetto ad Eclipse, le opzioni di personalizzazione sono più vaste e rispondono in modo migliore alle mie esigenze;

- Se da un lato Eclipse risulta più "leggero" in quanto a consumo di risorse, d'altro canto presenta diverse problematiche nell'esecuzione di alcune operazioni, come l'esportazione di file `.apk`. È capitato più volte infatti che l'IDE terminasse

[7] http://developer.android.com/sdk/index.html
[8] https://www.gradle.org/
[9] http://ant.apache.org/
[10] http://maven.apache.org/
[11] http://groovy.codehaus.org/

inaspettatmente il suo flusso (crash) oppure si bloccasse per un tempo indeterminato (freeze). Android Studio da questo punto di vista mi ha garantito maggiore affidabilità.

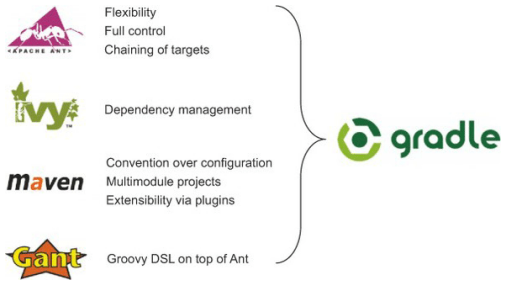


figura 3.6: Gradle combina tutte le migliori features dagli altri strumenti di sviluppo

Testing

Il principale strumento utilizzato in fase di verifica è sato **bugsnag**[12], il quale tramite una libreria permette la sua integrazione all'interno di un progetto Android. Il suo utilizzo consiste nella segnalazione e tracciamento di arresti anomali dell'applicazione. Ogniqalvolta all'interno di un'istanza dell'applicazione avviene un arresto anomalo bugsnag automaticamente invia una segnalazione al proprio server, riportando il tipo di errore, dove esso si è verificato e su quale dispositivo. Nella sua pagina web è possibile visualizzare una tabella con il *log* di tutti gli errori, raggruppati per tipologia, e il numero di occorrenze. Questo strumento si è rivelato di estrema importanza per il rilevamento e tracciamento di *bug* all'interno dell'applicazione.

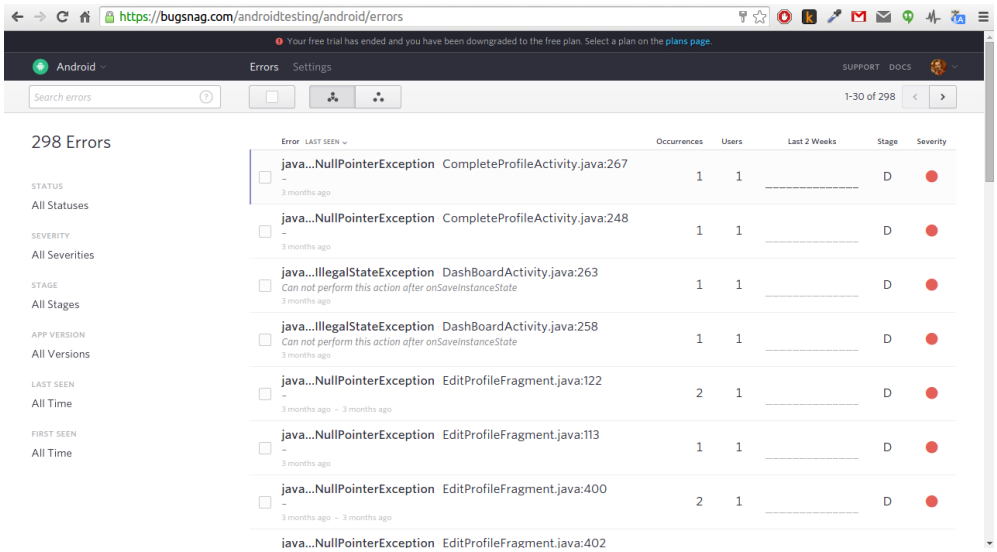


figura 3.7: Una schermata di bugsnag

[12]https://bugsnag.com/

3.3 Periodo di formazione

3.3.1 Apprendimento della tecnologia

In questa sezione descriverò come ho appreso la tecnologia di riferimento e attraverso quali processi, approfondendo nel dettaglio alcuni concetti indispensabili e di rilievo. Descriverò la differenza tra le mie conoscenze alla fine di questo periodo e all'inizio, fornendo evidenza di apprendimento.

3.3.2 Integrazione con le tecnologie aziendali

In questa sezione descriverò come, dopo uno studio della tecnologia legata al progetto di stage abbia integrato quest'ultima con le tecnologie già esistenti utilizzate all'interno dell'azienda.

3.4 Progettazione

3.4.1 Progettazione del flusso utente

In questa sezione descriverò la progettazione del flusso utente all'interno dell'applicazione Android, avvalendomi di diagrammi di attività e di flusso.

3.4.2 Progettazione del layout

In questa sezione descriverò la progettazione del layout delle diverse schermate dell'applicazione, giustificando le varie scelte secondo criteri di accessibilità e usabilità.

3.4.3 Progettazione architetturale

In questa sezione descriverò la progettazione dell'architettura software dell'applicazione, andando ad evidenziare la suddivisione in classi e pacchetti, le interazioni di quest'ultimi, le librerie esterne e i design pattern utilizzati; il tutto in relazione con le best-practise indicate dalla comunitò di sviluppatori Android.

3.5 Sviluppo e codifica

In questa sezione descriverò come ho realizzato il codice dell'applicazione aderendo alla progettazione attuata in precedenza.

3.6 Rifinitura e testing

In questa sezione descriverò come a fronte dello sviluppo e codifica dell'applicazione ho proceduto alla sua verifica, istanziando un sistema di bug tracking e testando l'applicazione con utenti reali.

3.7 Validazione del prodotto e rilascio

In questa sezione infine descriverò come a fronte dell'attività di rifinitura è stata istanziata un'attività di validazione mirata a verificare che il prodotto corrispondesse

alle aspettative e soddisfasse gli obiettivi preposti. Inoltre descriverò l'attività di rilascio e di pubblicazione di una versione sul play store di Google.

Capitolo 4

Valutazioni retrospettive

4.1 Bilancio sui risultati

In questa sezione darò evidenza di aver soddisfatto tutti gli obiettivi di stage, fornendo un consuntivo di quanto realizzato, confrontato con il preventivo di inizio stage.

4.2 Bilancio formativo

In questa sezione confronterò la mia formazione all'inizio dello stage con quella ottenuta alla fine, evidenziando quanto appreso e il tipo di maturazione ottenuta sul piano tecnico e metodologico.

4.3 Distanza tra contesto lavorativo e mondo accademico

In questa sezione fornirò un GAP di conoscenze, evidenziando gli elementi nulli sul quale il corso di laurea non è riuscito a formarmi. Fornirò dunque una critica costruttiva nei confronti del mondo accademico, confrontando il mio bagaglio in ingresso con quello in uscita e presentando dei suggerimenti in ottica di miglioramento del corso di laurea.

Appendice A

Gitflow

Gitflow è un modello di sviluppo basato su Git, ideato da Vincent Driessen e presentato nel 2010 in un suo celebre post.[1]. Sebbene sia pittosto complicato rispetto ad altri modelli, questo framework fornisce un robusto strumento per la gestione di progetti di grandi dimensioni.

Gitflow assegna ruoli specifici ai diversi branch, specificando quando e come essi debbano interagire tra di loro. Essendo basato su Git gli sviluppatori lavorano in locale e periodicamente effettuano *push* sul repository remoto centrale.

A.1 I branch principali

Nel repository principale si distinguono due rami principali:

- **master**, il ramo principale all'interno del quale risiede codice verificato e pronto per essere spedito in *production* e in particolare tutti i suoi nodi corrispondono a **release** del prodotto, ovvero ad una versione stabile, che può essere marcata o meno con un *tag*;

- **develop**, il ramo all'interno del quale avviene lo sviluppo vero e proprio del prodotto e dal quale il codice può essere rilasciato, effettuando dunque un *merge* con il ramo master

A.2 I branch di supporto

Al livello inferiore dei branch principali troviamo i branch di supporto, che assistono gli sviluppatori nelle operazioni quotidiane di sviluppo. Questi branch, a differenza dei due principali, hanno un tempo di vita limitato e vengono uniti nei rami principali o semplicemente scartati. I tre branch di supporto sono:

- **Feature**, che viene creato dal ramo `develop` ed unito solamente nel medesimo; la loro utilità, come dice la parola stessa, è quella di creare un ramo di sviluppo per un nuovo componente del sistema. Essi normalmente vanno tenuti in locale e non devono essere *pushati* sul repository remoto;

[1] http://nvie.com/posts/a-successful-git-branching-model/

- **Release**, che viene creato dal ramo `develop` ed unito sia nel medesimo che successivamente su `master`. Questo branch identifica le procedure che vengono istanziate all'atto della release. Quando uno sviluppatore decide di effettuare una release del prodotto apre innanzitutto un brach da `develop` chiamato `release-*`, esegue tutte le operazioni di *pre-release*, effettua il merge su `develop` ed infine il merge su `master`, marcando il nodo di merge eventualmente con un *tag*;
- **Hotfix**, che viene creato dal ramo `master` ed unito prima sia nel medesimo che successivamente su `develop`. Questi rami derivano dall'immediata necessità di risolvere un *bug* inserito all'interno del ramo master, senza dover effettuare una nuova release.

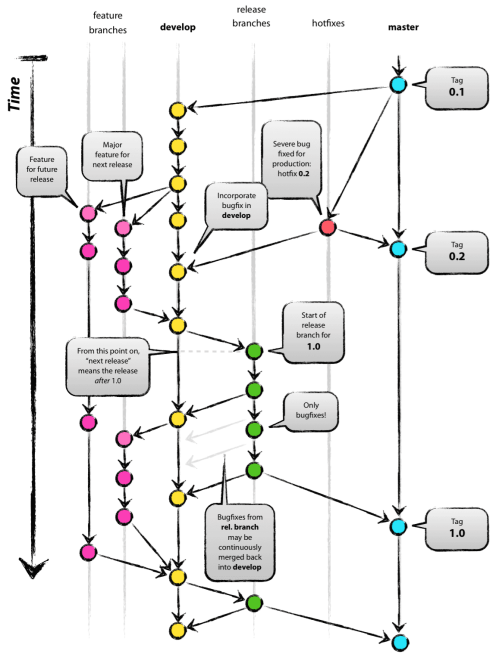


figura A.1: Modello Gitflow

Glossario

acceleratore Nell'ambito della startup l'acceleratore indica un'associazione che fornisce sostegno e assistenza all'azienda nella sua fase primordiale, "accelerando" il suo processo di sviluppo mettendo quest'ultima nell'ambiente giusto e a stretto contatto con il mondo esterno e la rete degli investitori. Normalmente queste associazioni organizzano un vero e proprio programma di accelerazione, fornendo un investimento iniziale all'azienda. In cambio solitamente l'acceleratore detiene una quota societaria.. 1, 31

agile Nell'ingegneria del software si riferisce ad una metodologia di sviluppo del software contrapposto ai modelli tradizionali (ad esempio *waterfall*), in cui si pone l'attenzione sul consegnare al cliente il prodotto in tempi brevi e con iterazioni veloci. Tale metodologia è rappresentata dal *Manifesto Agile*.[2]. 5, 31

Angular.js Si tratta di un framework per lo sviluppo web mantenuto dalla Google creato nel 2009. Favorisce un approccio di tipo dichiarativo allo sviluppo *client-side*, migliore per la creazione di interfacce utente, laddove l'approccio imperativo si presta maggiormente per la realizzazione della logica applicativa. Esso si ispira fortemente al design pattern MVC riducendo considerevolmente il codice necessario alla realizzazione di applicazioni HTML/javascript.. 8, 31

API Indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma. La finalità è ottenere un'astrazione, di solito tra l'hardware e il programmatore o tra software a basso e ad alto livello semplificando così il lavoro di programmazione.. 8, 31

back-end Nella sua accezione più generale rappresenta il componente software di un'applicazione che si occupa della gestione ed elaborazione dei dati, che in un secondo momento saranno *"spediti"* al front-end. Molto spesso si tratta di un programma a se stante, separato nettamente dal corrispettivo front-end, il quale viene gestito anch'esso come progetto distaccato.. iii, 31

Bootstrap È una raccolta di strumenti liberi 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, bottoni e navigazione, e altri componenti dell'interfaccia, così come alcune estensioni opzionali di JavaScript.. 8, 31

brainstorming Tecnica eseguibile in gruppo volta alla raccolta di idee per la soluzione di un problema.. 3, 31

[2] http://agilemanifesto.org/

CEO Traducibile in italiano come *amministratore delegato*, nel mondo delle startup è un componente del team che si occupa normalmente del lato *business* dell'impresa, è colui che fornisce un'interfaccia tra l'azienda e il mondo esterno.. 1, 32

Cloud Service Provider Si tratta di una struttura o un'organizzazione che fornisce servizi cloud ai clienti, a fronte di un contratto di fornitura.. 8, 32

commit Nell'ambito di un repository indica l'operazione con cui si genera una versione del proprio lavoro, aggiungendo un nodo al ramo di sviluppo.. 9, 32

CPO È un responsabile di alto livello all'interno di una startup che si occupa della gestione e coordinazione dei processi aziendali.. 1, 32

crash In informatica indica il blocco o la terminazione improvvisa, non richiesta e inaspettata di un programma in esecuzione, oppure il blocco completo dell'intero computer.. 24, 32

CTO È il responsabile del comparto tecnico all'interno di una startup che si occupa della valutazione e selezione delle tecnologie che possono essere applicate al prodotto o ai servizi che l'azienda produce.. 1, 32

freeze In informatica indica un comportamento anomalo di un'applicazione che normalmente non prosegue normalmente il suo flusso di lavoro ma resta bloccato in uno stato non rispondendo agli input ed occupando comunque le risorse allocate.. 24, 32

Git È un sistema software di controllo di versione distribuito, creato da Linus Torvalds nel 2005.. 9, 32

hashtag Si tratta di un tipo particolare di tag per creare delle *etichette*, formate da una parola (o più parole concatenate) antecedute dal carattere # (*hash*). Essi sono utilizzati principalmente come strumenti per permettere agli utenti di trovare più facilmente un messaggio collegato ad un argomento e partecipare alla discussione, ma anche per incoraggiare a partecipare alla discussione su un argomento indicandolo come interessante. Sostanzialmente sono dei collegamenti ipertestuali che fungono da etichette.. 21, 32

HTML5 È un linguaggio di markup per la strutturazione delle pagine web, pubblicato come W3C Reccomendation nell'Ottobre del 2014. Le novità introdotte dall'HTML5 rispetto all'HTML4 sono finalizzate soprattutto a migliorare il disaccoppiamento fra struttura, definita dal markup, caratteristiche di resa, definite dalle direttive di stile, e contenuti di una pagina web, definiti dal testo vero e proprio. Inoltre l'HTML5 prevede il supporto per la memorizzazione locale di grosse quantità di dati scaricati dal web browser, per consentire l'utilizzo di applicazioni basate su web anche in assenza di collegamento a Internet.. 8, 32

IDE In informatica un ambiente di sviluppo integrato è un software che, in fase di programmazione, aiuta i programmatori nello sviluppo del codice sorgente di un programma. Spesso l'IDE aiuta lo sviluppatore segnalando errori di sintassi del codice direttamente in fase di scrittura, oltre a tutta una serie di strumenti e funzionalità di supporto alla fase di sviluppo e debugging.. 22, 32

incubatore Il concetto di incubatore è molto simile a quello di *acceleratore*, indicando più precisamente il "luogo fisico" all'interno del quale risiede e lavora la startup.. 1, 33

merge Nell'ambito di un repository indica la procedura tramite la quale un *branch* viene unito in un altro, generalente nel branch che lo ha creato.. 9, 33

mockup Nell'ambito del design indica generalmente l'attività di realizzazione di una rappresentazione puramente grafica di un oggetto che dovrà in un secondo momento essere sviluppato. Questa attività è rivolta in particolare ai *designer*, i quali consegneranno l'output agli sviluppatori, che si occuperanno della sua implementazione. Un'esempio tipico è la realizzazione di una pagina web, in cui il designer si occupa di realizzare un "disegno" di come essa dovrà essere realizzata. In un secondo momento uno sviluppatore realizzarà l'opportuno codice HTML/CSS per la visualizzazione sul web.. 22, 33

MongoDB È un database non relazionale (NoSQL), orientato ai documenti. Classificato come un database di tipo NoSQL, MongoDB si allontana dalla struttura tradizionale basata su tabelle dei database relazionali in favore di documenti in stile JSON con schema dinamico (MongoDB chiama il formato BSON), rendendo l'integrazione di dati di alcuni tipi di applicazioni più facile e veloce.. 8, 33

Node.js È un framework *event-driven* di utilizzo *server-side* di Javascript. 7, 33

peer-to-peer È un'espressione che indica un'architettura logica di rete informatica in cui i nodi non sono gerarchizzati unicamente sotto forma di client o server fissi (clienti e serventi), ma sotto forma di nodi equivalenti o paritari (in inglese peer) che possono cioè fungere sia da cliente che da servente verso gli altri nodi terminali (host) della rete. Nell'ambito di CoffeeStrap indica che l'utente svolge sia il ruolo di "insegnante" che quello di "studente" per l'apprendimento linguistico.. 3, 33

refactoring In ingegneria del software, è una tecnica strutturata per modificare la struttura interna di porzioni di codice senza modificarne il comportamento esterno applicata per migliorare alcune caratteristiche non funzionali del software. I vantaggi che il refactoring persegue riguardano in genere un miglioramento della leggibilità, della manutenibilità, della riusabilità e dell'estendibilità del codice e la riduzione della sua complessità, eventualmente attraverso l'introduzione a posteriori di design pattern.. 23, 33

REST Riferisce ad un insieme di principi di architetture di rete, i quali delineano come le risorse sono definite e indirizzate. Il termine è spesso usato nel senso di descrivere ogni semplice interfaccia che trasmette dati su HTTP.. 8, 33

scrum È un framework di sviluppo agile del software iterativo ed incrementale in cui vengono enfatizzati tutti gli aspetti di progetto legati a contesti in cui è difficile pianificare in anticipo e in cui generalmente i requisiti cambiano molto spesso. Vengono istanziati meccanismi di controllo empirico, in cui al principio di *"command-and-control"* viene contrapposta una tecnica di gestione basata su rapidi **cicli di feedback**.. 5, 33

sprint Nella metodologia scrum viene inteso come il lasso di tempo che intercorre tra uno scrum meeting e il successivo. In particolare è la misura della dimensione di un'iterazione sul prodotto.. 5, 34

start-up Identifica la fase iniziale per l'avvio di una nuova impresa, cioè quel periodo nel quale un'organizzazione cerca di rendere profittevole un'idea attraverso processi ripetibili e scalabili. Inizialmente il termine veniva usato unicamente per indicare la fase di avvio di aziende nel settore internet o tecnologie dell'informazione.. 1, 34

UML In ingegneria del software, UML (*Unified Modeling Language*, "linguaggio di modellazione unificato") è un linguaggio di modellazione e specifica basato sul paradigma object-oriented. L'UML svolge un'importantissima funzione di "lingua franca" nella comunità della progettazione e programmazione ad oggetti. Gran parte della letteratura di settore usa UML per descrivere soluzioni analitiche e progettuali in modo sintetico e comprensibile a un vasto pubblico.. 22, 34

Bibliografia

Manifesto Agile. URL: http://agilemanifesto.org/.

Scrum Methodology. URL: http://scrummethodology.com/.

Riferimenti bibliografici

Siti Web consultati

Manifesto Agile. URL: http://agilemanifesto.org/.