

Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

Corso di Laurea in Informatica



## Realizzazione dell'app mobile Trains

Progetto di Mobile Programming

:  
**Matteo Marchiori**  
Matricola 1236329

:  
**Giovanni Peron**  
Matricola 1237783



# INDICE

Frontespizio	i
Indice	iii
Elenco delle figure	v
<b>1 INTRODUZIONE</b>	<b>1</b>
1.1 Obiettivi del progetto	1
1.2 Descrizione generale	1
1.3 Utenti target	1
<b>2 DESIGN DELL'APPLICAZIONE</b>	<b>2</b>
2.1 Interfaccia	2
2.2 Schermate dell'applicazione	3
2.2.1 Home	3
2.2.2 Stato del treno	4
2.2.3 Valuta il treno	5
2.2.4 Hai valutato	6
2.2.5 Il tuo Profilo	7
2.2.6 Informazioni	8
2.3 Gesture utilizzate	9
2.3.1 Gesture menù laterale	9
2.3.2 Gesture valutazione treno	9
<b>3 GAMIFICATION</b>	<b>10</b>
3.1 Tecnologie persuasive	10
3.2 Gamification e tipi di giocatore	12
3.3 Gamification: caratteristiche del sistema	14
3.4 Meccaniche di gamification	14
3.5 Meccaniche di gioco in Trains	16
3.6 Dinamiche di gioco e FBM (Fogg Behavior Model)	16
<b>4 IMPLEMENTAZIONE</b>	<b>18</b>
4.1 Ambiente di sviluppo	18
4.2 Tecnologie usate	18
4.3 Il framework Flutter	18
4.3.1 Descrizione	18
4.3.2 Vantaggi e svantaggi	19
4.3.3 Problemi riscontrati e soluzioni apportate	20
4.3.4 Comparazione con altri framework	20
<b>5 CONCLUSIONI</b>	<b>22</b>
5.1 Valutazione di Flutter	22
5.2 Persuasione e gamification	22

Bibliografia [23](#)

## ELENCO DELLE FIGURE

Figura 1	Schermata Home	3
Figura 2	Schermata Stato del treno	4
Figura 3	Schermata Valuta il treno	5
Figura 4	Schermata Hai valutato	6
Figura 5	Schermata Il tuo Profilo	7
Figura 6	Schermata Informazioni	8
Figura 7	Fogg Behavior Model	10
Figura 8	Architettura di Flutter	19



# 1

## INTRODUZIONE

La relazione descrive il progetto di Mobile Programming svolto da Matteo Marchiori e Giovanni Peron, il quale consiste in un'applicazione mobile con elementi di gamification.

### 1.1 OBIETTIVI DEL PROGETTO

Gli obiettivi del progetto sono:

- sviluppare un'applicazione mobile cross-platform;
- studiare e implementare alcune tecniche di gamification.

### 1.2 DESCRIZIONE GENERALE

Trains è un'applicazione mobile cross-platform utile agli utenti che usano i treni regionali per viaggiare da un posto all'altro. Dopo aver scelto il treno che vuole prendere, l'utente diventa un giudice e dà un voto alla carrozza del treno in cui si trova. Ogni utente può vedere la media dei voti dati al treno fino a quel momento, in modo da capire se conviene prendere quel treno, un altro, oppure cambiare mezzo di trasporto.

Gli elementi di gamification introdotti servono per coinvolgere l'utente e renderlo abituato all'uso dell'applicazione, di modo da farlo sentire soddisfatto e in modo da far crescere il numero di utenti che usano l'applicazione soprattutto in fase iniziale. Un altro obiettivo importante è quello di aiutare l'utente a capire se ci sono treni utili in fasce orarie inaspettate, per ottimizzare eventuali viaggi e da far usare di più i treni in alternativa a mezzi di trasporto più inquinanti.

### 1.3 UTENTI TARGET

Il pubblico di utenti che abbiamo considerato è generico e vario, perché chiunque può prendere un treno. L'applicazione vuole fornire un servizio ai viaggiatori che abitualmente prendono già il treno e coinvolgere e invogliare gli utenti meno abituali a usare un mezzo più eco-sostenibile rispetto ad altri.

# 2 | DESIGN DELL'APPLICAZIONE

In questo capitolo descriviamo gli aspetti di design visti a lezione che abbiamo considerato per la nostra applicazione.

## 2.1 INTERFACCIA

L'interfaccia dell'applicazione è stata implementata, tenendo in considerazione gli argomenti trattati nel corso, quindi abbiamo cercato di rispettare il più possibile:

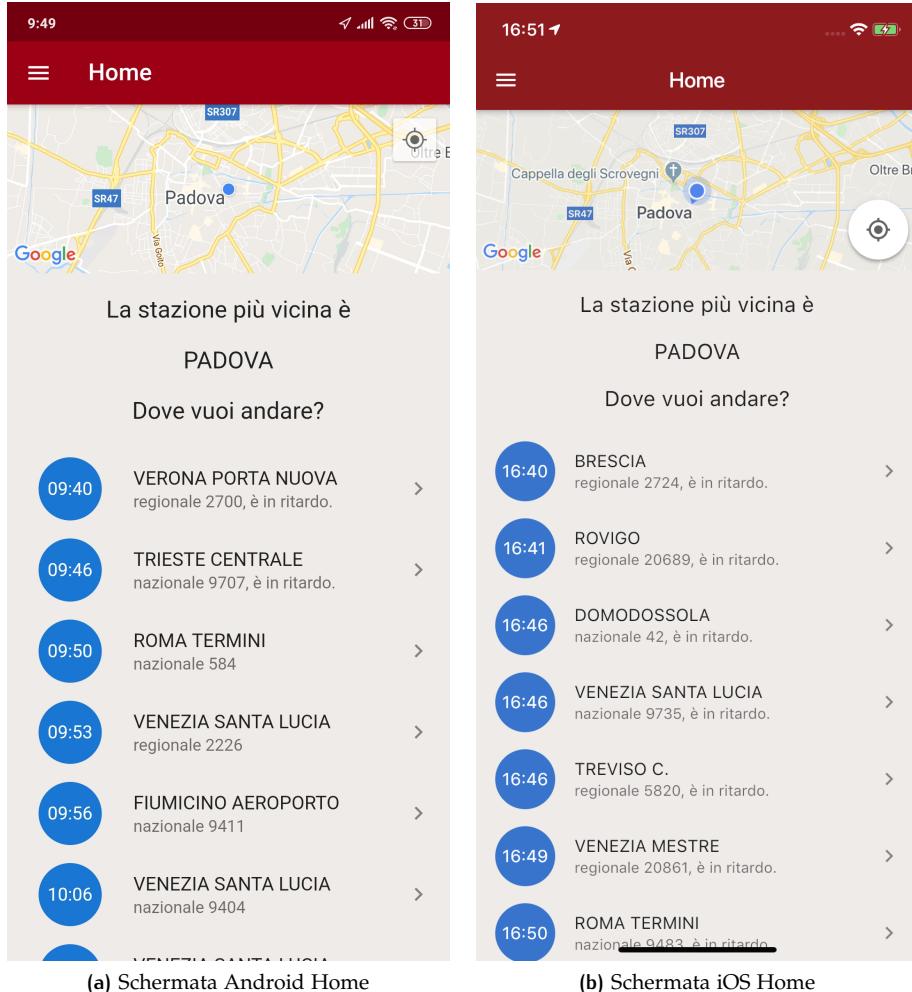
- le differenti dimensioni degli smartphones;
- la comfort zone (regola del pollice);
- la regola del content always on top;
- il fatto di evitare stack di pulsanti;
- il fatto di evitare l'inserimento di pulsanti a vantaggio di gesture intuitive;
- inserimento di contenuti just-in-time;
- dimensioni corrette degli elementi di interazione e degli spazi tra essi;
- l'utilizzo del progressive disclosure evitando il sovraffollamento dell'interfaccia;
- la scelta dei colori e delle dimensioni del testo corrette per garantire un buon livello di accessibilità.

Per mancanza di tempo abbiamo dovuto bloccare la visualizzazione delle schermate in modalità portrait, inoltre manca una riorganizzazione delle componenti per schermi più grandi (tablet ad esempio).

Siamo comunque riusciti a ottenere un'interfaccia con pochi pulsanti e di dimensione adeguata (rispettando sempre il limite minimo di dimensione di 7 mm e di distanza di 2 mm) e tutti i testi risultano abbastanza grandi e leggibili [Gaggi 2019b].

## 2.2 SCHERMATE DELL'APPLICAZIONE

### 2.2.1 Home



**Figura 1:** Schermata Home

Dalla schermata iniziale è possibile vedere la stazione ferroviaria più vicina, localizzata attraverso il GPS. Viene presentata una lista delle destinazioni raggiungibili tramite i treni in partenza dalla stazione più vicina identificata. Da questa lista di treni è possibile accedere a una pagina con più dettagli relativi al singolo treno. Questo è un esempio di interfaccia just-in-time che applica la progressive disclosure, perché vengono fornite le informazioni principali relative ai treni in partenza, mentre i dettagli per ognuno di essi possono essere visualizzati attraverso un'ulteriore interazione.

In questa schermata è possibile visualizzare un menù laterale, tramite una gestione di swipe a destra (si veda la sezione [2.3 a pagina 9](#)). Grazie a questo menù l'utente può raggiungere la pagina relativa al suo profilo, la classifica generale degli utenti, una pagina per rivedere il tutorial per la valutazione di un treno.

### 2.2.2 Stato del treno

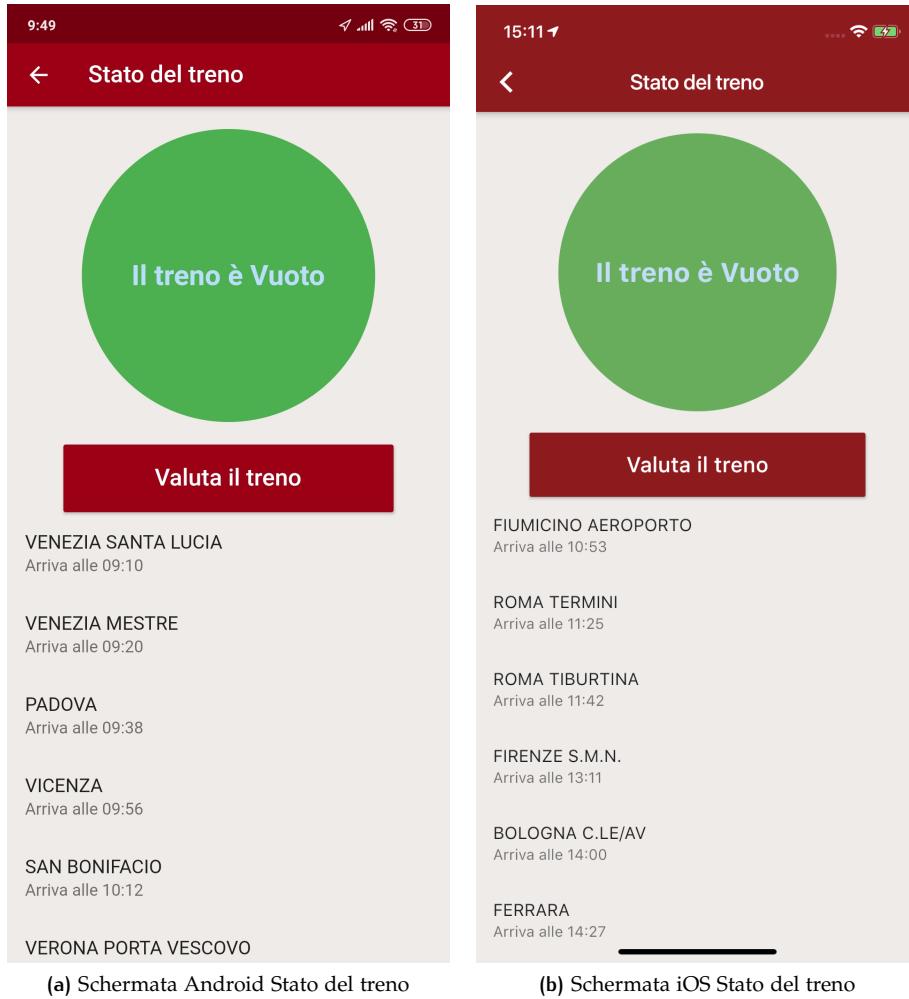
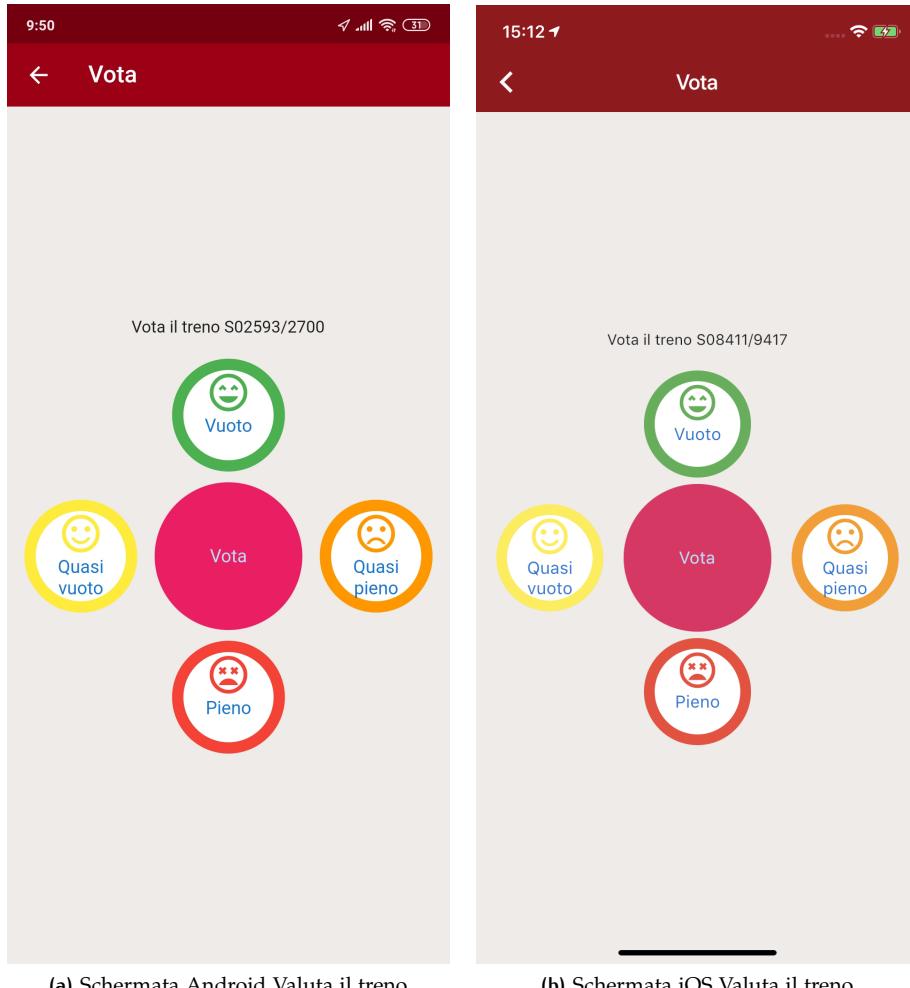


Figura 2: Schermata Stato del treno

In questa schermata l'utente può visualizzare i dettagli del treno selezionato, ovvero la valutazione media assegnata dagli utenti, una lista delle fermate del treno con gli orari di partenza previsti per le diverse fermate e un pulsante per proseguire alla schermata di valutazione del treno.

### 2.2.3 Valuta il treno



**Figura 3:** Schermata Valuta il treno

In questa schermata è presente un menù circolare che l'utente dovrà utilizzare per assegnare una votazione al treno, tra le quattro possibili. Per le prime tre valutazioni all'utente verrà proposto un tutorial veloce per spiegare la gestione di tipo press and drag da utilizzare per la valutazione (si veda la sezione [2.3 a pagina 9](#)). Una volta eseguita la valutazione l'utente verrà indirizzato alla pagina del profilo per poter visualizzare i suoi punteggi.

## 2.2.4 Hai valutato



(a) Schermata Android Hai valutato

(b) Schermata iOS Hai valutato

Figura 4: Schermata Hai valutato

È stata inserita questa schermata raggiungibile dall'utente solo in seguito ad una votazione. L'aggiunta di tale schermata è stata effettuata per includere all'interno dell'app un elemento di emotional design che potesse suscitare nell'utente un sentimento di gioia e soddisfazione nel vedere la simpatica mascotte dell'applicazione sorridere.

### 2.2.5 Il tuo Profilo

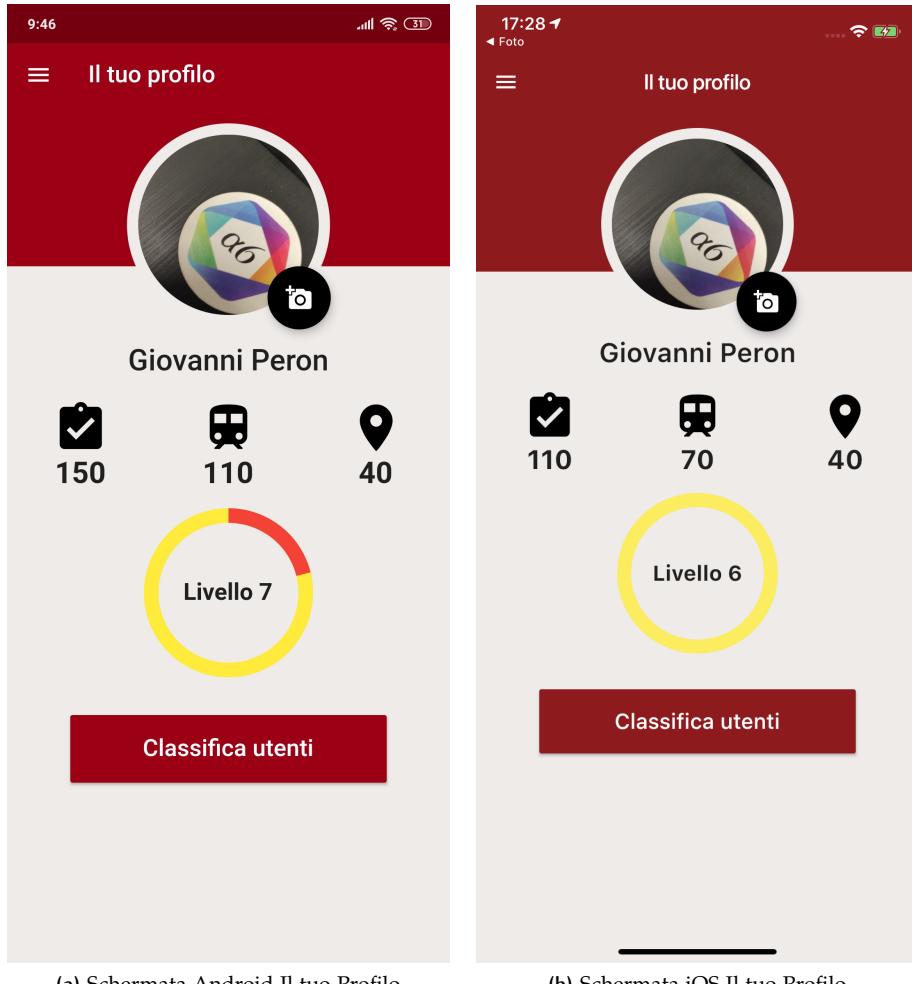


Figura 5: Schermata Il tuo Profilo

Nella schermata relativa al profilo, l'utente può visualizzare i suoi dati e i suoi punteggi. Nella parte alta dello schermo è presente l'immagine del profilo modificabile tramite un pulsante posto a lato di essa. Immediatamente sotto al nome del proprio account l'utente può visualizzare i punteggi relativi alle valutazioni eseguite, il numero di treni differenti valutati, e il numero di valutazioni eseguite da stazioni differenti. È presente inoltre un pulsante per poter accedere alla classifica generale.

## 2.2.6 Informazioni



(a) Schermata Android Informazioni

Figura 6: Schermata Informazioni

È stata inclusa una schermata contenente una descrizione dell'applicazione e le informazioni di base che l'utente deve sapere per poterla utilizzare. Sebbene queste informazioni dovrebbero risultare molto intuitive per l'utente è stato comunque pensato di aggiungere questa pagina per venire incontro agli utenti nel caso questi possano avere dubbi o non comprendere appieno il significato di qualche simbolo. In alternativa avremmo potuto utilizzare messaggi just-in-time ma è stata preferita questa soluzione per evitare messaggi troppo lunghi e per non interrompere troppo frequentemente l'esperienza dell'utente.

## 2.3 GESTURE UTILIZZATE

### 2.3.1 Gesture menù laterale

Per accedere al menù laterale evitando di inserire pulsanti è stato scelto di utilizzare una gestione di swipe a destra. Questo tipo di gestione è stata scelta perché consente all'utente di visualizzare il menù laterale tramite un gesto effettuabile dal lato sinistro della comfort zone, quindi sicuramente più comodo rispetto a un pulsante che avrebbe potuto prendere posizione nella parte alta dello schermo, e non in quella bassa per non creare uno stack di pulsanti in Android [Gaggi 2019b].

### 2.3.2 Gesture valutazione treno

Lo strumento per la valutazione del treno è stato realizzato attraverso un menù circolare, da cui è possibile fornire quattro voti. La votazione viene effettuata tramite una gestione press and drag che consente all'utente di scegliere il voto desiderato trascinando il simbolo corrispondente ad esso al centro dello schermo. Questa scelta vincola la disposizione e il numero delle opzioni del menù che non saranno più modificabili in futuro, perché si andrebbero a rompere le convenzioni che si creano con l'utente, tuttavia l'utilizzo di una gestione rende molto più veloce ed istantaneo il processo di valutazione del treno che è l'aspetto più importante dato che ipotizziamo che l'utente esegua tale azione in un ambiente precipitoso, frettoloso, come una stazione o un treno.

È stato scelto questo tipo di trascinamento dall'esterno all'interno perché, anche se forse potrebbe risultare meno comune rispetto allo spostamento verso l'esterno dello schermo, è stato pensato che il trascinamento verso l'interno può diminuire la probabilità di commettere errori. Presentare un unico oggetto centrale trascinabile verso l'esterno avrebbe reso la votazione dipendente dalla direzione del gesto data dall'utente (verso l'alto, il basso, a destra o a sinistra), mentre il fatto di dover prima scegliere l'oggetto e poi eseguire il trascinamento verso il centro dello schermo, consente di prestare più attenzione alla scelta del voto ed elimina la dipendenza dalla direzione della gestione che può essere facilmente sbagliata dall'utente per la fretta del gesto o per poca manualità.

Sono state scelte quattro valutazioni possibili, il numero di voti disponibili è volutamente pari per evitare che l'utente possa scegliere sempre un valore neutrale centrale, evitando quindi di pensare alla valutazione che sta esprimendo. Per i primi tre voti espressi dall'utente, viene fornito un tutorial just-in-time per fornire poche semplici istruzioni su come votare il treno su cui si è saliti. Dopo i tre voti il tutorial scompare, pur restando raggiungibile dal menù laterale.

# 3 | GAMIFICATION

Questo capitolo riassume le parti utili della tesi [Segato 2015], che descrive alcuni aspetti fondamentali delle tecniche di gamification.

## 3.1 TECNOLOGIE PERSUASIVE

Una definizione di tecnologie persuasive è la seguente:

[...]tecnologie interattive che mirano a cambiare il comportamento degli utenti attraverso la persuasione e l'influenza sociale senza usare inganni o forzature.  
[...]

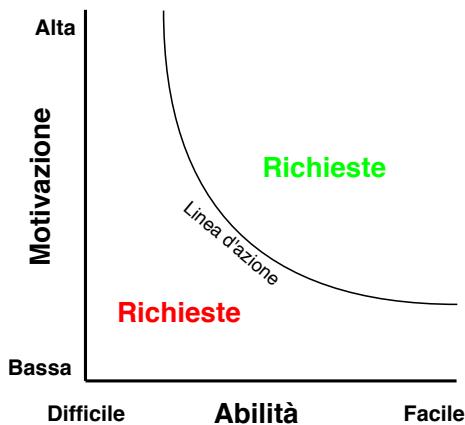


Figura 7: Fogg Behavior Model

B.J. Fogg ha ideato un modello per studiare il comportamento umano e come esso muta, prendendo in analisi le seguenti caratteristiche:

- motivazione dell'utente per svolgere determinati compiti;
- abilità per assumere un comportamento (risorse che l'utente ha a disposizione, come tempo, attenzione, capacità mentale);
- richieste da portare a termine, che in base a motivazione e abilità possono essere più o meno efficaci nell'influenzare un comportamento.

Oinas-Kukkonen e Harjumaa hanno invece creato il Persuasive Systems Design (PSD) model, con l'obiettivo di valutare i sistemi persuasivi e di descrivere il tipo di contenuto e di funzionalità software che dovrebbero essere presenti nel prodotto finale. In particolare PSD mette in evidenza:

- Information Technology (IT) non è mai neutrale rispetto ai comportamenti delle persone;

- se il sistema supporta l'assunzione di impegni, gli utenti probabilmente saranno più persuasi;
- un individuo che valuta attentamente il contenuto del messaggio persuasivo può essere approcciato con un percorso diretto, mentre un individuo meno riflessivo e che usa semplici indizi o stereotipi può essere persuaso attraverso un percorso indiretto;
- la persuasione è spesso incrementale: risulta più semplice iniziare le persone a fare una serie di azioni attraverso suggerimenti incrementali, piuttosto che con un unico suggerimento fornito una tantum;
- contenuti basati su informazioni non fidate o false impediscono l'obiettivo finale di cambiare le attitudini o i comportamenti;
- i sistemi persuasivi devono mirare a non essere intrusivi e ad essere semplici da usare.

Il modello PSD pone molta enfasi sul fatto che la selezione dei principi di progettazione rilevanti richiede un'attenta analisi del contesto per poter scegliere i momenti più opportuni in cui inviare i messaggi persuasivi.

Il contesto della persuasione è formato da:

**INTENTO :**

- il persuasibile, ovvero determinare chi è colui che si vuole persuadere;
- il tipo di cambiamento, in particolare se la persuasione mira al cambiamento di un'abitudine o di un comportamento che accade una volta sola.

**EVENTO :**

- contesto d'uso;
- contesto dell'utente;
- contesto tecnologico.

**STRATEGIA :**

- il messaggio da comunicare;
- il percorso da usare per raggiungere l'utente (diretto o indiretto).

Per poter essere in grado di progettare e valutare la persuasività di un sistema software bisogna comprendere sia il contenuto informativo che le funzionalità software. I principi di valutazione di sistemi persuasivi sono suddivisi in:

**PRIMARY TASK SUPPORT :**

- riduzione di comportamenti complessi in semplici attività;
- tunneling attraverso un processo o un'esperienza che può persuadere l'utente;
- personalizzazione di contenuti o servizi;
- self-monitoring delle performance o dello status dell'utente per raggiungere obiettivi;

- simulazioni per far osservare il collegamento tra causa ed effetto;
- prove di un comportamento per cambiare le loro attitudini o comportamenti nel mondo reale.

**DIALOGUE SUPPORT :**

- elogi per favorire l'apertura alla persuasione;
- ricompense per il completamento di richieste;
- promemoria degli obiettivi da raggiungere;
- suggerimenti per raggiungere gli obiettivi;
- similarità tra utente e sistema;
- liking: un sistema che è visualmente attraente;
- ruolo sociale.

**SYSTEM CREDIBILITY SUPPORT :**

- attendibilità;
- competenza;
- credibilità;
- real-world feel: un sistema che evidenzia i soggetti che ci lavorano;
- autorità;
- approvazione di terze parti;
- verificabilità attraverso fonti esterne.

**SOCIAL SUPPORT :**

- apprendimento sociale per similarità con altre persone;
- confronto sociale;
- normative influence attraverso pressione da persone simili;
- facilitazione sociale (altre persone mantengono il comportamento desiderato insieme a me);
- cooperazione;
- competizione;
- riconoscimento pubblico.

**3.2 GAMIFICATION E TIPI DI GIOCATORE**

Con il termine serious game ci si riferisce a quei giochi il cui scopo principale è educare e motivare gli utenti, attraverso la creazione di un'esperienza piacevole e accattivante. La componente ludica e di intrattenimento tipica dei giochi è quindi secondaria. Questi tipi di giochi richiedono all'utente di mantenere la concentrazione per un lungo periodo di tempo, ovvero sono immersive.

Il termine gamification è stato definito in molti modi diversi. Viene data la seguente definizione:

[...]

the use of game elements and game design techniques in non-game contexts.

[...]

Un sistema con gamification non è immersivo: il gioco viene svolto solo una volta ogni tanto, come gioco casuale, richiedendo quindi solo attenzione parziale da parte dell'utente. I tipi di giocatore che sono stati identificati sono i seguenti:

**SEEKERS** : giocatori a cui piace trovare qualsiasi cosa, dagli oggetti più strani e bizzarri a quelli più familiari; sono mossi dalla curiosità;

**SURVIVORS** : questi giocatori amano scappare dalle minacce spaventose, adorano il rischio;

**DAREDEVILS** : a questi giocatori piace superare piattaforme vertiginose, o correre ad alta velocità ma sempre mantenendo il controllo;

**MASTERMINDS** : in questa categoria rientrano tutti quei giocatori a cui piace risolvere puzzle ed escogitare strategie;

**CONQUERORS** : a questa categoria di giocatori piace sconfiggere nemici difficili;

**SOCIALIZERS** : in questa categoria rientrano tutti quei giocatori che amano andare in giro con altre persone fidate e aiutare altri giocatori;

**ACHIEVERS** : a questi giocatori piace collezionare qualsiasi cosa si possa collezionare.

Molti giochi e applicazioni utilizzano cooperazione e competizione, il supporto sociale e la pressione dei propri pari come motivatori per favorire il cambiamento di comportamento. Nonostante meccanismi basati sui gruppi siano promettenti, studi riportano risultati conflittuali sulla loro efficacia. L'influenza sociale può infatti variare in base alle condizioni e alle caratteristiche degli individui coinvolti.

Alcuni principi di design e osservazioni specifici per i giochi multiplayer sono:

- la maggior parte dell'influenza è limitata al più a un piccolo gruppo di amici;
- è più semplice per i giocatori coordinarsi all'interno di gruppi piccoli che hanno legami sociali oltre la competizione;
- visualizzare una rappresentazione online di sé in un gruppo può supportare l'attività di gruppo e creare un luogo online condiviso;
- alcuni giocatori mostrano interesse nel rappresentare un'immagine ideale di sé usando il loro avatar online;
- non è ragionevole pensare che tutti i giocatori scelgano un particolare stile di gioco e che non lo cambino mai;
- alcune persone possono essere particolarmente sensibili o non disposte a condividere informazioni su di sé. Occorre quindi dare ai giocatori il controllo per decidere quando condividere, quali informazioni e con chi.

### 3.3 GAMIFICATION: CARATTERISTICHE DEL SISTEMA

Dopo aver preso in considerazione i diversi tipi di giocatore e le relative caratteristiche, abbiamo deciso in base al target di utenti identificato in precedenza di dotare il sistema delle seguenti caratteristiche:

**PROFILO** : è presente una sezione profilo personalizzabile dall'utente, ma senza troppe opzioni, in modo da restare semplice e non sovraccaricare l'utente con complessità inutile;

**RANKING** : c'è una classifica in modo che gli utenti possano confrontare i propri punteggi in base ai voti espressi;

**PRIVACY** : non è obbligatorio registrarsi per votare un treno. È possibile giocare senza condividere i dati relativi al proprio profilo, restando anonimi relativamente alla classifica.

### 3.4 MECCANICHE DI GAMIFICATION

Zichermann descrive le principali game mechanics e descrive alcune delle più importanti game dynamics che si ritrovano nelle applicazioni web e mobile più popolari.

#### PUNTI RICOMPENSA :

- punti esperienza: guidano il giocatore e gli permettono di orientarsi nell'esperienza di gioco, di capire quali sono le attività più importanti. Non si perdono mai;
- punti vendibili: questi punti sono guadagnati e spesi dal giocatore per ricompense esterne (regali, monete, status) o interne al sistema e sono la base per la formazione di una economia virtuale;
- punti abilità: sono i punti che permettono al giocatore di guadagnare ricompense per le attività svolte e sono guadagnati portando a termine azioni specifiche;
- punti karma: questi punti portano un guadagno al giocatore solo se condivisi;
- punti reputazione: misurano il livello di attendibilità di un utente e sono usati per costruire un livello di fiducia tra le parti.

**LIVELLI** : fungono da marcatori per i giocatori per conoscere la loro posizione nell'esperienza e nel percorso di gioco. La loro difficoltà non è solitamente lineare: si tende ad iniziare con livelli più semplici e progressivamente si va verso quelli più complessi. Solitamente un gioco viene strutturato così che il passaggio di livello avvenga dopo aver accumulato un determinato quantitativo di punti e che dia accesso a nuovi contenuti.

**BADGE** : i badge marcano il completamento di obiettivi e il progresso nel gioco. In alcuni giochi possono anche rimpiazzare i livelli come marker del progresso. A livello di dinamiche, oltre che per mostrare uno status, le persone li

desiderano per poterli collezionare, per ragioni estetiche o anche provare la piacevole e improvvisa sorpresa di averne guadagnato uno nuovo.

**CLASSIFICHE** : le classifiche permettono di ordinare le performance degli utenti e di fare paragoni. Ne esistono di due tipi: le “infinite leader boards”, che comprendono tutti gli utenti del gioco, e le “no dissentive leader boards”, dove il giocatore viene posto al centro, senza badare alla sua posizione nel ranking, così che possa vedere solamente il giocatore prima e dopo di lui. La competizione risulta correlata all’aspirazione di diventare il migliore all’interno del proprio gruppo di amici.

**ECONOMIA DI GIOCO** : l’adozione di determinati tipi di punti permette la costruzione di vere e proprie economie all’interno del gioco, dove l’utente non può vincere a lungo senza comprare, guadagnare e consumare determinati beni virtuali.

**SFIDE** : le sfide corrispondono alle “missioni” che gli utenti devono compiere all’interno del gioco; danno un motivo al giocatore per continuare a partecipare al gioco, motivandoli a raggiungere risultati attraverso ricompense come trofei, badge da sbloccare. Inoltre, gli obiettivi generano confronto e competizione quando mostrati agli altri giocatori. È importante ricordare che a un giocatore novizio o ai primi livelli non vanno date sfide da livello master: è meglio dare sfide diverse per livelli diversi. Un particolare tipo di sfide sono quelle collaborative. Possono essere inserite nel gioco per creare esperienze di gruppo, dove i giocatori agiscono da soli ma i risultati vengono sommati insieme.

**ONBOARDING** : con onboarding si intende l’atto di introdurre un nuovo utente nel gioco. I primi minuti in cui un utente s’impegna in un gioco sono i più importanti, perché è in questo breve lasso di tempo che prende le sue decisioni. In questi piccoli istanti non bisogna dare spiegazioni o far leggere lunghi testi con regole o istruzioni, ma bisogna fargli provare l’applicazione. Inoltre in questa fase non bisogna chiedere all’utente di registrarsi: non c’è nulla allo stato attuale che lo obblighi a fornire dati personali ad un servizio che ancora non conosce. Il processo di onboarding rivela la complessità del gioco lentamente e non può fallire. Nel livello o (il tutorial) non ci devono essere scelte. Al giocatore vanno offerte azioni senza rischi, che non possono portare al fallimento, poi va ricompensato per aver completato l’azione con successo.

Altri due elementi di gamification che si ritrovano nelle applicazioni sono la personalizzazione e i feedback. La personalizzazione può apparire in diverse forme, ad esempio, lasciando decidere all’utente come vestire il proprio avatar nel mondo virtuale; anche solo il nome del giocatore sullo schermo può essere considerato un avatar, dando quindi possibilità di personalizzazione. Tuttavia se ci sono troppe scelte di personalizzazione, la soddisfazione dell’utente cala precipitosamente. I feedback sono le informazioni che indicano al giocatore dove si trova nell’esperienza di gioco e si vedono frequentemente nelle interazioni tra punteggi e livelli.

L’utilizzo di queste meccaniche risulta essere basilare per innescare gli activity loops, che caratterizzano i processi di gamification. Lo scopo di questi loop è quello di nutrire l’entusiasmo degli utenti al fine di convincerli a tornare ad utilizzare il sistema. Ne esistono due tipi: engagement loop e progression loop.

Engagement loop opera a livello di azioni individuali dell’utente: quando nell’utente la motivazione è sufficientemente forte, questa porterà l’utente stesso all’azione e quindi ad utilizzare il sistema. L’azione produrrà un feedback, il quale si tradurrà in una forma di motivazione per l’utente. Così il ciclo si ripeterà.

Il progression loop considera cosa accade all’utente durante l’utilizzo del sistema nel suo insieme. Questo loop dimostra la necessità di creare degli step intermedi bilanciati così da comunicare all’utente la facilità nel raggiungerli uno dopo l’altro. All’interno del gioco, solitamente questo si traduce nell’evoluzione del giocatore (newbie - expert - master). Il primo passo consiste in on-boarding, ovvero il processo di inserimento dell’utente nel gioco nel modo più veloce ed efficace possibile. Non appena l’utente ha imparato le regole ed inizia a raggiungere i livelli più alti, sono alternati momenti di riposo, con difficoltà minore, a momenti dove la difficoltà di gioco è più alta. Mantenere sempre la difficoltà alta renderebbe il gioco troppo difficile agli occhi dell’utente, stancandolo.

Le game mechanics sono quindi essenziali per innescare questi loop. Ci sono meccaniche di gioco che funzionano bene per una determinata tipologia di giocatori e meno bene per altre. Per questo motivo è importante considerare anche le dinamiche di gioco che sono l’evoluzione temporale e i pattern sia del gioco che dei giocatori che rendono il gioco più divertente. Buone dinamiche portano il giocatore nelle fasi successive al momento giusto, così che il giocatore si senta abile ed esperto. Se non ben progettate, portano a perdere giocatori nel tempo o per noia o per il dover affrontare sfide troppo complesse, che rendono il gioco meno divertente.

### 3.5 MECCANICHE DI GIOCO IN TRAINS

Per quanto riguarda Trains, abbiamo deciso di applicare le seguenti meccaniche di gioco:

- punti esperienza e punti abilità, che permettono al giocatore di salire di livello, in modo più facile all’inizio e più difficile man mano che il gioco prosegue, ma cercando di non fornire obiettivi impossibili;
- classifica globale per livello raggiunto (in futuro si potrebbero implementare più classifiche in base ai livelli e ai punti esperienza guadagnati dagli utenti);
- on-boarding con tutorial: l’utente viene introdotto al gioco senza troppe spiegazioni, e in modo che riceva immediatamente una gratificazione dopo aver votato un treno. Le istruzioni vengono visualizzate per i primi tre accessi, dopodiché scompaiono e diventano raggiungibili dal menù laterale.

### 3.6 DINAMICHE DI GIOCO E FBM (FOGG BEHAVIOR MODEL)

Le dinamiche di gioco create dall’attenta combinazione delle meccaniche sono un elemento molto importante perché permettono di modificare gli elementi chiave del FBM fino ad innescare il comportamento target. Le dinamiche offrono un modo

per motivare gli utenti attraverso feedback positivi, guadagnando badge, mostrando il proprio status, ecc. Forniscono anche due approcci per aumentare l'abilità dell'utente.

Il primo consiste nel far fare pratica all'utente con l'abilità richiesta, così da aumentarla fino a superare l'activation threshold.

Il secondo, invece, si concentra nel far percepire l'abilità come semplice. Per applicare questo approccio si possono utilizzare le seguenti strategie:

**DIVIDE-ET-IMPERA** : si spezza il lavoro complesso in task più piccoli e più semplici;

**COGNITIVE REHEARSAL** : si mostra come il lavoro è fatto e quanto è semplice;

**CASCADING INFORMATION** : istruzioni e informazioni sono rilasciate in piccoli frammenti per guidare l'utente in un task multi-fase.

# 4 | IMPLEMENTAZIONE

In questa sezione verranno descritte tutte le tecnologie utilizzate per lo sviluppo dell'applicazione.

## 4.1 AMBIENTE DI SVILUPPO

Per lo sviluppo dell'applicazione è stato utilizzato il sistema di controllo di versione Git, la realizzazione del progetto è stata eseguita da i seguenti calcolatori:

- OS: Windows 10 Home 64bit , Ubuntu 18.04, MacOS Catalina
- IDE: Microsoft Visual Studio Code, Android Studio, XCode

I dispositivi mobile utilizzati per sviluppo e il test dell'applicazione sono i seguenti:

- Acer To4 con Android 6.0 Marshmallow
- Xiaomi Mi 9T con Android 9.1 Pie
- iPhone XR con iOS (fornito dal dipartimento)

## 4.2 TECNOLOGIE USATE

Le tecnologie alle quali ci siamo appoggiati per lo sviluppo dell'applicazione sono:

**FRAMEWORK FLUTTER** è un framework cross platform di Google, lo descriviamo meglio in seguito;

**FIREBASE** platform as a service gestita da Google, fornisce funzionalità di analisi del comportamento degli utenti, memorizzazione di dati, gestione dei login e altro;

**SQlite** DBMS molto leggero per la gestione di basi di dati relazionali.

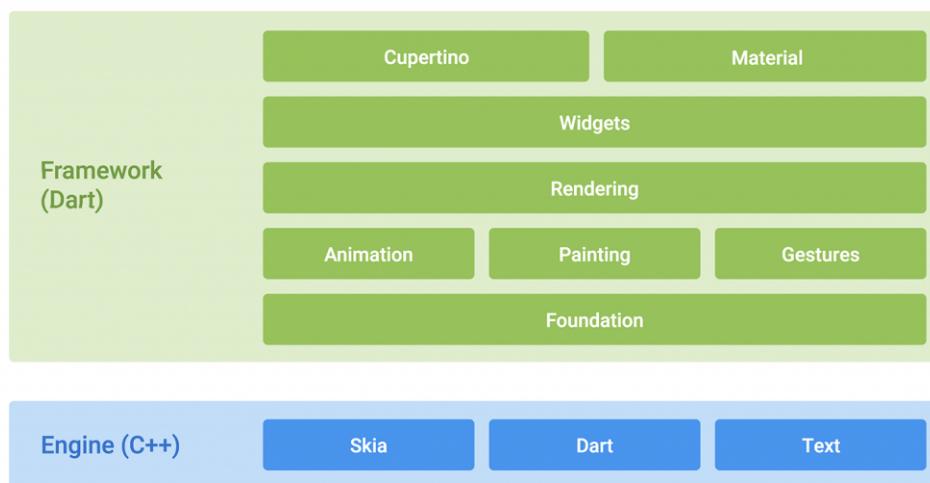
## 4.3 IL FRAMEWORK FLUTTER

### 4.3.1 Descrizione

Flutter è un framework cross-platform cross-compiled realizzato da Google, permette di realizzare applicazioni native eseguibili sia su Android che su iOS realizzando un singolo progetto in linguaggio Dart. La principale caratteristica di Flutter

è che l'interfaccia utente è composta unicamente da componenti Widget immutabili. I widget possono definire elementi strutturali (barre, pulsanti, menù), stilistici (font, colori) o di layout (padding), e così via; in pratica “tutto è un Widget”. Possono essere StatefulWidget, ovvero possedere uno stato ed essere quindi in grado di rispondere alle interazioni dell'utente, oppure StatelessWidget ovvero elementi statici.

Flutter è realizzato in C, C++, Dart e Skia (motore grafico). Le componenti vengono definite in Dart e renderizzate grazie ad un motore implementato in C++ che utilizza le librerie grafiche Skia. Il codice Dart viene compilato in codice nativo utilizzando la compilazione AOT (ahead-of-time) e viene eseguito grazie alla Dart Virtual machine. Il motore C/C++ invece viene compilato con NDK Android o con LLVM (Low Level Virtual Machine) in iOS, permettendo la compilazione in codice nativo [Gaggi 2018].



**Figura 8:** Architettura di Flutter

#### 4.3.2 Vantaggi e svantaggi

I principali vantaggi che il framework Flutter possiede sono:

**HOT RELOAD** consente di vedere e testare rapidamente i cambiamenti eseguiti sul codice mentre l'applicazione è in esecuzione, quindi rende lo sviluppo veloce;

**WIDGETS** possono essere utilizzati facilmente e sono definiti in stile Material design e Cupertino, permettono quindi la realizzazione di un interfaccia espressiva e flessibile e la loro riusabilità rende lo sviluppo più rapido;

**PERFORMANCE NATIVE** il framework essendo cross-compiled garantisce performance al livello di app native;

**DOCUMENTAZIONE** Flutter è ben documentato e possiede una community di utilizzatori attiva, inoltre è facilmente integrabile con Google Firebase il quale fornisce una rapida implementazione per l'archiviazione dei dati dell'applicazione e delle funzionalità di autenticazione degli utenti;

**IDE** sono disponibili vari plugin per il supporto in diversi IDE.

### 4.3.3 Problemi riscontrati e soluzioni apportate

Durante lo sviluppo dell'applicazione abbiamo riscontrato alcuni problemi legati alle performance dell'applicazione, che non ci soddisfacevano specie dal punto di vista delle animazioni. Un punto cruciale è la gestione dello stato nella gerarchia dei Widget: per fare in modo che l'applicazione abbia una resa grafica fluida è bene evitare il più possibile l'uso dei Widget stateful [Anonimo 2019].

Inizialmente abbiamo cercato di applicare design pattern noti per esperienze pregresse in modo inappropriato, ad esempio la dependency injection. Nelle app realizzate con Flutter è sbagliato realizzare le classi alte della gerarchia dei widget con uno stato e fare dependency injection sui widget figli, perché nel momento in cui lo stato di un widget genitore viene modificato, tutti i widget discendenti vengono ridisegnati, in particolar modo viene rieseguito il metodo build. Nel nostro caso avevamo implementato la schermata principale con uno stato formato da poche variabili che servivano per capire se un utente avesse mai eseguito l'accesso tramite il proprio account Google all'applicazione in precedenza oppure no e per conservarne alcuni dati di utilità. Queste variabili venivano in seguito iniettate nei widget discendenti per fornire le informazioni necessarie e renderli stateless.

Per risolvere il problema abbiamo deciso di sfruttare pesantemente l'architettura asincrona di Flutter e di eliminare completamente lo stato dove possibile. Usando in modo opportuno i FutureBuilder, oggetti appropriati per gestire l'asincronia nel metodo build (che viene invocato per costruire l'interfaccia), abbiamo ottenuto una resa molto più fluida.

Nei widget in cui non è stato possibile eliminare lo stato, abbiamo comunque fatto in modo di spostarlo il più in basso possibile nella gerarchia, di modo da mantenere delle buone performance.

Un altro problema è stata la gestione dei dati relativi ai voti degli utenti. Inizialmente abbiamo salvato direttamente i dati collegandoci a Firestore, un servizio offerto da Firebase per la gestione di database con una struttura dinamica orientata ai documenti. In questo caso però l'utente era costretto a eseguire il login con il proprio account Google per conoscere e salvare i punti guadagnati e il livello di esperienza raggiunto.

La soluzione in questo caso è stata quella di implementare un database locale con una struttura che emula quella del database su Firestore, ma in una versione relazionale, memorizzata attraverso l'uso di SQLite.

### 4.3.4 Comparazione con altri framework

Prima di usare Flutter avevamo valutato di usare un framework cross platform visto a lezione chiamato Appcelerator Titanium. Pur essendo un framework adatto allo sviluppo di un'app come Trains, ci sono stati diversi punti a sfavore che ci hanno portato a scegliere Flutter. Appcelerator Titanium è un framework di tipo interpretato, permette di ottenere app cross platform partendo da codice JavaScript [Gaggi 2019a]. È presente un interprete che consente di ottenere l'app con LAF nativo su Android e iOS. Pur essendo un framework con cui è possibile ottenere delle ottime performance (basso consumo di risorse e di batteria), ha un grosso punto debole relativo alla documentazione.

A un primo sguardo è presente molta documentazione relativa al framework, ma una volta che si inizia a usare ci si accorge che è molto sparsa e non ben organizzata. Appcelerator in particolare presenta diverse componenti, non solamente Titanium, che sono necessarie per l'implementazione di app con componenti lato server (database o altro) o per altre funzionalità (ad esempio Alloy è un framework MVC usabile con Titanium che permette di scrivere codice in modo più efficiente e ben organizzato). Il problema è che la documentazione di tutte le componenti è suddivisa in più siti, a volte è dispersiva, altre volte è presente in uno solo dei vari siti disponibili.

Un altro punto dolente è la quasi totale assenza di tutorial, che rende molto difficile un primo approccio al framework.

Infine i plugin che sono stati sviluppati per il framework non sempre sono completi, ad esempio quelli per l'integrazione con Firebase non lo erano, per cui è stato scelto Flutter, che al contrario dispone di una documentazione ben organizzata, di esempi e tutorial e di plugin funzionanti.

# 5 | CONCLUSIONI

## 5.1 VALUTAZIONE DI FLUTTER

Nel complesso valutiamo positivamente l'esperienza avuta con il framework Flutter. Permette di sviluppare app mobile anche abbastanza complesse in poco tempo, grazie alla documentazione fornita sul sito ufficiale, ai molti tutorial che si trovano in giro e ai diversi plugin che vengono sviluppati per gestire le diverse funzionalità messe a disposizione dai dispositivi mobile.

Alcuni punti sfavorevoli sono la difficoltà nell'implementare animazioni e le eccessive semplificazioni apportate nei tutorial, che potrebbero portare a fare errori che potrebbero compromettere le performance della app.

Probabilmente sarà possibile risolvere il primo problema man mano che nuovi plugin vengono rilasciati e che lo sviluppo del framework prosegue.

## 5.2 PERSUASIONE E GAMIFICATION

Per quanto riguarda lo studio delle tecniche di gamification, possiamo dire si tratti di un campo di ricerca interessante e utile nella persuasione degli utenti. Non abbiamo avuto modo di sperimentare tutte le tecniche illustrate, in primo luogo perché non tutte erano adatte al tipo di applicazione sviluppata (in particolare non trattandosi di un gioco immersive non sono stati presi in considerazione i giocatori survivors, daredevils e conquerors), in secondo luogo per il tempo destinato a conoscere meglio Flutter.

Sarebbe utile implementare le tecniche di gamification più orientate agli utenti che prediligono gli aspetti sociali, in particolare cooperazione, competizione e riconoscimento, ad esempio attraverso l'uso di Facebook per la gestione di una o più classifiche in base ai punti guadagnati dagli utenti.

# BIBLIOGRAFIA

## RIFERIMENTI BIBLIOGRAFICI

Gaggi, Ombretta

- 2018 *Flutter: approfondimento del framework crossplatform*, presentazione, Università degli Studi di Padova. (Citato a p. 19.)
- 2019a *I framework cross-platform*, presentazione, Università degli Studi di Padova. (Citato a p. 20.)
- 2019b *Mobile Design*, presentazione, Università degli Studi di Padova. (Citato alle p. 2, 9.)

Segato, Silvia

- 2015 *Climb The World: applicazione di tecniche persuasive ad un serious game*, tesi magistrale, Università degli Studi di Padova. (Citato a p. 10.)

## RIFERIMENTI WEB

Anonimo

- 2019 *StatefulWidget class: performance considerations*, dic. 2019, <https://api.flutter.dev/flutter/widgets/ StatefulWidget-class.html#performance-considerations>. (Citato a p. 20.)