Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

TRELLO GAMIFICATION: STUDIO DI UNO STRUMENTO DI SVILUPPO AGILE

Relatore:
Chiar.mo Prof.
PAOLO CIANCARINI

Presentata da:
SIMONE FAGGI

Sessione II
Anno Accademico 2017/2018

Sommario

L'elaborato presenta un caso di studio dello strumento di collaborazione agile Trello. Gli strumenti di collaborazione online sono sempre più utilizzati da aziende e team di sviluppatori software per organizzare il proprio lavoro e impostare un workflow. In questa tesi si analizza la piattaforma Trello, un software progettato per fornire uno strumento online a gruppi di persone e singoli individui. Esso permette di creare, raccogliere, visualizzare, organizzare e condividere ogni tipo di informazione in maniera rapida ed intuitiva. Viene presentato un metodo di studio di tale strumento che utilizza la gamification, una tecnica didattica innovativa che prevede l'utilizzo di elementi ludici in contesti non di gioco, in modo da favorire l'apprendimento da parte degli studenti. In particolare è proposto un progetto che consente agli studenti di avvicinarsi allo strumento di collaborazione Trello, giocando con esso ad uno dei giochi da tavolo più famosi al mondo: Monopoly.

Indice

Capitolo 1

Introduzione

La didattica è la scienza che si occupa della comunicazione e della relazione educativa. Lo scopo della teoria didattica è quello di fornire all'allievo un insegnamento efficace ed efficiente che faciliti l'apprendimento in termini di tempo e energie. Ciò si traduce nello sviluppo di strumenti e metodi di insegnamento innovativi.
Il metodo di insegnamento per antonomasia è la tipica lezione frontale in cui l'insegnante espone in maniera unidirezionale gli argomenti; il passaggio di nozioni tra insegnante e studente fa leva esclusivamente sulla capacità da parte del docente di farsi ascoltare.

Il caso di studio di questa tesi presenta invece una tecnica di insegnamento innovativa, nota come gamification, che pone le sue fondamenta sui metodi didattici del "role-playing", un tipo particolare di simulazione in cui i partecipanti diventano gli attori della situazione rappresentata, identificandosi in specifici personaggi e in determinati contesti.
Si presenta quindi Trellopoly, un progetto che applica tale tecnica, utilizzando elementi di gioco del Monopoly, per favorire la conoscenza e l'apprendimento di Trello, uno strumento di collaborazione online che ha lo scopo di semplificare la gestione delle attività individuali e di gruppo, utilizzato principalmente per progettazione e sviluppo di software con metodologia agile.

Nel primo capitolo viene introdotta la tecnica di gamification, trattandone origini, sviluppi e applicazioni didattiche, per poi concentrarsi sulle metodologie agili ed in particolar modo sulla tecnica agile Scrum. Quindi viene presentato Scrum Lego City, un attività ludica che, applicando la tec-

nica di gamification, è in grado di introdurre i concetti base del metodo di sviluppo agile Scrum.
Nel capitolo successivo sono discussi i principali strumenti di collaborazione, classificandoli in tre macro-categorie: controllo di versione, automazione dell'infrastruttura e supporto alle metodologie agili. È in quest'ultima categoria che si colloca Trello, a cui è dedicato il capitolo 4. Nel capitolo sono presentati gli ambiti di applicazione di Trello e viene spiegata dettagliatamente la sua struttura. Una sezione è dedicata alle API messe a disposizione da Trello, utilizzate per l'implementazione di Trellopoly.
Il capitolo 5 è dedicato alla presentazione del progetto Trellopoly. Tramite analisi dei requisiti, progettazione e architettura, è spiegato con quale logica si è deciso di implementare il progetto, soffermandosi su come si è scelto di rappresentare gli elementi di gioco del Monopoly tramite le componenti strutturali di Trello.
Infine, nel capitolo "Sviluppo e test" sono trattate le principali tecnologie utilizzate nell'implementazione del progetto, facendone una breve introduzione. Quindi è spiegato come tali tecnologie sono state integrate al progetto. È inoltre discusso un caso di applicazione (test) con un gruppo di studenti dell'Università di Bologna.
Nel capitolo 8 sono tratte le conclusioni e proposti alcuni spunti per gli sviluppi futuri.

Capitolo 2

Gamification e metodologie Agili

2.1 Gamification

Il principio alla base della gamification ("ludicizzazione" in italiano) è quello di utilizzare dinamiche e meccaniche di gioco per stimolare alcuni degli istinti primari di un essere umano: competizione, status sociale, voglia di compensi e successo.
Uno studio di Jane Mcgonigal del 2011 sull'influenza del gioco e della gamification sulla società [1], afferma che: "la componente ludica può agevolare la comprensione del mondo circostante e incitare comportamenti sociali virtuosi".

2.1.1 Definizione

Sono due le definizioni principali di "Gamification" [2]. La prima definizione è stata data da Deterding S. nel 2011, ed è la più citata. Deterding definisce il termine Gamification come: "Use of game design elements in non-game contexts" ("Utilizzo di elementi di gioco in contesti non di gioco"). Da notare l'utilizzo da parte di Detering del termine "game", diverso da "play". In inglese "play" è una forma libera, mentre "game" indica un gioco strutturato, in cui le regole prestabilite aggiungono competizione, caratterizzata da obbiettivi e premi. Con questa definizione, Detering mette in risalto come la Gamification utilizzi gli "elementi di gioco" per scopi differenti rispetto

all'impiego che ci si aspetterebbe.
La seconda definizione è stata data nel 2012 da Huotari e Hamari. Definiscono la Gamification come: "A process of enhancing a service with affordances for gameful experiences in order to support user's overall value creation", ovvero "Un processo di miglioramento dei servizi che supporti un esperienza di gioco al fine di valorizzare l'utente". Questa definizione tiene maggiormente conto della "natura" del termine Gamification, parlando di "miglioramento dei servizi" e concentradosi meno sui meccanismi di gioco e maggiormente su user-experience e valorizzazione dell'utente.

2.1.2 Gamification nella didattica

Gamification è quindi una strategia con la quale i processi ordinari vengono infusi con i principi di motivazione e di impegno ispirandosi alla teoria dei giochi.
Attirando l'attenzione di moltissime persone, soprattutto in ambienti virtuali, l'utilizzo di elementi di gioco in contesti educativi potrebbe giovare alla didattica. Capendo cos'è ed in particolare quali sono gli elementi che spingono così tante persone ad essere profondamente impegnati in questi ambienti virtuali, si potrebbe cercare di estrapolarli dai contesti ludici, e applicarli ad insegnamento e apprendimento.

I giochi, se si eliminano le differenze di genere e le varie complessità tecnologiche, condividono quattro caratteristiche: un obiettivo, delle regole, un sistema di feedback e la partecipazione volontaria alla sfida. La ludicizzazione, nella sua forma più rigorosa, è l'applicazione in un contesto didattico di tutto ciò che è prerogativa dei giochi: punti, livelli, classifiche e badge. Questo modo innovativo di fare didattica favorisce l'aumento dell'interattività, grazi al meccanismo premiale che spinge lo studente a raggiungere un obiettivo ben definito a priori; inoltre, aumenta il livello di consapevolezza di ciò che si sta facendo, in quanto gli studenti vengono messi di fronte a scenari di natura pratica che, generalmente, non vengono proposti durante le attività didattiche tradizionali.

2.2 Agile

Sebbene il Project Management esista da diversi anni, la gestione dei progetti per lo sviluppo software è un ambito di studio relativamente nuovo ed in continua crescita. Negli ultimi anni, infatti, si sono sviluppate metodologie di sviluppo software innovative.
In questa tesi sono di particolare interesse le metodologie che vengono definite "Agili", e che si sono evolute dal classico sviluppo "incrementale ed iterativo" (Incremental and Iterative Development, IID).
L'espressione "metodologia agile" (o sviluppo agile del software, in inglese "Agile Software Development", abbreviato in ASD) si riferisce a un insieme di metodi di sviluppo del software emersi a partire dai primi anni 2000 e fondati su un insieme di principi comuni, direttamente o indirettamente derivati dai principi del "Manifesto per lo sviluppo agile del software" [3] (Manifesto for Agile Software Development, impropriamente chiamato anche "Manifesto Agile") pubblicato nel 2001 da Kent Beck, Robert C. Martin, Martin Fowler e altri.
La gran parte dei metodi agili tenta di ridurre il rischio di fallimento sviluppando il software in finestre di tempo limitate chiamate iterazioni che, in genere, durano qualche settimana.

2.2.1 Manifesto for Agile Software Development

La formalizzazione dei principi su cui si basano le metodologie agili è stata oggetto del lavoro di un gruppo di progettisti software che si dettero il nome di "The Agile Alliance".
Come citato dal Manifesto: tra l'11 ed il 13 febbraio 2001, presso la stazione sciistica di "The Lodge at Snowbird", nelle montagne del Wasatch (Utah, US), l'Agile Alliance ha prodotto l'"Agile Software Development Manifesto", chiamato poi "Manifesto for Agile Software Development". I firmatari del Manifesto Agile sono (in ordine alfabetico): Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas; molti dei quali hanno anche sviluppato alcune delle metodologie

agili più famose.
Il Manifesto inizia così: "Stiamo scoprendo modi migliori di creare software, sviluppandolo e aiutando gli altri a fare lo stesso". Sono poi elencati i principi fondamentali da loro formulati secondo una formattazione che associa un influenza alle attività.
È importante notare che i principi di seguito elencati indicano sì che sono più importanti le voci che si trovano a sinistra, ma non vogliono detrarre valore a quelle di destra.

- **Gli individui e le interazioni più che i processi e gli strumenti**: le relazioni e la comunicazione tra gli attori di un progetto software sono la miglior risorsa del progetto.

- **Il software funzionante più che la documentazione esaustiva**: è più importante avere software funzionante che tanta documentazione, bisogna rilasciare nuove versioni del software ad intervalli frequenti, e bisogna mantenere il codice semplice e avanzato tecnicamente, riducendo la documentazione al minimo indispensabile.

- **La collaborazione col cliente più che la negoziazione dei contratti**: bisogna collaborare con i clienti oltre che rispettare il contratto perché la collaborazione diretta offre risultati migliori dei rapporti contrattuali.

- **Rispondere al cambiamento più che seguire un piano**: bisogna essere pronti a rispondere ai cambiamenti oltre che aderire alla pianificazione. Il team di sviluppo deve essere pronto, in ogni momento, a modificare le priorità di lavoro nel rispetto dell'obiettivo finale.

2.2.2 Metodologie Agili

Di seguito sono riportate ed introdotte con una breve descrizione, tre delle principali metodologie Agili che si sono sviluppate seguendo proprio i principi del Manifesto: Scrum, Test Driven Development e Extreme Programming.

Scrum

Scrum [4] è una metodologia sviluppata da Ken Schwaber e Jeff Sutherland ed è definito come "un framework che consente di risolvere problemi complessi di tipo adattivo e, al tempo stesso, di creare e rilasciare prodotti in modo efficace e creativo dal più alto valore possibile" .
Scrum non è un processo o una tecnica per costruire prodotti ma è piuttosto un framework all'interno del quale è possibile utilizzare vari processi e tecniche di sviluppo software. È costituito da "Scrum Team", ruoli, eventi, artefatti e regole ad essi associati. Ognuna di queste entità è essenziale per il successo e il buon utilizzo di Scrum.
Scrum si basa sulla "teoria empirica" [5], corrente filosofica nata nella seconda metà del Seicento, secondo cui la conoscenza umana è condizionata esclusivamente dai sensi e dall'esperienza. I pilastri che sostengono ogni implementazione del controllo empirico di processo sono: trasparenza, ispezione e adattamento. La trasparenza richiede che gli aspetti significativi del processo siano definiti da uno standard comune, in modo che gli osservatori ne condividano una comune comprensione. In altri termini, un linguaggio comune di riferimento deve essere condiviso da tutti i membri del team. Per raggiungere tale obbiettivo, è per esempio utile definire un glossario dei termini di progetto consultabile da tutti i membri del team. Chi utilizza Scrum deve ispezionare frequentemente gli artefatti e l'avanzamento verso l'obbiettivo prefissato. Tali ispezioni non dovrebbero essere tanto frequenti da intralciare il lavoro stesso, esse sono utili quando eseguite diligentemente da chi ha l'abilità e la competenza necessaria ad effettuarle rispetto ad un particolare stadio del lavoro. Se chi ispeziona verifica che uno o più aspetti del processo non sono accettabili, deve adattare il processo o il materiale ad esso relativo il più rapidamente possibile.
Scrum struttura il proprio sviluppo in cicli chiamati Sprint che durano da una a quattro settimane, al termine dei quali il team di sviluppo rilascia funzionalità immediatamente testabili. I cicli sono "timeboxed", ciò significa che hanno durata fissa nel tempo, non possono essere estesi e terminano anche se il lavoro non è stato ultimato. All'inizio di ogni Sprint il team seleziona i propri task da una lista di attività priorizzate e si impegna a completare tutte le attività selezionate entro la fine dello Sprint. Al termine dello Sprint

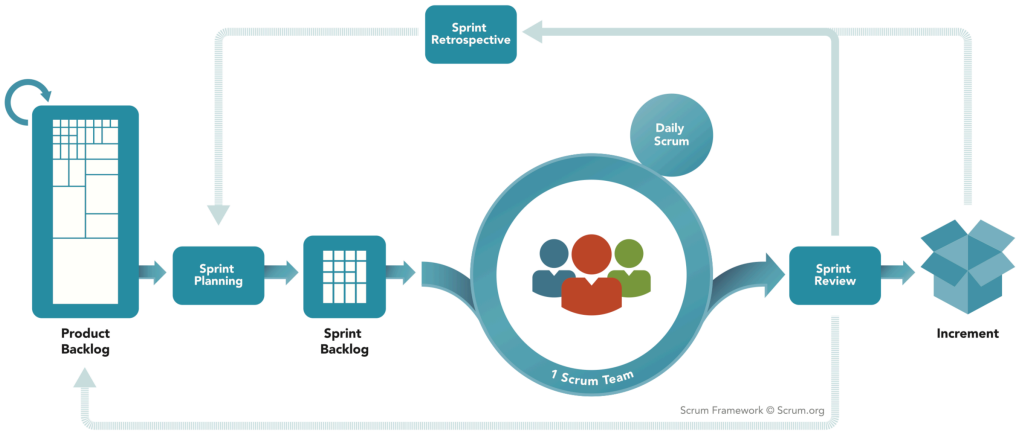

Figura 2.1: Scrum framework. Immagine tratta dal sito ufficiale di Scrum "www.scrum.org".

il team rilascia ciò che è stato completato (Done), vale a dire tutto ciò che rispetta un elenco di requisiti precedentemente definiti (Definition of Done). Per esempio, nel caso di una normale applicazione software, potrebbe voler dire una funzionalità integrata, funzionante, testata e rilasciabile.
Lo Scrum Team è formato da un Product Owner, dal team di sviluppo e da uno Scrum Master. Gli Scrum Team sono auto-organizzati (scelgono indipendentemente come svolgere il loro lavoro) e cross-funzionali (hanno le competenze per essere auto-organizzati).
Il modello su cui si sviluppa lo Scrum Team, è progettato per ottimizzare flessibilità, creatività e produttività.

Test Driven Development

Il Test Driven Development (TDD) [6], in italiano "sviluppo guidato dei test" è una metodologia agile di sviluppo software introdotta da Kent Beck, padre delle metodologie agili.
Il TDD è un metodo di programmazione in cui l'ordine delle tre attività principali, quali "coding", "testing" e "refactoring", sono intrecciate tra loro. Può essere suddiviso in tre fasi principali:

1. Scrivere un unico Unit Test relativo ad una funzionalità del programma.

2. Scrivere il codice che passi il test.

3. Fare refactoring del codice.

Durante la prima fase, detta "fase rossa", il programmatore scrive un test che il programma non deve passare, in quanto la funzionalità di cui si deve testare la correttezza, non è ancora stata implementata. Perché il test sia completo, deve essere eseguibile e, quando viene eseguito, produrre un esito negativo. In molti contesti, questo implica che debba essere realizzato uno "stub" minimale del codice da testare, necessario per garantire la compilabilità e l'eseguibilità del test. La fase rossa si conclude quando un nuovo test può essere eseguito e fallisce. Nella seconda fase, detta "fase verde", il programmatore sviluppa la quantità minima di codice necessaria per passare il test. Non è richiesto che il codice scritto sia di buona qualità o elegante; l'unico obiettivo esplicito è che funzioni, ovvero passi il test. Quando il codice è pronto, il programmatore lancia nuovamente tutti i test disponibili sul software modificato (non solo quello che precedentemente falliva). In questo modo, il programmatore ha modo di rendersi conto immediatamente se la nuova implementazione ha causato fallimenti di test preesistenti, ovvero ha causato regression nel codice. La fase verde termina quando tutti i test sono "verdi" (ovvero vengono passati con successo). La terza fase, detta "fase grigia" o di refactoring, il programmatore esegue il refactoring del codice per adeguarlo a determinati standard di qualità. Ne viene migliorata la struttura attraverso un procedimento basato su piccole modifiche controllate volte a eliminare o ridurre difetti oggettivamente riconoscibili nella struttura interna del codice. Dopo ciascuna azione di refactoring, i test automatici vengono nuovamente eseguiti per accertarsi che le modifiche eseguite non abbiano introdotto errori.

Il principio fondamentale del TDD è che lo sviluppo vero e proprio deve avvenire solo allo scopo di passare un test automatico che fallisce. In particolare, questo vincolo è inteso a impedire che il programmatore sviluppi funzionalità non esplicitamente richieste, e che il programmatore introduca complessità eccessiva in un progetto, per esempio perché prevede la necessità di generalizzare l'implementazione in un futuro più o meno prossimo.

I cicli TDD sono intesi come cicli di breve durata, al termine di ciascuno dei quali il programmatore ha realizzato un piccolo incremento di prodotto. L'applicazione reiterata del refactoring al termine di ogni ciclo ha lo scopo di creare codice di alta qualità e buone architetture in modo incrementale, tenendo però separati l'obiettivo di costruire software funzionante (fase verde) e quello di scrivere "buon codice" (fase grigia). La breve durata dei cicli TDD tende anche a favorire lo sviluppo di componenti di piccole dimensioni e ridotta complessità.

Extreme programming

Extreme Programming, espressione inglese per programmazione estrema, è una metodologia di sviluppo del software che enfatizza la scrittura di codice di qualità e la rapidità di risposta ai cambiamenti di requisiti. Appartenendo alla famiglia delle metodologie agili, prescrive lo sviluppo iterativo e incrementale strutturato in brevi cicli di sviluppo. L'Extreme Programming (XP) è stata ideata da Kent Beck durante il suo lavoro sul progetto Chrysler Comprehensive Compensation System (C3) [7]. Nel Marzo del 1996, Beck è diventato leader del C3 e iniziò a rifinire le metodologie di sviluppo che aveva utilizzato per quel progetto. Decise cos'ì, nel 1999, di scrivere e pubblicare il libro "Extreme Programming Explaitned" [7].

Il libro descrive la "programmazione estrema" come una disciplina di sviluppo del software che permette la produzione di software di qualità superiore in maniera produttiva. La programmazione estrema introduce una serie di valori, principi e pratiche di base: Pair Programming, Planning Game, Test Driven Development, Whole Team, Refactoring, Continuous Integration, Small Release, Coding Standard, Collective Ownership, Simple Design, System Metaphor, Sustainable Pace. Uno degli aspetti indubbiamente più interessanti dietro XP è l'utilizzo diffuso di TDD (Test Driven Development). Come detto, attraverso questa pratica è possibile testare ogni piccola componente software ancora prima che venga implementata e individuare possibili errori fin dalle prime fasi dello sviluppo.
Semplicità, pulizia e chiarezza del codice sorgente sono altri aspetti che Extreme Programming enfatizza e promuove. Si tende a scrivere soltanto il necessario e tutto ciò che non è indispensabile può essere aggiunto in un

secondo momento; tutto, inoltre, può essere migliorato o ristrutturato attraverso il refactoring. Una volta sviluppata una determinata funzionalità, il codice prodotto viene trasferito su un build server. A ogni build vengono avviati i test automatici, in modo da assicurare l'integrità del codice sorgente e dare la possibilità al team di accedere in qualsiasi momento a una versione funzionante del software. Questa pratica è chiamata Continuous Integration e la comunità XP raccomanda di eseguire build frequenti, anche una ogni ora, non meno di una volta al giorno.
Il Pair Programming (programmazione in coppia) è forse la pratica più conosciuta di XP. L'idea di base è che due sviluppatori lavorino insieme alla scrittura del codice: uno assume il ruolo di "guidatore" e l'altro di "navigatore" o "osservatore".

2.3 Stato dell'arte

Negli ultimi anni si sono sviluppate una grande quantità di tecniche di gamification per la didattica. In questa tesi sono di particolare interesse la metodologie Agili; viene quindi presentato il workshop Scrum Lego City, un'attività didattica che utilizza elementi ludici per far comprendere agli studenti i principi di base della metodologia Agile Scrum.

2.3.1 Scrum Lego City

"Scrum Lego City" è un attività didattica che è in grado di introdurre i concetti base del metodo di sviluppo agile Scrum, concentrandosi sul processo di sviluppo e non sui dettagli implementativi e/o sul prodotto finale. Questa attività è applicata nelle università europee e statunitensi, oltre che nelle scuole superiori; la declinazione specifica dell'attività stata sviluppata all'università di Goteborg, in Svezia.
È stata presentata anche all'Università di Bologna, nel corso di studi in Informatica.
L'obbiettivo del gioco è di costruire, con i lego, una città. Il lavoro è suddiviso in team, ognuno dei quali collabora per raggiungere lo stesso obbiettivo. La costruzione della città da parte dei team avviene seguendo i principi della

metodologia agile Scrum.
L'attività si sviluppa in tre fasi principali: presentazione iniziale, planning poker, iterazioni.

Presentazione

L'attività prevede inizialmente una breve presentazione da parte dei docenti della metodologia agile Scrum; non è necessario che i partecipanti abbiano già studiato o utilizzato questa tecnica, lo scopo è infatti introdurre agli studenti questa metodologia di sviluppo. Durante la fase di presentazione ci si sofferma principalmente sulla struttura di uno sprint, che consiste in scegliere le user story su cui lavorare, decomporre le storie in task e raffinarle, concepire l'architettura con cui realizzare le user story per poi implementarle, testarle e integrarle. Si presentano quindi i ruoli dei membri del team, quali Product Owner e Scrum Master.

Planning Poker

La prima attività che mettono in pratica i membri del team è il Planning Poker, con lo scopo di stimare lo sforzo richiesto, in termini di tempo, per completare le user story. Sono utilizzate carte che riportano dei numeri, rappresentanti l'effort stimato per ciascuna attività. I numeri sono quelli della sequenza di Fibonacci (1, 2, 3, 5, 7, 12, 19...), od altra sequenza simile, per indicare la maggiore incertezza nella stima delle attività caratterizzate da uno sforzo maggiore. Durante questa attività ogni membro del team pone una carta con la propria stima sul tavolo, a faccia in giù. Tutti rivelano la carta simultaneamente e, chi è più distante dal valore medio, giustifica la sua proposta. Si ripete l'attività per ogni user story, finché non si arriva a una decisione unanime. Si dà quindi del tempo al team da dedicare alla definizione del termine "fatto" (Definition of Done), in questo modo il gruppo ha dei criteri di accettazione del prodotto generali, da soddisfare prima che il prodotto sia esaminato in una review.

Iterazioni

Si inizia poi con le iterazioni, durante le quali avviene lo sviluppo e la revisione dei prodotti. Ogni iterazione ha, solitamente, la durata di 20 minuti, durata che può variare in base al tempo a disposizione del docente; è comunque consigliato effettuare almeno tre iterazioni. Durante le iterazioni, ogni team sceglie una storia e inizia a realizzarla. Quando la ritiene "fatta", secondo la definizione precedentemente data, viene consegnata per la fase di review. Durante lo sviluppo, i Product Owner intervengono, cambiando requisiti e mettendo in crisi il lavoro del team, durante la fase di revisione decidono se accettare il prodotto consegnato. In caso contrario danno indicazioni su come modificare il prodotto per la prossima consegna.

Capitolo 3

Strumenti di collaborazione

3.1 Origini

Collaborazione e comunicazione sono principi cardine per lo sviluppo software, in particolare per lo sviluppo agile. La distanza tra i team di lavoro e la mancata possibilità di incontri rendono difficile sia la collaborazione che la comunicazione.
Nascono cosi gli strumenti online per la collaborazione tra gruppi di lavoro [8]. Essi consentono l'agevolazione, l'automazione e il controllo dell'intero sviluppo progettuale. In particolare permettono di creare, raccogliere, organizzare e condividere informazioni utili alla gestione progettuale tra collaboratori.
Grazie a questi strumenti il team ha accesso in maniera simultanea al progetto su cui sta lavorando, avendo la possibilità di contribuire allo sviluppo, alla nascita di idee, di organizzare il lavoro in sotto progetti e di assegnarne ciascuno ad un membro del team. Si ha la possibilità di monitorare costantemente l'avanzamento del progetto e di creare conversazioni riguardo ad uno specifica istanza.

Esistono una moltitudine di strumenti di collaborazione online, divisibili in tre macrocategorie, diverse per caratteristiche tecniche e funzionali, correlate alle principali fasi del ciclo di sviluppo applicativo.

3.2 Strumenti di controllo di versione

Gli strumenti di gestione e automazione del codice sono tool che, attraverso un database, tengono traccia di tutte le modifiche ad un file o ad una serie di file, richiamando, all'occorrenza, specifiche versioni (versioning).
Nell'ingegneria del software il controllo di versione è qualunque pratica che tiene traccia e permette di controllare i cambiamenti al codice sorgente prodotti da ciascun sviluppatore, condividendone al tempo stesso, la versione più aggiornata o modificata da ciascuno di essi, mostrando costantemente lo stato di avanzamento del lavoro.
Gli sviluppatori software talvolta usano il controllo versione anche per i file di documentazione e di configurazione, oltre che per il codice sorgente.
Uno dei tool più famosi ed utilizzati è Git, un sistema distribuito di controllo versione, liberamente disponibile in modalità open source e studiato per gestire con velocità ed efficienza progetti di piccole o grandi dimensioni.

3.2.1 Git

Il "versioning" del codice è un aspetto fondamentale che caratterizza lo sviluppo di progetti software complessi.
Git è un progetto open source nato nel 2005 ad opera di Linus Torvalds [9]. È un Version Control System (VCS), ovvero un sistema per la gestione delle revisioni, nato con l'obiettivo di rendere agevole lo sviluppo condiviso e distribuito del kernel Linux.
Un VCS è un sistema software che permette di gestire lo storico delle versioni di un documento o di un progetto. Nel caso specifico di progetti software, questi strumenti sono anche definiti Source Control Systems (SCS), ovvero sistemi per il controllo del codice sorgente. Lavorando con un VCS è possibile salvare lo stato di avanzamento di una versione del progetto (o revisione). Essa verrà archiviata e associata ad un identificativo univoco. Ciò permette, in qualunque momento, di recuperare ed eventualmente ripristinare lo stato del progetto in un determinato istante.
La particolarità di un sistema come Git è l'architettura distribuita. In termini pratici, chiunque effettui il "checkout" di un file dal suo repository principale, possiede, a tutti gli effetti, una copia completa dello stesso (clone) sulla pro-

pria macchina. È dunque possibile svolgere le proprie attività offline, senza contattare il server centrale e senza collidere con lo sviluppo di altri programmatori. Git è pensato per funzionare in tutti i sistemi operativi basati su GNU/Linux, ma funziona anche in altri sistemi unix-like, tra cui BSD, Solaris e Darwin. Può essere utilizzato su Windows installando l'ambiente Cygwin [10] (emulazione POSIX).

3.3 Strumenti di automazione dell'infrasttutura

Questi strumenti vengono anche denominati tool per Infrastructure-as-Code (IaC) perché permettono di automatizzare il deployment dell'infrastruttura, amministrandola via software, tramite modelli e file di configurazione che descrivono i componenti, gli stack di risorse (hardware fisico o macchine virtuali) richiesti per far funzionare una singola applicazione, o anche un intero data center.
Nel mondo dell'Information Technology (IT) le esigenze e i carichi di lavoro cambiano continuamente e la richiesta di deploy delle applicazioni e delle infrastrutture è costante.

3.3.1 Terraform

Terraform [11] è un tool estremamente semplice e potente, che consente di creare, modificare e migliorare la produzione di infrastrutture in modo del tutto sicuro. Si tratta di uno strumento open source che permette di gestire file di configurazione che possono essere condivisi tra i membri dello staff e trattati esattamente come se fossero codice sorgente, pertanto possono anche essere modificati, revisionati, aggiornati ecc.
Di seguito sono descritti i principali passaggi necessari per costruire un'infrastruttura con Terraform, e i relativi vantaggi che il tool mette a disposizione degli utilizzatori.

1. **Definire l'infrastruttura come se fosse codice**: accresce la produttività e la trasparenza operativa e permette di condividere le con-

figurazioni, facilitando la collaborazione tra i membri del team. Inoltre è possibile tracciare l'intera storia delle versioni infrastrutturali e verificare la sua evoluzione.

2. **Progettare e pianificare preventivamente le modifiche**: con Terraform effettuare qualsiasi modifica è molto semplice, grazie alla possibilità di pianificare ogni cambiamento e di verificare le preview delle modifiche prima di applicarle. Il processo di progettazione si snellisce notevolmente proprio perché è possibile comprendere preventivamente gli eventuali effetti che anche il più piccolo cambiamento potrebbe causare all'infrastruttura.

3. **Creare un'infrastruttura riproducibile**: Terraform abilita gli operatori a riutilizzare facilmente le stesse configurazioni in luoghi diversi. Ciò permette di ridurre gli errori e di risparmiare tempo. Uno dei vantaggi di Terraform è che le configurazioni possono essere raggruppate in moduli e condivise in maniera semplice tra team e aziende.

3.4 Strumenti per il supporto alle metodologie Agili

Sono strumenti per la gestione condivisa del workflow management.
La teoria e le applicazioni del workflow management promuovono la gestione dei gruppi di lavoro collaborativi secondo il workflow model, o modello processuale. Un processo consiste in una o più attività ognuna delle quali rappresenta un lavoro da svolgere per giungere a un obiettivo comune.
Strumenti di questo tipo permettono ai team di avere chiara, in qualsiasi momento, la suddivisione del team in aree di sviluppo e lo stato di avanzamento dei progetti.

I più interessanti tool di questo tipo sono Trello, Asana e Kanbanchi.
Di seguito vengono introdotti con una breve descrizione Asana e Kanbanchi.
Verrà invece dedicato un capitolo a Trello, oggetto di studio di questa tesi.

3.4.1 Asana

Asana è un Software-as-a-Service (SaaS) rilasciato nel 2008, progettato e sviluppato per migliorare la collaborazione e la gestione del lavoro.
Aiuta i team a gestire progetti e attività con unico strumento, dando la possibilità di creare progetti, assegnare lavoro ai membri del team, specificare scadenze e comunicare direttamente con Asana tramite chat di gruppo. Include anche strumenti di reporting, file allegati, calendari e altro.
A maggio 2013, Asana ha lanciato "Organizations", un insieme di strumenti di reporting per aiutare i team a monitorare i progressi del progetto e gli strumenti di amministrazione. Nel 2016 ha aggiunto funzionalità di amministrazione tra cui gestione di membri, team e password e controlli di sicurezza.

Struttura

La struttura di Asana prevede quattro diversi livelli:

- **Organizzazione**
- **Team**
- **Progetti**
- **Task**

Un'*organizzazione* può avere diversi team al suo interno; ciascun team può gestire diversi progetti; ciascun progetto è composto da più task.
Se ci si iscrive ad Asana con l'email aziendale, si può creare un'organizzazione: tutti i colleghi che si iscrivono saranno in automatico collegati al gruppo di lavoro creato. Se l'organizzazione è molto strutturata, è possibile creare diversi *team* che rappresentano i comparti dell'azienda.

È possibile creare un *progetto* definendone il nome e inserendo una breve descrizione. In seguito si può condividere il progetto con uno o più team interni all'organizzazione. Una volta creato il progetto, si può assegnare un colore per distinguerlo e invitare i membri che devono prenderne parte. Ciascuno di loro riceverà una notifica email.

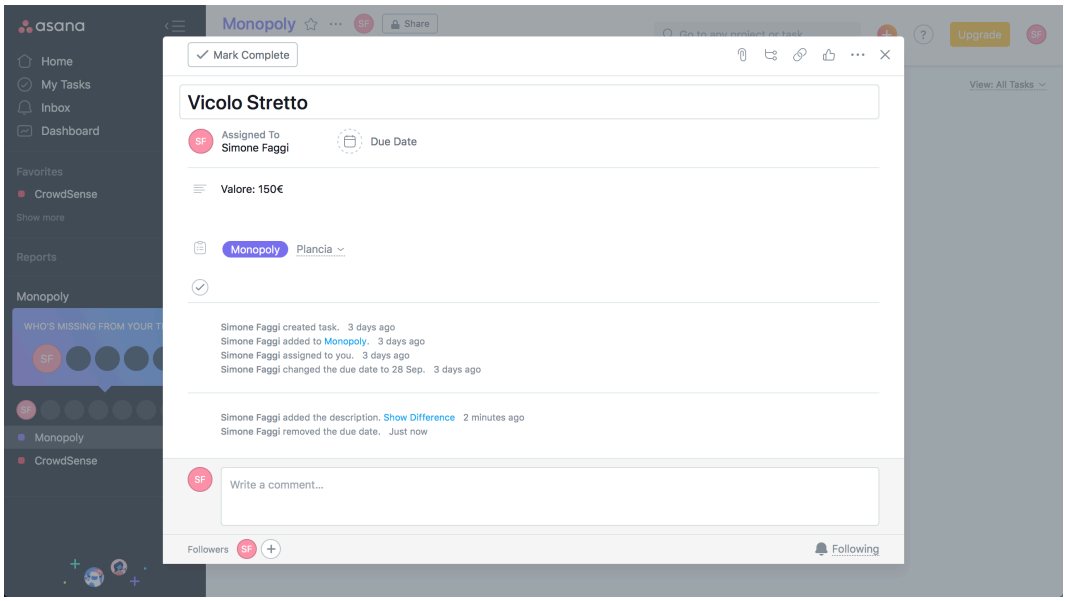

Figura 3.1: Esempio di task in Asana.

Il *task* (Figura 3.1) è l'unità di base su cui lavorare: può essere un'azione da fare, un'idea da proporre, un'attività da assegnare a qualcuno, una telefonata da ricordare.
Il task è assegnato ad una o più persone, può avere una data di scadenza e può appartenere ad una *sezione*. Le sezioni permettono di raggruppare i task, dividendoli ad esempio per fasi di sviluppo o per tipologia. Possono inoltre contenere dei "subTask" al loro interno, selezionabili come "fatti" (checkable) una volta eseguiti.

3.4.2 Kanbanchi

Kanbanchi è un software di supporto al lavoro di gruppo sviluppato da Google e presenta funzioni pensate per l'utilizzo in diversi contesti lavorativi, da un piccolo gruppo ad una grande impresa; è free ma presenta anche numerose "paid features".
Il software è organizzato in "cards" che possono rappresentare lavori, semplici informazioni o qualsiasi elemento del processo lavorativo. Le cards sono ordinate in colonne. L'interfaccia risulta molto semplice ed intuitiva ed offre una

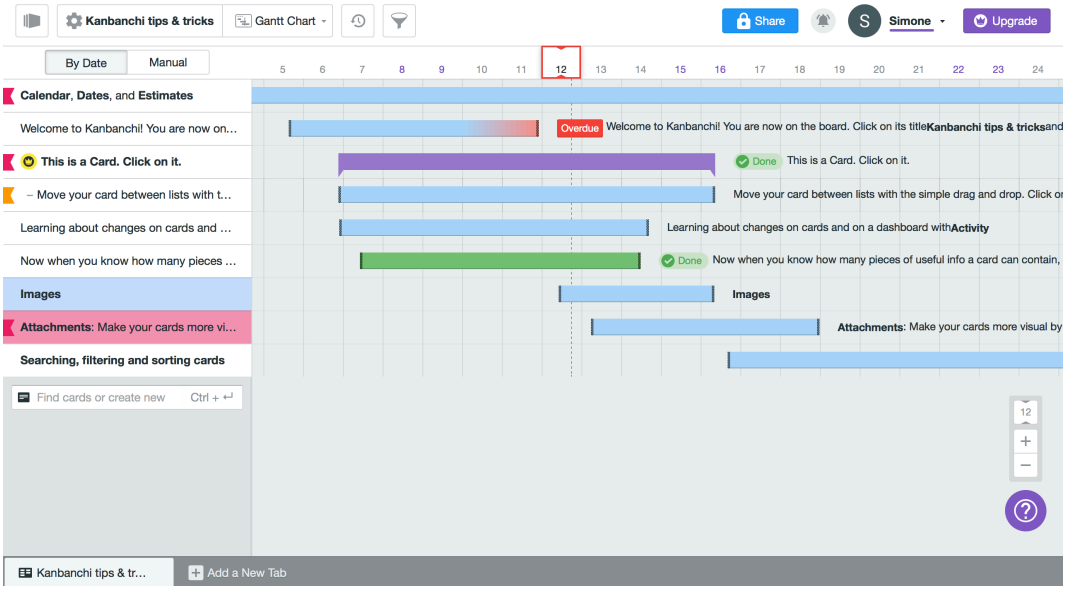

Figura 3.2: Gantt chart di tutorial su Kanbanchi

grande possibilità di personalizzazione, infatti è possibile organizzare le cards con colori e tag, possono contenere date, orari, files, links (convertiti dove possibile in Hyperlinks) e molto altro, come la percentuale di completamento di un determinato lavoro. L'utente ha anche la possibilità di condividere le cards con i propri contatti, il tutto semplificato dalla sinergia con Gmail e GoogleDrive.

Con le "paid features" troviamo molte altre funzioni, utili in contesti di lavoro specializzati. Tramite queste funzioni avremo la possibilità di visualizzare la nostra schermata come un "Gantt Chart", un sistema che ci permette di visualizzare lo svolgimento di più lavori in un grafico che mostra il completamento degli obbiettivi in relazione al tempo (utile per grandi gruppi di lavoro che svolgono più lavori contemporaneamente); ma anche funzionalità più immediate, come aggiungere alle cards un timer o il logo della compagnia, esportare il lavoro su Google Spreadsheets e organizzare le cards per priorità.
Tramite pochi semplici passaggi è possibile impostare notifiche per ogni genere di cambiamento che verrà apportato al lavoro da parte degli altri utenti o per scadenze imminenti; le notifiche verranno visualizzate in-app quando

un collaboratore: crea una card, ne cambia i destinatari, elimina una card, la commenta, la rinomina, ne cambia la descrizione, aggiunge allegati, sposta una card in un'altra lista, modifica le impostazioni di priorità.
La piattaforma è anche facile da inizializzare e richiede la registrazione dell'account (se si possiede un account Google il tutto avviene in pochi click).

Capitolo 4

Trello

4.1 Introduzione

Tra gli strumenti di supporto alle metodologie Agili spicca, per popolarità e ambiti di utilizzo, Trello.
Trello [12] è uno strumento di collaborazione online che nasce dal tentativo di semplificare la gestione delle attività individuali e di gruppo.
Fondata nel gennaio 2011 dall'azienda Fog Creek Software, Trello è stato progettato per fornire uno strumento online a gruppi di persone e singoli individui che permettesse di creare, raccogliere, visualizzare, organizzare e condividere ogni tipo di informazione in maniera rapida ed intuitiva.
Nel luglio 2014, il servizio Trello è stato scorporato da Fog Creek Software e divenne Trello Inc. [13]. A quel tempo, la società aveva raccolto $10,3 milioni in finanziamenti e aveva più di 4,75 milioni di utenti. All'inizio del 2017, con oltre 19 milioni di utenti, Trello è stata acquisita da Atlassian, azienda che già gestiva diverse applicazioni software e piattaforme online progettate per migliorare la produttività di piccole e medie imprese.

4.2 Ambiti di applicazione

Sono molti gli utilizzi che si possono fare con una piattaforma come Trello. Verrebbe da pensare, per come è stato presentato, che sia uno strumento che abbia il solo obbiettivo di fornire supporto a programmatori, sviluppatori,

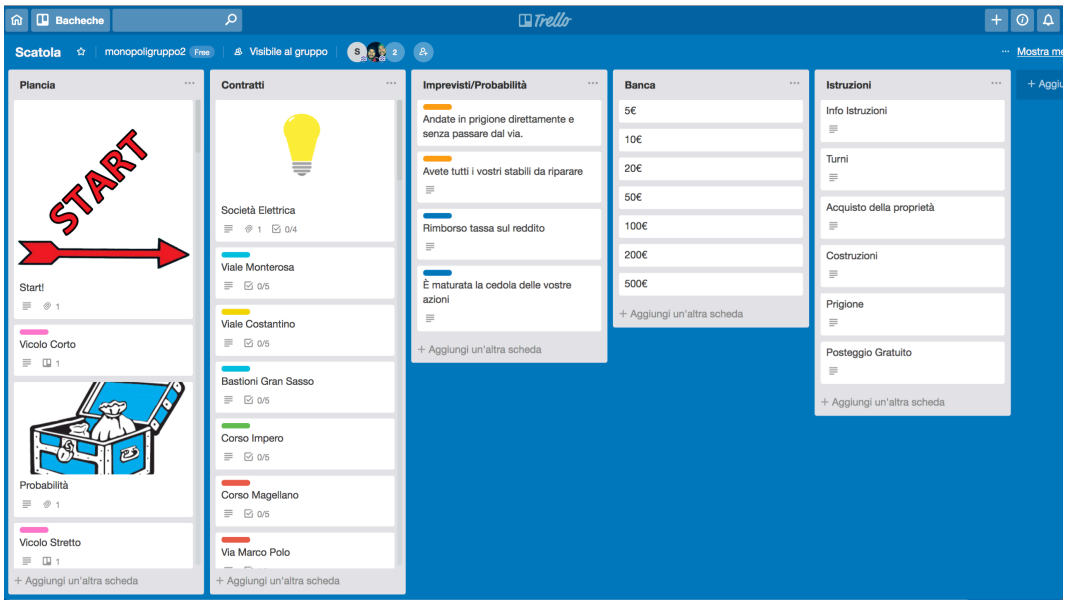

Figura 4.1: Bacheca di Trello con suddivisione in liste.

project manager e più in generale a team che sviluppano software. In realtà i milioni di utenti iscritti alla piattaforma, non sono tutti sviluppatori software. Trello è usato sia in ambito lavorativo, per l'organizzazione di piccoli-medi progetti, non necessariamente nell'ambito dell'IT, che nel quotidiano. Alcuni utenti infatti, pianificano la propria attività sportiva, utilizzando una lista per ogni giorno della settimana, le checklist per segnare il numero di ripetizioni di un esercizio ecc.., altri invece per segnarsi i migliori ristoranti di una città, usando le liste per rappresentare il tipo di cucina (e.g. mediterranea, giapponese, africana), altri ancora per decidere con un gruppo un itinerario di viaggio.
Insomma, gli ambiti di utilizzo di uno strumento come Trello sono molti, ed è probabilmente per questo che è uno degli strumenti di collaborazione più apprezzati.

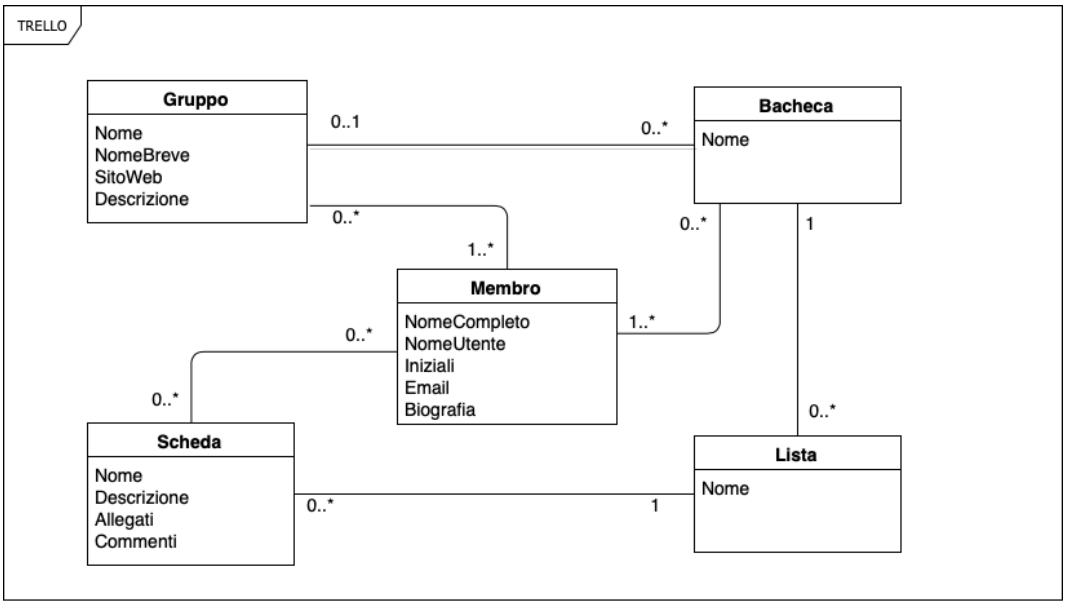

Figura 4.2: Diagramma UML delle classi di Trello.

4.3 Struttura

Bisogna pensare a Trello come una "meeting room" virtuale, in cui è possibile creare una quantità illimitata di lavagne, costituite da liste, su cui è possibile apporre dei "post-it".

La struttura di Trello è gerarchica: al livello più basso si posizionano le *Schede* (Cards), che contengono informazioni riguardo singoli task e possono essere paragonate ai post-it. Le schede possono essere ordinate e raggruppate in *Liste* e una serie di liste è posta su una *Bacheca* (Board). Come mostrato in Figura 4.1, la bacheca rappresenta la lavagna, divisa in colonne (Liste) sulle quali vengono "incollati" i post-it (Schede).
Entità al vertice sono i *Gruppi* (Organizations). Esistono due tipi di gruppi: Personal Team e Business Team. La differenza tra un personal e un business team (a pagamento) è che il secondo ha accesso a tool aggiuntivi per un maggior controllo delle bacheche e per le integrazioni con altre app.
Si può rendere una bacheca visibile solo ai membri del gruppo (privata) e dare la possibilità ai membri di unirsi alle bacheche del team.
Agli utenti aggiunti alle bacheche del gruppo possono essere assegnati privi-

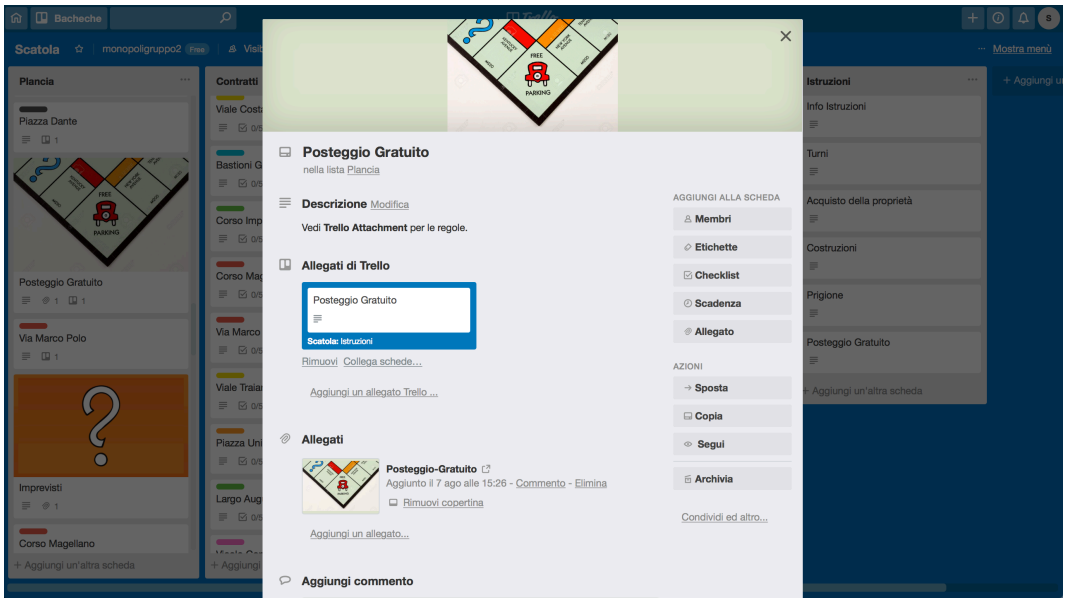

Figura 4.3: Back di una card di Trello.

legi differenti per permettere o meno di visualizzare, modificare o eliminare le schede di quella bacheca.

Le schede sono l'entità cardine intorno alle quali si sviluppa Trello: rappresentano un singolo task e sono composte da un titolo, possono contenere descrizione, allegati (link, foto, documenti), checklist, data di scadenza. È possibile inoltre assegnare uno o più membri ad ogni card ed aggiungere etichette personalizzate (colore, titolo) in modo da determinare facilmente il contesto a cui la card fa riferimento.

Ogni scheda ha due facce:

- **Front**: contiene un titolo e informazioni opzionali come immagini, etichette colorate e icone (Figura 4.1).
- **Back**: contiene informazioni dettagliate sulla scheda. Descrizioni associate (formattabili con il linguaggio di markup Markdown), allegati, immagini, checklist, data di scadenza e una sezione commenti. Un esempio è riportato in Figura 4.3.

Gli utenti possono, simultaneamente, creare e gestire bacheche e liste, aggiungere e muovere schede tra liste e bacheche differenti. Quando si hanno

nuove idee o dubbi riguardo uno specifico argomento, è possibile aggiungere commenti alle schede, creando conversazioni e coinvolgendo tutti o solo alcuni membri del team.
Trello offre la possibilità di integrare ad esso delle estensioni, i "power-ups", che offrono funzionalità aggiuntive: sincronizzazione delle date di scadenza con Google Calendar, monitoraggio del tempo di lavoro su ogni scheda, automazione di azioni eseguite più frequentemente e molto altro.
Inoltre, gli utenti possono configurare Trello per inviare notifiche via email, browser web o dispositivi mobili.

4.4 Power-Ups

I power-ups rendono le bacheche di Trello interattive.
Con aggiunta di funzionalità e integrazioni con piattaforme esterne, aiutano gli utenti di Trello a collaborare nel miglior modo possibile. In particolare permettono lo scambio di dati tra Trello e servizi esterni, offrendo un esperienza dinamica agli utenti. Molti sono resi disponibili direttamente da Trello, altri sono stati sviluppati da applicazioni terze.
Sono divisi in categorie in base ai loro scopi:

- **Dati analitici e report**: visualizzazione di statistiche riguardo schede e bacheche, ad esempio monitorare l'invecchiamento di una scheda.
- **Automazione**: automatizzare le azioni su schede e bacheche, ad esempio duplicare automaticamente schede.
- **Strumenti bacheca** : aggiunta di strumenti utili per l'organizzazione delle bacheche, ad esempio aggiungere un file Read Me per indicare istruzioni di utilizzo della bacheca.
- **Comunicazione e collaborazione**: rendere le interazioni tra utenti ancora più interattive, ad esempio avviare una chat video tra membri di un team.
- **Strumenti per sviluppatori**: integrazione con i più diffusi strumenti per sviluppatori, ad esempio Git.

- **Gestione file**: allegare file che si trovano su piattaforme di condivisone file come Dropbox e Google Drive.

- **Risorse umane e operazioni**: rintracciare automaticamente le date di scadenza per contratti, licenze software, assicurazioni e certificazioni dei dipendenti.

- **IT e gestione dei progetti**: integrazione con altri strumenti di collaborazione (e.g. Asana, Wrike e Jira).

- **Marketing e social media**: aggiungere alle schede post pubblicati sui più famosi social media, ad esempio è possibile allegare un "Tweet" alle schede.

- **Vendite e assistenza**: gestione di vendite di prodotti e assistenza ai clienti direttamente dalle schede.

4.5 Trello API

L'accesso programmatico ai dati di Trello può essere eseguito tramite le loro API. I dati vengono restituiti dalle API di Trello come una stringa di testo in JavaScript Object Notation (JSON) [14]. L'uso di JSON da parte di Trello, permette all'utente un controllo completo sui propri dati in maniera semplice.
A causa della natura nidificata di Trello (le schede sono in liste che, a loro volta, sono in bacheche), la maggior parte dei dati sono accessibili proprio come risorse nidificate.

Di seguito vengono elencate e descritte le API. Sono suddivise per entità e per ognuna di esse sono messi in risalto i più interessanti endpoint.
Gli endpoint riportati sono da posporre a "https://api.trello.com/1/".
Per Bacheche, Schede, Liste e Gruppi sono disponibili degli endpoint simili tra loro che permettono di compiere le stesse azioni su tutti gli oggetti. È possibile, ad esempio, richiedere informazioni sugli oggetti appartenenti ad un altro (e.g. le schede di una lista), aggiungere un oggetto ad un altro (e.g. la scheda ad una lista), creare ed eliminare oggetti.

Vengono riportati a titolo esemplificativo gli endpoint per le bacheche. Quelli per liste, schede e gruppi sono analoghi.

- Reperire la lista delle schede presenti sulla bacheca: **GET /boards /{idBoard}/cards**.
- Creare una nuova bacheca, specificandone il nome: **POST /boards/**.
- Eliminare una bacheca: **DELETE /boards/{idBoard}**.

4.5.1 Azioni

Le *Actions* vengono generate ogni volta che si verifica un'azione su Trello. Ad esempio, quando un utente cancella una scheda, viene generata un'azione "DeleteCard" che include: informazioni su scheda cancellata, lista in cui si trovava la scheda, utente che ha cancellato la scheda e l'id dell'azione. I dati relativi alle azioni possono essere recuperati utilizzando, tra le altre, le seguenti richieste:

- È possibile reperire la lista delle azioni effettuate su una scheda, lista o bacheca: **GET boards/{idBoard}/actions** elenca tutte le azioni per la bacheca data.
- Si può risalire all'utente che ha effettuato un azione tramite l'id dell'azione: **GET /actions/{idAction}/memberCreator**.

4.5.2 Gruppi

I gruppi rappresentano un insieme di membri e bacheche e, come detto, i membri possono avere permessi diversi (Amministratore/Normale). Con le richieste relative ai gruppi, è possibile, tra le altre:

- Reperire la lista dei membri appartenenti ad un gruppo: **GET /organizations/{idOrg}/members**.
- Reperire la lista dei membri che hanno ricevuto un invito al gruppo e a cui non hanno ancora risposto: **GET /organizations/{idOrg} /memersInvited**.

- Aggiornare i permessi di un membro all'interno del gruppo: **PUT /organizations/{idOrg}/members/{idMember}**.
- Aggiungere un logo al gruppo: **POST organizations/{idOrg}/logo** .
- Rimuovere un membro dal gruppo e da tutte le bacheche associate: **DELETE /organizations/{idOrg}/members/{idMember}/all**.

4.5.3 Bacheche

Ogni bacheca può appartenere o meno ad un gruppo e può contenere più liste.
Con le richieste relative alle bacheche, è possibile, tra le altre:

- Aggiungere/Rimuovere un membro alla bacheca, specificando, nel body della richiesta, l'email del membro da aggiungere/rimuovere: **[PUT/DELETE] /boards/{idBoard}/members**.
- Reperire le checklist presenti nelle schede all'interno della bacheca: **GET /boards/{idBoard}/checklists**.
- Creare una lista all'interno della bacheca specificata, indicando il nome della nuova lista: **POST /boards/idBoard/lists**.

4.5.4 Liste

Una bacheca può contenere diverse liste.
Ogni lista può essere archiviata (closed: true, ovvero non visibile) o meno (closed: false). Con le richieste relative alle liste, è possibile, tra le altre:

- Reperire l'elenco delle schede di una lista: **GET /lists/{idList}/cards**.
- Archiviare o ripristinare una lista: **PUT /lists/{idList}/closed**.
- Rinominare una lista: **PUT /lists/{idList}/name**.
- Archiviare tutte le schede di una lista: **POST /lists/{idList} /archiveAllCards**.

4.5.5 Schede

Le liste, in Trello, contengono le schede. Una scheda è associata esattamente ad una ed una sola lista.
Con le richieste relative alle schede, è possibile, tra le altre:

- Reperire l'elenco degli allegati presenti su una scheda: **GET /cards /{idCard}/attachments**. È possibile specificare nel body di quali allegati si vogliono ricevere informazioni.
- Reperire i membri che hanno votato per una scheda: **GET /cards/{idCard} /membersVoted**.
- Aggiornare un commento esistente, specificandone l'id: **POST /cards /{idCard}/actions/{idAction}/comments**.
- Spuntare un elemento della checklist: **PUT /cards/{idCard}/checklist /{idChecklist}/checkItem/{idCheckItem}**.
- Segnare una scheda come letta: **POST /cards/{idCard} /markAssociatedNotificationsRead**.
- Rimuovere un'etichetta da una scheda: **DELETE cards/{idCard} /idLabels/{idLabel}**.

4.5.6 Membri

Ogni utente registrato è un membro di Trello.
Utilizzando qualsiasi endpoint che contenga "**../members/{idMember}**", è possibile utilizzare **"me"** al posto dell'**"idMember"**, per ricevere informazioni riguardo l'utente attualmente loggato sul client che esegue la richiesta.

Con le richieste relative ai membri, è possibile, tra le altre:

- Reperire le bacheche a cui un membro è stato invitato: **GET /members/{idMember}/boardsInvited**.
- Reperire le notifiche che ha ricevuto un membro: **GET /members /{idMember}/notifications**.

- Associare un avatar al membro:

  POST /members/{idMember}/avatar.

- Aggiungere un nuovo "background" alle bacheche del membro:

  POST /members/{idMember}/boardBackgrounds.

- Eliminare una ricerca di un membro:

  DELETE members/{idMember}/savedSearches/{idSearch}.

4.6 Confronto con altri strumenti di collaborazione Agile

Gli strumenti di collaborazione trattati finora, hanno scopi, utilità e caratteristiche tecniche e funzionali differenti.
Si vuole fare un confronto (illustrato in tabella 4.1) tra Trello e i software descritti nel capitolo 2 da diversi punti di vista: piattaforme supportate, feature disponibili, prezzi e altro.

	Trello	Asana	Kanbanchi
About the product	Trello is a software tool that makes use of the idea of boards for projects and cards for tasks to create an efficient collaboration platform.	Asana is a popular task and project management tool that allows for an easier team collaboration and communication.	Kanbanchi is a dashboard application that is used to manage workflow.
Languages	20+ (Also Italian)	English	English
Supported devices	Windows Android iPhone/iPad Mac Web-based	Windows Linux Android iPhone/iPad Mac Web-based Windows Mobile	Windows Linux Mac Web-based Windows Mobile
Features	• Quick overview on front and back of cards • Easy organization with tags, labels and categories • Drag and drop functionality • Checklists, with progress meter • Uploading of files and attachments • Archiving of card records • Deadline reminders • Email notification • Assign tasks • Information retrieval and back-up • Developer API	• Activity feed • Add assignees, attachments, and hearts to tasks • Automatic updates to email/ inbox • Create custom calendars and views • Email bridge • My Tasks list and Focus Mode • Track tasks and add followers • Get notifications and reminders	• Activity stream • Attachments • Cards • Drag & drop cards between boards • Issue tracking • Kanban board • Search • Tablet and mobile access • Task management • Work with multiple cards
Popular client	Adobe, Tumblr, Trip Advisor, Fresh Direct, Anytime Fitness	CBS Interactive, Pinterest, Airbnb, Synthetic Genomics	Dolphin Research Center, Purcell Range Outfitters
Price	9,99$/month	9,99$/month	7,95$/month

Tabella 4.1: Confronto tra Trello, Asana e Kanbanchi. Tabella presa da: "https://comparisons.financesonline.com"

Capitolo 5

Trellopoly: requisiti e specifiche

5.1 Introduzione

Trellopoly è un progetto di gamification che prevede l'utilizzo dei componenti di Trello quali gruppi, bacheche, liste e schede, per giocare a Monopoly. Per raggiungere tale obbiettivo è stata sviluppata un'applicazione web responsive, esterna a Trello, che implementa dinamiche e funzionalità di gioco per fornire agli utenti una migliore user experience.
Per realizzare il progetto è stato quindi necessario lavorare su due fronti:

- Capire come utilizzare le componenti di Trello per giocare a Monopoly.
- Implementare un applicazione web che migliorasse la giocabilità e la user experience.

Per comprendere come utilizzare al meglio le componenti di Trello, sono state fatte diverse prove ed implementazioni logiche. È stato necessario anzitutto capire come legare tra loro gli oggetti che lo compongono ed associargli significati semantici: l'oggetto "bacheca" cosa può rappresentare nel Monopoly? qual'è l'oggetto in Trello che meglio rappresenta una casella sulla plancia? e quello per la pedina?
Le soluzioni trovate e descritte nelle sezioni successive sono solo degli esempi di come si possa implementare Monopoly utilizzando le componenti di Trello.

L'applicazione web è stata implementata utilizzando le API messe a disposizione da Trello introdotte nella Sezione 4.5 (Trello API). Grazie a queste

si è potuto sviluppare un'applicazione terza che gestisce le dinamiche di base del gioco.

Si è voluto trasferire sull'applicazione la minor logica di gioco possibile, in modo da far utilizzare il più possibile Trello ai giocatori.

5.2 Analisi dei requisiti

In riferimento al diagramma dei casi d'uso (Figura 5.1), sono due gli attori di Trellopoly: il client (il giocatore) e il server. Entrambi gli attori interagiscono con i due sistemi che lo compongono: L'app web e Trello.
L'applicazione web è utilizzata dal client per:

- Effettuare il Login con Trello: ogni altra operazione dipende da questa. Senza effettuare il login, infatti, non è possibile procedere.
- Scegliere il gruppo con cui giocare.
- Scegliere la pedina.
- Lanciare i dadi.

Il server gestisce la fase di login e il lancio dei dadi. Comunicando con il sistema Trello, inizializza la partita e muove il giocatore.
Il giocatore dovrà eseguire azioni specifiche su Trello, per tutte le attività che permettono lo svolgimento di una partita di Monopoly.
In particolare interagisce con Trello per:

- Acquistare il lotto sul quale si trova: ciò include il fatto che paghi la somma di denaro prevista alla banca.
- Pescare una carta (imprevisti/probabilità).
- Costruire case e alberghi.
- Pagare un giocatore.

Trellopoly è un sistema multipiattaforma, accessibile sia da desktop che da qualsiasi dispositivo mobile (tablet e smartphone)

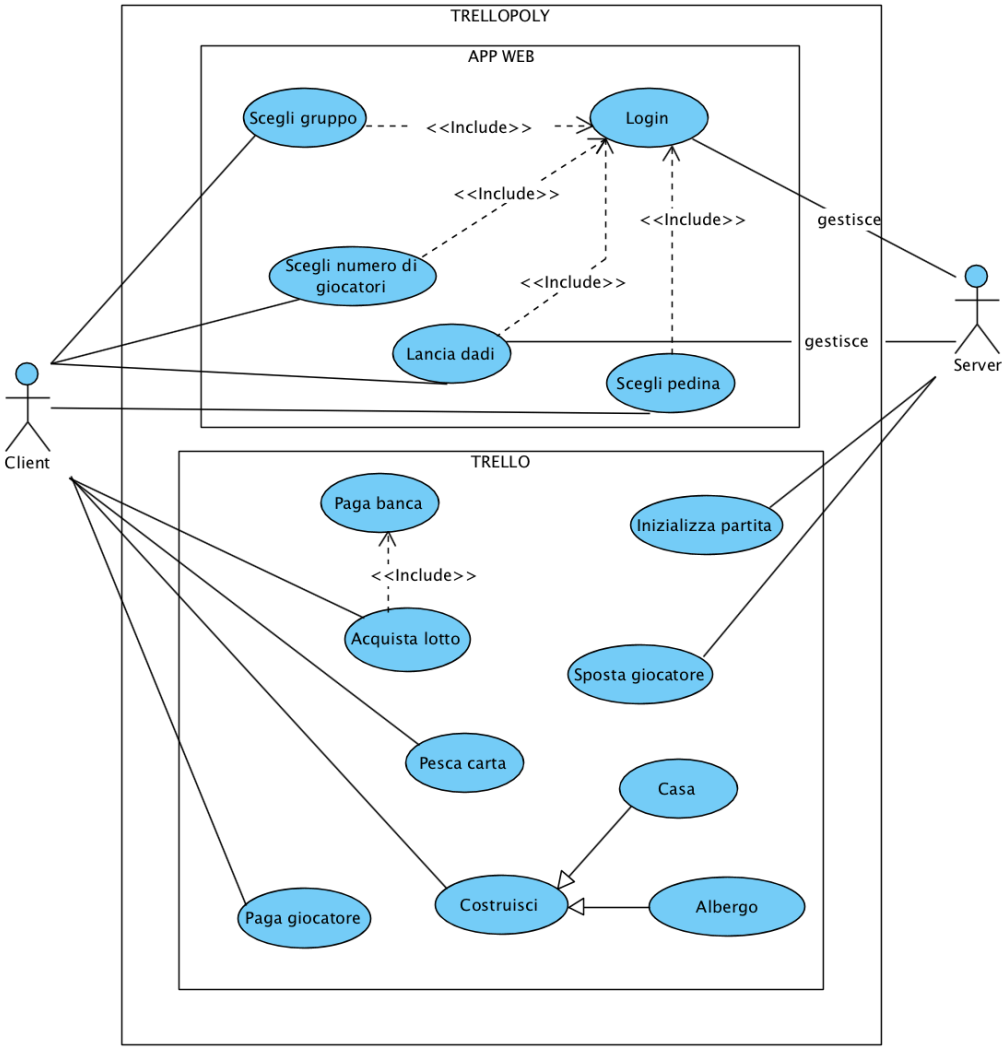

Figura 5.1: Diagramma dei casi d'uso.

Il sistema necessita di una corrispondenza di login contemporanea, su Trello e sull'app. In questo modo l'utente utilizza il desktop per visualizzare l'interfaccia di Trello e un dispositivo mobile per utilizzare l'applicazione.
L'utente sceglie il gruppo con cui giocare sull'applicazione; se quel gruppo ha già una partita in corso e l'utente non ne faceva parte, l'utente non può inserirsi. Altrimenti viene fatta scegliere la pedina con cui continuare il gioco.
Al termine della partita è lasciato il compito ad un membro del gruppo di impostare la partita come terminata, tramite l'apposito bottone.
Il lancio dei dadi è gestito dal server, esso permette a un solo giocatore alla volta di tirarli, solamente quando è il suo turno. Tutti i giocatori vedono il lancio dei dadi durante tutta la partita, così da sapere chi li ha lanciati e per conoscerne il risultato.

Capitolo 6

Progettazione e architettura

In questo capitolo sono descritti i principi di progettazione utilizzati durante lo sviluppo del progetto.
Una sezione spiega come si è scelto di utilizzare le componenti di Trello, l'altra la logica di progettazione dell'applicazione.

6.1 Componenti di Trello

6.1.1 Gruppi

Ogni gruppo in Trello rappresenta un insieme di persone che giocano a Monopoly nella stessa partita.
I giocatori di ogni partita appartengono allo stesso gruppo, ed ogni giocatore può appartenere a più gruppi. Ogni membro del gruppo ha i privilegi da amministratore, in modo che ogni utente possa modificare, aggiornare o archiviare liste e schede. Ogni gruppo ha le proprie bacheche con cui può interagire per effettuare diverse attività previste dal gioco.
Un esempio di gruppo costruito per giocare in due persone è mostrato in Figura 6.1.

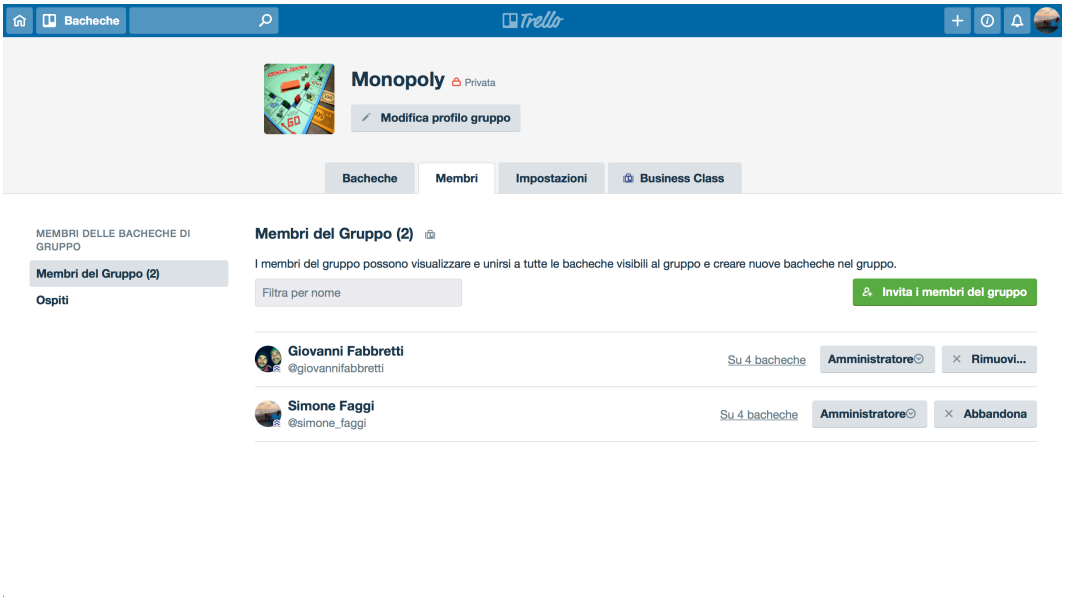

Figura 6.1: Esempio di gruppo su Trello.

6.1.2 Bacheche

Le bacheche rappresentano le entità di gioco del Monopoly. Esistono due "tipi" di bacheche:

- Scatola
- Giocatore

La bacheca "Scatola", come intuibile dal nome, rappresenta la scatola di gioco del Monopoly. Al suo interno ci sono infatti gli oggetti che compongono la scatola del Monopoly (e.g. plancia di gioco, contratti).
La bacheca "Giocatore" è la dashboard di un partecipante. Ognuno ha la propria bacheca sulla quale visualizza il suo stato di avanzamento nel gioco (e.g. posizione, contratti acquistati) e sulla quale può innescare azioni di gioco. Infine, lo sfondo delle bacheche "Giocatore" rappresenta graficamente la pedina associata.

6.1.3 Liste

Ogni lista contiene gli oggetti del Monopoly. Le liste sono state utilizzate per rappresentare i contenitori degli oggetti che compongono il gioco del Monopoly. Esistono due insiemi di liste: quelle per la scatola e quelle per il giocatore.
La scatola è composta dalle seguenti liste, ognuna rappresentate l'oggetto corrispondente.

- **Plancia**: la plancia è composta dalle 40 caselle previste dal gioco del Monopoly. Contiene le celle su cui i giocatori possono posizionarsi.
- **Contratti**: i contratti sono le carte rappresentanti i lotti che compongono la plancia. Sono acquistabili dai giocatori, nel qual caso li visualizzano nella loro bacheca.
- **Imprevisti/Probabilità**: contiene le carte "imprevisti" e "probabilità".
- **Banca**: la banca è una lista contenente tutti i tagli di banconote disponibili.
- **Istruzioni**: la lista "istruzioni" contiene informazioni sull'utilizzo delle liste sopra citate. Non contiene istruzioni sul gioco del Monopoly.

Un esempio di liste della bacheca scatola è mostrato in Figura 4.1.

La bacheca giocatore è costruita come segue ed è riportata in figura 6.3:

- **Contratti**: lista dei contratti acquistati dal giocatore.
- **Posizione**: contiene la scheda che rappresenta la posizione attuale del giocatore. È quindi una lista composta sempre da una ed una sola scheda.
- **Soldi**: elenco delle banconote possedute dal giocatore.

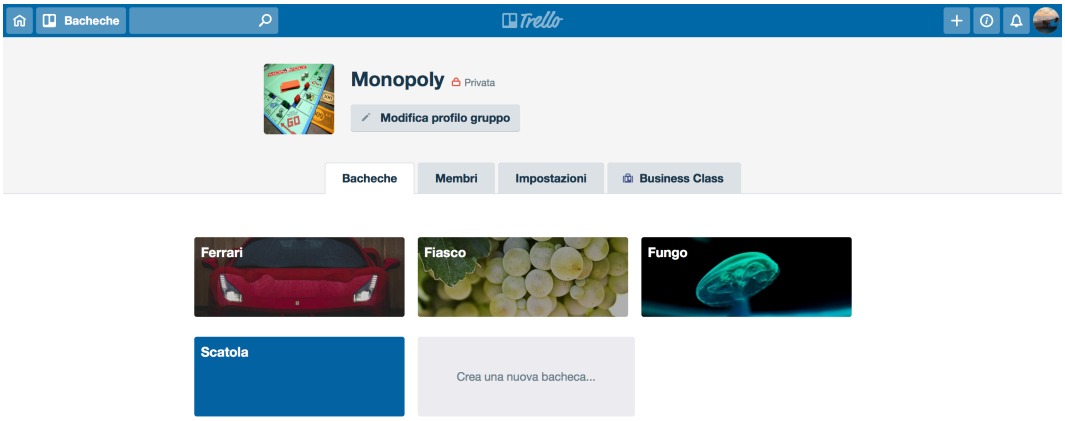

Figura 6.2: Esempio di bacheche su Trello, rappresentante bacheche Giocatore e bacheca Scatola.

6.1.4 Schede

Essendo la schede componente base di Trello, con esse si è deciso di rappresentare gli oggetti fisici del gioco del Monopoly: le carte, le caselle e le banconote.

Carte

Esistono tre tipi di carte nel Monopoly: i contratti, gli imprevisti e le probabilità.
Sono state rappresentate come segue:

- Contratti: la scheda che rappresenta un contratto è composta da:
  - Nome: indicante la via o la società corrispondente.
  - Descrizione: prezzo del lotto, costo di una casa e ricavi.
  - Checklist: la checklist è costituita di cinque elementi, quattro per le case e uno per l'albergo. Ogni item "checked" indica che è stato costruito il numero "checked" di case/alberghi.

Figura 6.3: La bacheca di un giocatore.

- Imprevisti e Probabilità: hanno entrambe la stessa struttura (Esempio in Figura 4.1):
  - Titolo: riporta il titolo dell'imprevisto/probabilità
  - Descrizione (opzionale): approfondisce lo scopo della carta.
  - Etichetta: di colore differente (arancione per gli imprevisti, blu per le probabilità).

Caselle

Le caselle sono le celle sulla plancia.
Sono 40, ognuna rispettiva ad una posizione che può essere assunta dal giocatore. Ogni casella è composta da (Figura 4.2):

- Nome: Indica via/società/imprevisti/probabilità/prigione.
- Descrizione: indica il valore di acquisto del contratto corrispondente.
- Etichetta: rappresenta la colorazione della casella sulla plancia.

- Allegato: link alla scheda che rappresenta il contratto corrispondente nella lista Contratti.

Banconote

La banconota è composta unicamente dal titolo. Esso indica il valore della banconota corrispondente (Figura 4.1).

6.2 Regole per l'utilizzo delle componenti di Trello

Ogni componente in Trello può essere utilizzata per giocare. Ci sono però dei vincoli di utilizzo, dovuti alla natura degli oggetti in Trello e alle regole imposte dal Monopoly:

- Le schede nella lista "Plancia" vengono *copiate* nella lista "Posizione" del giocatore che si trova su quella casella. Ciò è dovuto dal fatto che, nel Monopoly, su una stessa casella si possono trovare più giocatori.

- Le schede nella lista "Contratti" della scatola vengono *spostate* nella lista "Contratti" del giocatore che lo compra. Infatti ogni contratto può appartenere ad un solo giocatore.

- Gli imprevisti/probabilità, una volta usati, vanno *archiviati*. In questo modo non è possibile ripescare la stessa carta più di una volta.

- Le banconote vanno *copiate* nella lista "Soldi" di un giocatore, in modo che la lista "Banca", della Scatola, abbia sempre disponibilità economica.

6.3 Applicazione Web

Come detto, il compito dell'applicazione è quello di aggiungere giocabilità e dinamiche di gioco.
L'idea di base è che l'utente utilizzi un desktop per visualizzare la propria

Figura 6.4: Home di Trellopoly

bacheca su Trello e un dispositivo mobile per giocare tramite l'applicazione. Trellopoly comunica con Trello tramite le API messe a disposizione dallo stesso e con queste dà la possibilità agli utenti di effettuare azioni su Trello visibili direttamente sul proprio desktop.
L'app richiede il login con Trello tramite il quale è disponibile il token dell'utente, utile per effettuare le richieste alle API di Trello.

Sono due gli scopi principali dell'applicazione: inizializzare la partita e lanciare i dadi.

Inizializzazione

Una volta effettuato il login, si ha la possibilità di inserire il nome del gruppo con cui giocare. Il nome del gruppo corrisponde al "Nome Breve" associato al gruppo, reperibile su Trello andando sulla pagina del gruppo e cliccando su "Informazioni su questo gruppo".
Se non è presente una partita in corso con quello stesso gruppo, l'utente può scegliere il numero di giocatori (da un minimo di due ad un massimo che corrisponde alle bacheche di tipo "giocatore"). In caso contrario, se l'utente era un giocatore della partita in corso, rientra nella partita, altrimenti gli

viene chiesto di inserire il nome di un altro gruppo.
Scelto il numero di giocatori, si entra in una "sala d'aspetto"; la stanza è stata creata e si rimane in attesa dell'ingresso dei giocatori rimanenti. Non appena si raggiunge il numero richiesto, viene mostrata contemporaneamente a tutti i giocatori la schermata per scegliere la pedina con cui giocare. Durante il caricamento della pagina la partita precedente viene "resettata"
Il reset consiste nel "mettere a posto la scatola" della partita precedente e inizializzare la nuova partita. In particolare, i contratti dei Giocatori vengono rimessi nella scatola, la posizione e i soldi archiviati e la Scatola ritorna allo stato iniziale.
I giocatori, a questo punto, scelgono la pedina con cui giocare. L'applicazione distribuisce quindi i contratti alle pedine coinvolte e posiziona quest'ultime nella casella della plancia "Start!".
I giocatori hanno così, sulla propria bacheca di Trello, i contratti e il denaro iniziali e la posizione settata a quella di partenza (Start!).

Lancio dei dadi

Scelta la pedina, ai giocatori viene presentata un interfaccia che permette di lanciare il dado.
È l'applicazione che gestisce i turni e sceglie l'ordine di lancio dei dadi.
Tutti i giocatori hanno l'impressione di vedere gli stessi dadi. Ad ogni turno infatti, ogni giocatore vede il lancio dei dadi da parte degli altri giocatori.
Il lancio dei dadi provoca lo spostamento sulla plancia del giocatore per un numero corrispondente di caselle. Il giocatore sulla sua bacheca di Trello vede cambiare la sua posizione, e, in base a dove capita, decide quali azione compiere (e.g. acquistare un terreno, andare in prigione); azioni da compiere su Trello tramite desktop.

6.3.1 Statistiche

La sezione "Statistiche" è raggiungibile tramite l'apposito bottone situato sulla navbar. Tramite esso si accede alla sezione relativa alle statistiche di gioco. Questa serve per rappresentare lo stato di avanzamento di gioco. Mostra due istogrammi:

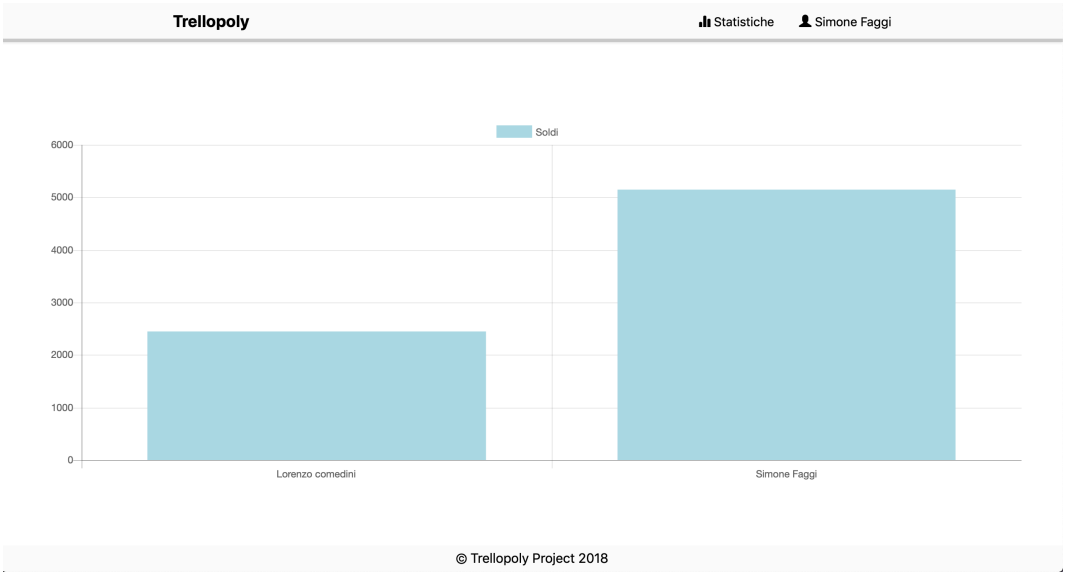

Figura 6.5: Sezione statistiche relativa ai soldi

- **Soldi**: rappresenta la situazione economica dei giocatori in questione, mostrando quantità di denaro di ogni giocatore e differenza tra questi.
- **Contratti**: mostra la quantità di contratti posseduti da ciascun giocatore.

Tramite questi istogrammi si è voluto rappresentare la gamification del progresso delle partite. In altri termini, la corrispondenza tra progresso di sviluppo da parte di programmatori e progresso di gioco da parte dei giocatori.

Burndown chart

È inoltre interessante mostrare come un burndown chart potrebbe rappresentare lo sforzo (effort) effettuato e rimanente durante il corso di una partita.
È una funzionalità che non è stata implementata, se ne vuole quindi dare un esempio esplicativo, tramite la figura 6.7, di come un burndown chart potrebbe rappresentare lo sforzo durante le partite.
La figura mostra lo sforzo di una partita di tre giocatori con una durata prevista di 3 ore. La partita termina in circa 2 ore e 20 minuti.

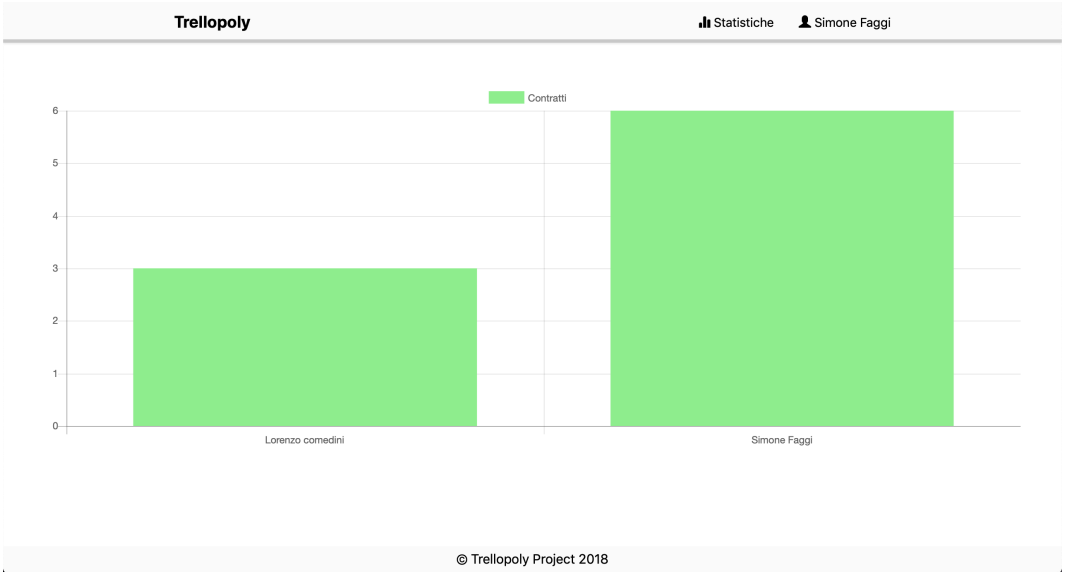

Figura 6.6: Sezione statistiche relativa ai contratti

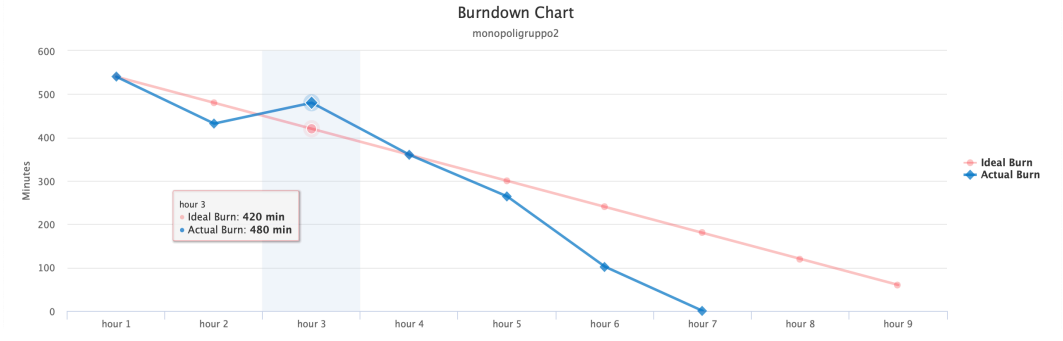

Figura 6.7: Burndown chart

6.4 Comunicazione tra le parti

L'idea di base del progetto è che l'utente utilizzi due dispositivi, avendo accesso a due interfacce differenti: una per effettuare azioni su Trello tramite l'applicazione web, l'altra per effettuarle su Trello stesso. L'utente interagisce contemporaneamente, tramite dispositivo mobile e pc, con l'applicazione web e con Trello.

6.4.1 Client-Server

Il client (utente) comunica con il server dell'applicazione tramite l'interfaccia della stessa. Ad ogni azione effettuata dall'utente, il client effettua una richiesta al server.
Il server comunica con il client in due modi:

- Rispondendo alle richieste del client
- Mandando messaggi broadcast

I messaggi broadcast inviati dal server a tutti i client servono per sincronizzare l'interfaccia di tutti i client connessi. Ogni client riceve lo stesso messaggio e si comporta esattamente allo stesso modo. Sono utilizzati principalmente per la sincronizzazione del lancio dei dadi e la gestione dei turni, ma anche per avvisare gli utenti in attesa dell'arrivo di tutti i giocatori.

6.4.2 Server-Trello

Il server dell'applicazione, ricevuta la richiesta dal Client, effettua la richiesta relativa a Trello tramite le API precedentemente descritte. Quindi restituisce una risposta al client.

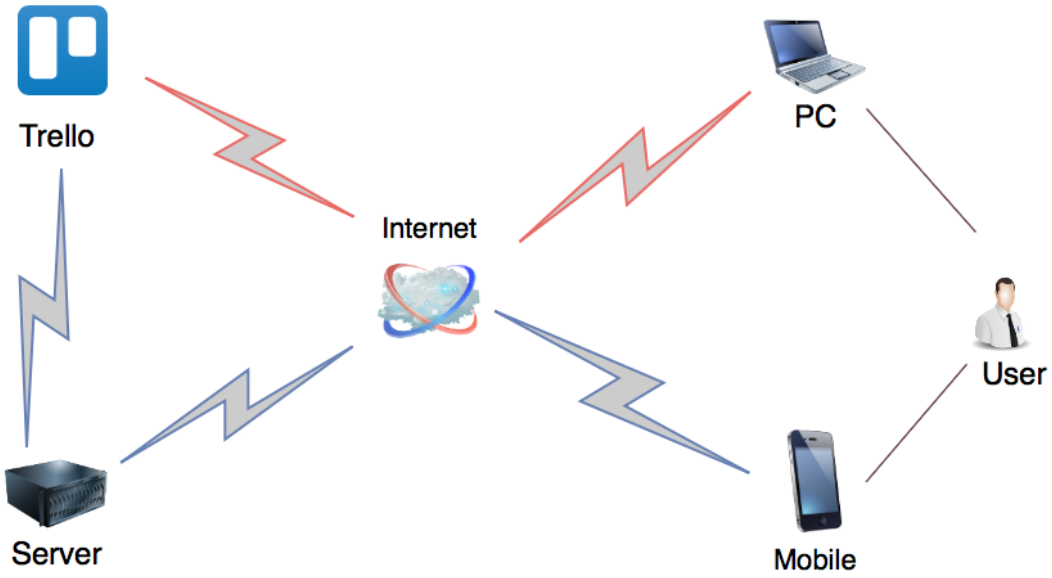

Figura 6.8: Comunicazione tra le componenti di Trellopoly

Capitolo 7

Sviluppo e test

7.1 Tecnologie utilizzate

7.1.1 Node.js

Node.js [15] è un ambiente open source cross-platform per applicazioni server-side scritte in Javascript.
Il modello di networking su cui si basa Node.js non è quello dei processi concorrenti (multithreading), ma I/O event-driven: ciò vuol dire che Node richiede al sistema operativo di ricevere notifiche al verificarsi di determinati eventi, e rimane quindi in ”sleep” fino alla notifica stessa: solo in tale momento torna attivo per eseguire le istruzioni previste nella funzione di callback, eseguita una volta che il risultato dell'elaborazione del sistema operativo è disponibile.
Node ha un unico thread che gestisce tutte le richieste, ciascuna richiesta viene prelevata da una "Event queue". Le operazioni sono effettuate su thread separati e paralleli e non accessibili tramite programmazione. Terminate le operazioni, i risultati sono messi nell' Event Queue come eventi di callback.

7.1.2 NPM: Node Package Manager

NPM è il principale software utilizzato per maneggiare i moduli di Node.js e consente di condividere il codice per problemi tipici tra gli sviluppatori JavaScript. La filosofia alla base di NPM è quella che se un problema è

stato già risolto da altri programmatori non ha senso doverlo sviluppare per conto proprio, poiché è possibile utilizzare la soluzione condivisa messa a disposizione di tutti. NPM oltre a consentire il riuso del codice, consente di tenerlo costantemente sotto controllo in modo da aggiornarlo e migliorarlo.
NPM è interamente scritto in JavaScript ed è stato sviluppato da Isaac Z. Schlueter. Utilizzato prevalentemente lato server, NPM può essere utilizzato con browerify sul front-end.
NPM suddivide il codice in package o moduli.
Caratteristica di NPM è la semplicità di utilizzo: con il comando

```
npm init
```

un mini-wizard guida l'utente alla creazione del file "package.json", un file descrittore del progetto e delle sue dipendenze. Contiene informazioni quali:

- Autore
- Nome
- Versione
- Descrizione
- Entry point
- Dipendenze

I nuovi pacchetti possono essere installati tramite il comando:

```
npm install <library>  --save
```

Con questo comando, viene scaricata la nuova libreria e aggiunta al progetto aggiornando le dipendenze nel file package.json.

7.1.3 Express.js

Express è un framework per Node. In particolare consente di semplificare e velocizzare lo sviluppo di un server scritto in Node.
È utile al programmatore per implementare le API REST. Il suo punto di

forza sono le ”rotues”, ”strade” in italiano.
Esse hanno proprio il compito di instradare le richieste del client ai giusti endpoint del server. Ciò permette di avere una architettura di progetto solida e ben strutturata.

7.1.4 Web Socket Server

Utilizzare le WebSocketServer API permette l'interazione tra client e server senza la necessità di ricorrere a continue chiamate Ajax o altri meccanismi simili, bensì mantenendo una canale costantemente aperto che consenta lo scambio bilaterale di messaggi.
Questa tecnologia si basa su un protocollo (simile ad HTTP nell'handshake iniziale) che può essere gestito in modo trasparente dal programmatore tramite l'utilizzo di alcune librerie software.
La comunicazione si basa sull'utilizzo di un oggetto di tipo WebSocket, che può essere dichiarato in JavaScript lato client come segue:

```
var host = location.origin.replace(/^http/, 'ws');
var ws = new WebSocket(host);
```

Una volta definito l'oggetto socket, vediamo un esempio di codice che mostra i metodi principali per la gestione lato client della comunicazione via WebSocket su JavaScript.

```
\\On open connection Client-Server
ws.onopen = function() {
   console.log('websocket is connected...')
}


\\Handle incoming message
ws.onmessage = function(msg) {
   if (msg.data == "Inizializzo la partita...") {
     $("#error").html(msg.data);
     moveBar()
    getBoards(localStorage.getItem("organization"),
```

```
    , localStorage.getItem("token"));
  } else if (msg.data == "Attendo giocatori..."){
    $("#error").html(msg.data);
  } else {
    var msgParse = JSON.parse(msg.data)
    if (msgParse.resultDice != null) {
      $("#resultDice").text(msgParse.resultDice)
    }
    if (localStorage.getItem("idPlayer") == msgParse.id) {
      localStorage.setItem("myTurn", true)
      $("#launchDice").show()
    } else {
      localStorage.setItem("myTurn", false)
      $("#launchDice").hide()
    }
  }
}
```

Lato server invece, la comunicazione viene gestita come segue.

```
//set listening port
var wss = new WebSocket.Server({port: 40510})

//handle communication
wss.on('connection', function (ws) {
  ws.on('message', function (message) {
    console.log('received: %s', message)
  })
  ws.send("Connected")
})
```

L'oggetto WebSocket permette il "push" di messaggi broadcast a tutti i client connessi. Questa funzionalità è stata sfruttata nel progetto per implementare:

- **Sincronizzazione inizio partita**: una volta connesso il numero di giocatori previsti, il messaggio broadcast avvisa i client dell'inizio della partita. Quindi a tutti i giocatori in attesa viene mostrata contemporaneamente la schermata di gioco.

- **Lancio dei dadi sincronizzato**: ad ogni lancio del dado, viene inviato un messaggio broadcast a tutti i client contenente il risultato del lancio e il nome del giocatore che lo ha effettuato. Ciò permette anche di gestire i turni di gioco.

Di seguito il codice per inviare un messaggio broadcast.

```
function sendBroadcast(msg) {
   var wss = app.get("wss")
   wss.clients.forEach(function each(client) {
     client.send(msg);
   });
}
```

7.1.5 Ajax

Ajax [16], acronimo di "Asynchronous JavaScript and XML", è una tecnica di sviluppo software per la realizzazione di applicazioni web interattive (Rich Internet Application).
Lo sviluppo di applicazioni HTML con Ajax si basa su uno scambio di dati in background fra web browser e server, che consente l'aggiornamento dinamico di una pagina web senza esplicito ricaricamento da parte dell'utente.
Ajax è asincrono. I dati sono quindi richiesti al server e caricati in background senza interferire con il comportamento della pagina attualmente attiva.
Normalmente le funzioni richiamate sono scritte con il linguaggio JavaScript.
Ajax usa:

- XHTML e CSS per la grafica.

- DOM aggiornato dinamicamente.

- XML con XSLT, oppure HTML preformattato, per lo scambio di dati.

- Un'istanza della classe XMLHttpRequest, che consente al browser di dialogare in modo asincrono con il server.

7.1.6 jQuery

jQuery [17] è una libreria JavaScript che ha lo scopo di rendere la manipolazione di documenti HTML rapida e dinamica, la gestione degli eventi, le animazione e le funzionalità Ajax più semplici da implementare. Tutto ciò grazie ad un'API facile da usare e supportata da una moltitudine di browser web. Le sue caratteristiche permettono agli sviluppatori JavaScript di astrarre le interazioni a basso livello tra interazione e animazione dei contenuti delle pagine. L'approccio di tipo modulare di jQuery consente la creazione semplificata di applicazioni web e versatili contenuti dinamici.

```
$('#searchBarContainer > input').on('keypress', function(e) {
  if (e.keyCode == 13) {
    localStorage.setItem("organization", $(this).val());
    window.history.pushState({}, '', "/");
    window.history.pushState({}, '', "organization=" +
     + localStorage.getItem("organization"));
    loadOrganization(localStorage.getItem("is_log"))
  }
});
```

Lo snippet di codice sopra riportato, mostra come è possibile manipolare un elemento del DOM. Viene selezionato l'elemento che, nell'"index.hmtl", ha id=sarchBarContainer ed è figlio di un tag <input/>. Quando viene premuto un tasto sulla tastiera (keypress) che corrisponde al tasto "invio" (keycode=13) viene eseguito il codice che modifica l'url ed effettua la ricerca del testo inserito.

7.2 Server

Per lo sviluppo del server si è utilizzato il sistema Node+Express. Si è fatto quindi ampio uso delle routes.

Di seguito viene riportato il modulo ”main.js”, file che rappresenta il primo bivio che incontrano le richieste effettuate dal Client.

```
var express = require('express');
var router = express.Router();

router.use('/login', require('./login'));
router.use('/game', require('./game'));
router.use('/init', require('./init'));

module.exports = router;
```

Il comando router.use instrada qualsiasi tipo di richiesta HTTP nella route di competenza. Ad esempio, facendo riferimento al file “main.js”, le richieste del client che iniziano con “/login” vengono instradate verso il modulo “login.js”. Il modulo login.js contiene gli endpoint che gestiscono la fase di login con Trello: il server effettua le richieste a Trello e riceve un token associato all'utente.
Il modulo game.js gestisce invece la fase di gioco (lancio dei dadi e turni).
il modulo init.js gestisce la fase di inizializzazione spiegata precedentemente.

Il Server mantiene in memoria una struttura "gameMatches" contenete informazioni riguardo le partite. In particolare associa ad ogni partita i seguenti campi:

- **name**: il nome del gruppo con cui si sta giocando quella partita.
- **players**: la lista degli ID dei giocatori che partecipano.
- **nop**: number of players, (il numero di giocatori impostato da chi ha creato la stanza)
- **noc**: number of connected (il numero di giocatori connessi)a

- **boardLen**: numero di board disponibili. Essendo una board riservata alla Scatola, si potra giocare in (boardLen - 1) giocatori.
- **listPlanciaId**: ID della lista "Plancia" nella bacheca "Scatola". Utile per le richieste che il server fa a Trello.
- **startCardId**: ID della scheda "Start" nella lista "Plancia", utile durante l'inizializzazione per posizionare tutti i giocatori sulla casella stessa.
- **isStart**: indica se la partita è iniziata o meno.

```
Esempio di struttura "gameMatches":

{  name: 'monopoligruppo2',
     players: [ {
       5b2e14ec52806859fd0aebe8,
       5b3f10ed52813460af0abea4
      }],
     nop: 2,
     noc: 2,
     boardLen: 4,
     listPlanciaId: '5b8945a748e0061becbc4756',
     startCardId: '5b8945a748e0061becbc47ae',
     isStart: true
}
```

Di seguito è riportato il codice con cui, durante il "reset" prima dell'inizio di una nuova partita, vengono messi i contratti nella Scatola.

```
function moveContratti(idListPlayer, idListScatola, idBoardScatola,
, token, callback) {
  var options = {
```

```
    method: 'POST',
    url: 'https://api.trello.com/1/lists/' + idListPlayer +
    + '/moveAllCards',
    qs: {
      idBoard: idBoardScatola,
      idList: idListScatola,
      key: "XXX",
      token: token
    }
  };

  request(options, function(error, response, body) {
    if (error) throw new Error(error);
    var data = JSON.parse(body);
    callback()
  });
}
```

Il Server riceve una richiesta HTTP dal Client e viene chiamata la funzione

```
moveContratti(...)
```

Quindi il Server effettua una richiesta HTTP di tipo POST a Trello e, se termina con successo, continua il normale flusso di istruzioni chiamando la callback.

7.3 Client

Il client dell'applicazione è scritto in Javascript e fa uso della libreria jQuery.
Il compito del client è esclusivamente quello di gestire la visualizzazione della pagina web. In particolare modifica l'aspetto della pagina in base alle informazioni acquisite dal server. Il client, per ogni azione dell'utente, effettua

una richiesta HTTP al server. Il server elabora la risposta e la restituisce al client. A questo punto, con le informazioni e i dati ricevuti, il client modifica l'aspetto della pagina web, rendendo disponibili o meno determinate funzionalità all'utente finale.
Di seguito è riportato il codice con cui il client invia al server il numero di giocatori scelto dall'utente. Il server aggiorna la struttura "gameMatches" inserendo nel campo "nop" della relativa partita il numero di giocatori, decidendo se far attendere ad ogni al client connesso l'ingresso di altri giocatori.

```
$("#numberOP").on("change", function(select) {
  var nop = select.currentTarget.value
  $(".input-field").hide()
  $("#searchBarContainer").hide()
  $.ajax({
    url: '/api/init/nop?nop=' + nop + "&organization=" +
    + localStorage.getItem("organization") +
    + "&id=" + localStorage.getItem("idPlayer"),
    success: function(res) {
      if (res.success) {
        if (res.isStart) {
          moveBar()
          getBoards(localStorage.getItem("organization"),
           localStorage.getItem("token"));
        } else {
          initGame(localStorage.getItem("organization"),
           localStorage.getItem("token"))
        }
      } else {
        $("#players").html("");
        $("#error").html("<p>" + res.message + "</p>");
      }
    },
    error: function(err) {
```

```
      console.log("Error: " + err);
    }
  });
})
```

Il client rimane quindi molto "alleggerito" per quanto riguarda la fase computazionale e l'applicazione web risulta facilmente utilizzabile tramite un qualsiasi calcolatore.

7.4 Test

Il progetto è stato testato con un gruppo di ragazzi dell'Università di Bologna. Durante il test si è cercato di analizzare l'efficacia della tecnica di gamification che si è scelto di sviluppare, sia in termini didattici, sia in termini di giocabilità.
Il gruppo, composto da quattro studenti, è stato scelto in modo che due di questi si trovassero nello stesso luogo e i restanti in luoghi differenti. In questo modo è stato possibile comprendere ed analizzare le difficoltà di comunicazione che la distanza tra i giocatori avrebbe introdotto.
Del gruppo di studenti in questione fa parte anche il sottoscritto, che conosceva in maniera piuttosto approfondita Trello; degli altri ragazzi invece, uno ne aveva già sentito parlare e gli altri due non lo conoscevano.

Facendo riferimento allo scopo del gioco, quello di far avvicinare i giocatori a Trello insegnandoli ad utilizzare le sue principali funzionalità, essi hanno espresso di aver imparato ad utilizzare le funzionalità che lo svolgimento del gioco prevede di adoperare. Il gruppo ha comunque riferito che il gioco è ben strutturato ma migliorabile in termini di giocabilità. Al termine del gioco, infatti, è stata aperta una discussione durante la quale sono state espresse le impressioni da parte dei giocatori, lasciando spazio a consigli per migliorarne proprio la giocabilità. Alcune delle idee tratte dalla discussione sono state sviluppate nei giorni seguenti (e.g. lancio dei dadi visibile a tutti i giocatori), altre verranno sviluppate in seguito e descritte nel capitolo 8 (Conclusioni e

sviluppi futuri).
Per quanto riguarda la solidità del sistema, il gioco è risultato fluido e non ci sono stati problemi rilevanti nella gestione dell'interazione Applicazione-Trello.

Capitolo 8

Conclusioni e sviluppi futuri

Il progetto realizzato non è completo; è quindi da considerarsi un prototipo. È funzionante ma l'applicazione sviluppata non è un software stabile e rilasciabile. Saranno necessari miglioramenti da diversi punti di vista quali stabilità del codice e funzionalità.
Di seguito sono riportate alcune delle funzionalità aggiuntive che potranno essere introdotte nel progetto.

8.1 Statistiche per costruzioni

La sezione "statistiche" dell'applicazione attualmente consente la visualizzazione di istogrammi relativi a situazione economica e numero di contratti. In futuro potranno essere integrate statistiche per case/alberghi. L'istogramma mostra il numero di case e/o alberghi costruiti da ogni giocatore.
Con l'aggiunta di questa statistica, si avrà la possibilità di monitorare l'avanzamento della partita in modo più accurato, in modo da prevedere lo sforzo effettuato e mettere in risalto quello rimanente tramite un burndown chart.

8.2 Burndown chart

È stato presentato, nel capitolo 6, un esempio di burndown chart relativo allo sforzo della partita. È interessante l'integrazione di un burndown chart nell'applicazione, costruibile a partire dalle statistiche della partita rilevate

durante lo svolgimento della stessa e riportate nella sezione "statistiche".
Il burndown chart sarà costruito facendo una valutazione sulla situazione della partita e stimando lo sforzo rimanente di ciascun giocatore.

8.3 Distrubuzione carte

In questa versione, l'applicazione prevede la distribuzione dei contratti e dei soldi durante l'inizializzazione del gioco. Una funzionalità aggiuntiva potrebbe essere quella di distribuire dinamicamente le carte imprevisti e probabilità nel caso in cui un giocatore venga posizionato sulla casella relativa della plancia. Ciò prevede l'aggiunta di una lista sulla bacheca del giocatore che contenga le carte pescate. In questo modo le schede imprevisti e probabilità non andranno archiviate ma riposizionate nella scatola durante la fase di reset.
La lista aggiuntiva potrebbe permettere di far tenere al giocatore la carta per uscire di prigione

8.4 Direttamente in prigione

Questa funzionalità prevede che il giocatore venga posizionato direttamente in prigione nel caso in cui si trovi sulla casella della plancia "In Prigione!". Attualmente è il giocatore a doversi posizionare sulla suddetta casella.

8.5 Visualizzazione della plancia

Una visualizzazione interattiva della plancia da parte dei giocatori migliorerebbe certamente la giocabilità. La funzionalità prevista permette ad ogni giocatore di visionare una plancia dinamica su cui vede muoversi le pedine dei giocatori. Questa funzionalità potrebbe essere implementata sia sull'applicazione che su Trello. Nel secondo caso si potrebbe creare un terzo tipo di bacheca che contenga tante liste quante sono le caselle della plancia e tante schede quanti sono i giocatori in questione. Le schede vengono spostate tra le

liste dall'applicazione, posizionando il giocatore sulle caselle che gli spettano. Sulla plancia potranno essere visualizzate anche le costruzioni, rendendo il gioco ancora più dinamico e interattivo.

8.6 Conclusioni

In questa tesi è stato presentato un sistema di gamification per la didattica che ha l'intento di favorire studenti e non ad avvicinarsi allo strumento di collaborazione agile Trello. Il progetto che è stato sviluppato è solo un esempio di come si possa costruire un gioco a partire da uno strumento di collaborazione online.
Per lo sviluppo del progetto ci si è serviti di un account Trello per usufruire delle API messe a disposizione da Trello stesso. È stato utilizzato un account "Free", che permette di accedere a tutte le API messe a disposizione e dà la possibilità di implementare i power-ups (sezione 4.4). Trello permette agli utenti "Free" di utilizzare un solo power-up per ogni bacheca, mentre ad un account "Business" di attivare illimitati power-up sulle proprie bacheche. Lo sviluppo di power-ups per l'implementazione del progetto avrebbe permesso di aggiungere funzionalità e bottoni direttamente sulle bacheche e quindi di far utilizzare agli utenti solamente la piattaforma Trello. Poiché gli utenti avrebbero utilizzato esclusivamente Trello, si sarebbe riscontrato un notevole miglioramento sia nell'apprendimento da parte degli utenti che nella giocabilità e nel coinvolgimento degli studenti. Di contro, ogni giocatore che volesse partecipare all'attività didattica sarebbe stato obbligato ad avere un account "Business" a pagamento.
È per questo motivo che si è scelto di sviluppare un'applicazione terza che migliorasse la giocabilità e la dinamicità di gioco, in modo da coinvolgere maggiormente gli utenti. È stato quindi discusso lo sviluppo e la progettazione di tale applicazione, specificando logica e architettura del sistema. Esso è stato pensato e implementato in modo che l'utente interagisca il più possibile con Trello e per far sì che imparasse ad utilizzare le principali funzionalità fornite da esso.

Bibliografia

[1] Giuseppe Caroli Stefano Cencetti Giuseppe Fantori Luca Conti. Quando il gioco «fa» salute. *Il sole 24ore*, page 17, 2012.

[2] Sebastian Deterding, Miguel Sicart, Lennart Nacke, Kenton O'Hara, and Dan Dixon. Gamification. using game-design elements in non-gaming contexts. In *CHI'11 extended abstracts on human factors in computing systems*, pages 2425–2428. ACM, 2011.

[3] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. Manifesto for agile software development. 2001.

[4] Ken Schwaber and Jeff Sutherland. The scrum guide. *Scrum Alliance*, 21, 2011.

[5] Willard van Orman Quine. Two dogmas of empiricism. In *Can Theories be Refuted?*, pages 41–64. Springer, 1976.

[6] Kent Beck. *Test-driven development: by example*. Addison-Wesley Professional, 2003.

[7] Kent Beck and Erich Gamma. *Extreme programming explained: embrace change*. addison-wesley professional, 2000.

[8] Filippo Lanubile, Christof Ebert, Rafael Prikladnicki, and Aurora Vizcaíno. Collaboration tools for global software engineering. *IEEE software*, 27(2), 2010.

[9] Linus Torvalds and Junio Hamano. Git: Fast version control system. *URL http://git-scm. com*, 2010.

[10] Jeffrey Racine. The cygwin tools: a gnu toolkit for windows. *Journal of Applied Econometrics*, 15(3):331–341, 2000.

[11] Yevgeniy Brikman. *Terraform: Up and Running: Writing Infrastructure as Code*. O'Reilly Media, Inc., 2017.

[12] Trello Documentation. Trello documentation, 2011.

[13] Jason R Rich. *Working in the Cloud: Using Web-based Applications and Tools to Collaborate Online*. Que Publishing, 2017.

[14] Douglas Crockford. The application/json media type for javascript object notation (json). Technical report, 2006.

[15] Stefan Tilkov and Steve Vinoski. Node. js: Using javascript to build high-performance network programs. *IEEE Internet Computing*, 14(6):80–83, 2010.

[16] Jesse James Garrett et al. Ajax: A new approach to web applications. 2005.

[17] David Sawyer McFarland. *JavaScript & jQuery: the missing manual*. " O'Reilly Media, Inc.", 2011.