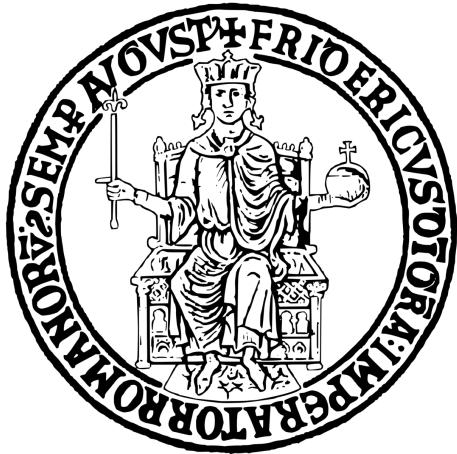


UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO
II

Scuola politecnica e delle scienze di base
Dipartimento di ingegneria elettrica e tecnologie
dell'informazione



Corso di Laurea in Informatica
Interazione Uomo-Macchina
Anno Accademico 2020/21

Specifiche, progettazione, implementazione, validazione e
testing di usabilità di un'applicazione Android

Team di sviluppo:
Quirile Alessandro N86002559
Telese Mauro N86002755

Committenti:
Prof. Cutugno Francesco
Prof. Grazioso Marco
Prof. Origlia Antonio

Marzo 2021

Indice

1 Presentazione dell'idea progettuale	4
1.1 L'applicazione	4
1.2 Le metodologie utilizzate	4
1.2.1 Gamification	4
1.2.2 Micro-interaction	5
1.2.3 Fidelizzazione	5
2 Individuazione del target degli utenti	7
2.1 Individuazione di pattern e variabili comportamentali	10
2.2 Sintesi delle caratteristiche e degli obiettivi rilevanti	12
2.3 Personas-type	12
2.3.1 Mariavittoria Ortona	13
2.3.2 Salvatore Ascione	14
2.3.3 Rebecca Giamundo	15
2.3.4 Giovanni Arpa	16
2.4 Personas goal e business goal	17
2.4.1 Primary personas-type goal e business goal	17
2.4.2 Secondary personas-type goal e business goal	18
2.4.3 Negative personas-type goal e business goal	19
3 Studio della valutazioni a priori dell'usabilità	20
4 Analisi delle funzionalità	23
4.1 Modello funzionale	23
4.1.1 Requisiti funzionali	23
4.1.2 Requisiti non-funzionali	23
4.1.3 Requisiti di dominio	24
4.2 Impiego delle risorse e pianificazione del lavoro	25
4.3 Modellazione dei casi d'uso	29
5 Prototipazione	30
5.1 Funzionale	30
5.2 Visuale	34
5.2.1 Fedeltà media	36
5.2.2 Fedeltà alta	47
5.3 Sistemica	57
5.3.1 Di analisi	57
5.3.2 Di design	63
5.4 Delle attività	69

6 Valutazione sul campo dell'usabilità soggettiva e oggettiva	75
6.1 Valutazioni euristiche	77
6.2 Test di usabilità	78
6.2.1 Test di compito	79
6.2.2 Test di scenario	87
6.3 5-seconds test	88
6.4 Analisi di coerenza delle icone	89
6.5 Analisi di coerenza dei colori	90
6.6 Analisi della legibility	92
6.7 Analisi della readability	93
7 Confronto coi competitor	94
7.1 Chi sono i nostri competitor?	94
7.2 Analisi esterna	94
7.3 Analisi interna	95
7.4 SWOT	95
8 Analisi architettura del sistema	96
8.1 Il client Android	98
8.2 Il server	98
8.3 Le tecnologie impiegate	98
8.3.1 Perché Java?	102
8.3.2 Perché un web framework?	103
8.3.3 Perché MongoDB?	103
9 Note finali	106
10 Glossario	108

Capitolo 1

Presentazione dell'idea progettuale

In questa documentazione tecnica verrà descritta l'idea progettuale rispondendo ai punti cardine del suo sviluppo: cosa deve fare l'applicazione, a chi essa è destinata e in che modo è stata valutata l'usabilità del prodotto.

1.1 L'applicazione

Vamos! è un sistema informativo user-centered distribuito finalizzato ad offrire una moderna piattaforma mobile per la scoperta, la ricerca di attrazioni, l'apprendimento ludicizzato e la visualizzazione di recensioni scritte dagli altri utenti per scegliere la migliore soluzione alla propria necessità. L'applicazione è pensata per essere di facile utilizzo ed usufruisce delle tecniche di gamification, delle tecniche di micro-interaction e di metodologie di fidelizzazione per stimolare gli utenti. Attraverso l'utilizzo dell'applicazione l'utente può collezionare titoli, sbloccare obiettivi, aumentare di livello e guadagnare gettoni virtuali da usare come interscambio con le strutture convenzionate per usufruire di sconti o ingressi gratuiti: tramite un altro applicativo client il bigliettario, o il responsabile di un'attrazione, può scannerizzare un codice QR fornito da **Vamos!** per validare i pagamenti virtuali.

1.2 Le metodologie utilizzate

In questo paragrafo verranno mostrate le principali metodologie utilizzate per lo sviluppo del prodotto.

1.2.1 Gamification

Sono state adottate metodologie di gamification usufruendo di elementi mutuati da giochi o di tecniche proprie di game design in contesti non necessariamente ludici. In particolare: punti esperienza da accumulare, ricompense sotto forma di titoli da esibire, rinforzo positivo e rinforzo negativo. Questo principalmente per fomentare l'istinto primitivo dell'uomo basato sulla competizione, sull'affermazione del proprio status sociale e della ricerca del successo, seppur virtualizzato. Spesso quando si parla di gamification si pensa ad un riferimento al mondo videoludico, ma in realtà non è necessariamente così: i giochi esistono per intrattenere e per apportare diversi benefici al fruitore, come ad esempio un aumento delle abilità psico-motorie; la gamification, invece, è giustificata qualora si voglia o sia necessario motivare un insieme di utenti attraverso meccaniche più o meno complesse: avatar, barre di progresso, punti, obiettivi da raggiungere, titoli, badge, oggetti virtuali, leaderboard e così via. I badge servono per mostrare il raggiungimento di un obiettivo, le barre di progresso (anche testuali) per mostrare all'utente lo stato del sistema e supportare un internal locus of control rendendo più chiaro quanto ci sia ancora da fare per il raggiungimento di uno specifico obiettivo, i leaderboard per

avere un controllo più diretto. Sebbene la gamification adotti meccaniche di gioco note, è necessario comunque trovare un'intersezione tra quelle che sono gli obiettivi dell'utente (user goal) e i business goal.

1.2.2 Micro-interaction

Le micro-interaction possono essere di grande impatto nella user experience. Essenzialmente sono delle coppie trigger-feedback che aumentano il locus of control del sistema, ne aumentano la facilità d'uso e la prevenzione degli errori. Un trigger può essere scaturito a seguito di un'azione dell'utente, ad esempio quando interagisce coi componenti grafici dell'interfaccia come bottoni, oppure dal sistema stesso, ad esempio quando esso incontra e soddisfa un sistema predeterminato di condizioni. Una volta che una micro-interaction è stata scatenata fornisce feedback continui e consistenti all'utente che possono portare ad una modifica più o meno radicale dell'interfaccia grafica. A tal proposito sono stati impiegati feedback con un intervallo di tempo Δt piuttosto breve che consenta di ridurre al minimo il tempo del golfo della valutazione, ovvero il tempo necessario all'utente affinché percepisca lo stato del mondo e lo interpreti valutandone il risultato. La scelta di rendere minima la distanza temporale Δt tra l'azione dell'utente e del feedback del sistema aiuta non solo l'utente a diminuire i tempi del golfo della valutazione, ma è stata scelta anche come metodologia di rinforzo positivo o negativo: sia nell'essere umano che nei cani durante le fasi di addestramento cinofile è spesso usato un clicker; il suo suono, a seguito di un'azione del cane, rinforza il comportamento dell'animale se il lasso di tempo Δt è minimo. In generale, l'interfaccia è curata per fornire all'utente il miglior affordance possibile, così da ridurre il tempo necessario del golfo dell'esecuzione e da rendere, cioè, quanto più intuitivo possibile il prodotto.

1.2.3 Fidelizzazione

L'obiettivo principale della fidelizzazione è il mantenimento del cliente in modo tale che continui ad usare il nostro prodotto e pensi a noi come miglior alternativa per il suo obiettivo, rifiutando gli altri competitor qualora ce ne siano. Indipendentemente dal grado di maturità del prodotto, la fidelizzazione è essenziale soprattutto per attrarre nuovi clienti, che sono mediamente più difficili da convincere perché non conoscono ancora cosa il prodotto offre in termini pratici, differentemente dai clienti abituali che già hanno dimestichezza con esso. Inoltre è stato dimostrato che il tasso di conversione di un utente abituale tc_{cab} , ovvero il rapporto tra il numero di visitatori che accedono ad una certa risorsa e il numero di visitatori che compiono un'effettiva azione su quella risorsa, è tra il 60-70% mentre quella di un nuovo utente è del 5-20%, avendo quindi molte meno probabilità che un nuovo utente ritorni ad utilizzare quel prodotto. Formalmente:

$$\begin{cases} 6 \leq tc_a \\ 0.5 \leq tc_n \leq 2 \end{cases}$$

In più è stato verificato che il profitto di un prodotto (in termini di impiego o anche di tempo trascorso con esso) aumenta del 25-95% quando i tassi di fidelizzazione aumentano anche solo del 5%. È bene sottolineare, tuttavia, che la fidelizzazione va costruita: un utente non diventa fedele da un giorno all'altro ma si tratta di un rapporto che l'utente instaura col prodotto in sé, esattamente come il rapporto tra due innamorati. Occorre quindi che l'utente diventi dapprima un utente soddisfatto, ad esempio in termini di funzionalità ed usabilità del sistema ed infine fedele. Bisogna quindi trovare un modo di salvare un contatto dell'utente, ad esempio un'e-mail lasciandogli in cambio un lead magnet che fa da incentivo: è essenziale quindi costruirlo in modo tale che sia efficace, gratuito, specifico, di valore, di immediata fruizione e gratificante, come ad esempio dei punti bonus iniziali, un avatar o un titolo esclusivo e così via. Come visto, la fidelizzazione è veramente importante

ma a volte non basta ed occorre potenziarla con la tecnica della segmentazione: essa consiste nel suddividere i clienti in gruppi simili per certe caratteristiche (come ad esempio la zona di residenza o il genere) ad esempio per creare piccoli eventi esclusivi locali o per altre strategie più o meno complesse soprattutto quando il numero di clienti per un certo prodotto è alto.

Capitolo 2

Individuazione del target degli utenti

È importante che il sistema venga progettato in modo tale che sia incentrato sull'utente, come mostrato in Figura 2.1. Il sistema, inoltre, dovrebbe essere di facile utilizzo e non dovrebbe richiedere all'utente grossi sforzi in termini di memoria a breve termine, ma dovrebbe aiutarlo a concentrarsi soltanto sulle parti realmente essenziali dell'interfaccia attraverso una scelta strategica sui colori, sulle forme dei pulsanti e sulla loro posizione sullo schermo.

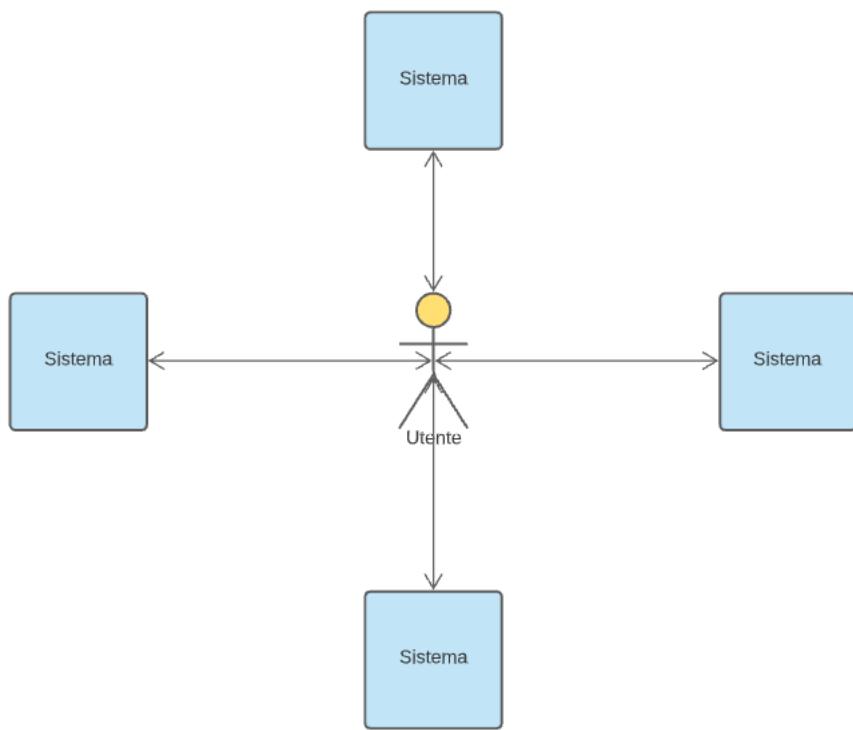


Figura 2.1: Sistemi user-centered

Per far ciò è necessario definire chi sarà il target di utenti a cui è destinato il prodotto. L'utente andrà studiato sia da un punto di vista cognitivo, inteso come animale sociale con percezione ed appercezione, ma anche da un punto di vista fisico o motorio. La letteratura medica studia una varietà di profili psicologici piuttosto interessanti; la diversità degli esseri umani sono naturali e determinanti e lo saranno ancora di più quando, nelle sezioni successive, verranno definite le personas: in quel caso una personas non solo sarà determinata dalle sue condizioni cognitive, fisiche e motorie, ma anche da caratteristiche proprie che lo identificano in quanto uomo: nome, cognome, età, una lingua, una formazione ed, in generale, la sua storia personale. Lo studio, inoltre, sarà puntualizzato tenendo in considerazione anche i rapporti che hanno le personas le une con le altre, analizzando quindi il profilo

inter-personale e non solo limitandosi a quello intra-personale, valutandone quindi i comportamenti sociali. Infine occorre definire quale sarà il ruolo dell'utente all'interno del contesto in cui lo si pensa.

La pandemia COVID-19 ha portato notevoli conseguenze non solo fisiche ma anche psicologiche che si manifestano con diversi gradi di intensità attraverso il cosiddetto stress post-traumatico caratterizzato da diversi sintomi: di tipo cognitivo (difficoltà di concentrazione, scarsa memoria e demotivazione), di tipo emotivo (stress, irritabilità, ansia) e di tipo fisico (stanchezza, disturbi del sonno). Tra i sintomi comportamentali quello meno indagato è l'isolamento, ovvero l'evitamento di relazioni sociali a causa di paure irrazionali che, oltre alle varie problematiche psicologiche, causa un aumento dei livelli di fibrinogeno associato a maggiori rischi cardiovascolari. Questa dimensione emotiva si riferisce ad uno stato di smarrimento ed implica la voglia di continuare a rimanere al sicuro nella propria abitazione o nel proprio confinamento. Se diverse persone sono riuscite gradualmente a ritornare ad una presunta normalità, molte altre hanno faticato e faticano più del previsto, sviluppando in alcuni casi una vera e propria patologia chiamata sindrome della capanna (o del prigioniero) che rappresenta uno dei principali contraccolpi psicologici in seguito alle esperienze dell'ultimo anno. La sindrome della capanna si manifesta con la paura di uscire e abbandonare la propria comfort-zone. Per diverse ragioni, ma anche a causa di meccanismi del tutto inconsci quali ansia, paura, stress e frustrazione, la sindrome della capanna può essere un vero e proprio ostacolo nella vita quotidiana, ma esiste una soluzione a questo problema: è stato dimostrato che i soggetti maggiormente colpiti sono nella fascia d'età giovanile dai 15 ai 25 anni; il Dott. Alfonso Piccoli, internista e psicoterapeuta, ha dimostrato che per fronteggiare questo fenomeno è necessario riconoscere la propria paura per praticare la "consapevolezza nel qui e ora" e affrontare il quotidiano attraverso piccoli step. La sindrome della capanna, procede il Dott. Piccoli, tesse una ragnatela di bugie e scuse al di sopra della ragione: il senso critico della persona è assopito da una continua ricerca di scuse che lo portano ad evitare il problema e quindi a mai risolverlo, sebbene non implichi una diminuzione degli interessi verso cui una persona era legata (apatia): l'affetto dalla sindrome della capanna è semplicemente una persona normale che ha paura di abbandonare il proprio confinamento per una paura irrazionale, certe volte perché mancano gli stimoli adatti. Per sentire quella protezione di cui si necessita è necessario compiere piccole azioni remunerative o che in generale diano una sollecitazione in questo senso. È necessario, inoltre, mantenere attivo il corpo e la mente attraverso piccoli obiettivi non per forza impegnativi ma che diano un senso di completezza una volta raggiunti. Infine, ma non per importanza, il COVID-19 ha costretto le istituzioni ad effettuare diversi tagli sui programmi scolastici a causa del minor numero di ore a disposizione: a tal proposito, **Vamos!** si propone anche un intento educativo in questo senso, spronando l'utente alla conoscenza riempiendo le proprie lacune o ampliando le proprie conoscenze in maniera divertente, utendo quindi l'utile al dilettevole e il dilettevole al necessario. In Figura 2.2 viene mostrato attraverso il formalismo di Eulero-Venn la relazione che sussiste tra la popolazione e gli affetti dalla sindrome della capanna. Formalmente $\forall A, B \subset \mathbb{N}$

$$\begin{cases} A := \text{Popolazione} \\ B := \text{Affetti da sindrome della capanna} \end{cases} \implies \emptyset \neq B \subseteq A \neq \emptyset$$

In particolare il fatto che $B \neq \emptyset$ è dimostrato dalle considerazioni scientifiche e dalle ricerche citate in questo capitolo, mentre il fatto che $A \neq \emptyset$ è dimostrato dal fatto che esistono 60 milioni di persone in Italia. Se poi si considera la totalità di essi che possiede uno smartphone (scelta giustificata dal fatto che il sistema che si sta progettando è un'applicazione su smartphone) allora la cardinalità di A sarà approssimativamente pari a 56 milioni di unità, ovvero $|A| \approx 56 \cdot 10^6$ dal momento che è dimostrato che il 93% degli italiani possiede uno smartphone,

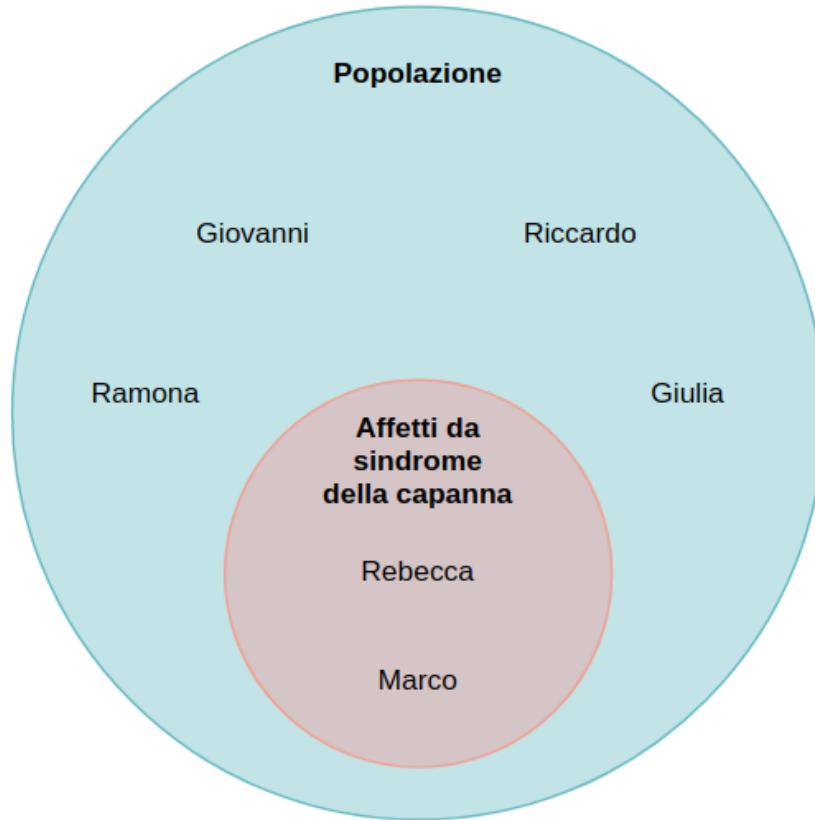


Figura 2.2: Diagramma di Eulero-Venn che rappresenta la popolazione affetta da sindrome della capanna

A partire dal diagramma in Figura 2.2, è stato individuato un altro insieme C che rappresenta il bacino di utenti affetti da sindrome della capanna appassionati di arte ed attrazioni turistiche, come mostrato in Figura 2.3. Formalmente:

$$\begin{cases} A := \text{Popolazione} \\ B := \text{Affetti da sindrome della capanna} \\ B \subseteq A \end{cases} \implies C \subseteq B$$

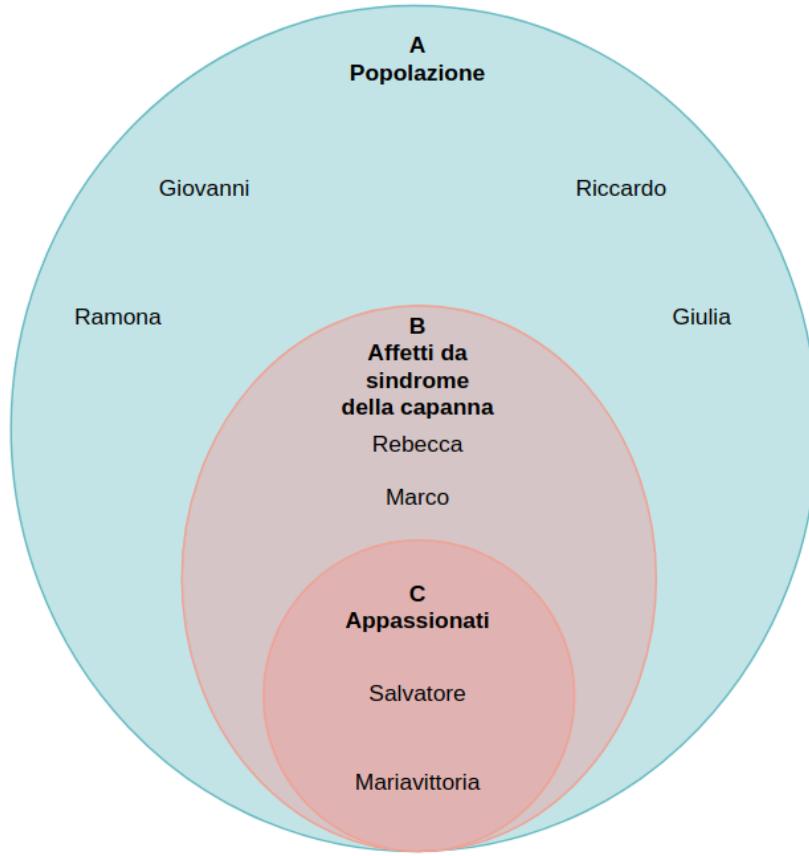


Figura 2.3: Diagramma di Eulero-Venn che rappresenta la popolazione affetta da sindrome della capanna appassionata di arte e attrazioni turistiche

Su queste considerazioni si basa la progettazione di **Vamos!** quindi sulla popolazione affetta dalla sindrome che sia appassionata all'arte e alle attrazioni turistiche. L'utente, in quanto affetto da sindrome della capanna ma appassionato di arte, si sente frustrato ed impotente perché vorrebbe ampliare la propria conoscenza, scoprendo cose nuove e superare la propria paura, ma è ostacolato dalla malattia stessa. Il conoscimento, l'analisi e la creazione delle personas è costituita non solo sugli aspetti già discussi, ma anche sulla definizione di obiettivi (goal). Nel nostro caso tutti gli elementi del pool di utenti *C* individuato vogliono sentirsi motivati ad uscire di casa (experience goal) poiché tramite l'uso dell'applicazione viene fomentata la sua fame di conoscere, istigando le sue passioni (arte e attrazioni, appunto) e alimentando l'innato senso di competizione dell'uomo; inoltre vogliono poter conoscere nuove attrazioni, lasciare recensioni o leggerne, documentarsi, scoprire, aumentare di livello e gareggiare nella leaderboard (end goal), cercando di diventare una persona più serena superando il proprio timore (life goal) spinto dagli espedienti descritti. I goal sono gli obiettivi che un utente intende raggiungere attraverso delle azioni ad essi necessari, chiamati operators. I metodi sono le sequenze di operatori disponibili atti al raggiungimento di un singolo obiettivo.

2.1 Individuazione di pattern e variabili comportamentali

Sulla base di quanto detto finora è possibile definire in maniera più netta quali siano le variabili comportamentali (behavioural variables) degli elementi appartenenti all'insieme *C* della Figura 2.3. In particolare $\forall x \in C$

1. Attività (activity) - x non riesce a uscire di casa, trascorre molto tempo allo smartphone e alimenta la propria passione per le attrazioni turistiche o l'arte in generale leggendo e documentandosi;

2. Atteggiamenti (attitudes) - passando molto tempo allo smartphone (e a causa della sua giovane età), x ha una buona conoscenza del dominio tecnologico. Da quanto già detto, si presume abbia utilizzato già qualche applicazione educativa, oppure abbia giocato ad almeno un videogioco (assunzione plausibile poiché è dimostrato che circa il 43% degli italiani, ovvero delle persone nell'insieme A, gioca ai videogiochi, ovvero circa 260 milioni di italiani. Percentuale che, in relazione al fatto che x rimanga a casa, può essere maggiorata considerando l'insieme C).
3. Attitudini (aptitudes) - la sindrome della capanna di cui x soffre non comporta alcuna problematica a livello cognitivo. Quindi, da questo punto di vista, x ha le stesse capacità cognitive di una persona sana (quindi capacità cognitive nella norma).
4. Motivazioni (motivation) - x è motivato all'uso del prodotto poiché appassionato di arte e interessato alle attrazioni che una certa città ha da offrire. Inoltre, come spiegato dal Dott. Piccoli, il malato ha piena conoscenza del suo problema ed è motivato a superarlo.
5. Abilità (skill) - la giovane età di x può implicare una buona conoscenza del dominio tecnologico e del dominio del prodotto.

La Figura 2.4 mostra la distribuzione delle variabili comportamentali tramite un diagramma pseudo-statistico.

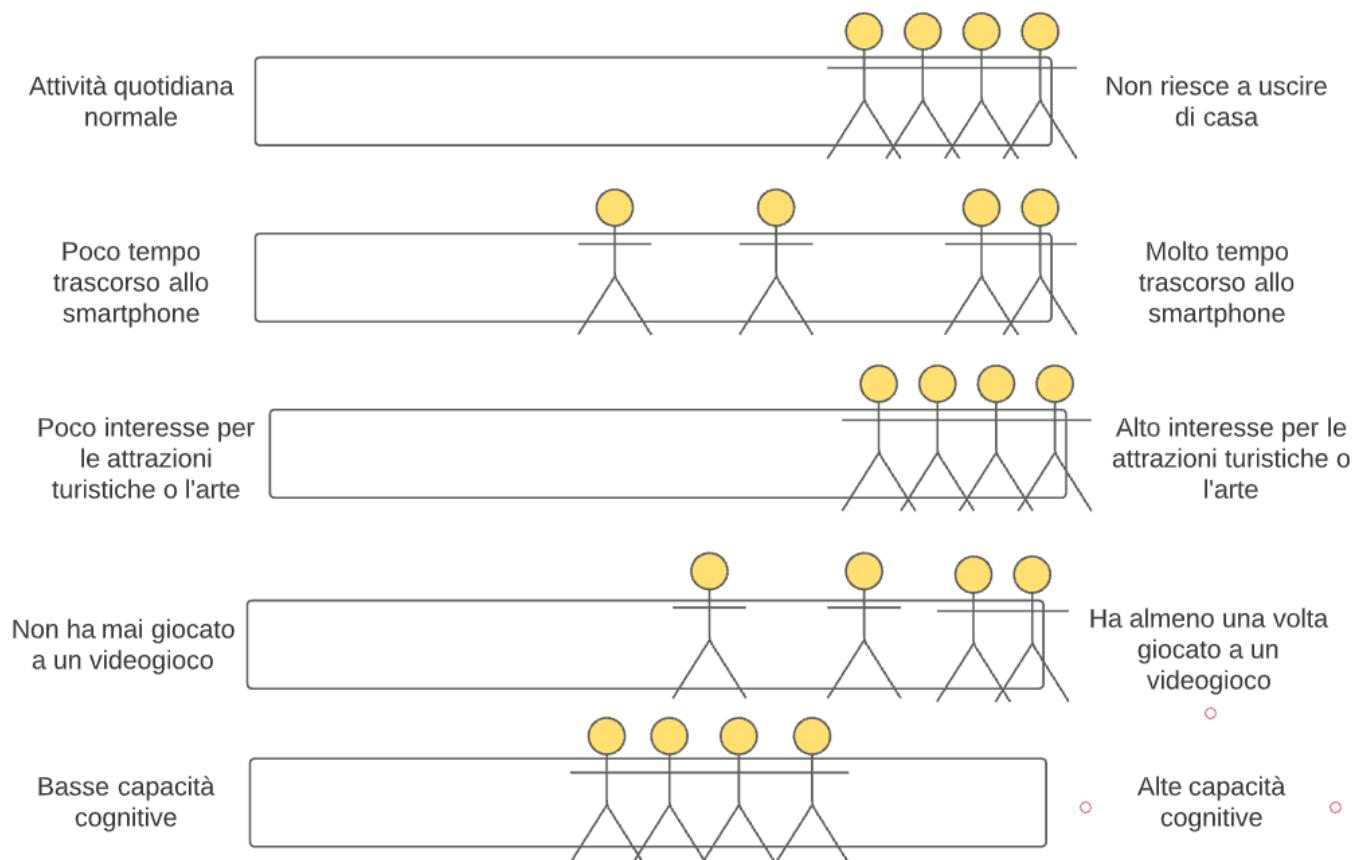


Figura 2.4: Distribuzione delle variabili comportamentali degli elementi dell'insieme C

Per consentire uno studio a raggruppamenti e quasi generalizzato delle personas è possibile definire $\rho \in EQ(A)$ una relazione di equivalenza su A, cioè sull'intera popolazione italiana, tale che $\forall x, y \in A$

$$x\rho y \iff "x \text{ e } y \text{ sono due persone affette dalla sindrome e appassionate}"$$

giustificando l'esistenza dell'insieme

$$[x]_\rho := \{y \in A : x\rho y\} \neq \emptyset$$

Tale insieme conterrà tutti gli elementi che sono equivalenti ad x , ovvero la totalità delle persone affette dalla sindrome e appassionate di arte e attrazioni turistiche. Per tale costruzione se un certo fenomeno è stato analizzato studiando x , rappresentante della classe, allora esso sarà plausibilmente riscontrabile in tutti gli altri elementi equivalenti a x che, per costruzione, sono tutti e soli gli elementi di C. Ciò consente, tra le varie cose, di estendere il diagramma mostrato in Figura 2.4 analizzando non più soltanto quattro elementi di C, ma astraendo all'intera cardinalità dell'insieme. In altre parole, se una soluzione può funzionare su un qualsiasi elemento x di C, allora essa potrà funzionare altresì su tutti gli altri elementi y che siano equivalenti a x e quindi, per costruzione, su tutti gli elementi di C.

2.2 Sintesi delle caratteristiche e degli obiettivi rilevanti

Adesso è necessario sintetizzare i dati a partire dalle considerazioni sulle variabili e i pattern comportamentali degli step precedenti. Sintetizziamo tali caratteristiche mediante una lista puntata. Quindi, considerando x il rappresentante della classe di equivalenza $[x]_\rho$ definita precedentemente, allora:

- x non riesce a uscire di casa passandovi tutto il tempo tra studio, smartphone e qualche videogioco;
- x trascorre molto tempo al telefono di cui ha una buona conoscenza di dominio e digitale in generale;
- x studia, è informato, non ha menomazioni cognitive e ha interesse verso tutto ciò che è arte o tutto ciò che possa rappresentare un'attrazione turistica nelle varie città;
- x è una persona, poiché appartenente all'insieme A, per questo motivo ha un forte (ed istintivo) spirito di competizione e di affermazione del proprio status sociale. Per questo motivo le tecniche di gamification adottate, e discusse nel capitolo precedente, potrebbero attecchire in maniera piuttosto interessante.
- x desidera uscire dallo stato di torpore e superare la sua paura irrazionale.

Si noti che l'ultimo obiettivo è realizzabile poiché dimostrato dal Dott. Piccoli e dal momento che $C \subseteq A$, vale a dire che esiste un'immersione $\Psi : x \in C \hookrightarrow x \in A$. Per questo motivo la realizzazione di questo desiderio è demandato all'implementazione di una funzione di transizione $\tau : x \in C \mapsto x \in A \setminus B \setminus C$ dove il codominio della funzione τ rappresenta la popolazione italiana "sana". La costruzione dell'applicazione τ è uno degli obiettivi principali del progetto.

2.3 Personas-type

A partire dalle constatazioni e dalle analisi compiute finora, possiamo costruire le personas-type in modo tale da priorizzare gli elementi dell'insieme C e definire quale tra i suoi elementi deve essere il target primario del design. Ogni personas individuata in questa fase sarà sintetizzata attraverso una scheda che non ha solo la funzione di fornire una sintesi delle caratteristiche principali, ma anche una funzione per comunicare la ricerca con terzi o col team di sviluppo.

2.3.1 Mariavittoria Ortona

La prima personas che individuiamo è Mariavittoria Ortona, di 17 anni, studentessa a tempo pieno presso il liceo scientifico statale Carlo Urbani di Roma. Rappresenta la primary personas-type, cioè quel singolo elemento dell'insieme C i cui bisogni devono essere completamente e felicemente soddisfatti e con cui il sistema deve empatizzare nel migliore dei modi. Mariavittoria è giovane, va bene a scuola, è amichevole, dolce e competitiva. Sa usare internet, infatti è una ragazza molto social. Ama l'arte in tutte le sue forme e da sempre è un'appassionata di siti storici, come attrazioni turistiche, musei e punti d'interesse. Afferma che da quando il Covid è arrivato in Italia ha iniziato a soffrire della sindrome della capanna e si sente psicologicamente confusa ed è costretta a causa della sua paura irrazionale a restare a casa passando il suo tempo interessandosi alla storia e all'arte. È una persona tendenzialmente introversa, razionale e piuttosto empatica. Vorrebbe tanto riuscire a superare la sua paura, imparare qualcosa di nuovo sull'arte e sui punti di interesse nelle varie città e restare in contatto con le persone e competere con esse anche virtualmente. Tali caratteristiche sono sintetizzate in Figura 2.5.

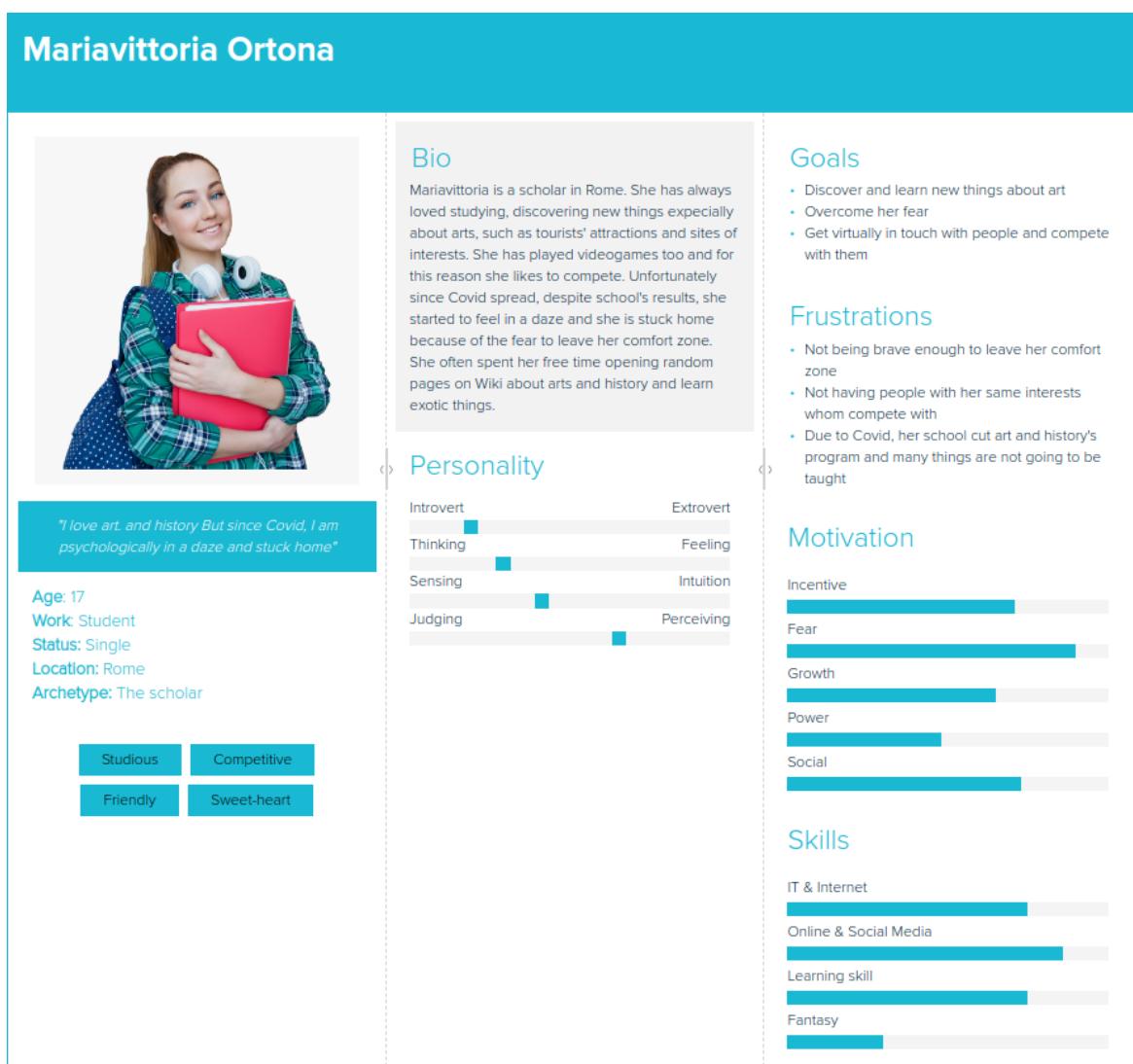


Figura 2.5: Personas card per Mariavittoria Ortona

2.3.2 Salvatore Ascione

Adesso occorre definire la secondary personas-type, ovvero un'altra personas (cioè un altro elemento dell'insieme C) che abbia qualche necessità in più rispetto a Mariavittoria ma che possano comunque essere soddisfatte in maniera piuttosto agile senza sconvolgere l'abilità che ha il sistema di soddisfare le necessità di Mariavittoria. Chiamiamo Salvatore Ascione il secondary personas-type. Salvatore è un ventenne napoletano ed attualmente frequenta il secondo anno dell'Accademia delle Belle Arti. È un ventunenne sfrontato che ama viaggiare e muoversi, è una persona dinamica e loquace, ma da quando si è ammalato di Covid ha iniziato a soffrire anche della sindrome della capanna. Anche Salvatore, come Mariavittoria, vorrebbe poter affrontare e superare la sua paura ed uscire di casa (che ricordiamo, in accordo al Dott. Piccoli, può essere combattuta trovando un impulso che consente di uscire dalla comfort-zone). In più vuole sempre distinguersi dalle altre persone e puntare ai massimi risultati. Una sua schematizzazione è data dalla scheda in Figura 2.6.

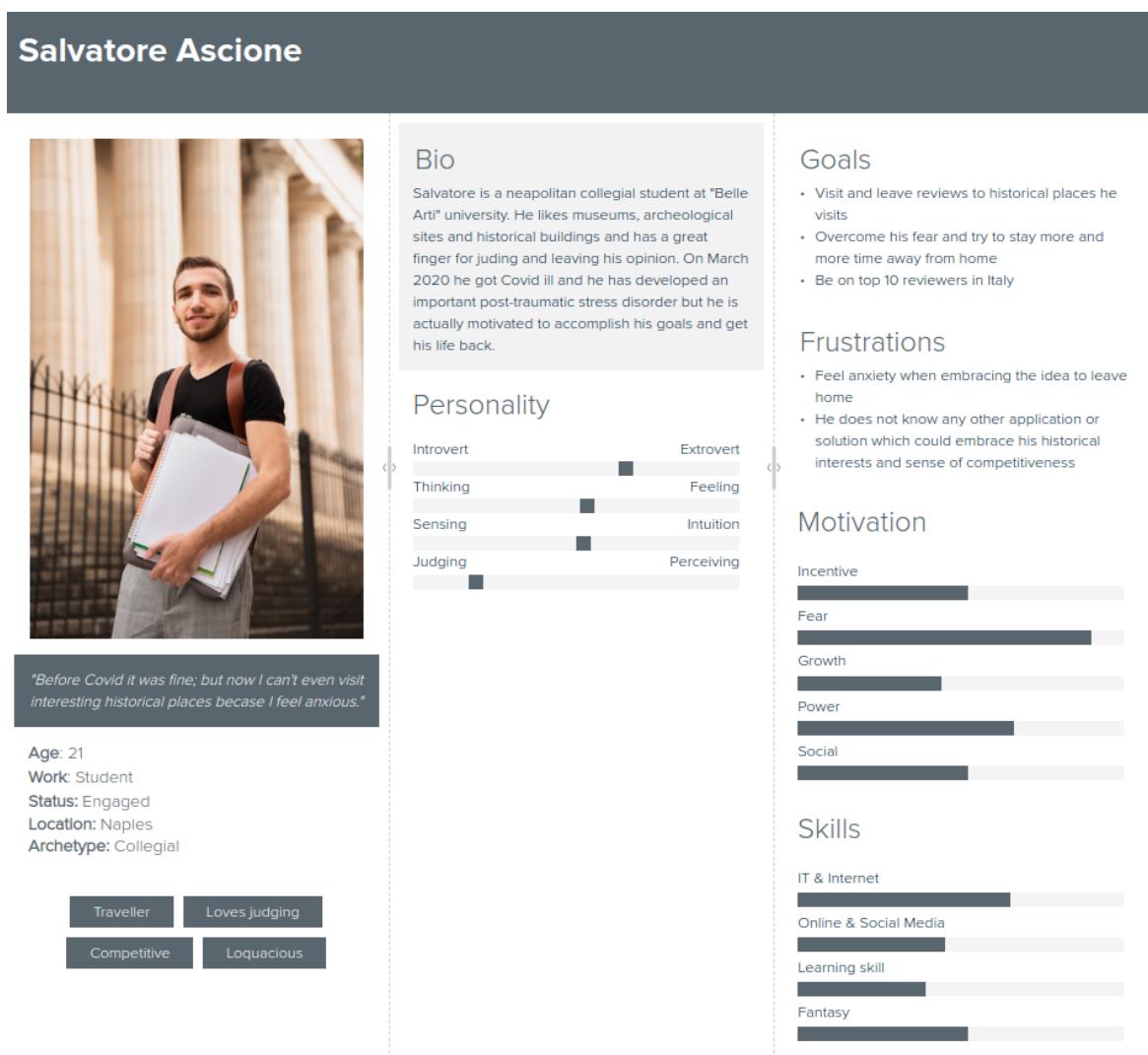


Figura 2.6: Personas card per Salvatore Ascione

2.3.3 Rebecca Giamundo

Finora abbiamo costruito le personas target del prodotto in maniera primaria, secondaria o supplementare. Occorre tuttavia definire anche la personas che possa schematizzare un utente che sebbene abbia sviluppato la sindrome della capanna come conseguenza al covid, non è per nulla interessato all'arte, alla storia, o a siti d'interesse in generale. Formalmente sono tutti e soli gli elementi dell'insieme $B \setminus C$ del diagramma in Figura 2.3. A tal proposito, introduciamo Rebecca Giamundo. La professoressa Giamundo è un'insegnante austera e determinata. Insegna matematica alla statale di Milano, sebbene sia prossima alla pensione. Da ottobre si è ammalata di Covid e ha sviluppato qualche paura irrazionale a causa della sindrome della capanna, anche se non sente eccessivamente il suo peso. I suoi bisogni sono disgiunti rispetto agli obiettivi del sistema, infatti desidera portare ai massimi livelli le sue competenze matematiche, ottenere un dottorato di ricerca in fisica e visitare Barcellona. Non è una persona social e non passa quasi mai il suo tempo online o in generale allo smartphone. La sua card è in Figura 2.7.

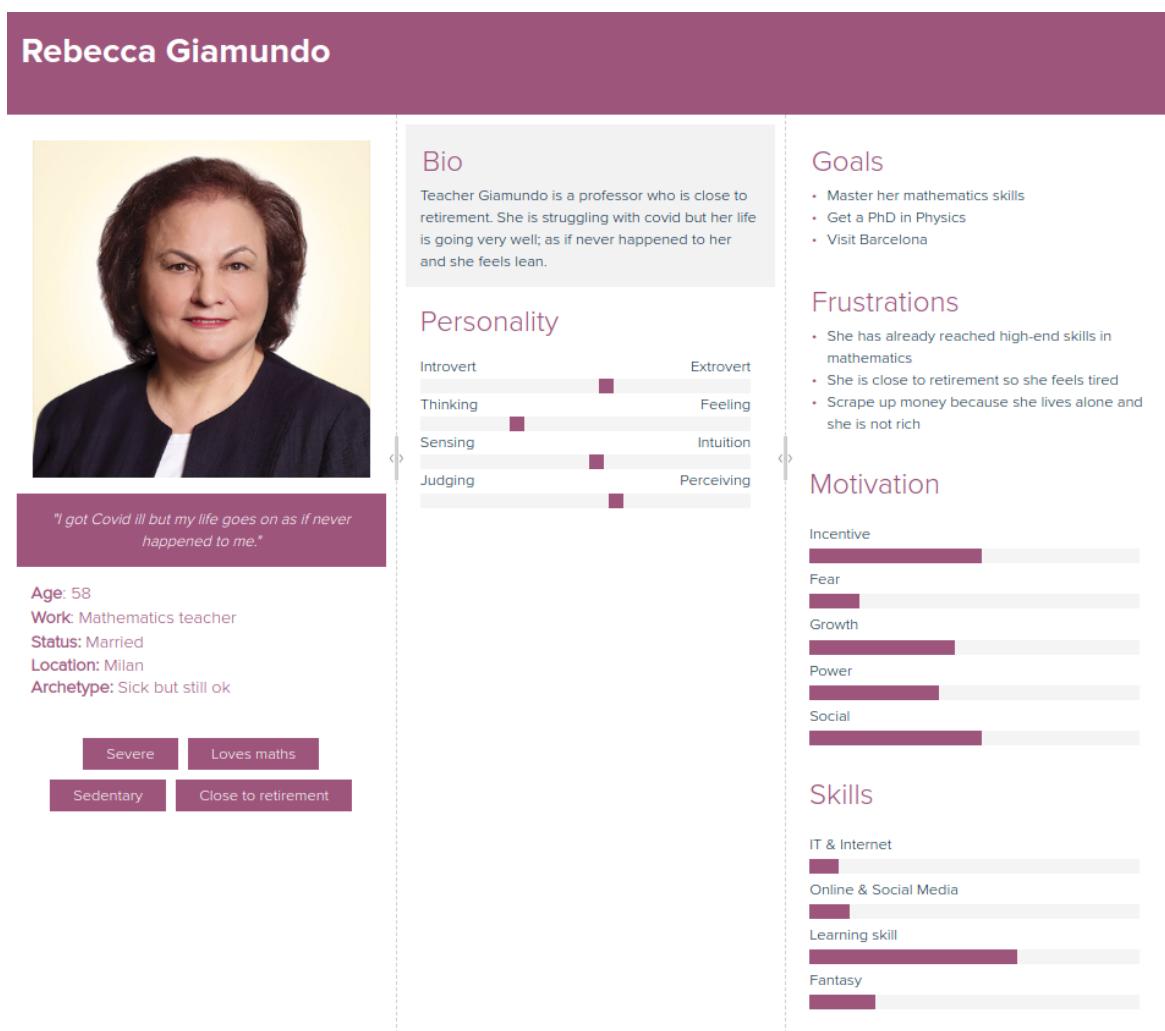


Figura 2.7: Personas card per Rebecca Giamundo

2.3.4 Giovanni Arpa

Definiamo adesso ovvero un rappresentante degli elementi appartenenti all'insieme $A \setminus B$ della Figura 2.3. Chiamiamo questo tipo di personas Giovanni Arpa. Il dottor Arpa è un software engineer quarantenne. È educato, sorridente e riesce sempre a mettere di buonumore i suoi colleghi. È una persona completamente sana: mai ammalata di Covid e, nella fattispecie, non ha mai sofferto realmente la sua presenza motivo per cui non ha mai sviluppato la sindrome della capanna: la sua vita sta procedendo come se non fosse mai successo nulla. Vorrebbe vincere il premio Turing, sviluppare nuove applicazioni interessanti e imparare la chitarra, sebbene il premio sia duro da vincere e abbia le mani mediamente piccole per questo strumento musicale. Le sue caratteristiche sono sintetizzate in Figura 2.8.

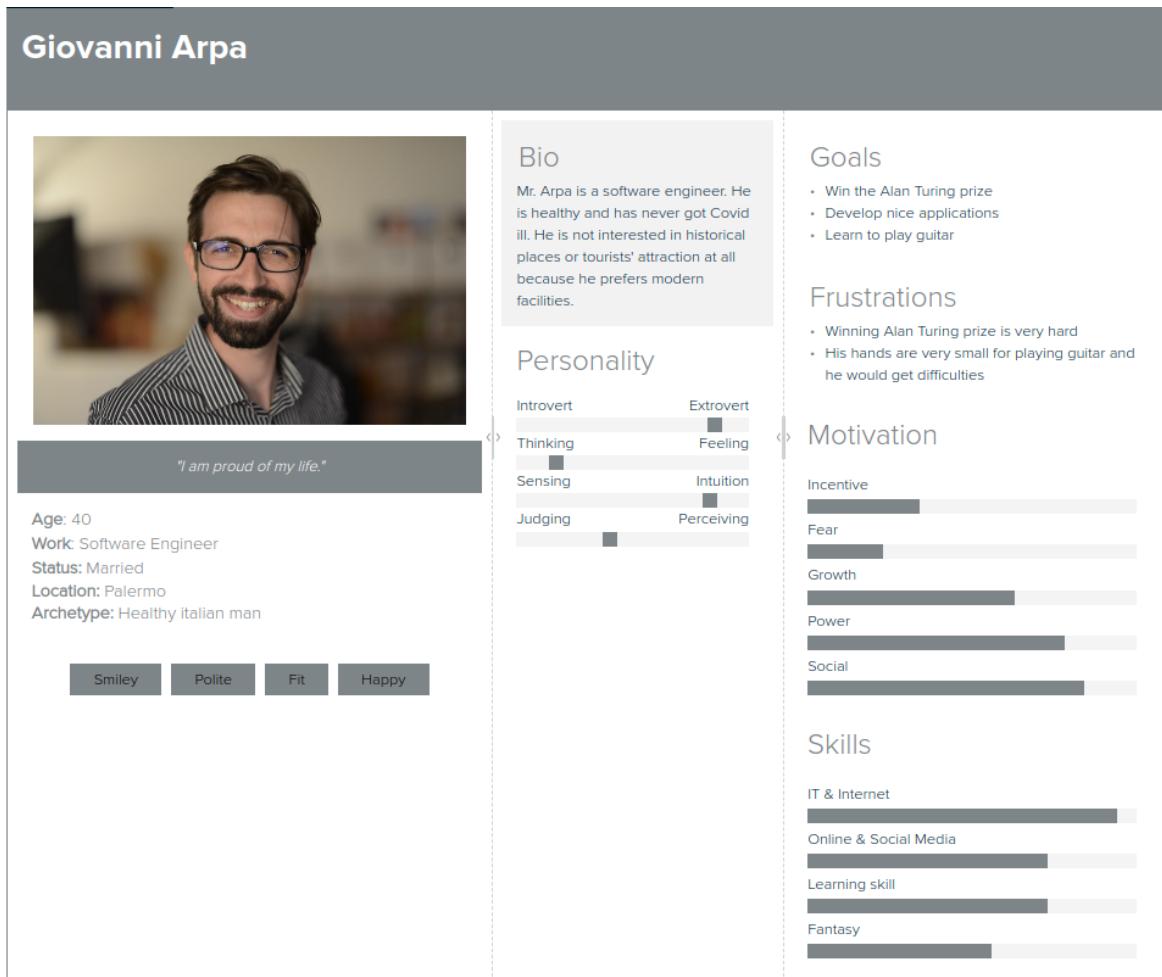


Figura 2.8: Personas card per Giovanni Arpa

2.4 Personas goal e business goal

In questa sezione verrà mostrata la relazione che sussiste tra le necessità di ciascuna personas-type definita al punto precedente e quelle del sistema.

2.4.1 Primary personas-type goal e business goal

Per definizione i bisogni e le necessità del primary personas-type, cioè Mariavittoria, devono essere completamente soddisfatte. Come si nota in Figura 2.9, gli obiettivi di Mariavittoria sono completamente soddisfatti dal momento che rappresentano l'intersezione con l'insieme degli obiettivi del sistema.

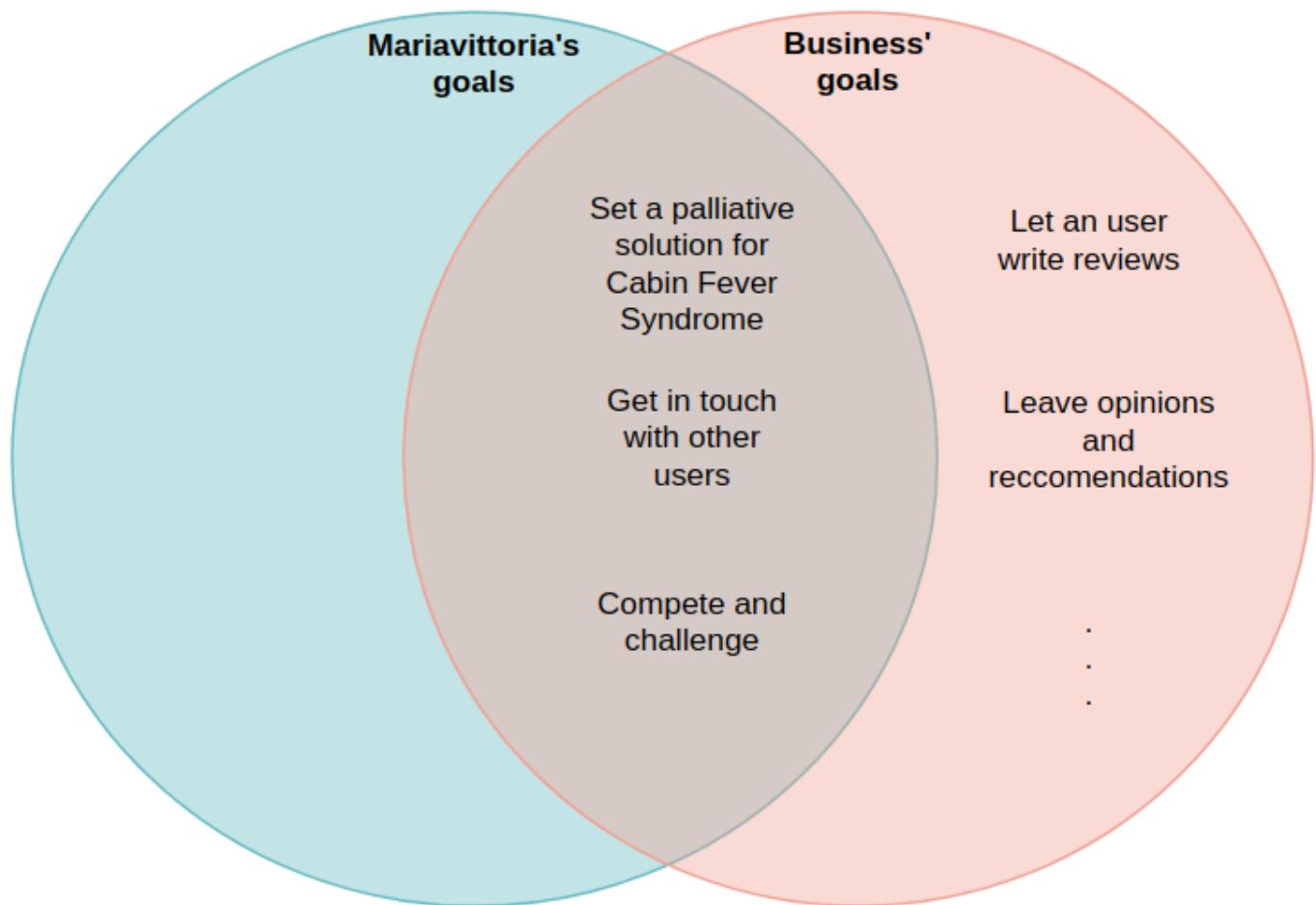


Figura 2.9: Primary personas goal e Business goal

2.4.2 Secondary personas-type goal e business goal

Anche gli obiettivi del secondary personas-type, cioè Salvatore Ascione, devono essere soddisfatti in maniera agile dal sistema senza perdere di efficienza e completezza rispetto alle necessità di Mariavittoria. In Figura 2.10 è mostrata la relazione tra l'insieme degli obiettivi di sistema, quelli di Salvatore e quelli di Mariavittoria. Se ne evince che le necessità di Salvatore sono un'estensione di quelle di Mariavittoria.



Figura 2.10: Primary e Secondary personas-type goal e Business goal

2.4.3 Negative personas-type goal e business goal

Le necessità delle personas che non sono target del prodotto, in particolare quelle di Rebecca Giandomo e Giovanni Arpa, non possono essere pienamente soddisfatte dal sistema. In Figura 2.11, infatti, è evidente che i tre insiemi sono a due a due disgiunti.

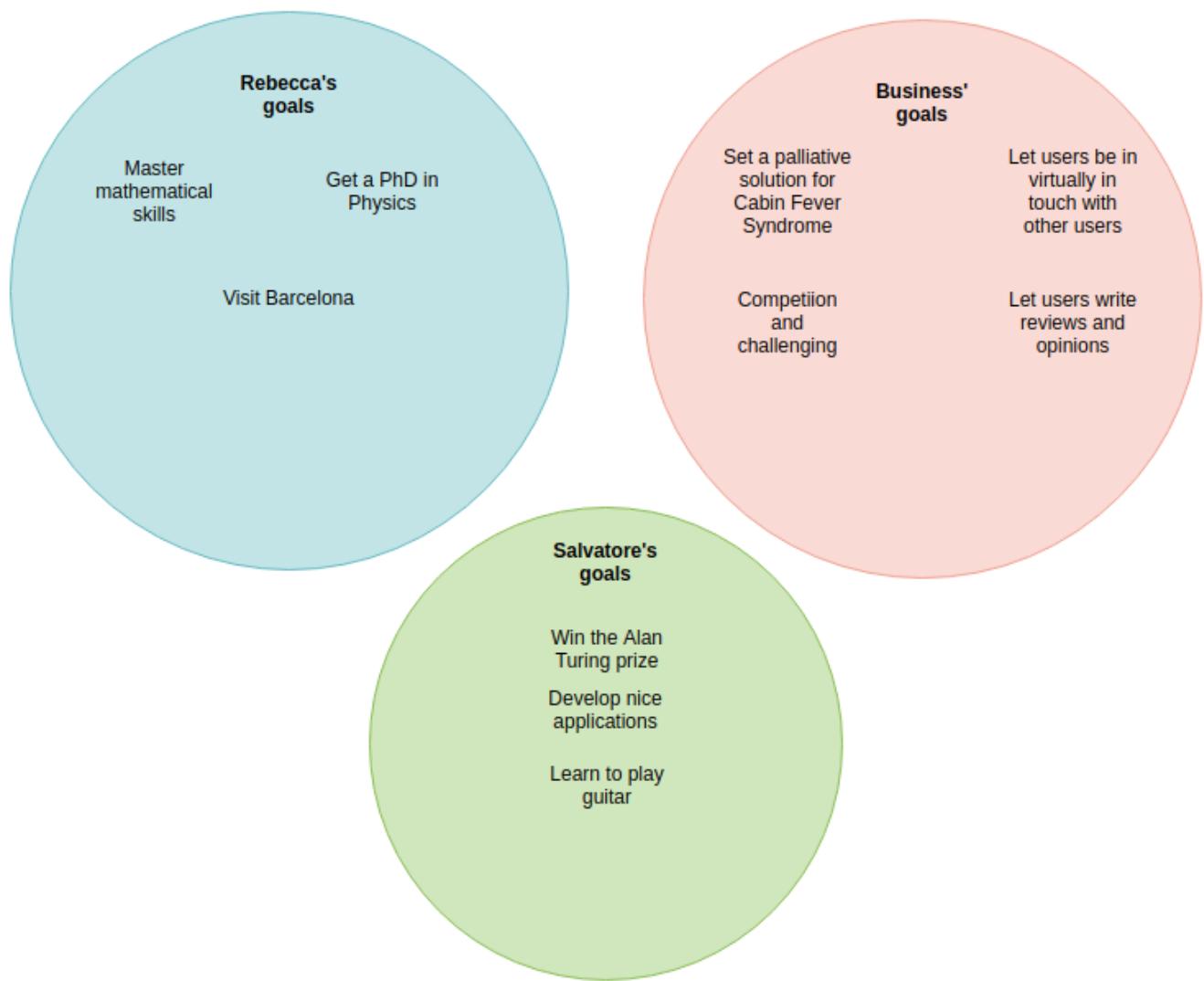


Figura 2.11: Necessità delle negative personas disgiunte rispetto agli obiettivi del prodotto

Capitolo 3

Studio della valutazioni a priori dell'usabilità

L'uomo spesso interagisce con oggetti, non necessariamente tecnologici, che spesso interpreta come di difficile utilizzo. L'analisi che consiste di individuare *dove* l'utente riscontra una difficoltà è necessaria per eliminarla e quindi ricercare l'usabilità. In questo capitolo affronteremo la questione di valutare l'usabilità del prodotto a priori, ovvero verificare mediante metodi scientifici, e a partire dal prodotto non ancora esistente, se il prodotto, ancora in uno stato primordiale, potrà essere usabile.

Poiché in questa fase il prodotto finale non esiste ancora, ma si è ancora in una fase di prototipazione, la valutazione dell'usabilità sarà informale, utilizzando sketch e prototipi. Principalmente sono stati impiegati prototipi di ruolo (*role prototype*) e quelli che consentano di valutare l'interazione uomo-macchina (*look&feel prototype*). Quelli relativi agli aspetti implementativi e algoritmici (*implementation prototype*) sono stati ignorati in questa prima fase, ma sono stati presi in considerazione dal team di sviluppo per scegliere le migliori strutture dati che potessero garantire un tempo di esecuzione minimo. In Figura 3.1 viene mostrata la natura dei prototipi impiegati in questa prima fase di valutazione a priori. Il cerchio giallo rappresenta l'orientamento dello scopo dei prototipi impiegati in questa fase: essi, come detto, sono stati costruiti in modo tale da valutare in maniera equa il ruolo (del prodotto nella vita del suo utente) e l'interfaccia, per massimizzare l'usabilità del prodotto e ricercare le pratiche di buon design sull'interazione uomo-macchina.

Sono stati effettuati test formativi poiché atti a dare forma a quello che sarà il prodotto finale. Sono particolarmente adatti nelle fasi iniziali di progettazione, quando il design concept è ancora grezzo, facendo sperimentare i prototipi di carta mettendo in luce quelli che sono i difetti macroscopici del sistema, senza entrare troppo nel dettaglio. Le modifiche prototipi effettuate in questa fase sono estremamente rapide da compiere: è sufficiente cancellare ciò che nel prototipo di un'interfaccia non convince e modificare quella piccola parte. Inoltre, per non riscontrare l'effetto mascheramento secondo cui altri difetti vengano nascosti da quelli già individuati, le modifiche da effettuare devono essere compiute in maniera istantanea. Per cui questo tipo di valutazione ha richiesto più di un'unica interazione coi valutatori.

Naturalmente, alla fine di questa valutazione a priori sull'usabilità è stato richiesta un'intervista a ciascuno degli utenti coinvolti nella valutazione, per constatare punti critici, debolezze ma anche punti forti su cui fare perno.

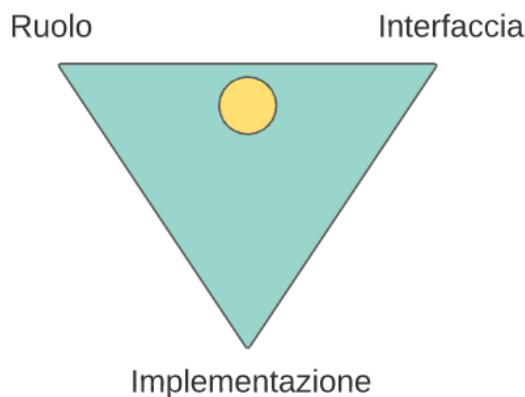


Figura 3.1: Spazio dei prototipi in relazione al loro scopo

I prototipi che sono stati impiegati in questa fase, inoltre, sono stati costruiti in modo tale da essere dinamici ma anche parzialmente interattivi, per venire in contro all'esigenza di costruire un sistema user-centered: durante le primissime fasi è stata compiuta una prototipazione a schizzo come mostrato in Figura 3.2 per contastare l'aspetto macroscopico del sistema e avere una vista generale delle sue funzionalità e interfacce principali. Una volta che il prototipo a schizzo è stato validato, rimuovendo aspetti deboli o poco convincenti, sono stati costruiti prototipi dinamici e interattivi attraverso animazioni su Adobe Xd: l'utente valutatore, infatti, poteva guardare ciascuna delle schermate principali del prodotto e, cliccando opportunamente sui bottoni che vi comparivano, poteva assistere alla transizioni oppure animazioni di vario genere. Durante lo sviluppo di questo progetto l'Italia è stata in zona rossa, per cui non ci si poteva recare presso appositi laboratori di usabilità né tantomeno assembrarsi; ad ogni modo i test sono stati effettuati tutti a domicilio prima dell'avvento della zona rossa, perseguitando test di usabilità informali: l'utente valutatore è stato messo dinanzi ad un sistema prototipizzato a fedeltà media ed attraverso una registrazione dei suoi movimenti, espressioni facciali e frasi tramite una webcam e successivamente analizzate dal team di sviluppo per trovare punti critici da gestire oppure movimenti corporei che potessero presupporre perplessità su qualche funzionalità del prodotto.

L'idea è stata accolta positivamente da tutti gli utenti sottoposti alla valutazione. In definitiva essa, seppur introduttiva e a priori, ha consentito la nascita del prodotto vero e proprio e la successiva valutazione dell'usabilità a posteriori in modo soggettivo e oggettivo che verrà quindi raffinata nei capitoli successivi.

Una specifica nota dolente evinta grazie alla valutazione a-priori dell'usabilità mediante schizzi o prototipi usa e getta è relativa alla parte di registrazione: un paio di valutatori ha riscontrato difficoltà nel capire che l'account, dopo aver sottomesso i dati nel form, andasse verificato cliccando sul link di verifica inviato alla loro casella di posta elettronica. Il motivo principale di questa difficoltà era dovuto al fatto che il messaggio informativo, ovvero quel messaggio che indicava all'utente di verificare il proprio account accedendo e cliccando il link di verifica nella sua casella email, era molto poco evidente, nel senso che era nascosto tra le altre componenti dell'interfaccia. Una videocamera frontale posizionata vicino ai prototipi di carta o ai wireframe di Adobe Xd ha catturato le mimiche facciali che ben rappresentavano questa difficoltà. L'analisi ha consentito di posizionare in maniera più evidente e leggibile il messaggio informativo.

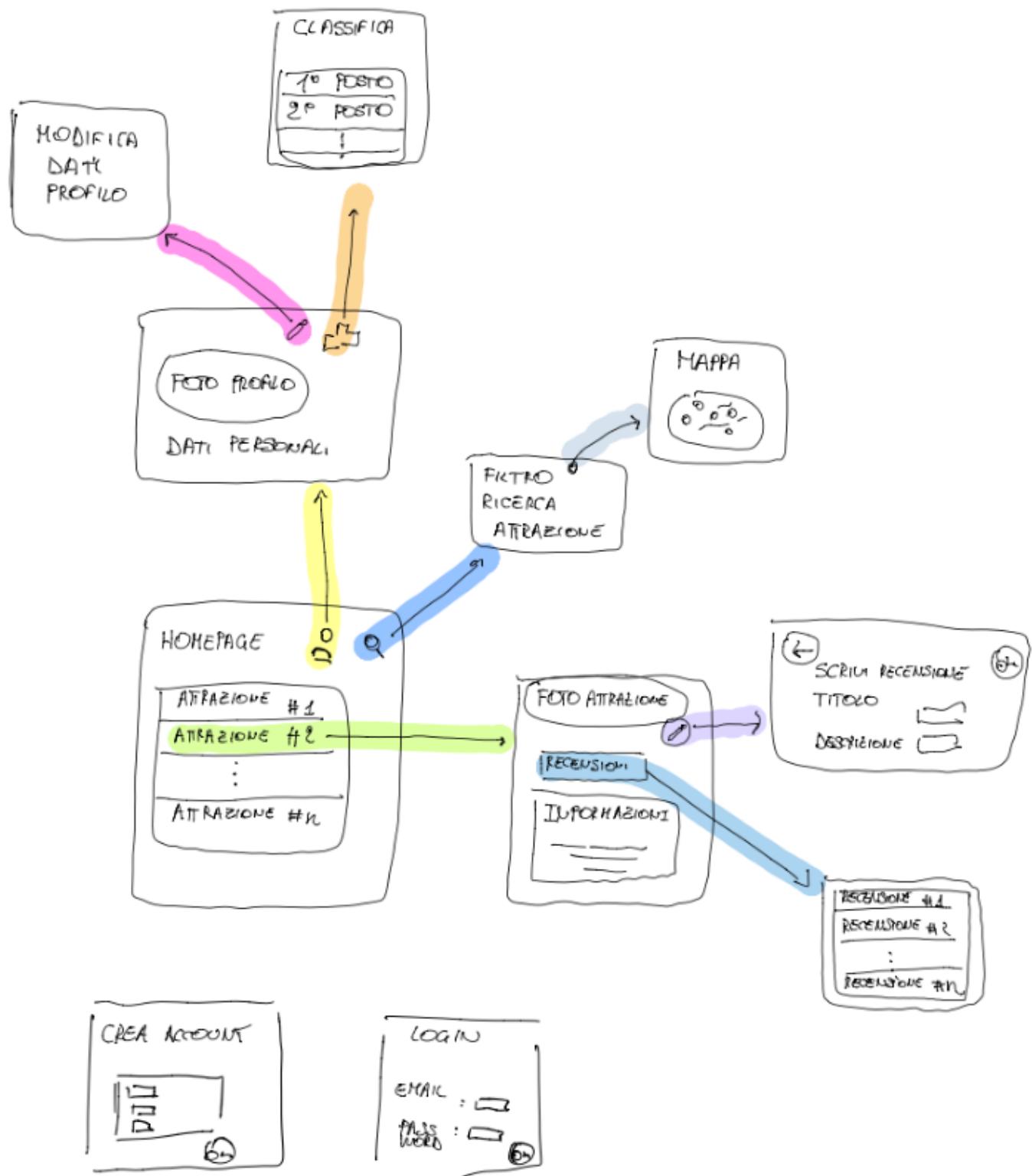


Figura 3.2: Prototipazione iniziale "a schizzo" del prodotto

Capitolo 4

Analisi delle funzionalità

In questo capitolo verranno analizzate e formalizzate le funzionalità principali, costruendo le gerarchie funzionali del prodotto e dettagliando la pianificazione del lavoro nella sua totalità.

4.1 Modello funzionale

Di seguito è riportata la classificazione dei requisiti ricavati mediante interviste, scenari e ricerche combinati tra loro. I requisiti utenti così raccolti sono stati successivamente analizzati e studiati per trasformarli nei requisiti di sistema.

4.1.1 Requisiti funzionali

- **Visualizzazione delle attrazioni turistiche/luoghi d'interesse:** l'utente deve poter visualizzare tali strutture memorizzate in un database e ricercarle;
- **Visualizzazione delle attrazioni turistiche/luoghi d'interesse su mappa:** l'utente deve poter visualizzare tali strutture anche su mappa, per rendersi conto della posizione di tali strutture rispetto alla propria, aumentando il livello di percezione di sé nello spazio;
- **Lettura di recensioni:** l'utente deve poter visualizzare le recensioni eventualmente lasciate per le strutture;
- **Creazione di un account:** l'utente deve poter creare un proprio account, l'utente mobile deve poter creare un proprio account per accedere alle funzionalità esclusive per questa classe di utenti (scrivere una recensione), ma può anche decidere di utilizzare l'applicazione come ospite non autenticato;
- **Pubblicazione di recensioni:** l'utente registrato deve poter scrivere e pubblicare una recensione per una certa struttura. Una recensione sarà intesa nella forma: voto (numero intero tra 1 e 5), titolo e corpo della recensione (con lunghezza massima per evitare casi di flood). Inoltre durante la pubblicazione di una recensione, l'utente può decidere di farlo anonimamente;
- **Acquisire punti e titoli:** l'utente registrato, utilizzando l'applicazione e rilasciando recensioni, deve poter acquisire punti per salire di livello o per ottenere titoli che servono per competere nella leaderboard nazionale.

4.1.2 Requisiti non-funzionali

- **User-centered:** il sistema deve essere volta ad una progettazione user-centered;
- **Interfaccia Mobile:** il Software deve essere disponibile per le interfacce Mobile con Sistema Operativo Android;

- **Comunicazione per mezzo di http(s)**: per sfruttare la potenza crescente del www, il software deve poter impiegare il protocollo http per effettuare le tipiche operazioni offerte (POST, PUT, DELETE, GET) avvalendosi di un'architettura RESTful;
- **Usabilità**: il Software deve essere facile da usare per l'utente;
- **Compatibilità**: il Software dovrebbe essere compatibile col maggior numero di dispositivi possibile, per abbracciare un pubblico maggiore, in termini di cardinalità e, di conseguenza, avere più end-user;
- **Velocità di risposta**: il Software dovrebbe rispondere in maniera molto rapida agli input dell'utente, soprattutto per quanto riguarda la visualizzazione di strutture; la risposta dovrebbe giungere entro 3 secondi nel 90% dei casi;
- **Impiego di Maps API**: per implementare il requisito di "visualizzazione di strutture su mappa", il Software dovrà impiegare un'API per Maps;
- **Affidabilità su diversa grandezza di dispositivi**: il Software dovrebbe funzionare sia su uno schermo molto piccolo che su uno molto grande, senza perdere in termini di qualità delle interfacce;
- **Multi-lingua**: è richiesto che l'app sia *moderna*, per questo motivo può avere senso introdurre una seconda lingua oltre l'italiano, come l'inglese, per abbracciare un pubblico più ampio;
- **Coerenza dei colori**: per aumentare il livello di usabilità, il Software dovrebbe utilizzare colori nitidi che non affaticino la vista ma che facciano trattenere l'attenzione dell'utente sulle parti significative dell'applicazione;
- **Paginazione**: il Software deve mostrare le strutture in maniera paginata¹, così da evitare carichi alla rete e rallentamenti di sistema;
- **http secure**: il protocollo utilizzato dalle operazioni che trattano dati sensibili - ad esempio login e registrazione - dovrebbe essere https per una maggiore sicurezza di trasmissione.

4.1.3 Requisiti di dominio

- **TC 159/SC 4**: il Software deve essere standardizzato rispetto all'ergonomia dell'interazione con l'utente e tutte le persone che lo hanno progettato e mantenuto. Le aree di standardizzazione comprendono l'ergonomia dell'hardware, del software, inclusa la progettazione dello human-computer interaction e i metodi e processi dello human-centered design, inclusa l'ingegneria dell'usabilità e i metodi di progettazione partecipativa;
- **ISO 13407 human-centered design**: il Software va progettato con un'accezione "umano-centrica";
- **GDPR**: il Software deve garantire l'adempimento della *General Data Protection Regulation*, cioè del regolamento generale sulla legge dei dati sensibili; questo perché l'utente sottemetterà dei dati personali, come email e password;
- **Privacy-by-design**: il Software è stato sviluppato in base al Regolamento Europeo sulla Privacy che impone alle organizzazioni e ai sistemi che trattano dati personali in modo che sia conforme da subito alle regole della privacy, spostando la responsabilità del corretto trattamento tramite strumenti informatici idonei sul titolare e sul responsabile del trattamento, quando identificato;
- **UE 679/2016**: il Software dovrà essere conforme ai requisiti imposti dal regolamento UE 679/2016 il quale prevede che tutti i dati personali siano protetti, compresa l'eventuale pseudonomizzazione dei dati personali - quando necessaria - e la cifratura dei dati stessi;

¹Mostrare sottoinsiemi di strutture

- **ISDP 10003:2015:** il Software dovrà aderire alle norme vigenti EU in tema di trattamenti di dati personali

4.2 Impiego delle risorse e pianificazione del lavoro

Il Project Management è consistito nella gestione del progetto sia in termini di risorse che in termini di benessere psicologico del team operativo. La gestione verteva sul:

- **Definire un obiettivo da raggiungere:** consegnare il progetto
- **Pianificare un modo per conseguirlo:** abbiamo usato il modello a cascata (*waterfall model*);
- **Definire risorse:** il team a disposizione era composto dagli studenti Quirile Alessandro e Telese Mauro;
- **Effettuare checkpoint ovvero date significative;**
- **Effettuare valutazioni periodiche.**

Le *key-question* a cui un project management dovrebbe rispondere sono le seguenti:

- **Cosa e quando realizzare** - bisogna l'applicazione entro Marzo 2021;
- **Quanto costerà** - 0 euro (sono stati usati servizi gratuiti);
- **Come è stato misurato l'andamento del progetto** - attraverso apposite metodologie e cronodiagrammi;
- **Cosa deve fare l'applicazione, a chi essa è destinata e in che modo è stata valutata l'usabilità del prodotto.**

Il primissimo passo effettuato è stato scegliere il modello che al meglio potesse rispondere al nostro working plan. È stato scelto il modello a cascata per numerosi motivi:

- **Supera l'approccio "code and fix"** - obbliga gli sviluppatori ad una disciplina e ad una pianificazione più accurata
- **Supera le difficoltà dei modelli agili** - ad esempio un modello come SCRUM avrebbe necessitato di un team più grande, di ceremonie e artefatti: troppo complicato
- **Gli attributi intrisechi del modello sono ottimali** - comprensibilità² alta, visibilità³ eccellente, supportabilità⁴ alta, accettabilità⁵ discreta, affidabilità⁶ massima, robustezza⁷, manutenibilità⁸ ottima e rapidità⁹ scarsa

Tuttavia, gli svantaggi di questo modello sono:

- **Sequenzialità** - ogni fase genera un documento per quella successiva e quindi è molto importante che non si commettano errori fase per fase

²Quanto è facile interpretare il tutto

³A che punto si è con lo sviluppo?

⁴Quanti strumenti di supporto ci sono?

⁵Le persone del team sono contente nel ruolo assegnato?

⁶Quale sarà la qualità del software prodotto?

⁷Quanto è robusto il software a cambiamenti in corso d'opera?

⁸Quanto è robusto il software a cambiamenti futuri?

⁹Quanto tempo ci vuole per il deploy?

- **Delicatezza della fase di Requirement Collection** - la fase di raccolta dei requisiti col cliente è, nel modello a cascata, estremamente delicata ed andrebbe chiesto al cliente qualsiasi cosa non sia completamente chiara
- **Richiede ordine e pianificazione (estrema)** - questo modello forza gli sviluppatori a un ordine zelante

Dopo aver scelto il modello che al meglio si presti alla causa, occorre definire il *Work Breakdown Structure*, ovvero una struttura ad albero che descrive le fasi del progetto. Esso è rappresentato in Figura 4.1

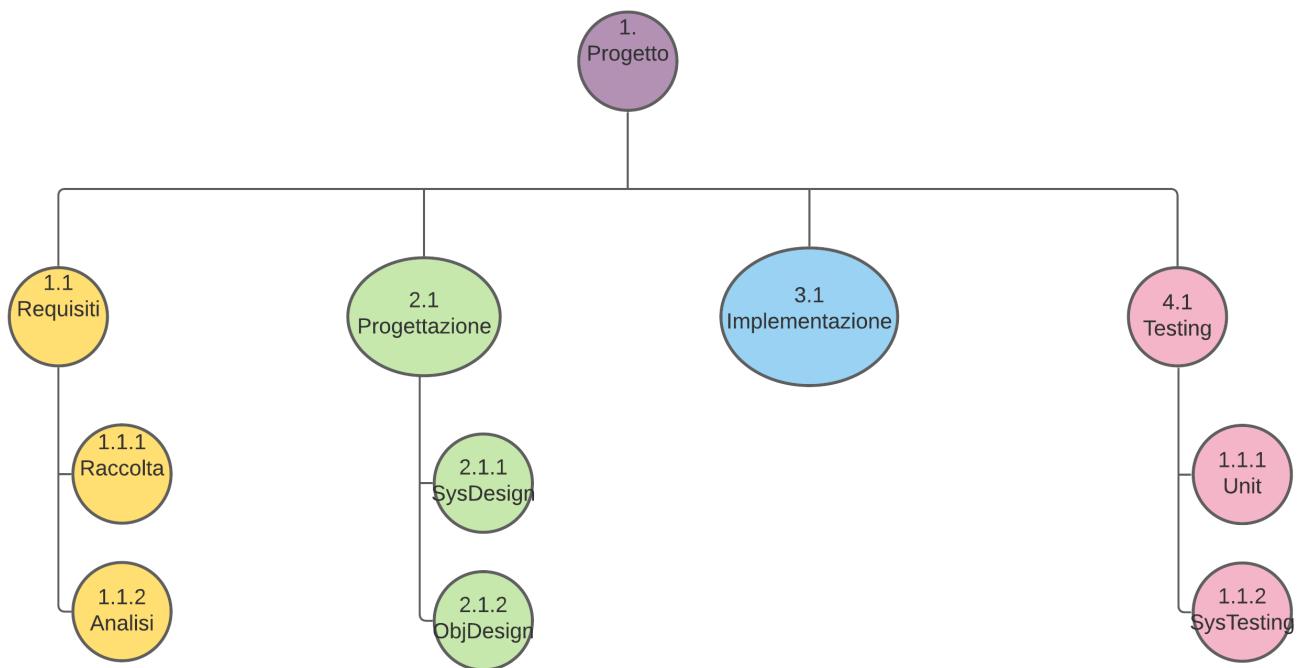


Figura 4.1: Project Management - Work Breakpoint Structure

Di seguito è definito il diagramma di PERT che consente di specificare dipendenze fra *task* e calcolare un percorso critico che non dovrebbe mai subire rallentamenti. Il diagramma di PERT è essenzialmente un grafo orientato che descrive le attività del progetto ed è mostrato in Figura 4.2

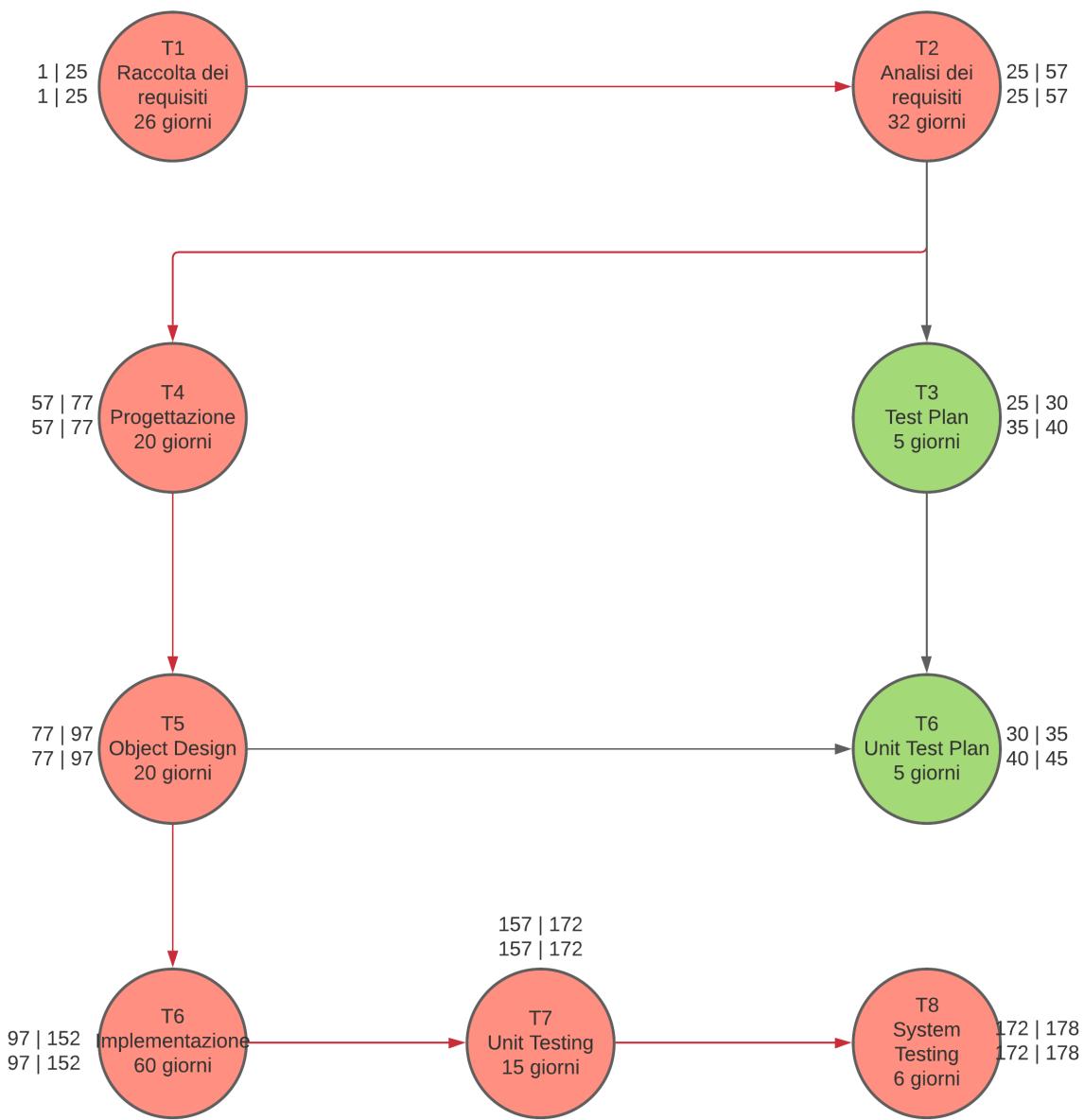


Figura 4.2: Project Management - Diagramma di PERT

I nodi e gli archi marcati in rosso sono quelli che costituiscono il cammino critico, ovvero quel percorso le cui attività non dovrebbero mai essere ritardate (e per cui *best case* e *worst case* coincidono)

L'attività di testing va intesa come system testing, functional testing, integration testing, unit testing e usability testing che accompagnano ogni fase costruttiva del percorso, come mostrato in Figura 4.3. È importante parallelizzare le fasi costruttive con quelle distruttive in primis per ragioni di tempo (è impensabile spendere sei mesi, ad esempio, in costruzione ed altri sei in distruzione), ma anche perché - avendo scelto il modello a cascata - era essenziale assicurarsi che non vi fossero errori generati in una fase e immessi in input alla fase successiva.

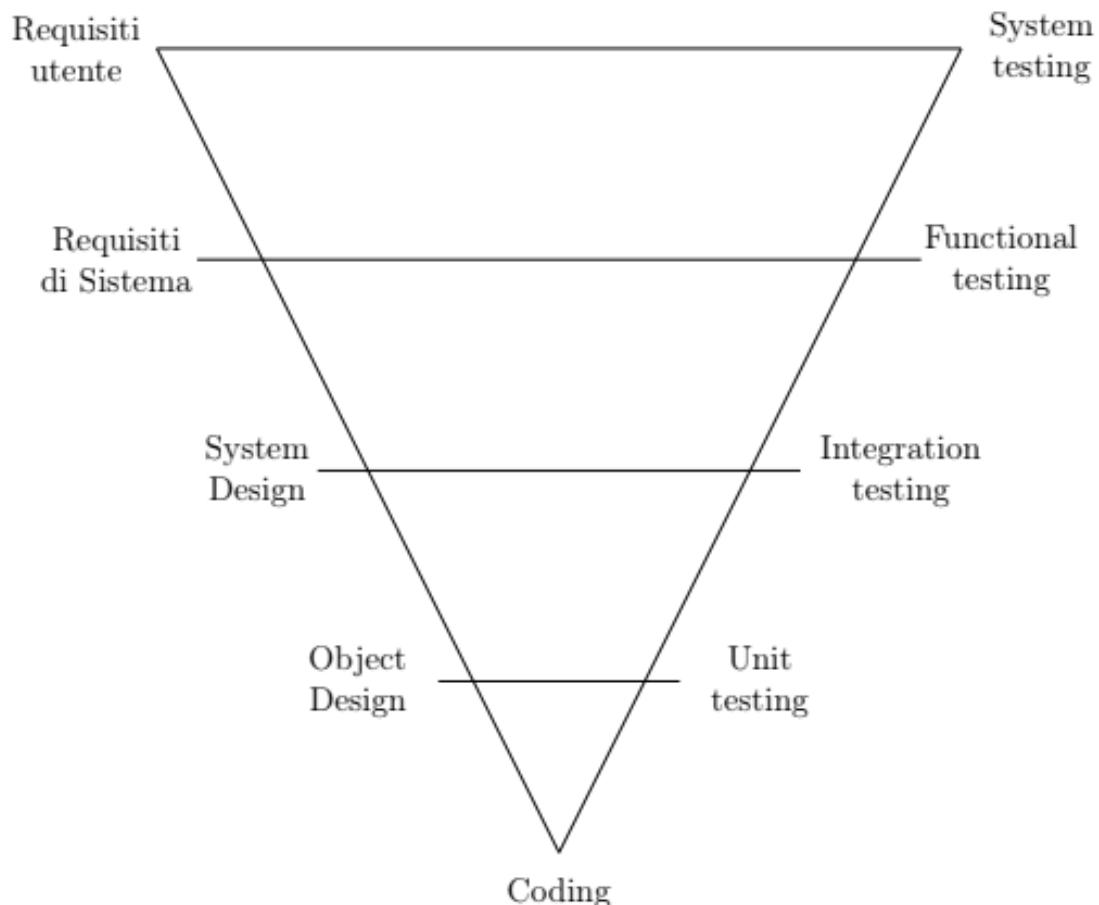


Figura 4.3: Project Management - Costruzione e Distruzione

4.3 Modellazione dei casi d'uso

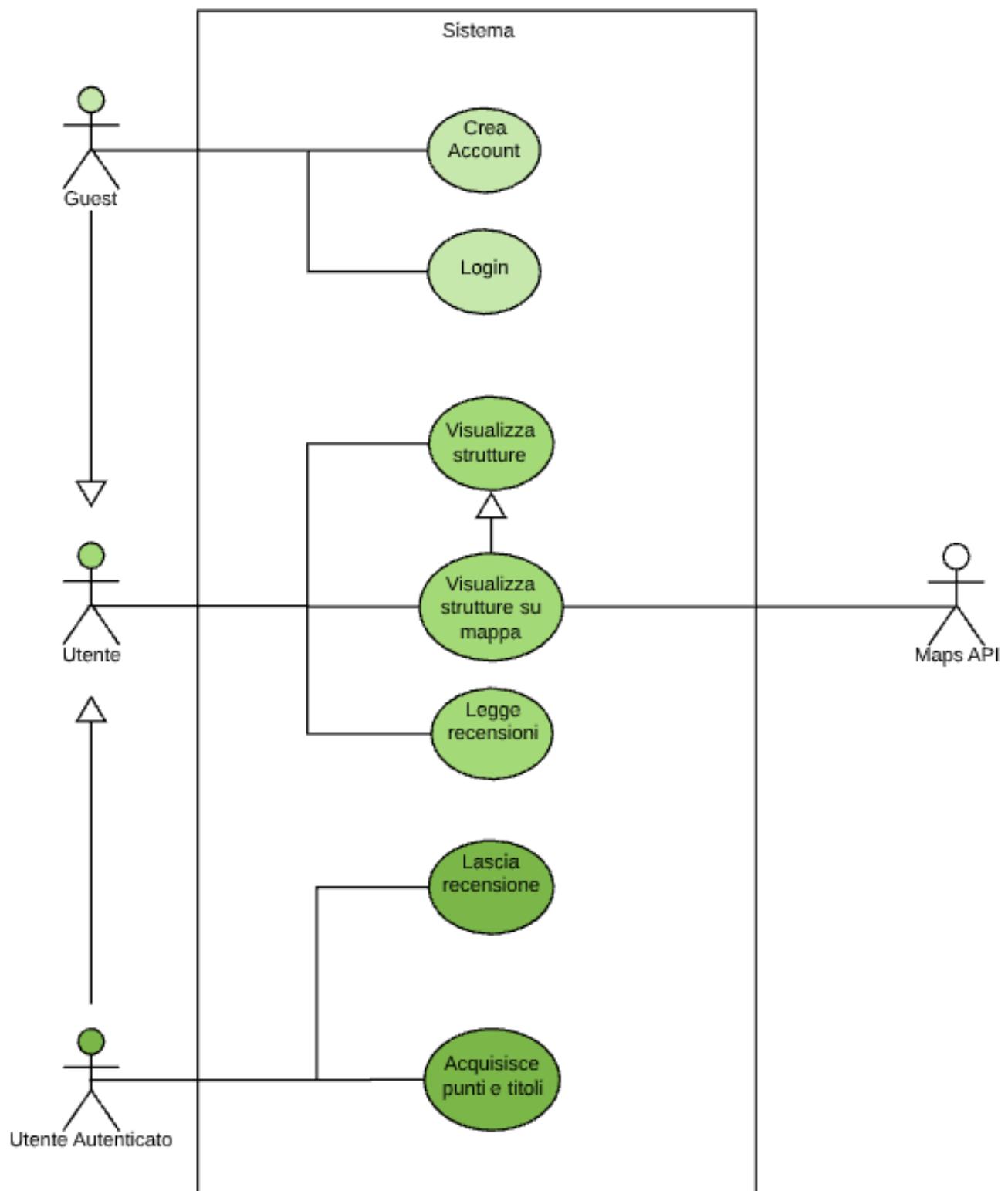


Figura 4.4: Use Case Diagram

Capitolo 5

Prototipazione

In questo capitolo verrà esibita la prototipazione del sistema, intesa come prototipazione funzionale, visuale e sistemica.

5.1 Funzionale

Per venire in contro all'esigenza di progettare un sistema user-centered è importante che nell'ampio ventaglio di possibilità di costruzione di prototipi ci si concentri soprattutto su quelli di ruolo e di interfaccia, in particolare che siano interattivi e a bassa o alta fedeltà. Al di là di questa classificazione bisogna costruire i prototipi anche in base alla loro fedeltà al prodotto finale, alla loro durata e alla loro completezza. Se la bassa fedeltà non costituisce un prototipo meno utile, è importante invece ricercare la completezza funzionale: un prototipo orientato in orizzontale riesce, anche in modo poco fedele, a fornire e schematizzare all'utente tutte le funzionalità di cui il sistema si compone; quello verticale, invece, realizza in maniera compiuta un insieme limitato di funzionalità. È importante quindi cercare di massimizzare sia il livello di dettaglio del sistema ma anche il numero di funzionalità mostrate dai prototipi, in modo tale da avere un set di completezza. La Figura 5.1 mostra alcuni schizzi iniziali e approssimativi delle principali funzioni e schermate dell'applicazione; è una bozza che serve principalmente a fissare le idee e non tiene conto solamente delle schermate più importanti. Le frecce indicano, sebbene in modo non sistematico, alcune possibili sequenze di interazione. Ad esempio la freccia in giallo che parte dalla voce "Profilo" serve per visualizzare il profilo dell'utente dalla homepage dell'applicazione, che mostra invece una lista approssimativa delle strutture nelle vicinanze. La freccia in rosa dalla schermata del profilo, invece, porterà ad una schermata che implementa una delle funzionalità dell'applicazione calata in un contesto di gamification, mostrando la propria posizione e l'andamento della classifica nazionale, basata principalemtnre sui livelli cumulabili al rilascio delle recensioni. La freccia blu scuro nella homepage serve invece per ricercare le strutture, mentre quella in verde serve per visualizzare le informazioni della struttura cliccata, della quale si potranno leggere le recensioni (freccia azzurra) oppure scriverne una (freccia viola). Questo schizzo, sebbene sia a bassa fedeltà e principalmente è un prototipo statico, riesce comunque ad essere evolutivo e fornisce tutte le funzioni del prodotto finale, anche se in modo semplificato, ricercando quindi la completezza funzionale orizzontale.

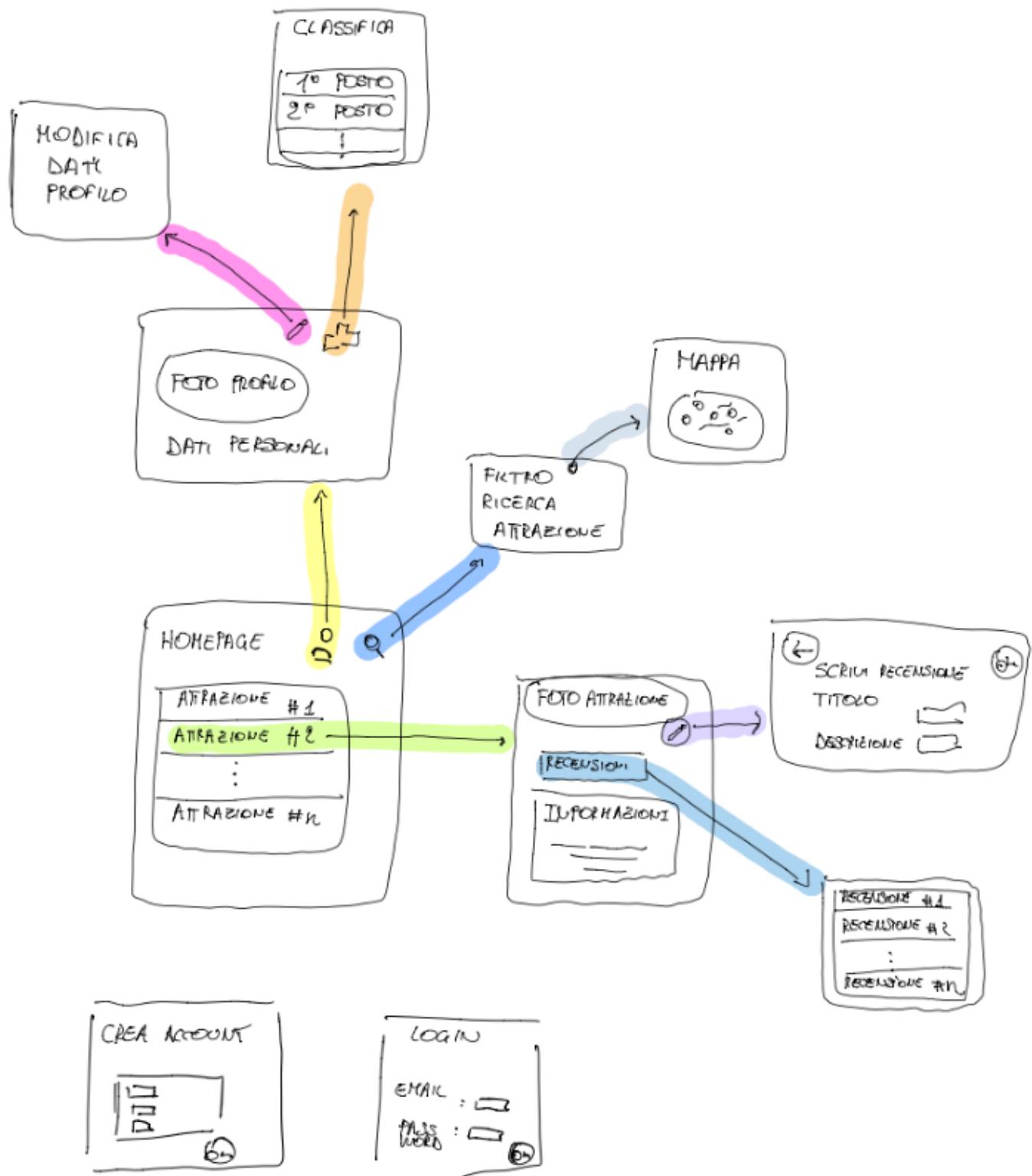


Figura 5.1: Prototipazione "a schizzo" del prodotto per la completezza orizzontale

La Tabella 5.1 sintetizza le funzionalità ricoperte dalla Figura 5.1. Si noti che poiché tale Figura rappresenta un prototipo orizzontale dell'intero sistema, sono esibite tutte le funzionalità del sistema stesso.

Tabella 5.1: Verifica della prototipazione orizzontale

Obiettivo	Verificare che il prototipo è di tipo orizzontale		
	Funzionalità valutata	Risultato atteso	Risultato ottenuto
Crea Account	✓	✓	✓
Login	✓	✓	✓
Visualizza strutture	✓	✓	✓
Visualizza strutture su mappa	✓		✓
Legge recensioni	✓		✓
Scrive recensione	✓		✓
Acquisizione punti/titoli	✓		✓
Notes			

Un prototipo con fedeltà più alta e un maggior grado di completezza verticale è invece dato dallo schema in Figura 5.21. Questo prototipo, infatti, fornisce una rappresentazione più fedele di quella che sarà la schermata finale, mostrando una sola funzionalità ma esemplificata in dettaglio, ricercando quindi la completezza funzionale verticale. Nella fattispecie, viene dettagliata l'interfaccia e la relativa funzionalità di login: come si nota, infatti, sono presenti due campi ed il pulsante per compiere il login.

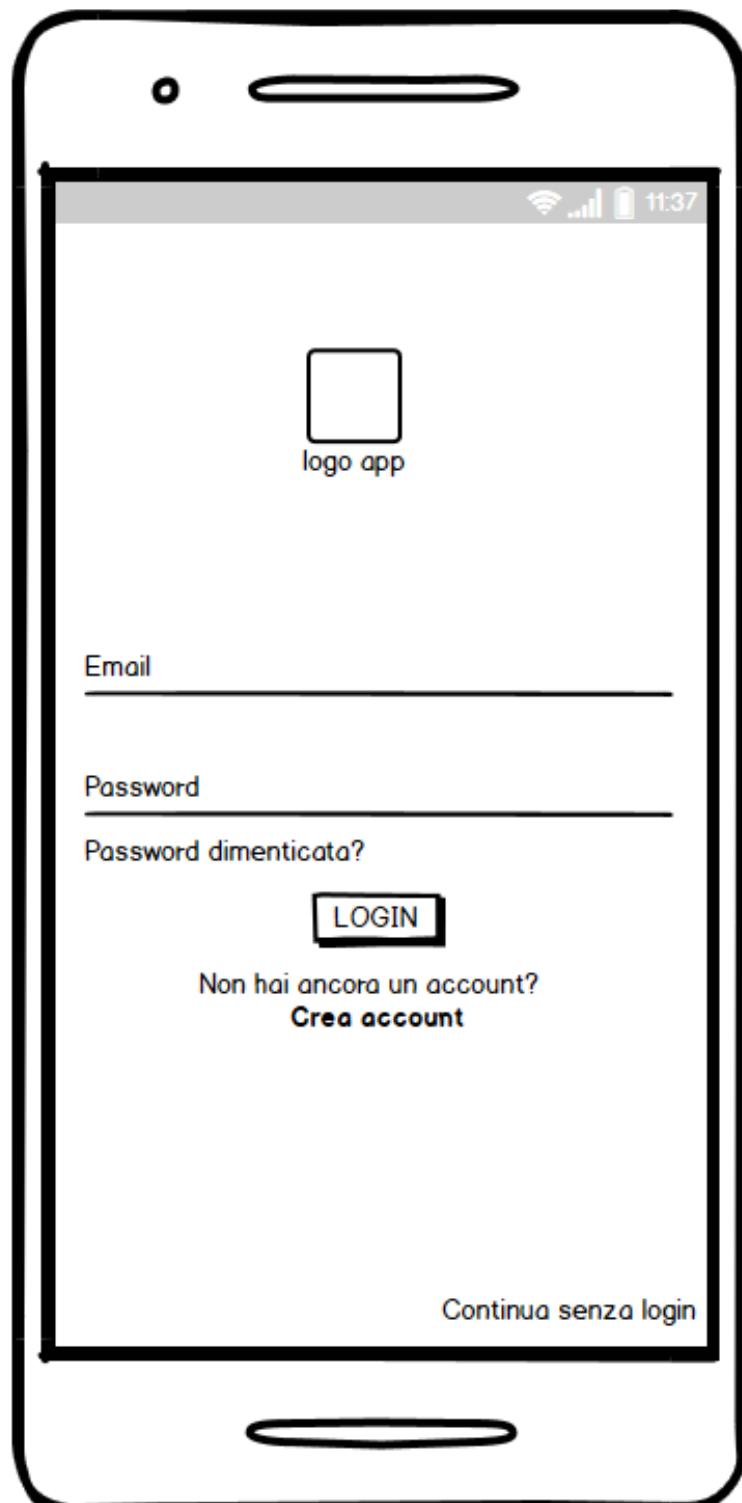


Figura 5.2: Prototipo verticale per il login

5.2 Visuale

In questo paragrafo sono mostrate le principali schermate del prodotto a seconda della loro fedeltà col prodotto finale.

5.2.1 Fedeltà media

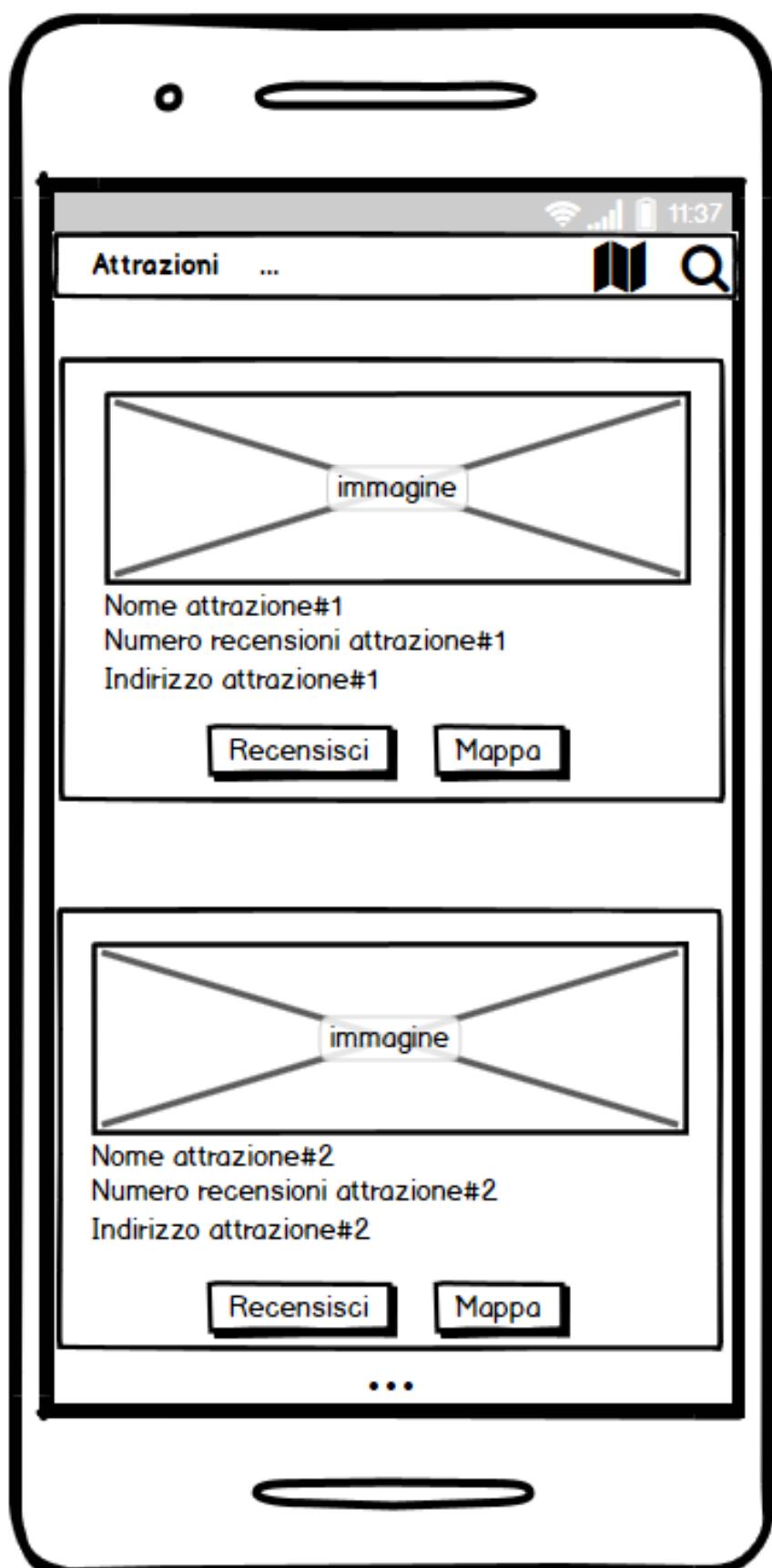


Figura 5.3: Mockup per la homepage



Figura 5.4: Mockup per il caso d'uso Cerca Strutture



Figura 5.5: Mockup per il caso d'uso Visualizza Strutture su Mappa



Figura 5.6: Mockup per il caso d'uso Cerca Strutture su Mappa

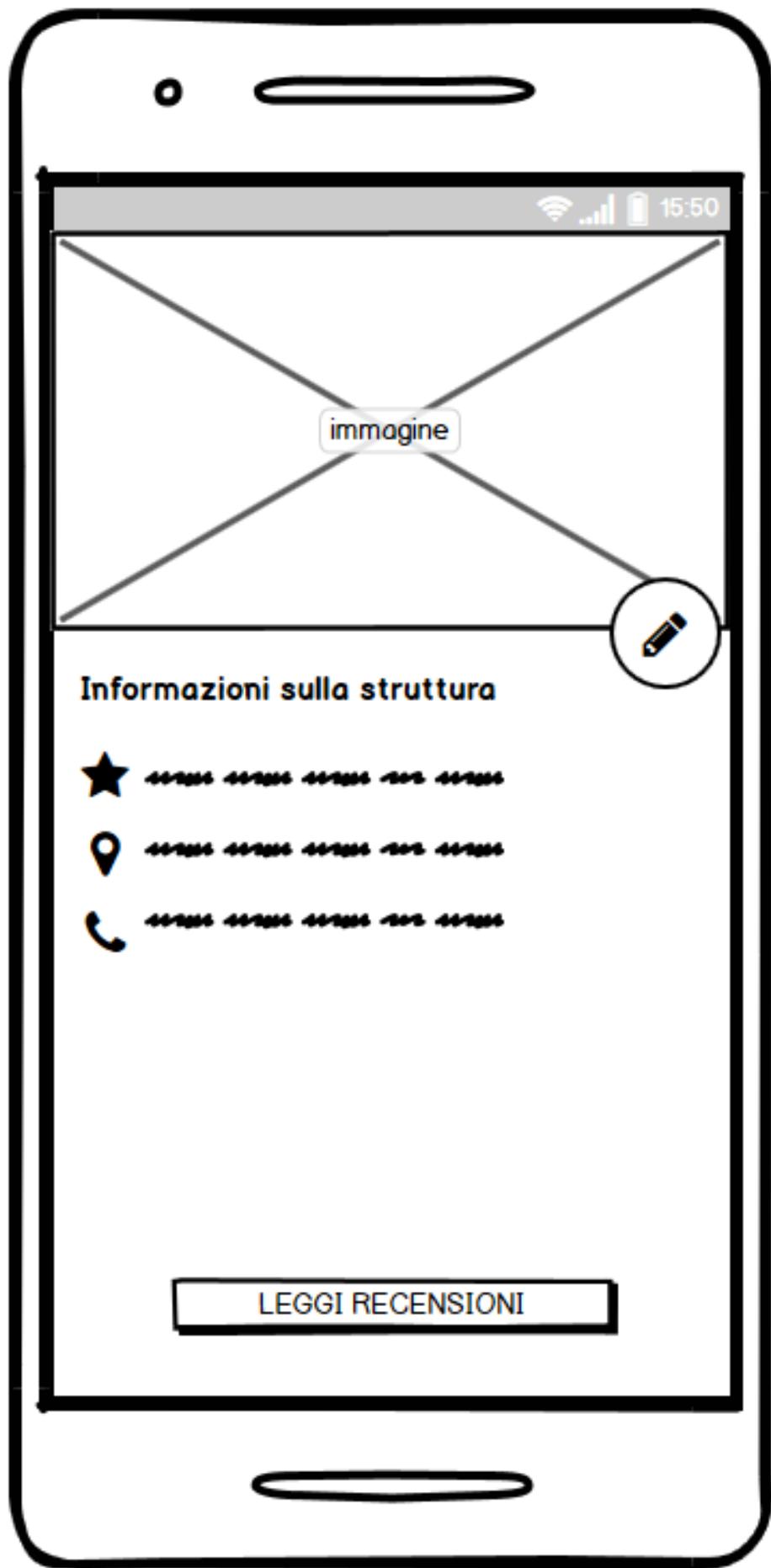


Figura 5.7: Mockup per l'overview di un'attrazione

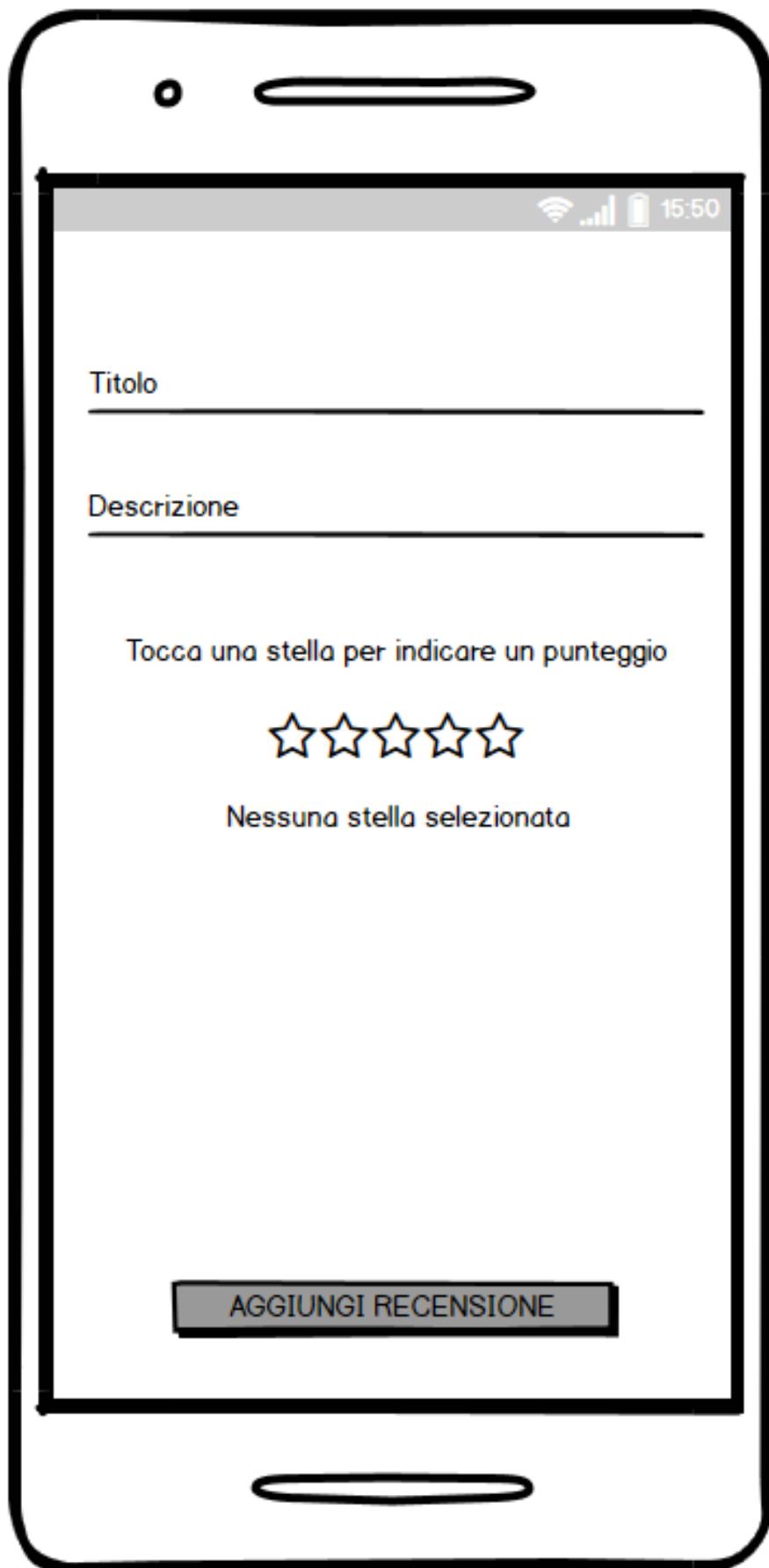


Figura 5.8: Mockup per il caso d'uso Lascia Recensione

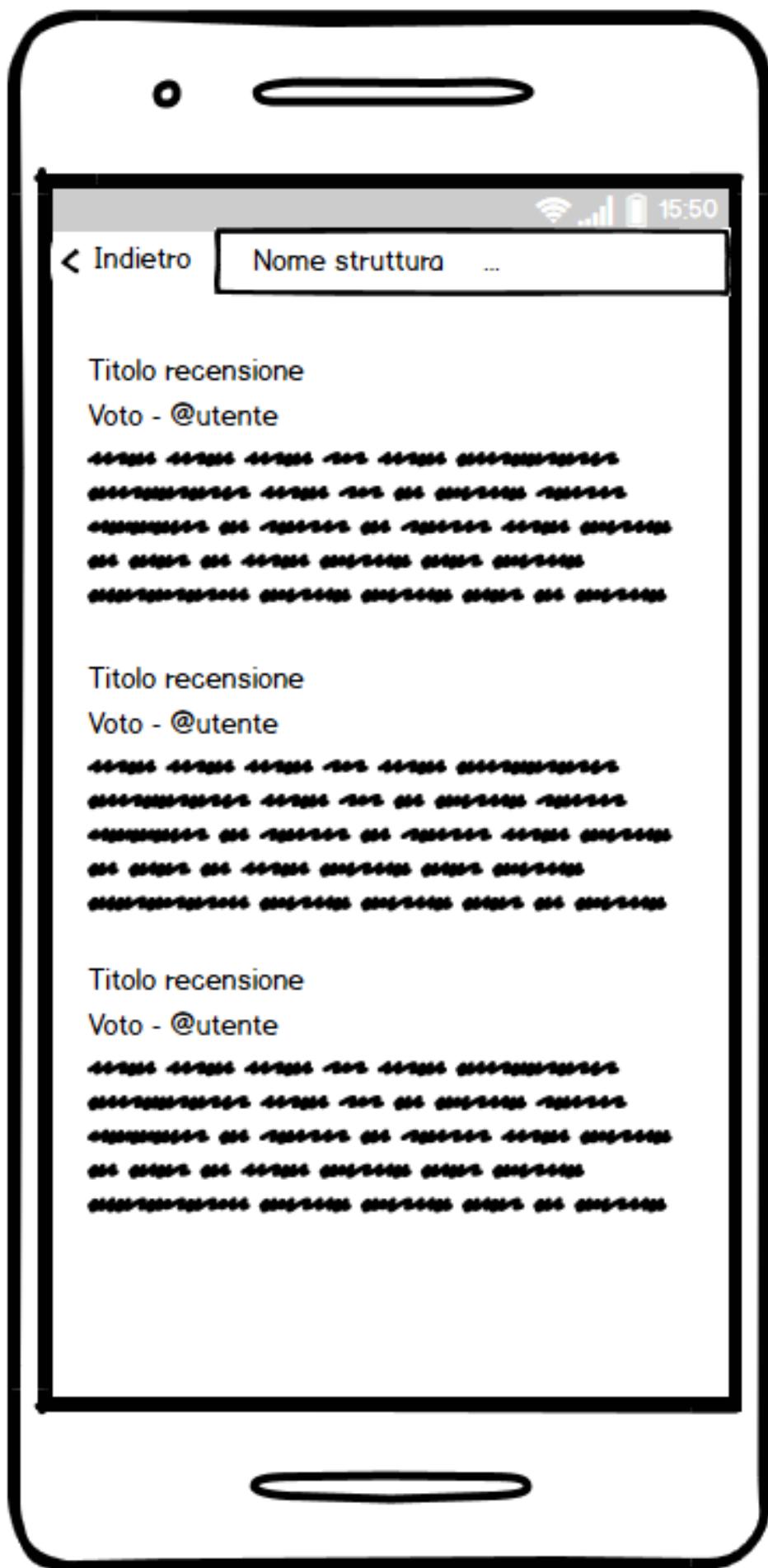


Figura 5.9: Mockup per la lista di recensioni

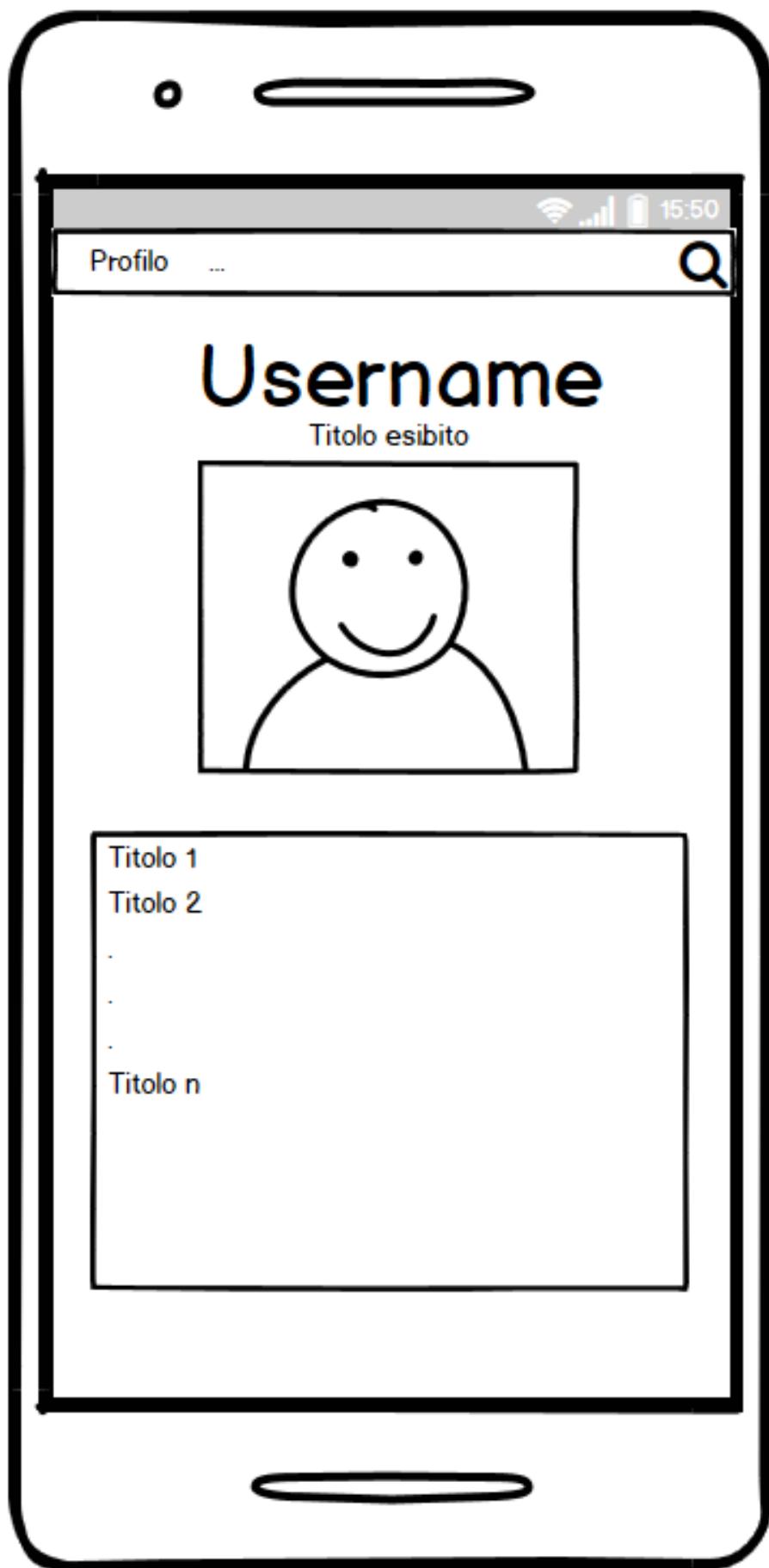


Figura 5.10: Mockup per il profilo

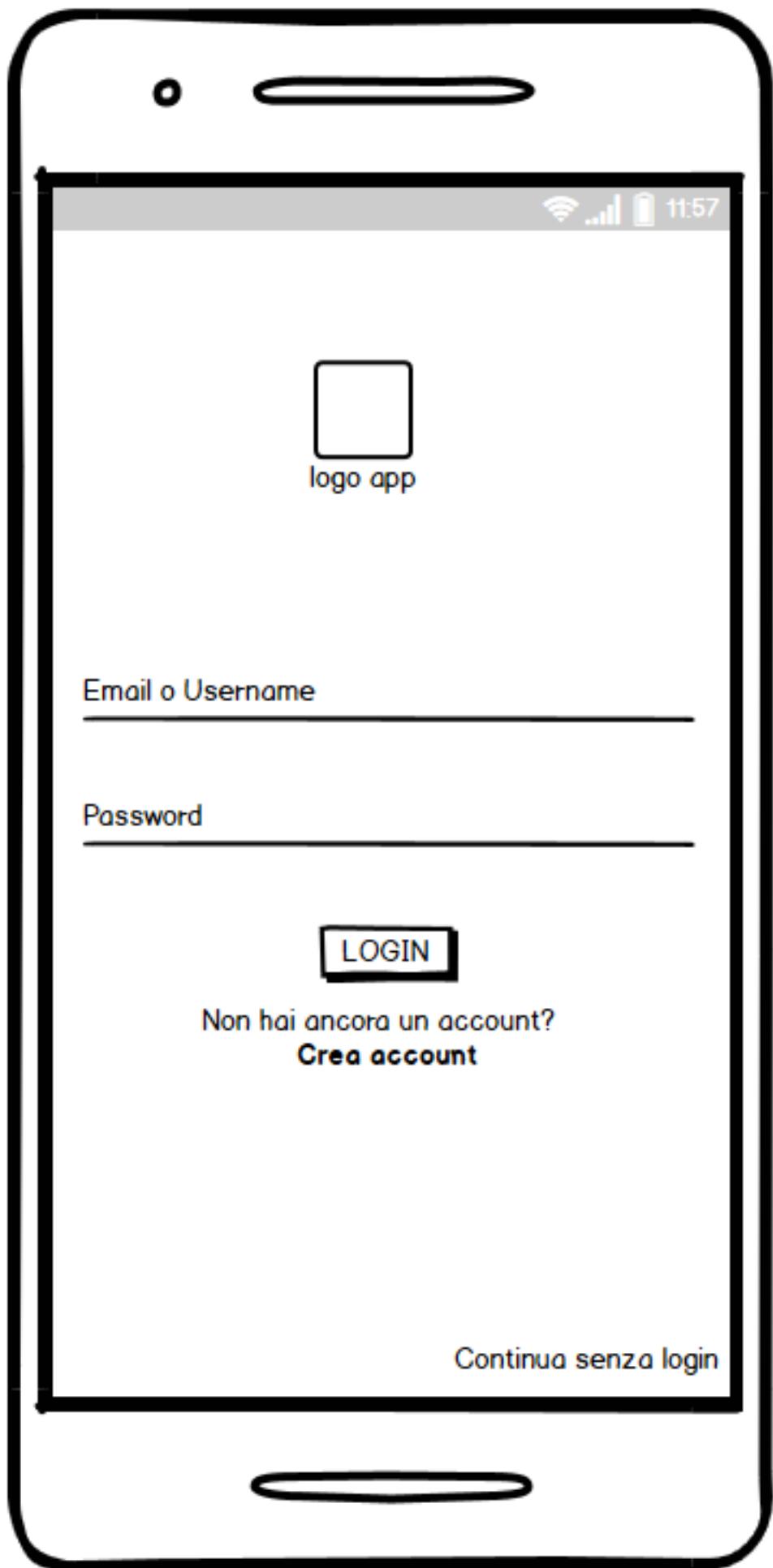


Figura 5.11: Mockup per il caso d'uso Login

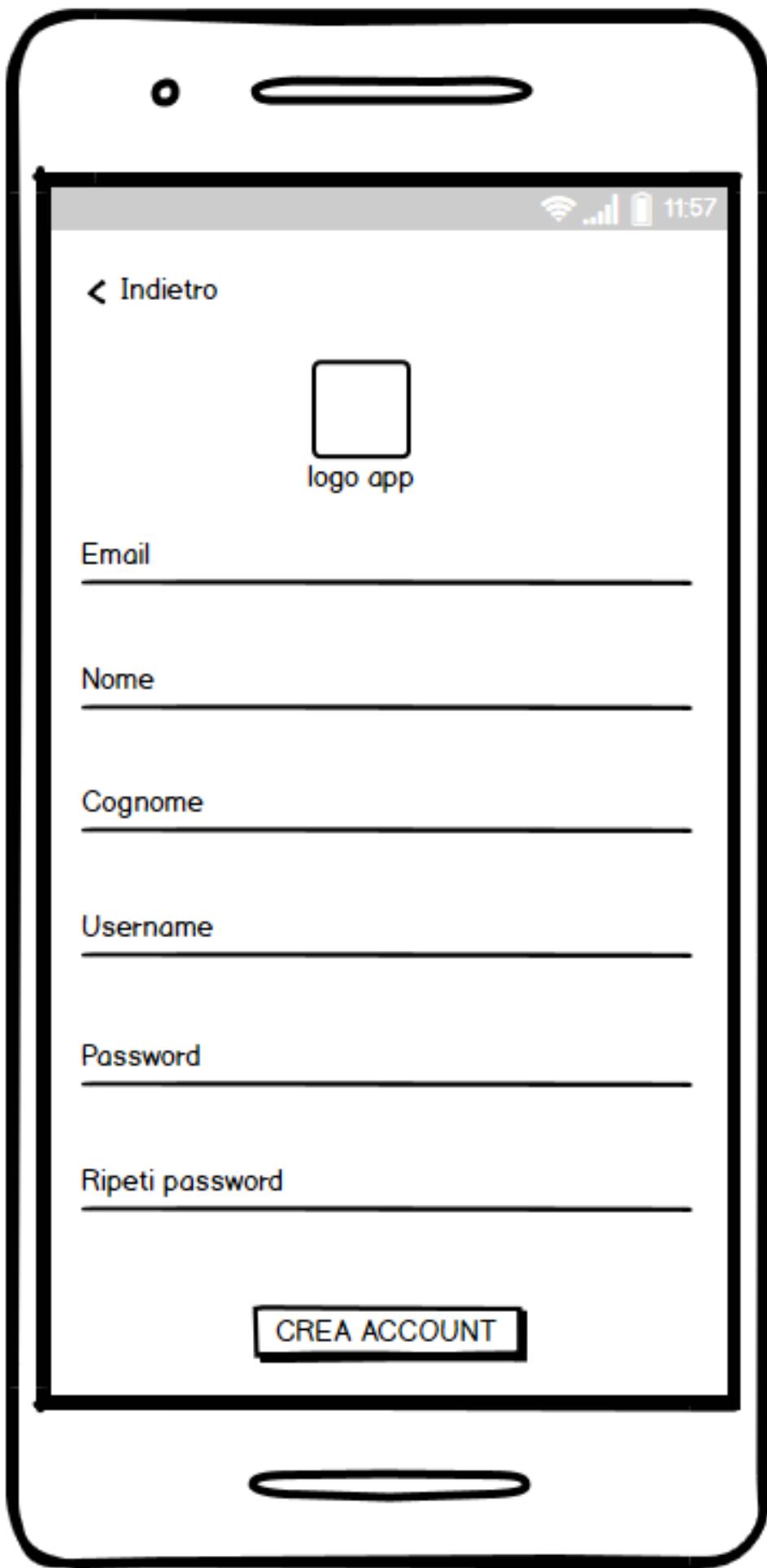


Figura 5.12: Mockup per il caso d'uso Crea Account

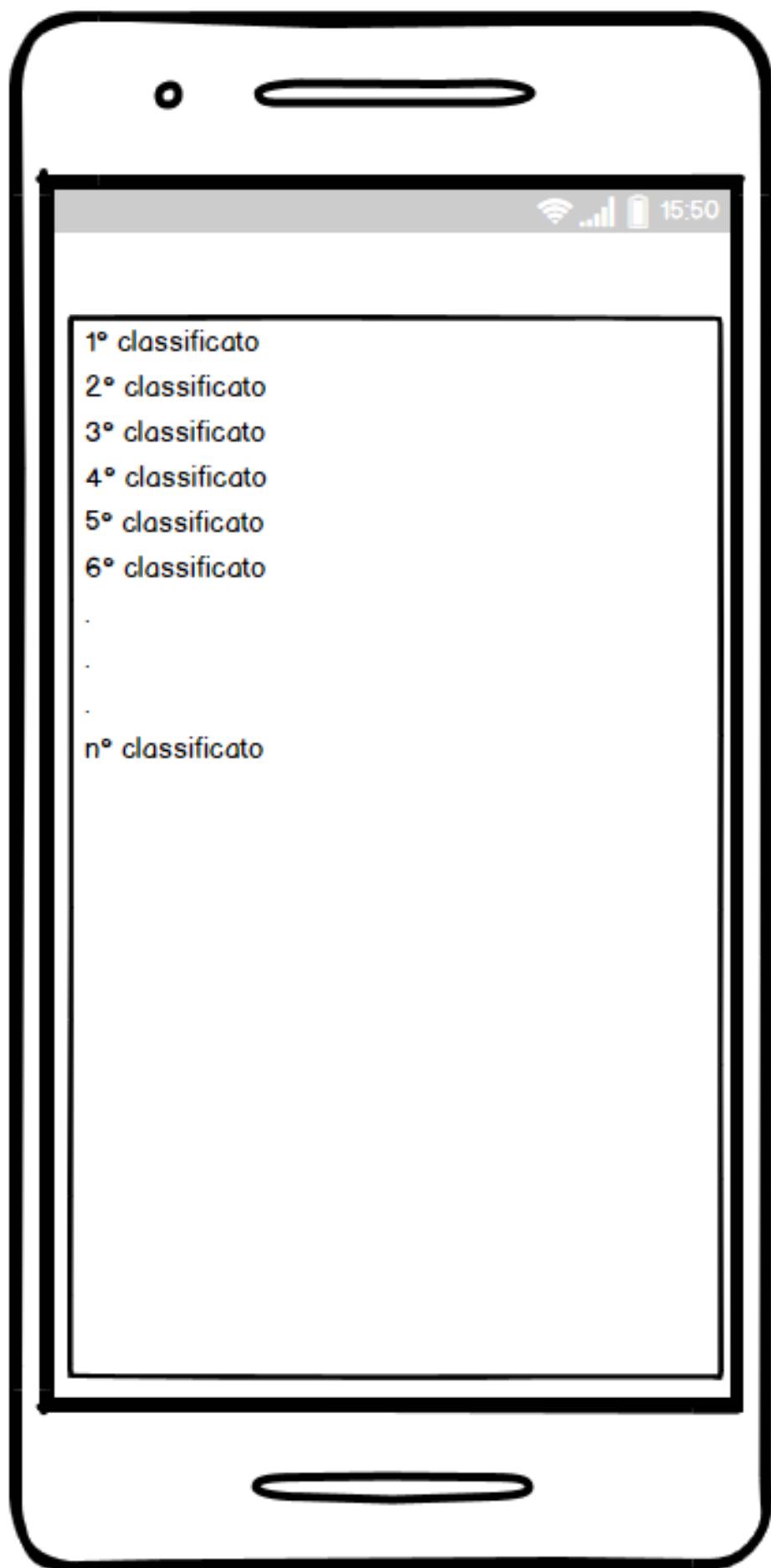


Figura 5.13: Mockup per la classifica

5.2.2 Fedeltà alta

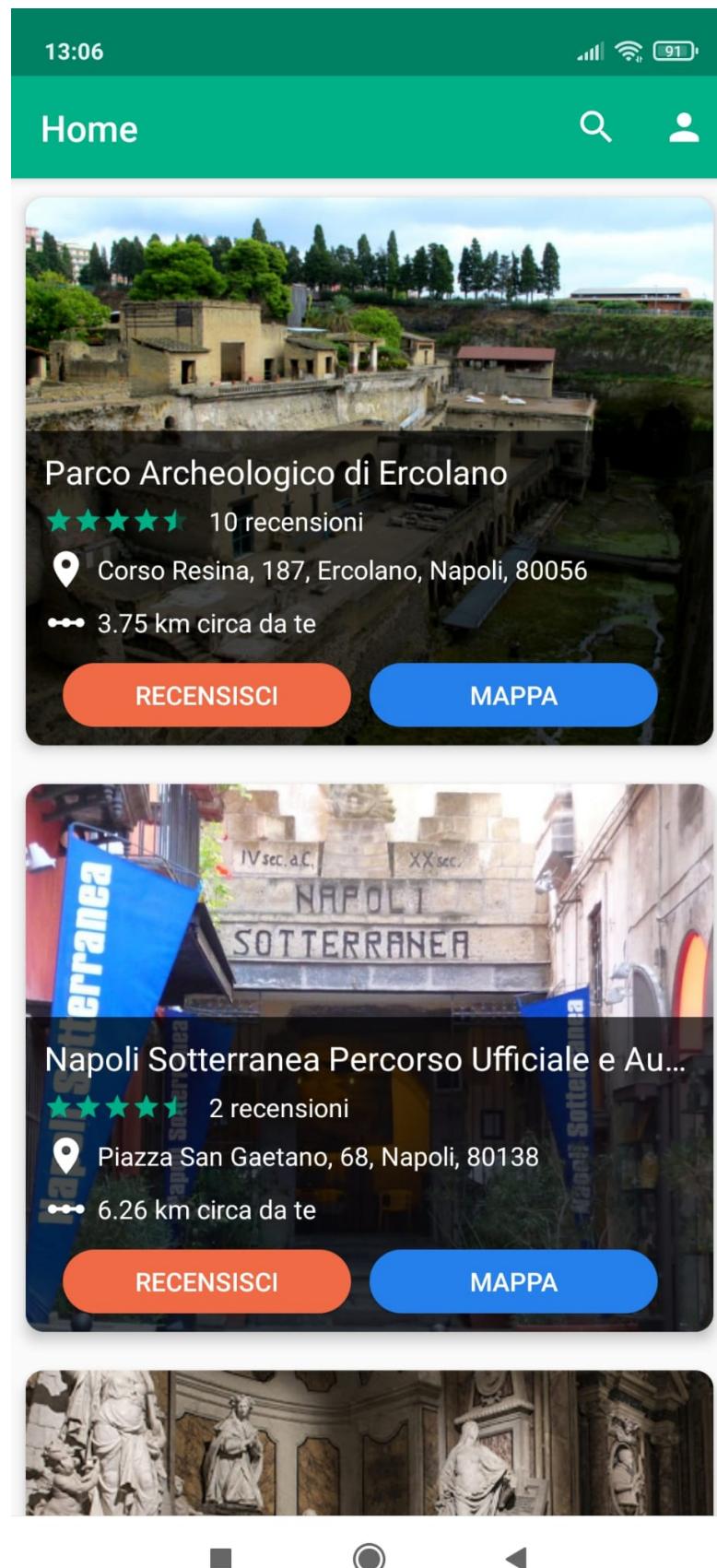


Figura 5.14: Mockup per la homepage

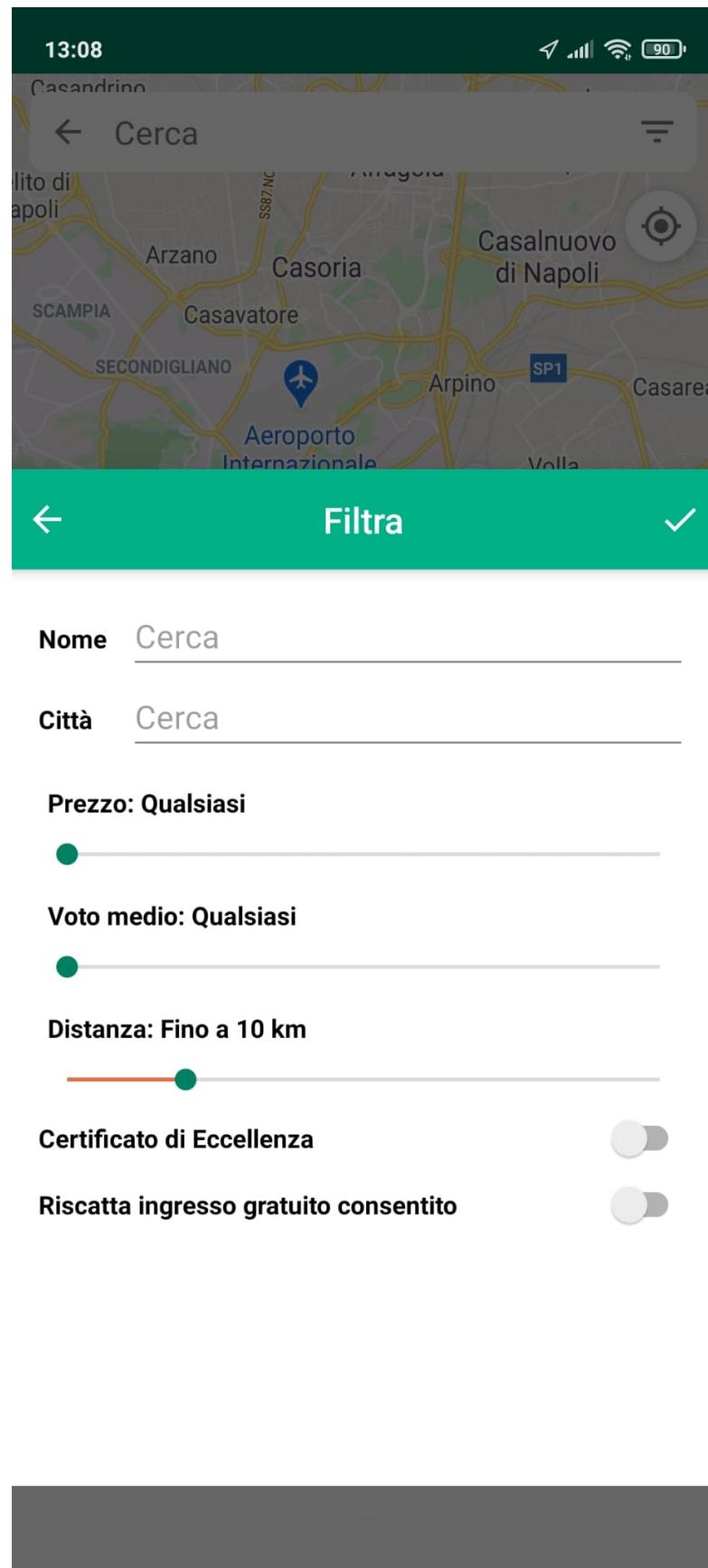


Figura 5.15: Mockup per la ricerca di strutture su mappa

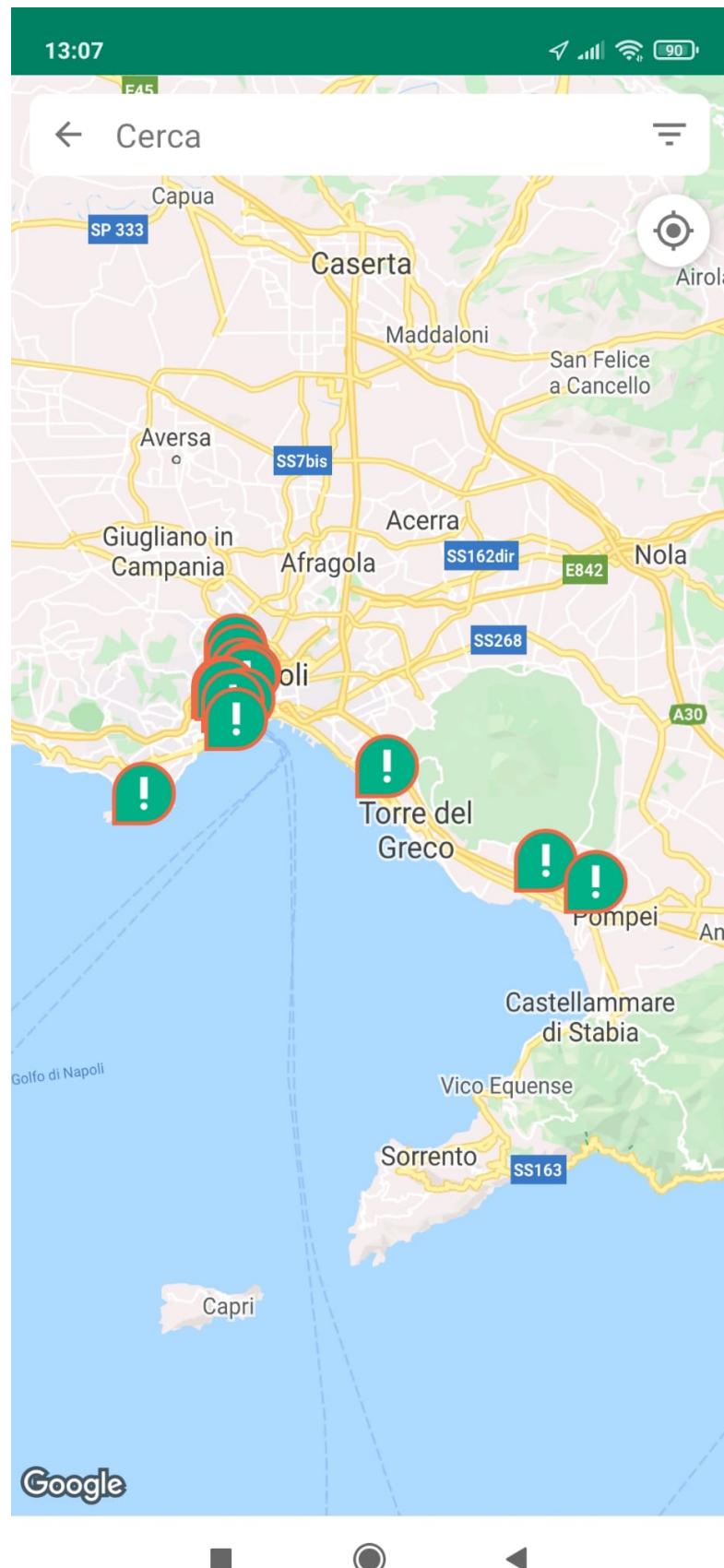


Figura 5.16: Mockup per la mappa

The mockup displays a mobile application interface for a historical site. At the top, there is a large image of a stone archway with two red cylindrical columns, labeled "Parco Archeologico di Ercolano". Below the image, the title "Parco Archeologico di Ercolano" is shown, followed by a green circular icon with a white pencil symbol.

Quello che gli utenti dicono

★★★★★ 10 recensioni

- Il luogo si trova nei pressi della mia abitazione. È raggiungibile anche a piedi ed è un luogo carino
- Il luogo è tranquillo e la gente gentile
- Un tuffo nella storia

[Leggi Recensioni >](#)

Informazioni sulla struttura

Certificato di Eccellenza

Corso Resina, 187, Ercolano, Napoli, 80056

Riscatta ingresso gratuito (26 gettoni)
Insufficienti: 10 mancanti

Orari di apertura

Lunedì: 08:30 – 17:00
Martedì: 08:30 – 17:00
Mercoledì: 08:30 - 17:00
Giovedì: 08:30 - 17:00
Venerdì: 08:30 - 17:00
Sabato: 08:30 - 17:00
Domenica: 08:30 - 17:00

Figura 5.17: Mockup per l'overview di una struttura

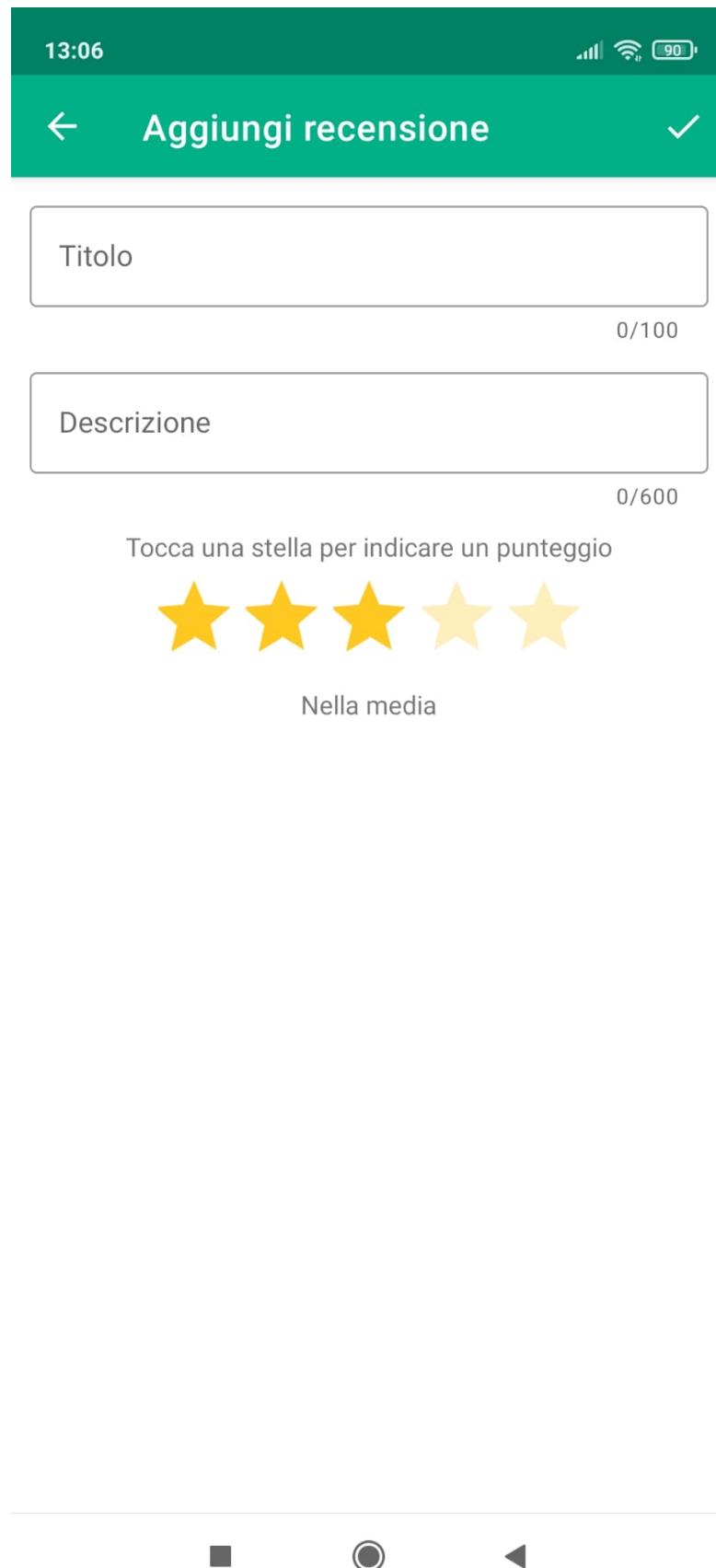


Figura 5.18: Mockup per la scrittura di una recensione

The mockup displays a list of reviews for the Archaeological Park of Ercolano. Each review includes the reviewer's name, profile picture, rating, date, and a short comment. There are also like and dislike counts and a share icon.

- Vicino casa**
★★★★★ 22-3-2021
 **Alessandro Quirile - daiquiri**
Super utente
Il luogo si trova nei pressi della mia abitazione. È raggiungibile anche a piedi ed è un luogo carino
È stato utile?  1  0 
- Bella esperienza**
★★★★★ 22-3-2021
 **Annamaria Mangone - Rodari99**
Utente medio
Il luogo è tranquillo e la gente gentile
È stato utile?  3  1 
- Da visitare**
★★★★★ 22-3-2021
 **Paolo Ascolese - Paolo99**
Nuovo membro
Un tuffo nella storia
È stato utile?  2  1 
- Arte, cultura, secoli di storia**
★★★★★ 23-3-2021
 **Marco Barese - mimmopanini98**
Nuovo membro
Questo posto racchiude secoli di storia. Incantevole essere immersi o commersi dall'arte. Ci tornerà sicuramente


Figura 5.19: Mockup per la lettura di recensioni

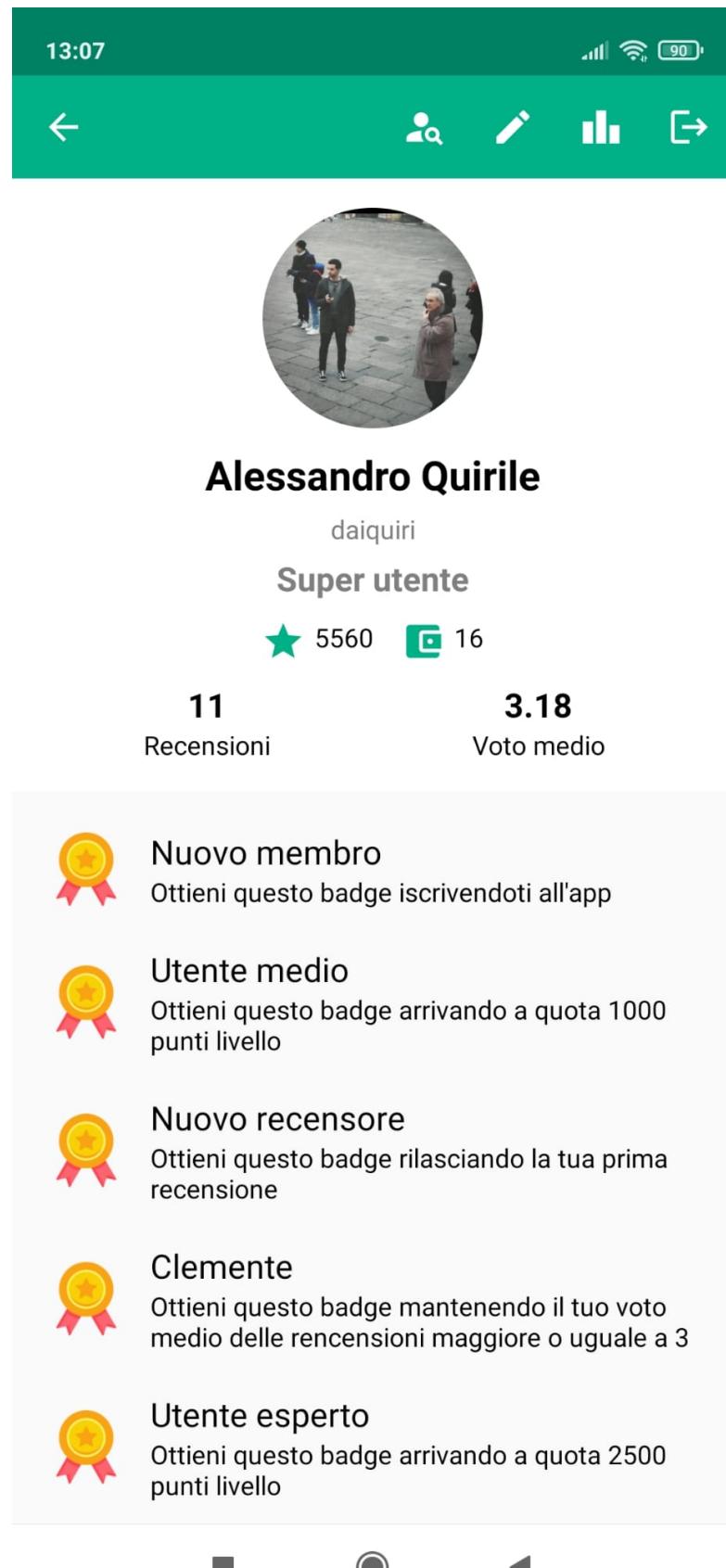


Figura 5.20: Mockup per il profilo

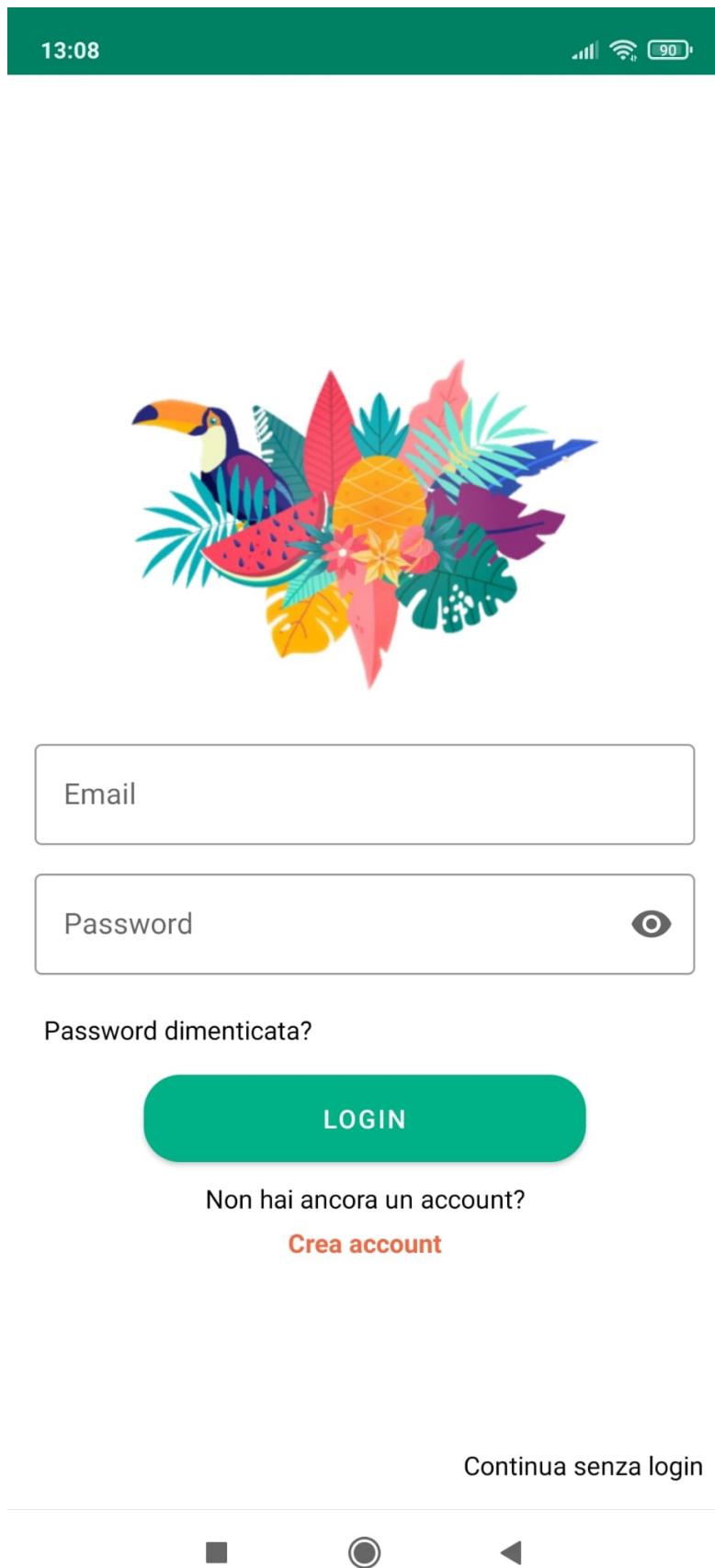


Figura 5.21: Mockup per il login

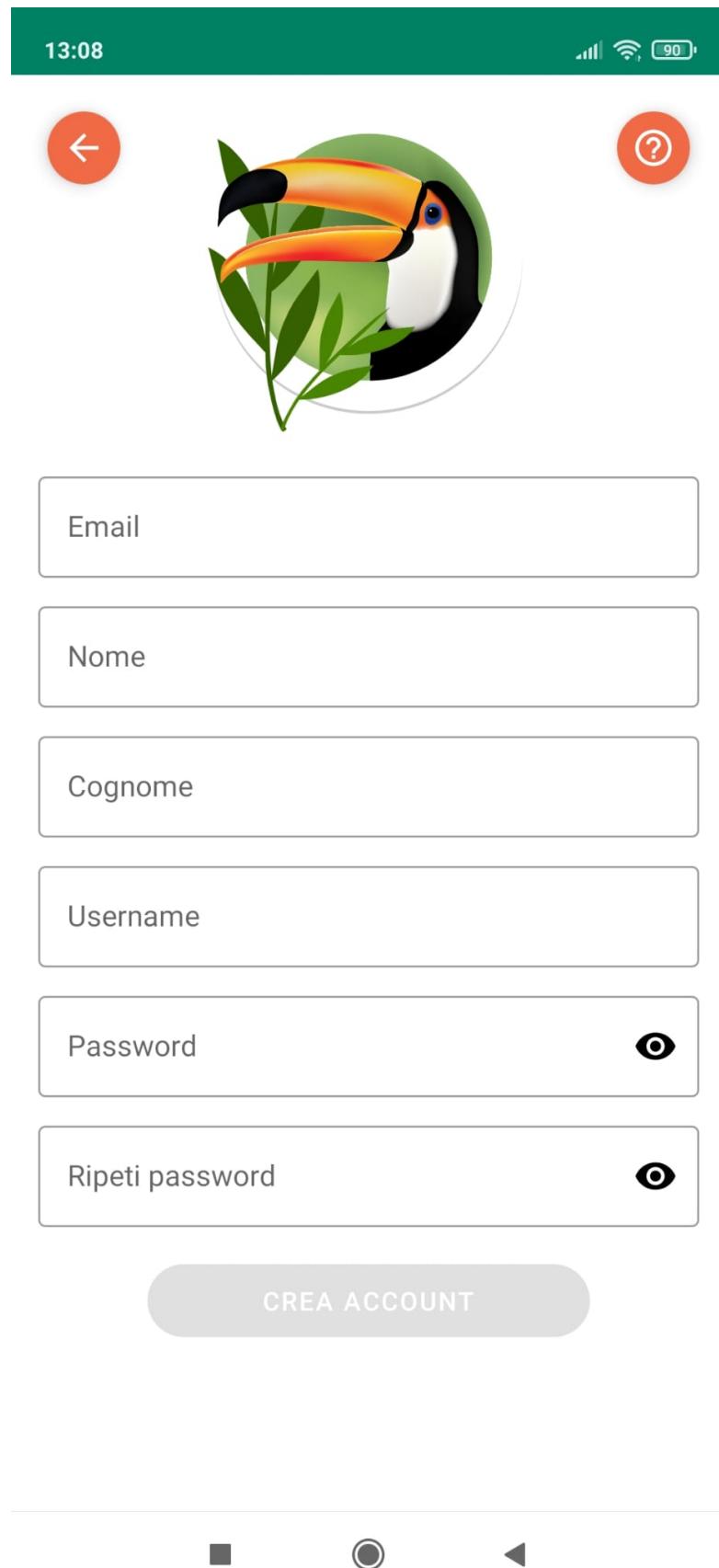


Figura 5.22: Mockup per la registrazione

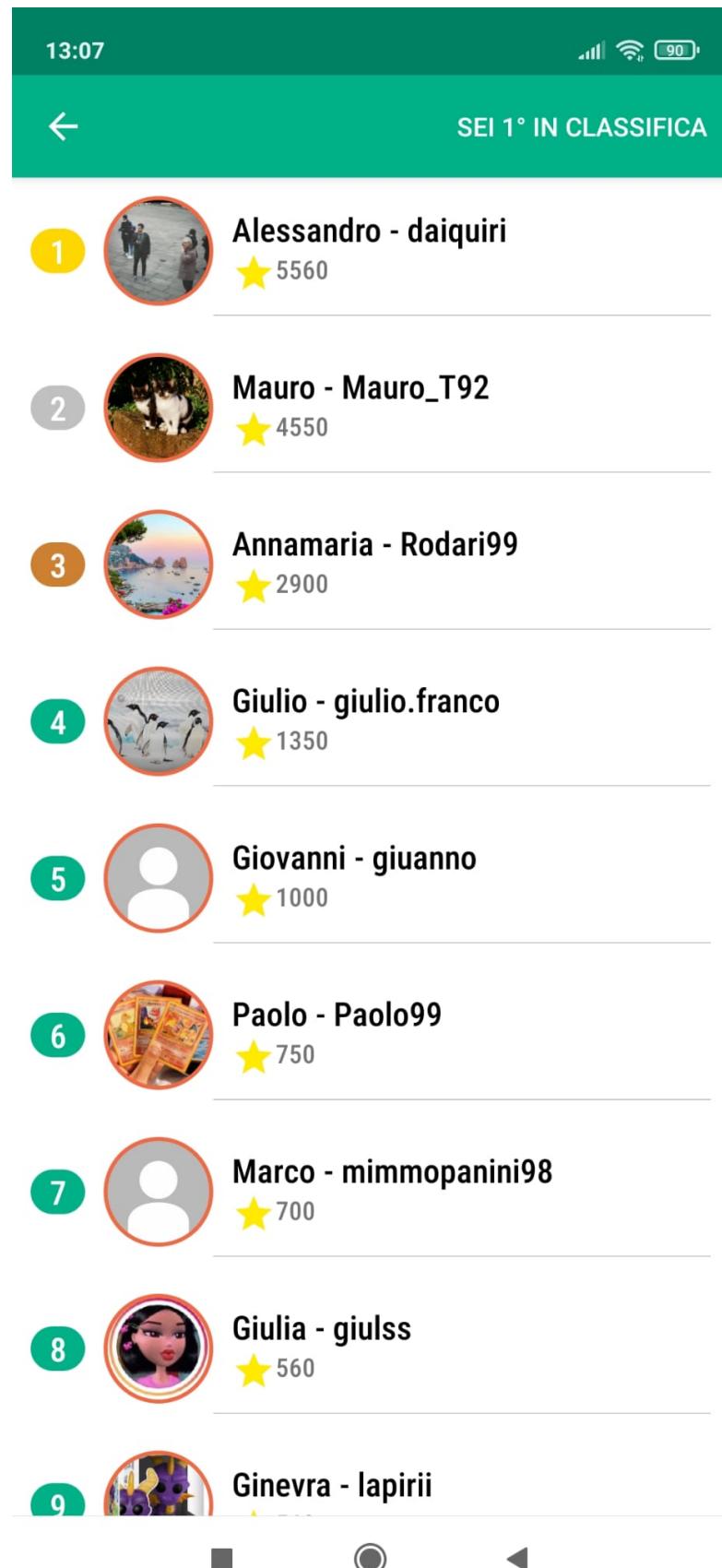


Figura 5.23: Mockup per la classifica

5.3 Sistemica

In questo paragrafo presentiamo gli statechart UML, che sono diagrammi molto utili per modellare sistemi di vario tipo, tra cui l'interazione tra uomo e macchina. Vengono tipicamente utilizzati per modellare il dialogo e riescono a descrivere con relativa semplicità una transizione di stati per giungere ad un qualche obiettivo. Non vanno però confusi con gli automi a stati finiti: gli stati di uno statechart, infatti, possono essere ricorsivi, differentemente dagli stati di un automa che sono astrazioni atomiche.

Sono presentati prima gli statechart di analisi, che hanno il compito di analizzare il problema da un punto di vista più astratto e generale e poi quelli di design, che derivano dai precedenti ma sono calati in un contesto algoritmico/tecnologico.

5.3.1 Di analisi

Crea Account

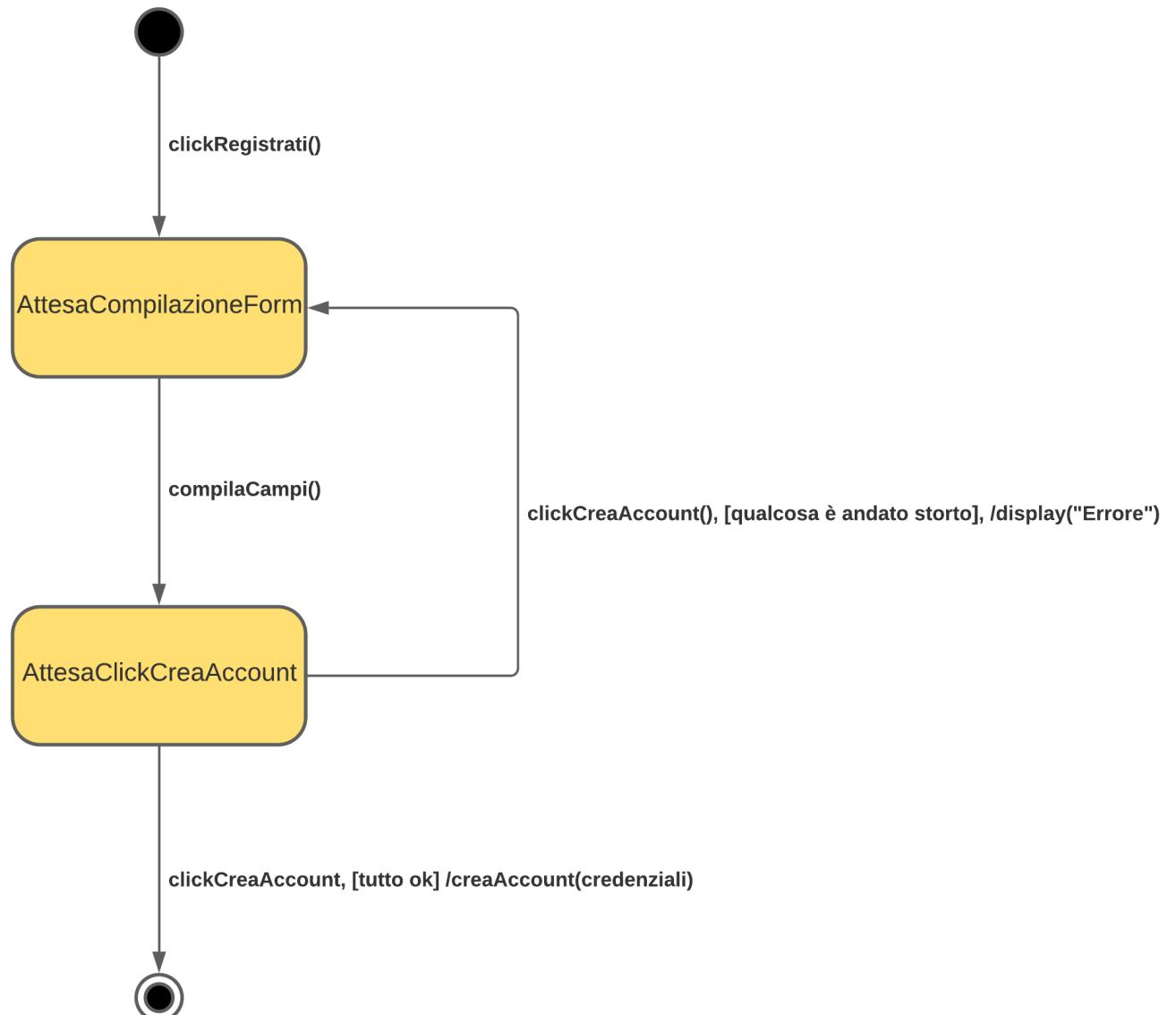


Figura 5.24: Statechart Diagram d'Analisi per il caso d'uso Crea Account

Login

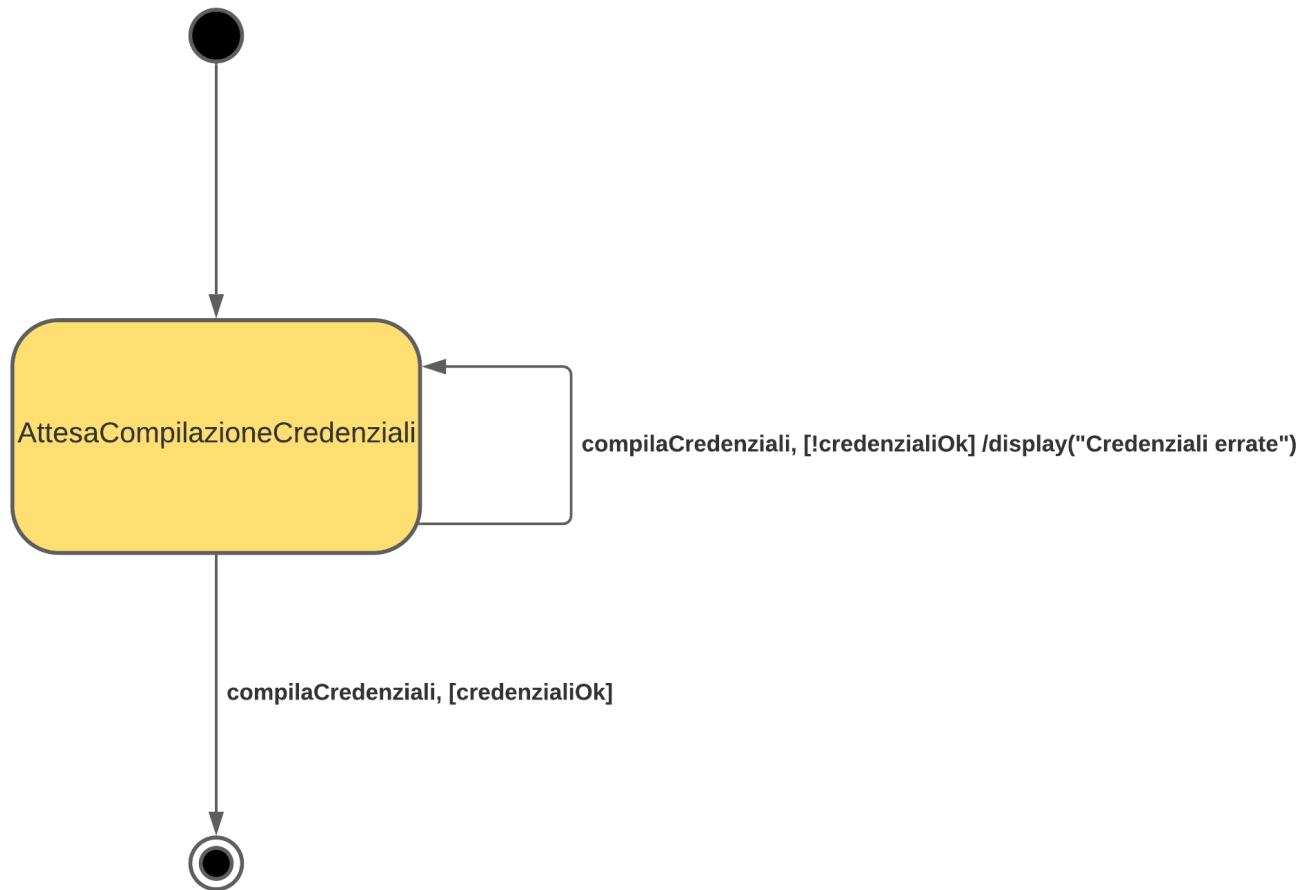


Figura 5.25: Statechart Diagram d'Analisi per il caso d'uso Login

Visualizza strutture

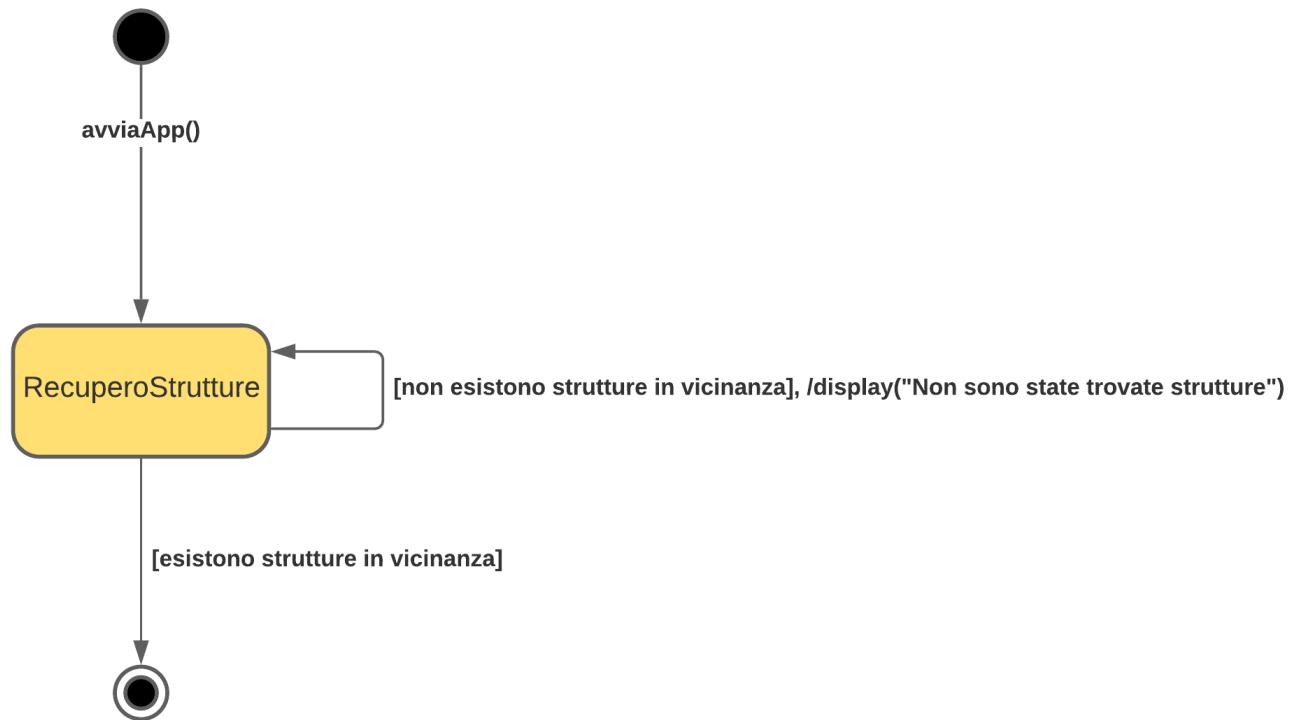


Figura 5.26: Statechart Diagram d'Analisi per il caso d'uso Visualizza Strutture

Visualizza strutture su mappa

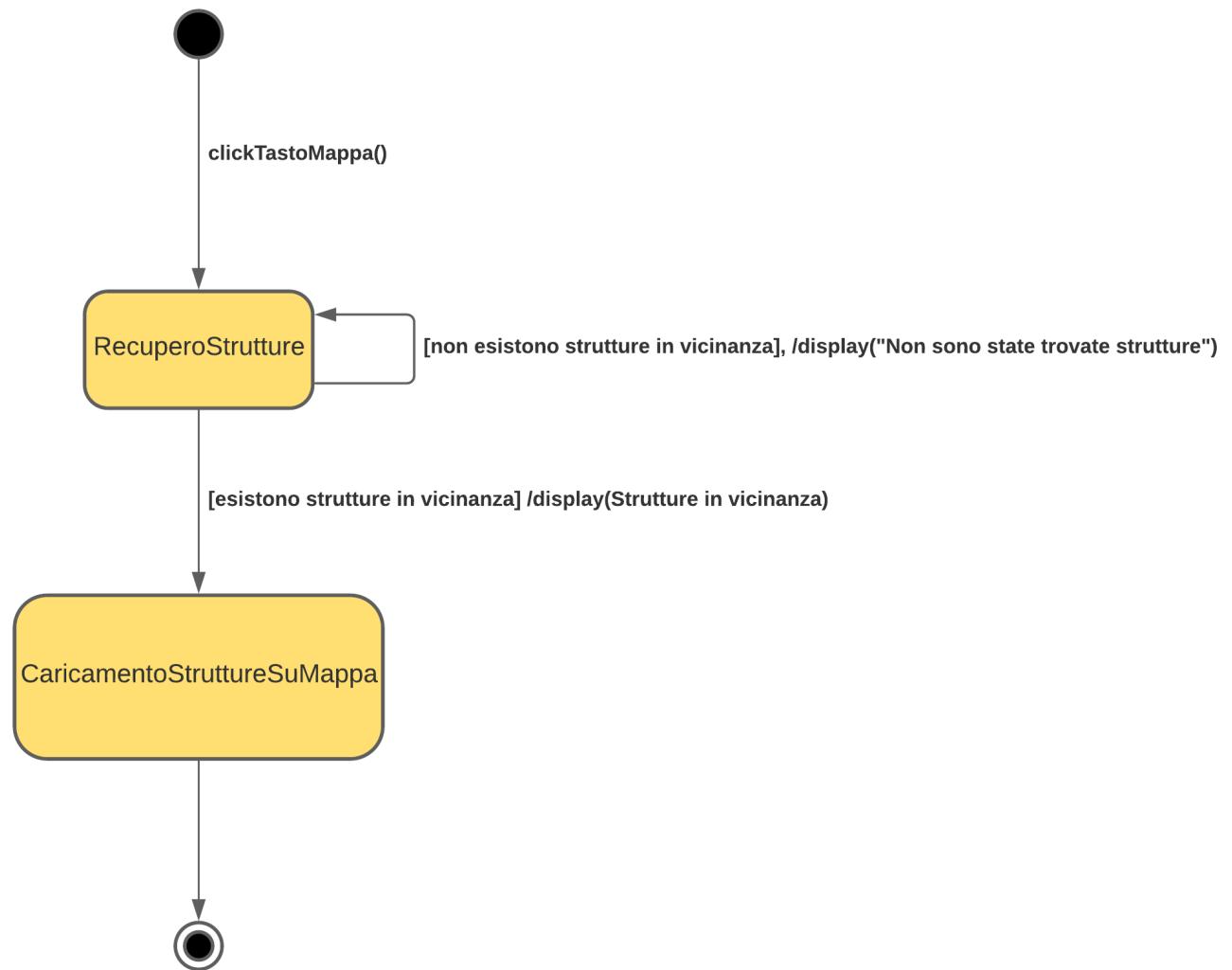


Figura 5.27: Statechart Diagram d'Analisi per il caso d'uso Visualizza Strutture su Mappa

Legge recensioni

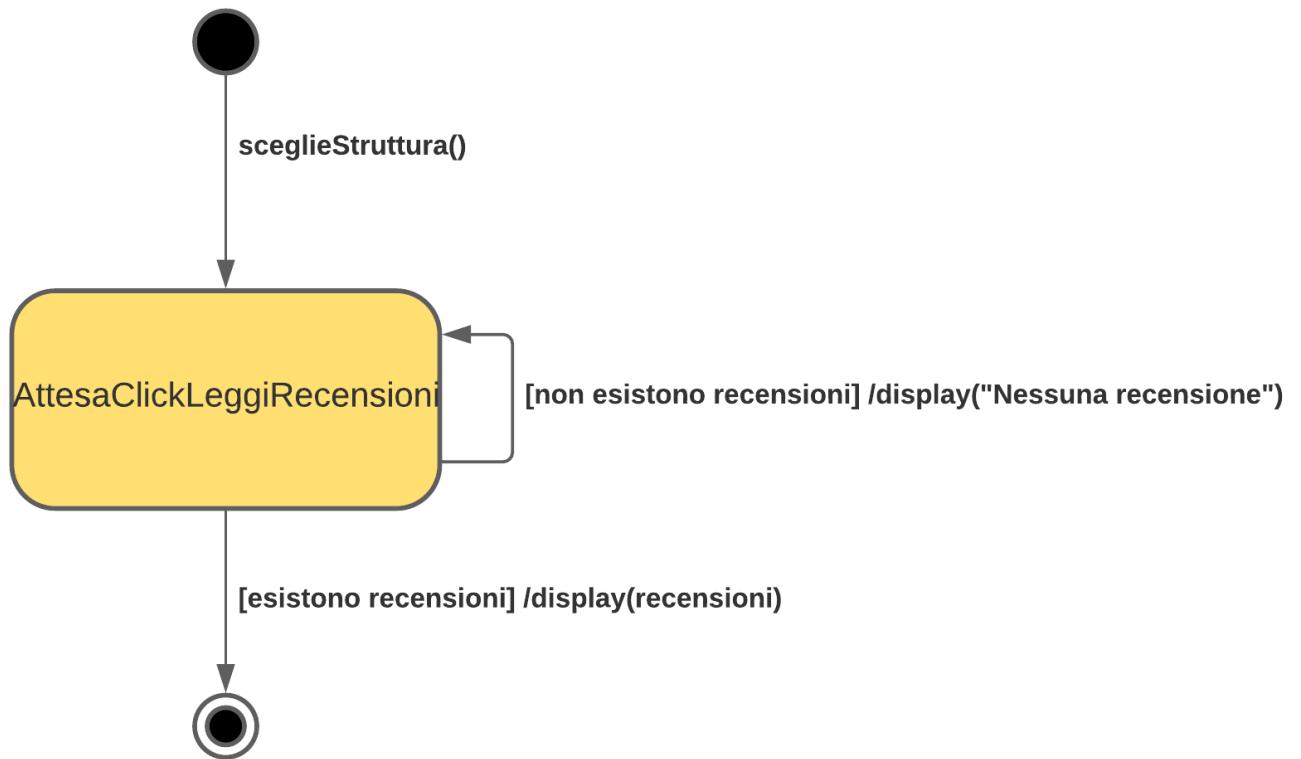


Figura 5.28: Statechart Diagram d'Analisi per il caso d'uso Legge Recensioni

Lascia recensione per acquisire punti o titoli

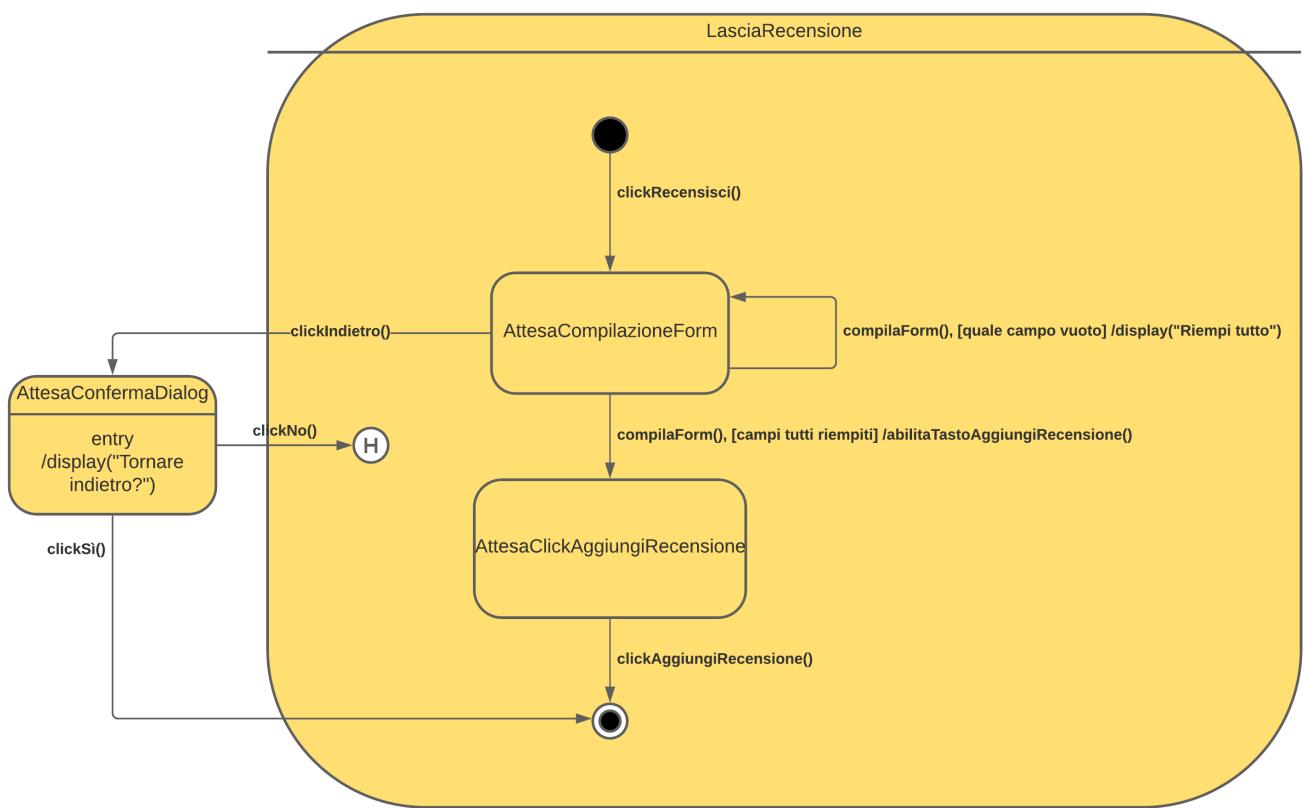


Figura 5.29: Statechart Diagram d'Analisi per il caso d'uso Lascia Recensione

5.3.2 Di design

Crea Account

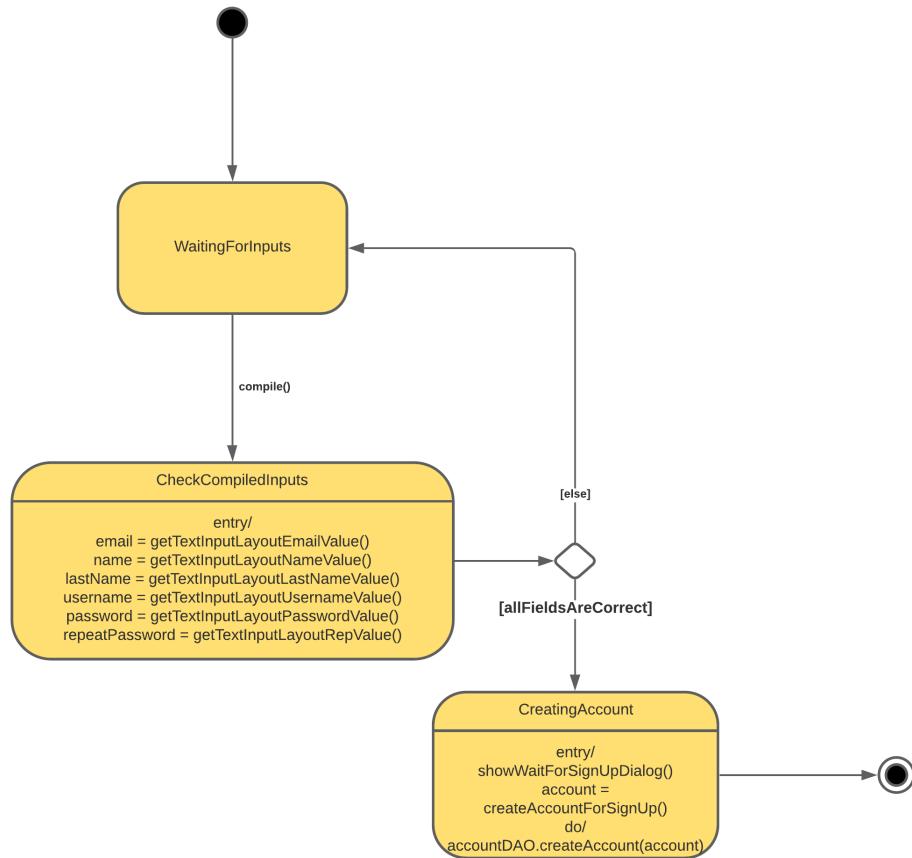


Figura 5.30: Mobile - Statechart Diagram di Design per il caso d'uso Crea Account

Login

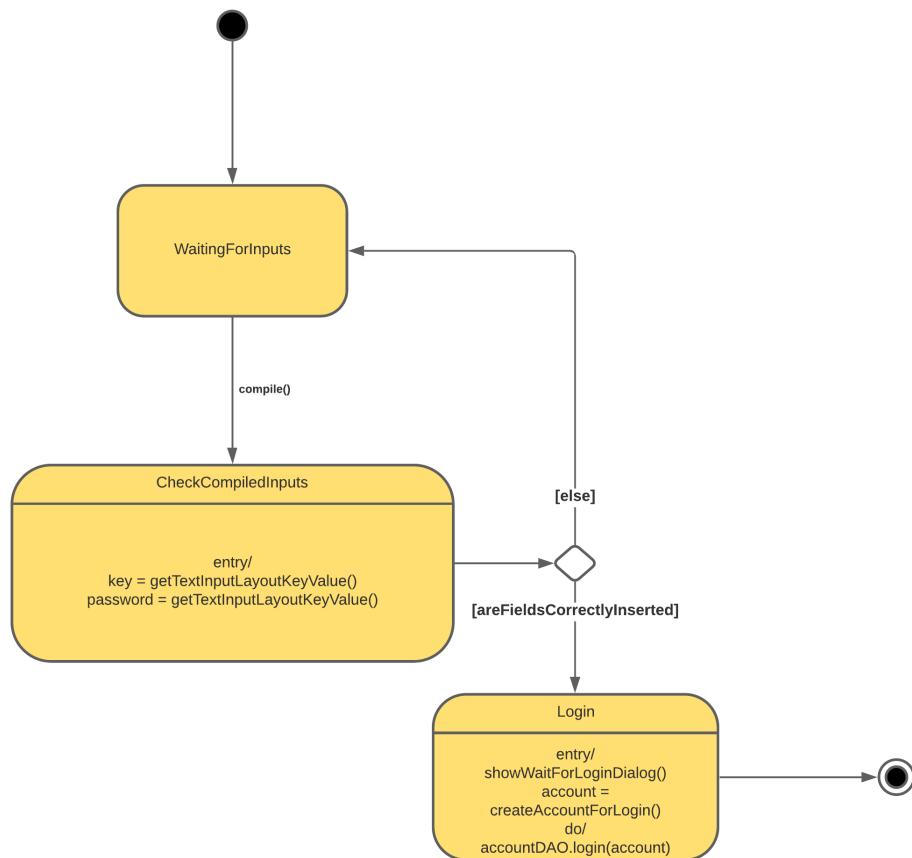


Figura 5.31: Mobile - Statechart Diagram di Design per il caso d'uso Login

Visualizza Strutture

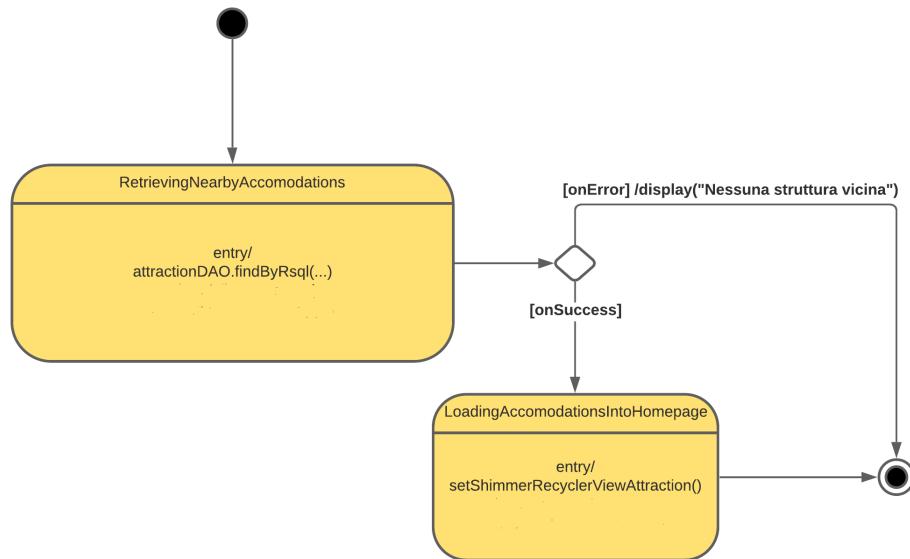


Figura 5.32: Mobile - Statechart Diagram di Design per il caso d'uso Visualizza Strutture

Visualizza Strutture su Mappa

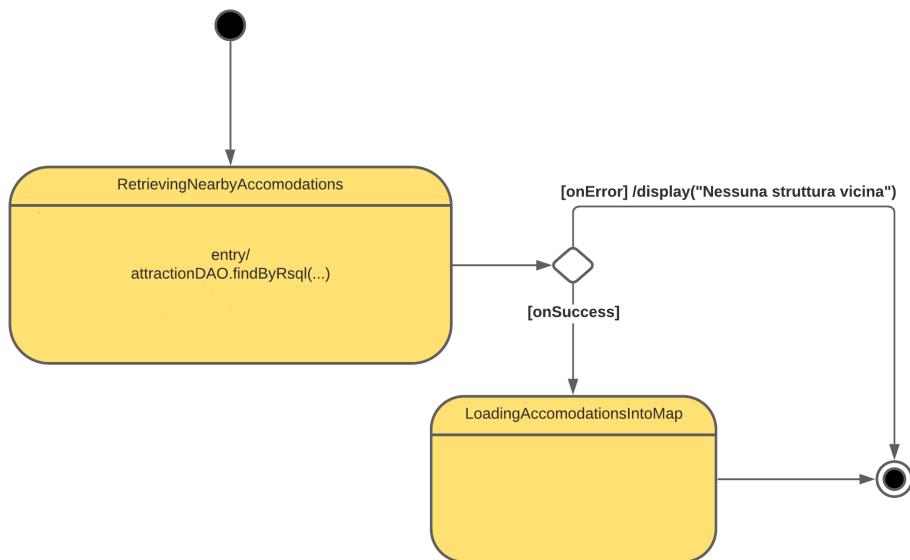


Figura 5.33: Mobile - Statechart Diagram di Design per il caso d'uso Visualizza Strutture su Mappa

Legge Recensioni

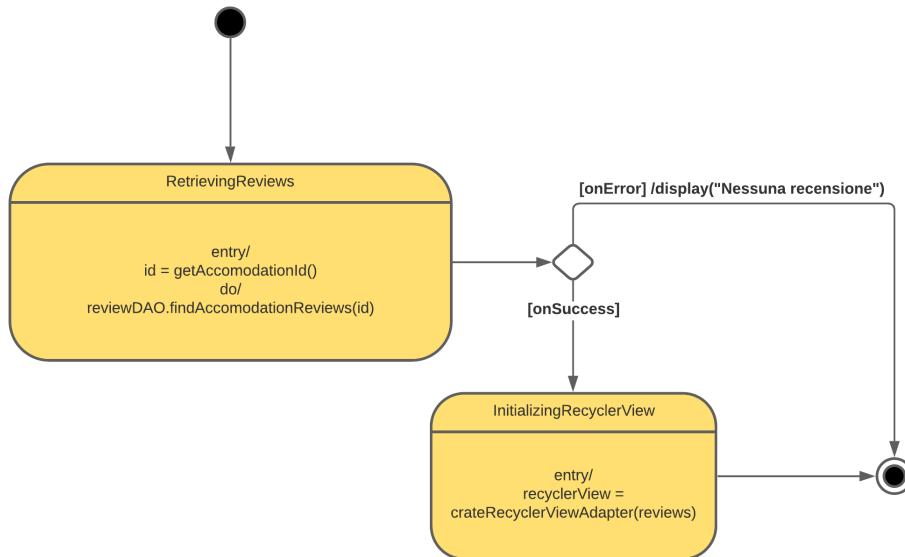


Figura 5.34: Mobile - Statechart Diagram di Design per il caso d'uso Legge Recensioni

Lascia recensione per acquisire punti o titoli

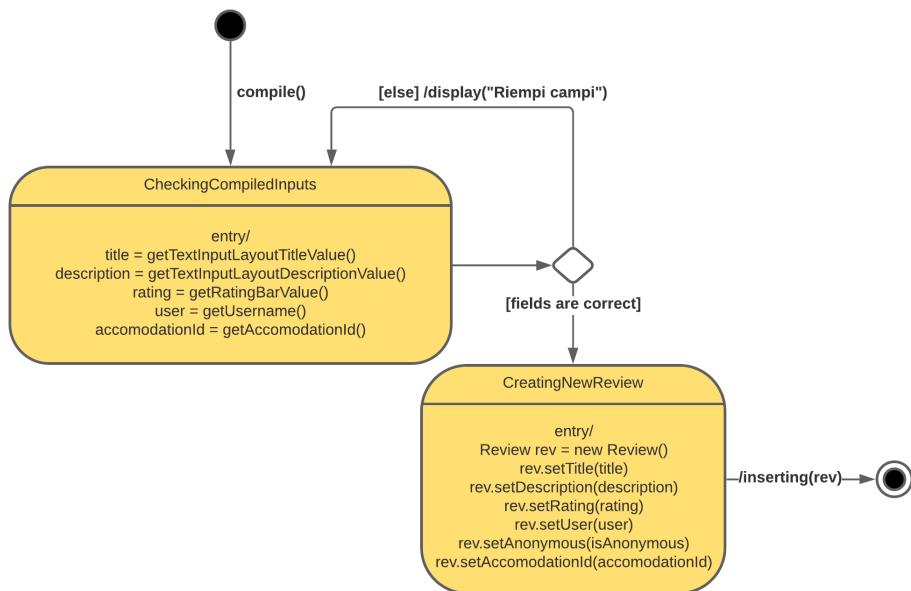


Figura 5.35: Mobile - Statechart Diagram di Design per il caso d'uso Lascia Recensione

5.4 Delle attività

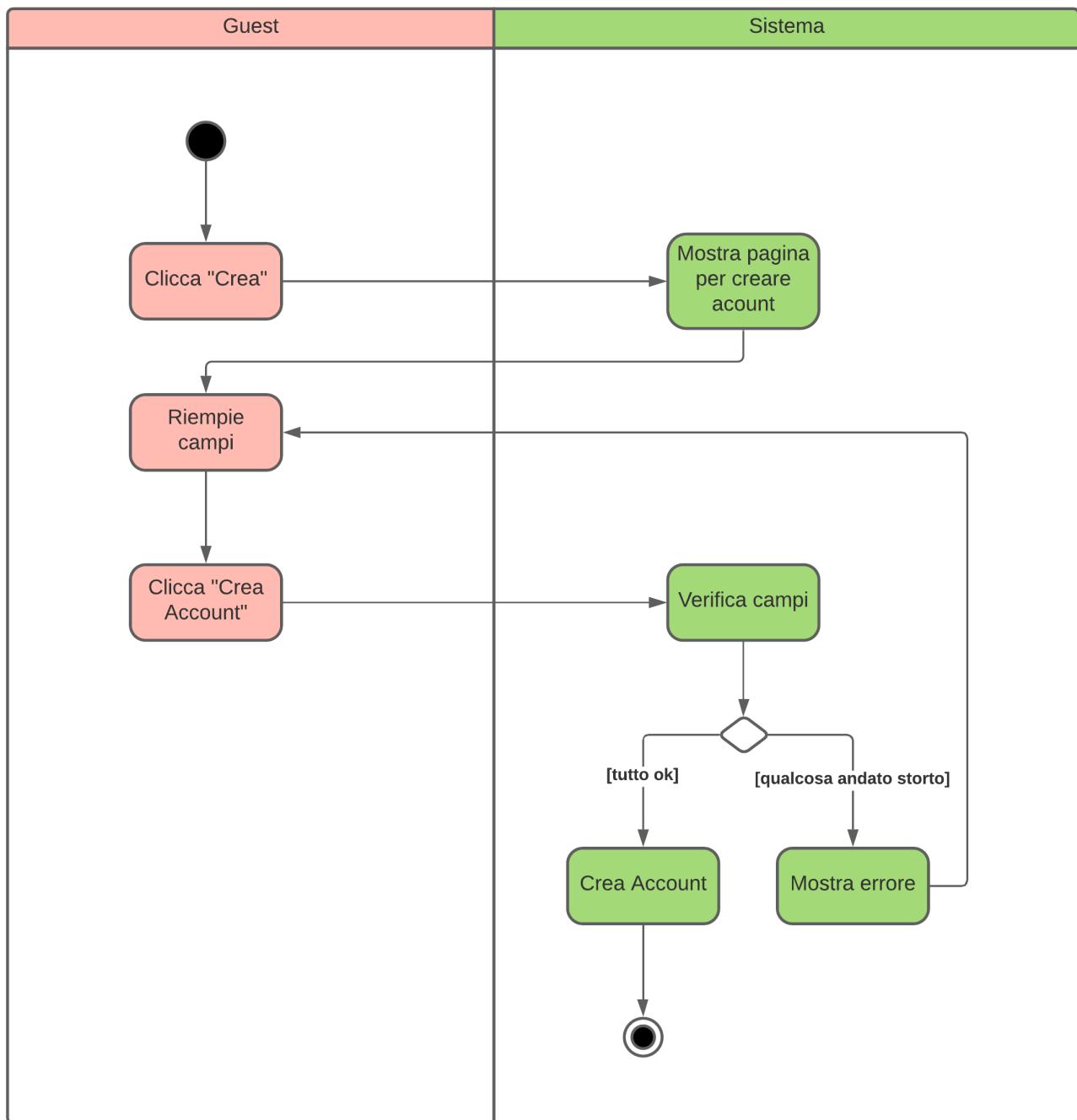


Figura 5.36: Mobile - Activity Diagram per il caso d'uso Crea Account

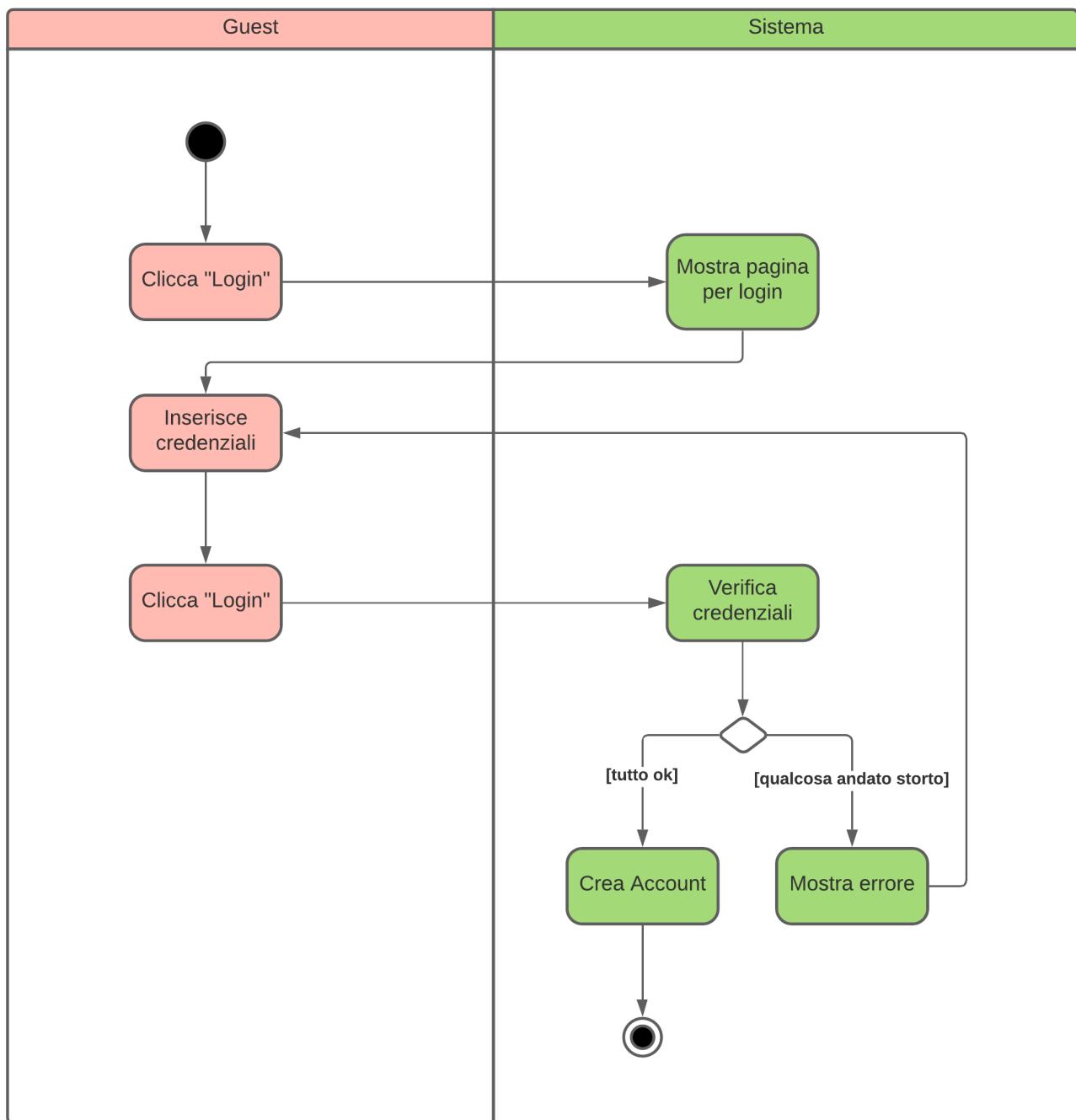


Figura 5.37: Mobile - Activity Diagram per il caso d'uso Login

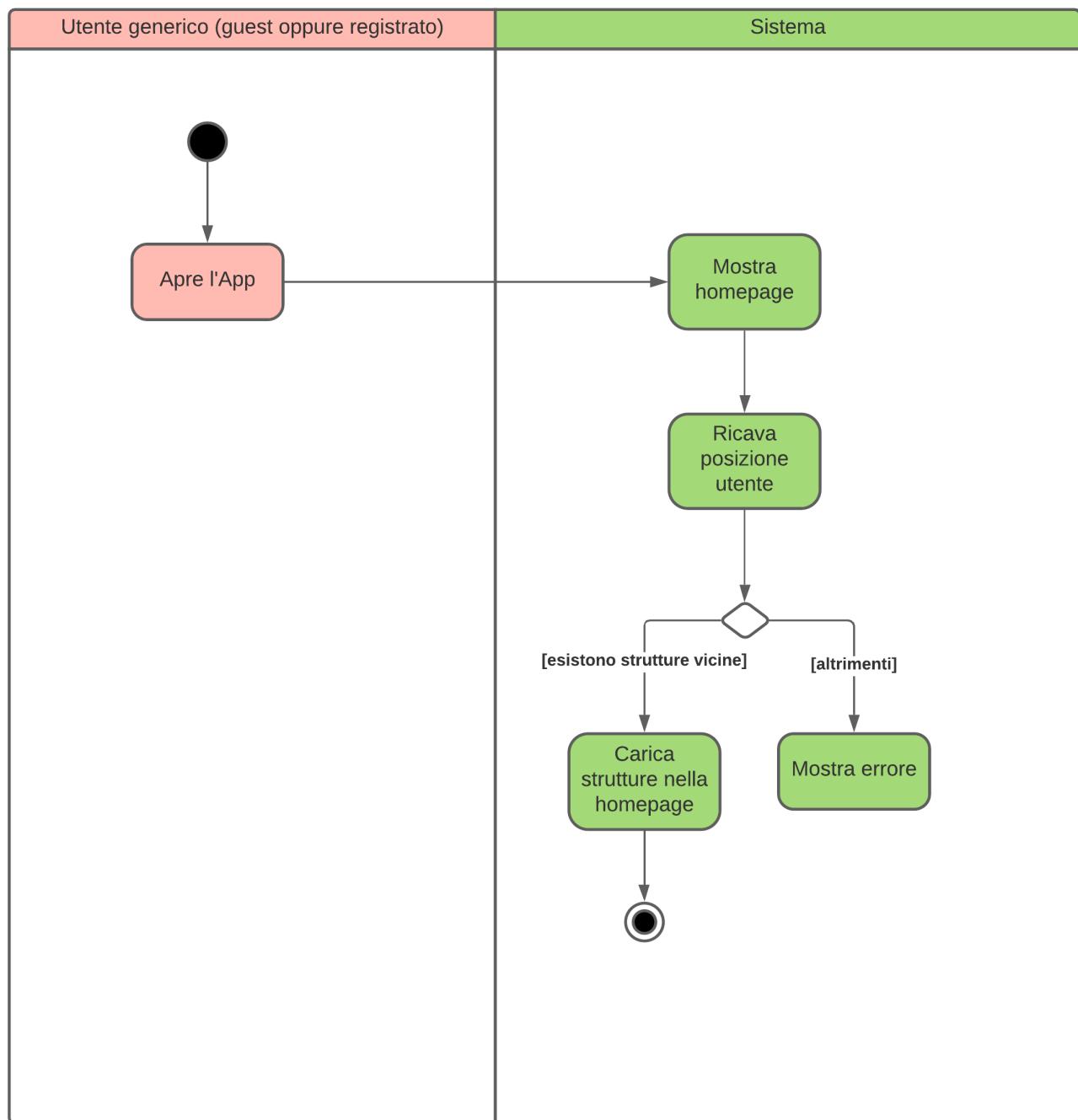


Figura 5.38: Mobile - Activity Diagram per il caso d'uso Visualizza Strutture

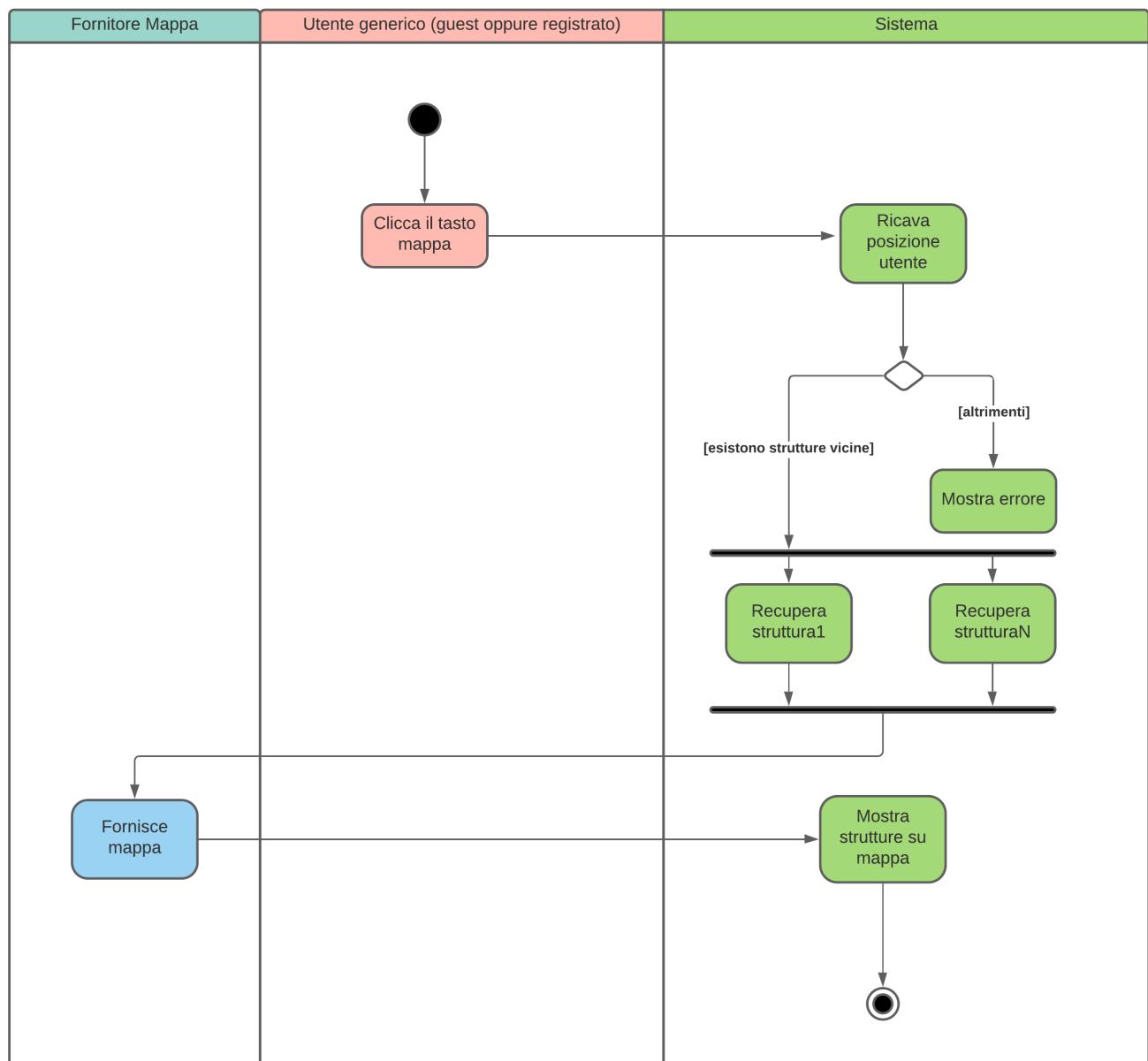


Figura 5.39: Mobile - Activity Diagram per il caso d'uso Visualizza Strutture su Mappa

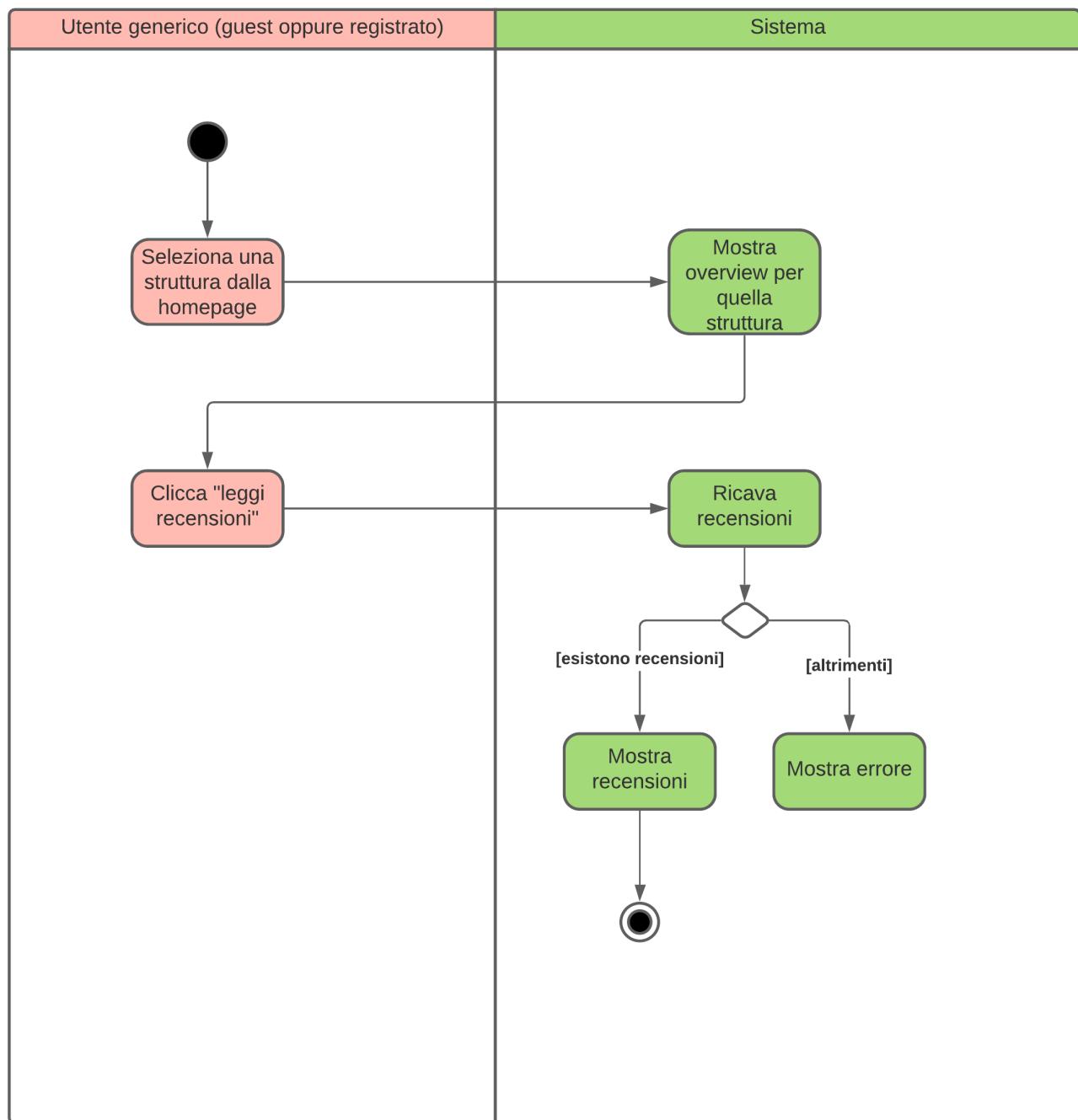


Figura 5.40: Mobile - Activity Diagram per il caso d'uso Legge Recensioni

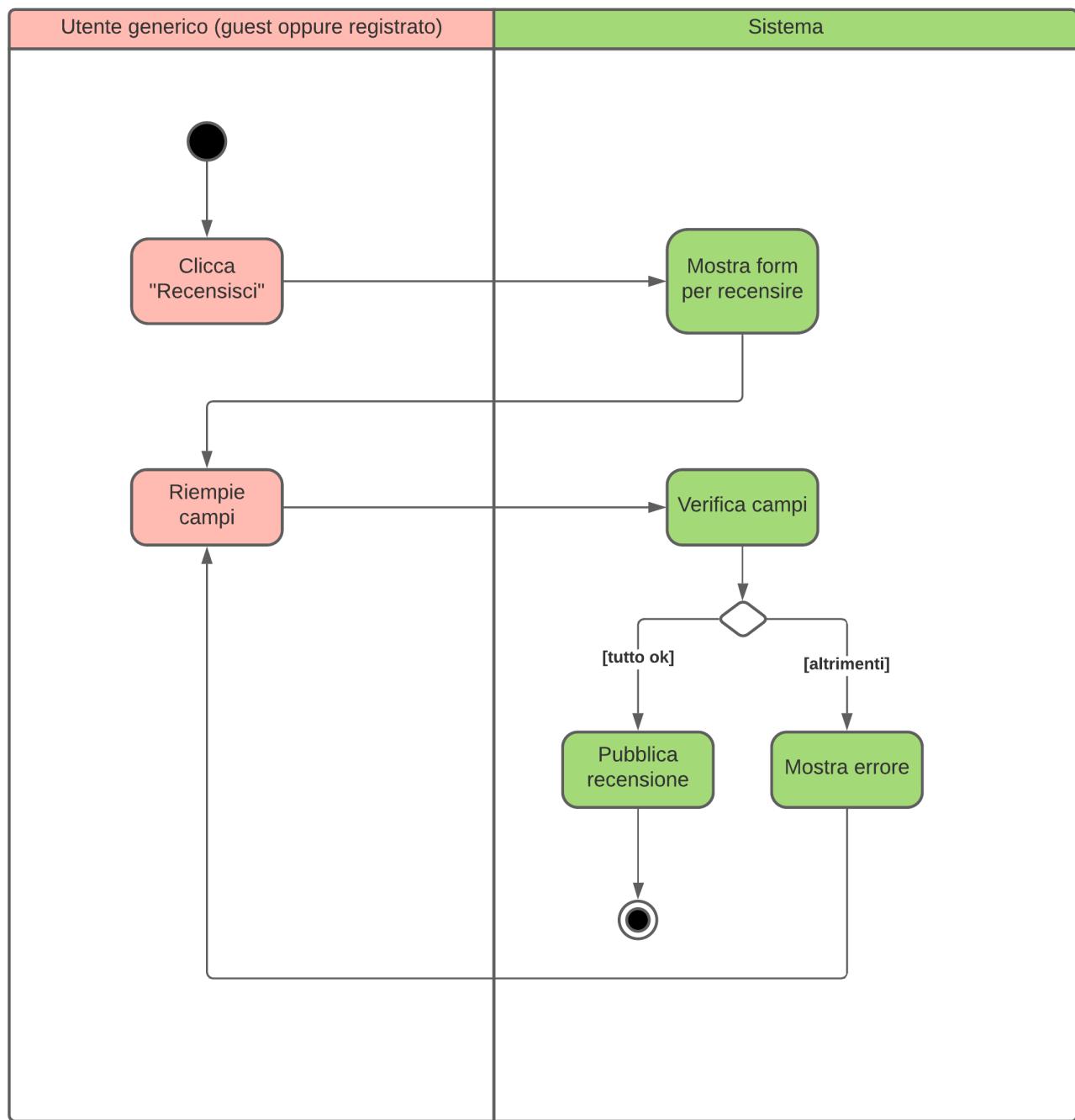


Figura 5.41: Mobile - Activity Diagram per il caso d'uso Lascia Recensione

Capitolo 6

Valutazione sul campo dell'usabilità soggettiva e oggettiva

In questo capitolo viene valutata l'usabilità del software in maniera soggettiva ed oggettiva attraverso specifiche tecniche scientifiche riconosciute nell'ambito dell'ingegneria del software e dell'interazione uomo-macchina. Non bisogna dimenticare, comunque, che la fase della valutazione dell'usabilità richiede tempo ed è un passo fondamentale per il progetto. Sono state seguite le best-practise introdotte da Schneiderman, conosciute con il nome delle "otto regole d'oro". Una schematizzazione di quanto compiuto è rappresentato nella Tabella 6.1 che verifica la coerenza del sistema rispetto a tali regole.

Tabella 6.1: Coerenza del sistema rispetto alle otto regole d'oro di Schneiderman

Obiettivo	Valutare se il sistema rispetta le otto regole d'oro di Schneiderman		
	Regola	Risultato atteso	Risultato ottenuto?
	Battersi per la coerenza	I termini dovrebbero essere coerenti in situazioni simili e familiari per l'utente	✓
	Consentire agli utenti abituali l'uso di scorciatoie	Talvolta alcune operazioni ricorrenti potrebbero diventare tediose se si è costretti a fare sempre le stesse operazioni	✓
	Offrire feedback informativi	Occorrerebbe fornire un feedback immediato da parte del sistema per ogni azione dell'utente, coerentemente con l'azione intrapresa	✓
	Progettare i dialoghi dando la precedenza alle forme di chiusura	Per una maggiore semplicità di lettura	✓

Continua alla pagina successiva

Tabella 6.1: Coerenza del sistema rispetto alle 8 regole d'oro di Schneiderman

	Offrire gestioni semplici di errori	Gli errori dovrebbero essere gestiti in maniera semplice e tale da non spaventare l'utente e il sistema dovrebbe fornire un meccanismo semplice per informare l'utente e gestire l'errore stesso	✓
	Permettere azioni reversibili	In questo modo l'utente capisce che una qualsiasi azione può essere reversibile e sarà quindi spinto a provare funzioni ad egli ancora sconosciute	✓
	Supportare il locus of control	L'utente dovrebbe essere sempre in grado di conoscere cosa sta succedendo al sistema in ogni istante di tempo	✓
	Riduci il carico di memoria a breve termine e i movimenti del corpo	Non dovrebbe essere richiesto all'utente di impiegare la propria memoria a breve termine né di compiere grossi spostamenti col dito sul display per compiere azioni quotidiane	✓
Notes	Relative al design delle interfacce		

6.1 Valutazioni euristiche

Oltre a verificare la coerenza del sistema rispetto alle regole introdotte da Shneiderman, occorre valutare l'usabilità anche per mezzo delle euristiche di Nielsen e per mezzo di test di usabilità. Attraverso le euristiche di Nielsen è stato possibile valutare l'usabilità soggettiva del sistema. Esse infatti sono regole generali e dipendenti dall'esperienza e dall'abilità del singolo utente valutatore.

Tabella 6.3: Valutazione mediante euristica di Nielsen

Obiettivo	Valutare l'usabilità tramite euristiche di Nielsen		
	Euristica	Risultato atteso	Risultato ottenuto?
	Visibilità dello stato del sistema	Impiego di feedback in tempi ragionevoli per informare l'utente	✓
	Corrispondenza tra il mondo reale e il sistema	Il sistema dovrebbe parlare la lingua dell'utente, in termini di simboli, frasi e concetti familiari	✓
	Libertà e controllo da parte dell'utente	Fornire all'utente dei pulsanti di Undo e Redo	✓
	Consistenza e standard	Seguire lo standard	✓
	Prevenzione degli errori	Impiegare metodologie di prevenzione di errore come white-list e chiedere conferma prima che un'azione critica venga eseguita	✓
	Riconoscere e non ricordare	Minimizzare l'uso della memoria dell'utente	✓
	Flessibilità ed efficienza	Metodologie di accelerazione per procedure ricorrenti	✓
	Design minimalista	Messaggi brevi e interfacce pulite	✓
	Gestione degli errori in maniera semplice	I messaggi di errore dovrebbero essere espressi in un linguaggio naturale	✓
	Supporto	Possibilità dell'utente di avere supporto	✓
Notes	Valutazioni soggettive e da complementare con i test di usabilità		

Con questo tipo di valutazione è possibile ottenere buoni risultati a costo zero impiegando molti valutatori per lo stesso sistema e che non comunichino tra loro, cercando quindi di avere valutazioni indipendenti le une dalle altre. Per questo motivo la valutazione mediante euristica di Nielsen deve

necessariamente essere accompagnata da una valutazione più oggettiva attraverso i test di usabilità. Essi, infatti, sono complementari e andrebbero effettuati insieme durante questa fase del progetto. La relazione che intercorre tra le valutazioni euristiche di Nielsen ed i test di usabilità è mostrata in Figura 6.1.



Figura 6.1: Confronto tra valutazione euristica di Nielsen e test di usabilità

6.2 Test di usabilità

Come detto, occorre effettuare anche un test di usabilità. Abbiamo scelto un gruppo di utenti target del sistema (formalmente, un sottoinsieme di C della Figura 2.3) e abbiamo chiesto loro di svolgere separatamente gli stessi compiti i quali erano scritti su carta per non influenzare ciascun valutatore con il proprio tono di voce. Oltre ai singoli utenti che facevano da cavie, sono state coinvolti anche un facilitatore per dirigere e gestire la prova e due osservatori che registravano attraverso una videocamera i comportamenti dell'utente, come espressioni facciali, movimenti, prossemica e annotavano comportamenti rilevanti. A questo punto della fase di progettazione, non sono stati impiegati i prototipi, che erano stati usati nella fase di valutazione del sistema a priori, ma l'applicazione vera e propria e finita. È stato chiesto tassativamente all'utente di pensare ad alta voce mentre gli era chiesto di svolgere qualche compito (metodologia think aloud). La registrazione e l'annotazione su carta degli osservatori circa le espressioni facciali dell'utente sono servite per effettuare un'analisi a posteriori poiché non è fattibile analizzare tali comportamenti in tempo reale poiché non si avrebbe visibilità su quale sia effettivamente il problema che l'utente sta riscontrando in quel momento. Una volta che la registrazione è terminata, il team esperto visualizzerà gli appunti e le registrazioni e, qualora siano stati riscontrati problemi da qualche parte, si re-itererà per raffinamenti successivi e l'eliminazione di tali punti critici. Per quanto riguarda la registrazione audio/video sono state impiegate due videocamere: una propria dello smartphone su cui si eseguiva l'applicazione (in par-

ticolare la fotocamera frontale), una webcam che riprendeva il corpo dell'utente (per esaminare la prossemica) ed un'altra webcam che riprendeva sia l'utente che il sistema. A causa delle restrizioni Covid, i test di usabilità sono stati effettuati presso il domicilio di ogni utente che doveva valutare l'applicazione, in un contesto informale poiché impossibilitati a recarci presso un laboratorio di usabilità professionale.

Rispetto alle attività condotte dagli utenti durante le prove, i test di usabilità in questa fase sono classificabili in test di compito e test di scenario. Nel primo caso l'utente svolge un singolo task che permette di specificare una funzione specifica del sistema. Nel secondo caso, invece, all'utente viene indicato un obiettivo da raggiungere attraverso una serie di compiti elementari, senza però indicarli in maniera esplicita. Per test di scenario più complessi, è possibile costruire una "storia" al contorno. Entrambi i compiti, tuttavia, sono stati dati ad ogni utente per iscritto ed in modo chiaro in modo tale che ogni valutatore si troverà nella medesime condizioni: alcune spiegazioni date a voce possono essere alterate dal tono di voce o da altri fattori che inevitabilmente influenzano la psicologia di chi ascolta. Naturalmente oltre ad una tale misura qualitativa è opportuno calcolare il tasso di successo di ciascun compito, ovvero la percentuale dei compiti portati al termine.

6.2.1 Test di compito

Ciascuno dei compiti riportati nelle seguenti tabelle, per ogni utente valutatore dell'usabilità, ha riportato un tasso di successo pari a 1. In altre parole, tutti gli utenti sono stati in grado in maniera completa e non parziale a svolgere il compito assegnato. Formalmente, su una taglia di 10 persone

$$sr = \frac{\#successi}{\#valutatori} = \frac{10}{10} = 1$$

portando quindi la probabilità di successo, in percentuale, al 100%.

Tabella 6.5: Compito - Crea Account

Task ID	1		
Task Name	Crea Account		
Task Description	Verifica la funzionalità Mobile <i>Crea Account</i>		
	Input	Risultato atteso	Risultato ottenuto
	Riempie i campi con dati validi	Registrazione effettuata e caricamento della schermata successiva	Superato
	Lascia uno dei campi vuoti	Login fallito e visualizzazione di un messaggio di errore	Superato
	Riempie i dati ma inserendo un pattern email non valido	Login fallito e visualizzazione del messaggio "Pattern email non valido"	Superato
	Riempie i dati ma password e ripeti password non coincidono	Login fallito e visualizzazione del messaggio di errore "Le password non coincidono"	Superato
	Riempie i dati già sono occupate da un altro utente	Login fallito e visualizzazione del messaggio di errore	Superato
Notes			

Tabella 6.7: Compito - Login

Task ID	2		
Task Name	Login		
Task Description	Verifica la funzionalità Mobile <i>Login</i>		
Input	Risultato atteso	Risultato ottenuto	
	Riempie i campi con dati validi	Login effettuato e caricamento della schermata successiva	Superato
	Lascia il campo email vuoto	Login fallito e visualizzazione di un messaggio di errore	Superato
	Lascia il campo password vuoto	Login fallito e visualizzazione di un messaggio di errore	Superato
	Lascia entrambi i campi vuoti	Login fallito e visualizzazione di un messaggio di errore	Superato
Notes	Riempie i dati con credenziali errate		
	Per compiere il login, l'utente mobile deve già avere un account verificato		

Tabella 6.9: Compito - Visualizza Strutture

Task ID	3		
Task Name	Visualizza Strutture		
Task Description	Verifica la funzionalità Mobile <i>Visualizza Strutture</i>		
	Input	Risultato atteso	Risultato ottenuto
	Apre l'app e vi sono strutture vicine	Visualizzazione delle strutture effettuata	Superato
	Apre l'app ma non vi sono strutture vicine	Visualizzazione delle strutture fallita e comparsa messaggio di errore	Superato
	Seleziona una delle voci dalla homepage e vi sono strutture vicine	Visualizzazione delle strutture sottoforma di lista	Superato
	Seleziona una delle voci dalla homepage ma non vi sono strutture vicine	Visualizzazione delle strutture fallita e comparsa messaggio di errore	Superato
	Clicca la lente di ingrandimento per filtrare le strutture vicine	Visualizzazione delle strutture corrispondenti effettuata	Superato
Notes	L'utente deve aver concesso i permessi per la geolocalizzazione		

Tabella 6.11: Compito - Visualizza Strutture su Mappa

Task ID	4		
Task Name	Visualizza Strutture su Mappa		
Task Description	Verifica la funzionalità Mobile <i>Visualizza Strutture su Mappa</i>		
	Input	Risultato atteso	Risultato ottenuto
	Clicca il pulsante a forma di mappa dalla lista di strutture vicine	Visualizzazione delle strutture su mappa effettuata	Superato
	Clicca il pulsante a forma di mappa dalla lista di strutture ma non ve ne sono di vicine	Visualizzazione delle strutture fallita e comparsa messaggio di errore	Superato
	Clicca il pulsante a forma di lente di ingrandimento dalla mappa	Visualizzazione delle strutture corrispondenti su mappa effettuata	Superato
	Clicca il pulsante a forma di lente di ingrandimento dalla mappa ma non vi sono strutture	Visualizzazione delle strutture corrispondenti su mappa fallita e comparsa messaggio di errore	Superato
Notes	L'utente deve aver concesso i permessi per la geolocalizzazione		

Tabella 6.13: Compito - Legge Recensioni

Task ID	5		
Task Name	Legge Recensioni		
Task Description	Verifica la funzionalità Mobile <i>Legge Recensioni</i>		
	Input	Risultato atteso	Risultato ottenuto
	Seleziona una struttura dalla homepage e clicca "Legge Recensioni"	Visualizzazione delle recensioni effettuata	Superato
	Seleziona una struttura dalla homepage e clicca "Legge Recensioni" ma non vi sono recensioni	Visualizzazione delle recensioni fallita e comparsa messaggio d'errore	Superato
Notes			

Tabella 6.15: Compito - Lascia Recensione (e acquisisce punti/titoli)

Task ID	7		
Task Name	Lascia Recensione (e acquisisce punti/titoli)		
Task Description	Verifica la funzionalità Mobile <i>Lascia Recensione</i>		
	Input	Risultato atteso	Risultato ottenuto
	Seleziona una struttura poi clicca "Recensisci" (o il pulsante a forma di matita dalla overview) e riempie il form	Scrittura recensione effettuata	Superato
	Seleziona una struttura poi clicca "Recensisci" (o il pulsante a forma di matita dalla overview) e riempie il form ma lasciando qualche campo vuoto	Visualizzazione delle recensioni fallita e comparsa messaggio d'errore	Superato
	Seleziona una struttura poi clicca "Recensisci" (o il pulsante a forma di matita dalla overview) e riempie il form ma eccedendo ai limiti di caratteri nel titolo	Visualizzazione delle recensioni fallita e comparsa messaggio d'errore	Superato
	Seleziona una struttura poi clicca "Recensisci" (o il pulsante a forma di matita dalla overview) e riempie il form ma eccedendo ai limiti di caratteri nella descrizione	Visualizzazione delle recensioni fallita e comparsa messaggio d'errore	Superato
	Seleziona una struttura poi clicca "Recensisci" (o il pulsante a forma di matita dalla overview) e riempie il form ma eccedendo ai limiti di caratteri nel titolo e nella descrizione	Visualizzazione delle recensioni fallita e comparsa messaggio d'errore	Superato
Notes	Per lasciare una recensione, occorre avere un account ed essere autenticato		

CAPITOLO 6. VALUTAZIONE SUL CAMPO DELL'USABILITÀ SOGGETTIVA E OGGETTIVA 86

Ciascuna delle precedenti tabelle di compiti può essere sintetizzata nella seguente tabella:

	Compito 1	Compito 2	Compito 3	Compito 4	Compito 5	Compito 6
Valutatore 1	S	S	S	S	S	S
Valutatore 2	S	S	S	S	S	S
Valutatore 3	S	S	S	S	S	S
Valutatore 4	S	S	S	S	S	S
Valutatore 5	S	S	S	S	S	S
Valutatore 6	S	S	S	S	S	S
Valutatore 7	S	S	S	S	S	S
Valutatore 8	S	S	S	S	S	S
Valutatore 9	S	S	S	S	S	S
Valutatore 10	S	S	S	S	S	S

6.2.2 Test di scenario

Durante la valutazione dell'usabilità per mezzo di test di scenario sono stati ipotizzati i seguenti scenari:

1. Cambia la tua immagine di profilo;
2. Guarda chi è il primo classificato nella leaderboard;
3. Cerca il profilo di Alessandro Quirile e Mauro Telese e valuta chi dei due ha scritto più recensioni.
4. Cerca le strutture entro 30km che abbiano almeno 4 stelle;
5. Scrivi una recensione per un'attrazione a piacere.

Anche in questo caso tutti gli utenti scelti come valutatori hanno mostrato un tasso di successo pari a 1, come sintetizzato dalla seguente tabella:

	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5
Valutatore 1	S	S	S	S	S
Valutatore 2	S	S	S	S	S
Valutatore 3	S	S	S	S	S
Valutatore 4	S	S	S	S	S
Valutatore 5	S	S	S	S	S
Valutatore 6	S	S	S	S	S
Valutatore 7	S	S	S	S	S
Valutatore 8	S	S	S	S	S
Valutatore 9	S	S	S	S	S
Valutatore 10	S	S	S	S	S

6.3 5-seconds test

Un altro test che è stato compiuto a prodotto finito è il cosiddetto "5-seconds test". Il 5-seconds test è una tecnica utilizzata nel testing di usabilità ed ha come obiettivo quello di catturare la prima impressione che un gruppo più o meno ampio di valutatori ha vedendo ciascuna interfaccia per soli cinque secondi. Allo scadere dei cinque secondi, al valutatore viene sottratta l'interfaccia e gli viene chiesto se ricorda, in maniera anche vaga, cosa ci fosse su quell'interfaccia o, per meglio dire, a cosa servisse quell'interfaccia e cosa rappresentasse.

L'applicazione ha superato con successo il 5-seconds testing per ciascuna interfaccia principale e per ogni valutatore, come sintetizzato dalla seguente Tabella, dove 1 rappresenta l'homepage, 2 l'overview di un'attrazione, 3 la schermata per scrivere una recensione, 4 la mappa, 5 la pagina di profilo, 6 la classifica, 7 la pagina di registrazione, 8 quella di login e 9 quella per applicare filtri di ricerca.

6.4 Analisi di coerenza delle icone

È importante scegliere le icone che siano più evocative per l'utente. Ad esempio se si dovesse progettare un'icona per la funzione "Cerca" non avrebbe senso utilizzare un'icona a forma di Wi-Fi, ma occorrerebbe progettare un'icona che abbia come semantica quella di ricercare/trovare. La Tabella 6.17 mostra una valutazione della coerenza delle icone effettivamente presenti nell'applicazione rispetto al loro significato usuale o standard.

Tabella 6.17: Analisi di coerenza delle icone

Obiettivo	Valutare la coerenza delle icone presenti nell'applicazione rispetto al loro significato usuale		
	Icona	Significato usuale/standard	Significato effettivo nell'app
	Lente di ingrandimento	Cercare qualcosa	Come da standard
	Omino	Pagina personale/pagina profilo	Come da standard
	Freccia indietro	Tornare alla schermata precedente	Come da standard
	Omino con lente di ingrandimento	Cercare un altro utente	Come da standard
	Matita	Scrivere o modificare qualcosa	Come da standard
	Classifica	Mostrare una classifica in base ad un qualche criterio	Come da standard
	Porta con freccia verso l'esterno	Logout	Come da standard
	Mappa	Aprire una mappa o qualche informazione geografica	Come da standard
	Pollice in su	Approvazione	Come da standard
	Pollice in giù	Disapprovazione	Come da standard
	Orologio	Fornisce informazioni temporali	Come da standard
	Telefono	Fornisce informazioni telefoniche	Come da standard
	Occhio	Mostrare/oscurare la propria password in un form	Come da standard
	Bandierina	Segnalare	Come da standard
	\$	Prezzo o tariffari	Come da standard

6.5 Analisi di coerenza dei colori

Lo studio dei colori non è banale: occorrerebbe trovare la miglior combinazioni di colori in base ad alcuni criteri più o meno complessi come l'obiettivo del prodotto, la finalità ed i suoi mezzi. In Tabella 6.19 è valutata la coerenza dei colori utilizzati nell'app con il loro significato psicologico. Come si legge, il verde rappresenta la natura, l'ambiente ecologico, la gioventù ed è un colore equilibrante poiché neutro nello spettro visivo: in altre parole non è né eccessivamente stimolante come la maggior parte dei colori forti (ad esempio il rosso) né troppo poco stimolante, come la maggior parte dei colori freddi (bianco, grigio e così via). Per questo motivo il verde è stato utilizzato come colore principale dell'applicazione. Inoltre, per le funzionalità attorno a cui ruotano le meccaniche dell'applicazione - come la pubblicazione di una recensione - è stato utilizzato un colore stimolante, che infonde ottimismo e creatività come l'arancione. Il blu, inoltre, è stato utilizzato in combinazione con l'arancione per contrastare e creare una sinergia, mentre il bianco è stato utilizzato come colore accessorio per aumentare il senso di luminosità delle interfacce e far esaltare i colori.

Tabella 6.19: Analisi di coerenza dei colori

Obiettivo	Valutare la coerenza dei colori utilizzati nell'applicazione rispetto al loro significato psicologico		
	Colore	Significato psicologico	Impiego effettivo nell'app
	Verde	Natura, ambiente ecologico, generatività, benessere, fiducia, pace, respiro, interiorità, vita, vegetazione, gioventù, equilibrante, neutro nello spettro visivo	Colore principale dell'applicazione
	Arancione	Stimolante, energizzante, ottimismo, creatività	Utilizzato per i pulsanti "recensisci"
	Blu	Calmante, compromesso, in sinergia con l'arancione	Utilizzato per il pulsante "mappa"
	Bianco	Pulizia, luminosità	Colore accessorio

La Figura 6.2 rappresenta la paletta dei colori utilizzata nell'applicazione.

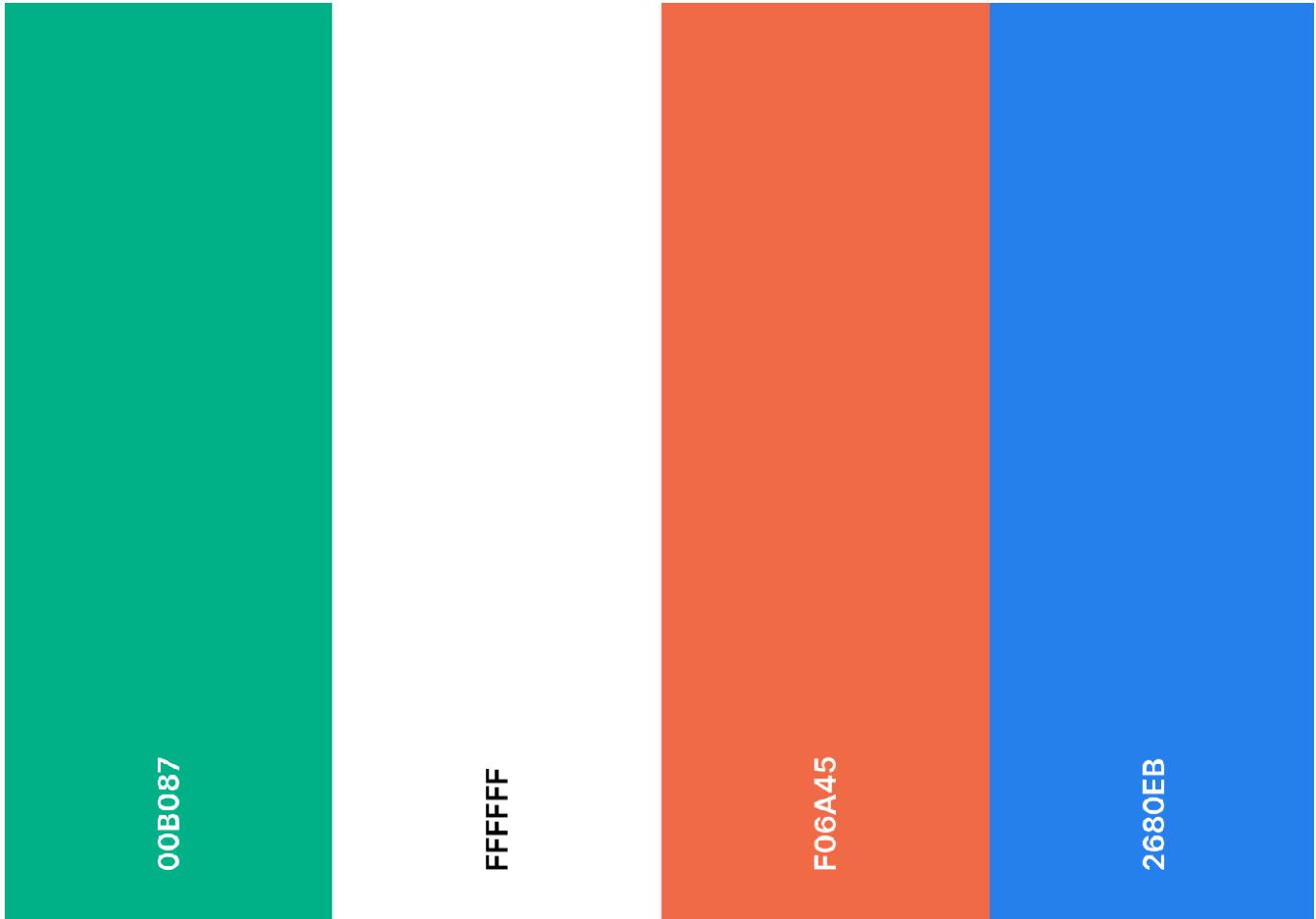


Figura 6.2: Palette di colori di Vamos!

6.6 Analisi della legibility

Per aumentare il grado che un sistema ha di essere user-friendly, è importante ricercare gli espedienti che aumentino il più possibile il suo livello di legibility, in modo tale da consentire all'utente di distinguere più facilmente le parole. La seguente Tabella analizza il grado di legibility del sistema.

Tabella 6.21: Analisi della legibility

Obiettivo	Valutare la legibility del sistema		
	Regola	Risultato atteso	Risultato ottenuto?
	Utilizzare font standard e senza grazie, come il roboto per Android	Bisognerebbe utilizzare font standard per non affaticare la vista del lettore	✓
	Utilizzare una dimensione per il font tra il 10 e il 12	Bisognerebbe utilizzare una dimensione per il font che sia consona alla lettura	✓
	Utilizzare in maniera coerente le maiuscole e le minuscole	La leggibilità di un testo dipende da vari fattori, tra cui l'uso coerente di maiuscole e minuscole	✓
	Evitare il corsivo	Evitare il corsivo perché potrebbe portare ad un effetto di aliasing su alcuni schermi	✓
	Evitare testi lunghi	Bisognerebbe evitare testi lunghi	✓
	Preferire caratteri scuri su sfondi bianchi e viceversa	Bisognerebbe utilizzare in maniera coerente le tinte delle scritte	✓
	Non affiancare scritte di tinte spettralmente lontane	Bisognerebbe evitare di affiancare scritte che siano lontane nello spettro visivo, come ad esempio verde e blu, per evitare problemi di messa a fuoco visiva	✓
	Non veicolare tutta l'informazione per mezzo di colori	Per gli utenti daltonici	✓

6.7 Analisi della readability

Occorre analizzare la readability del sistema per verificare la semplicità con cui l'utente acquisisce le informazioni. In questa fase viene utilizzata una funzione $f : \mathbb{N}^3 \rightarrow [0; 100]$ che implementa l'indice di leggibilità (index of readability) di un sistema definita come

$$f(\#frasi, \#lettere, \#parole) = 89 + \frac{300 \cdot \#frasi - 10 \cdot \#lettere}{\#parole}$$

La seguente Tabella analizza il grado di readability del sistema.

Tabella 6.23: Analisi della readability

Obiettivo	Valutare la readability del sistema		
	Regola	Risultato atteso	Risultato ottenuto?
	Usare parole precise ma semplici	Bisognerebbe usare parole precisi ma semplici che forniscano in maniera ottimale l'informazione	✓
	Omettere parole inutili	Soprattutto preposizioni articolare	✓
	Costruire periodi/frasi brevi	Di solito sono più facilmente interpretabili	✓
	Esprimere idee simili in forma simile	Ricercare quindi la coerenza	✓
	Preferire la costruzione positiva a quella negativa	Usualmente è più chiara e diretta	✓

Capitolo 7

Confronto coi competitor

L'analisi dei competitor risulta necessaria ed utile per diversi aspetti: in primis fornisce informazioni sulle altre applicazioni simili a **Vamos!** le quali possono essere utilizzate non solo per indebolire i concorrenti ma per portare numerosi vantaggi e migliorie al proprio prodotto. Un'analisi dei punti di forza dei concorrenti, infatti, può fornire informazioni utili su sistemi o meccaniche di funzionamento che possono essere implementate anche in **Vamos!** mentre un'analisi dei punti deboli può fornire informazioni su quali caratteristiche enfatizzare per sbaragliare la concorrenza.

7.1 Chi sono i nostri competitor?

L'applicazione **Vamos!** può essere vista essenzialmente come un'app per escursioni turistiche, dal momento che fornisce informazioni e consigli sulle più belle attrazioni nelle zone limitrofe all'utente. Per questo motivo i nostri competitor principali possono essere applicazioni sullo stile di TripAdvisor.

7.2 Analisi esterna

L'analisi esterna è l'analisi compiuta a partire da fonti esterne come brochure o cataloghi, siti internet pubblici, news e media, fiere, eventi, convegni, dati pubblici come bilanci e report e clienti comuni. L'analisi della concorrenza esamina l'offerta sul territorio rivolgendo l'attenzione alle caratteristiche delle imprese direttamente concorrenti (competitor), cioè tutte quelle che si rivolgono allo stesso target di persone o simile.

Un'analisi di questo tipo occorre per comprendere le diversità e le analogie più rilevanti dei prodotti o dei servizi della concorrenza. L'obiettivo che ci si pone è quello di evidenziare i nostri punti di forza (strength) e quelli di debolezza (weakness) rispetto ai concorrenti. Le informazioni raccolte dall'analisi esterna consentono di individuare gli elementi su cui poter fondare il proprio vantaggio competitivo, cioè la combinazione di risorse che assicuri al prodotto un successo imprenditoriale nel medio/lungo periodo. Il vantaggio competitivo raggiunto è il vantaggio competitivo di costo, anche detto leadership di costo, che è la strategia che riesce nell'organizzare il processo produttivo al minor costo possibile in termini di risorse: il deploy di **Vamos!** infatti è stato compiuto senza alcun importo da parte degli sviluppatori, dal momento che i servizi di cui necessita sono tutti servizi gratuiti (free tier). L'altro vantaggio competitivo raggiunto è quello di differenziazione: **Vamos!** si distingue dai concorrenti dal momento che è l'unica applicazione a calarsi in un contesto ludico con tecniche di gamification ed è l'unica applicazione, tra le concorrenti, ad ammettere un accesso gratuito alle strutture tramite gettoni virtuali guadagnate semplicemente utilizzando l'applicazione in maniera più o meno assidua.

7.3 Analisi interna

L'analisi interna è l'analisi compiuta a partire da fonti interne come esperienza personale, database, rappresentanti del prodotto, contatti di business, report privati di vendita e dipendenti commerciali.

È stata utilizzata l'analisi SWOT per valutare i punti di forza e debolezza, in maniera più precisa, del prodotto rintracciando le minacce e le opportunità che derivano dall'ambiente esterno oppure punti di forza e punti di debolezza che derivano dall'ambiente interno, come mostrato nel paragrafo successivo

7.4 SWOT

Questo paragrafo mostra i punti evinti dalla precedente analisi esterna ed interna disponendoli nella cosiddetta matrice SWOT.

Focus interno - punti di forza (del prodotto):

1. Conforme alle best practise della human-computer interaction e dell'ingegneria del software
2. Gamification
3. Gettoni virtuali per accessi gratuiti
4. Focus mirato alle attrazioni turistiche
5. Sistema modulare e altamente manutenibile: è facile cambiare qualsiasi aspetto tecnologico per stare al passo coi tempi ed anticipare il cambiamento

Focus interno - punti di debolezza (del prodotto):

1. Disponibilità economica da parte del team di sviluppo scarsa
2. Server e database ospitati in ambienti free tier
3. Applicazione client non ancora disponibile per iOS, ma esclusiva per Android

Focus esterno - opportunità (dall'esterno che favoriscono la crescita del prodotto):

1. Inserimento di nuovi membri nel team di sviluppo
2. Aumento del budget disponibile e conseguente miglioramento delle risorse e dei servizi utilizzati dall'app
3. Nascita del prodotto come soluzione palliativa ad un problema moderno causato dal Covid

Focus esterno - minacce (dall'esterno che favoriscono la decrescita del prodotto):

1. Mercato competitivo
2. Deploy del prodotto in tempi Covid ed in particolare durante la zona rossa, rendendo quasi impossibile la visita di attrazioni nei primi momenti

Capitolo 8

Analisi architettura del sistema

Questo capitolo è stato inserito per completezza e per avere una visione più concreta dell'architettura del sistema.

L'obiettivo principale della scelta di un'architettura è quello di impostare la visione complessiva del sistema in termini di visibilità dall'esterno. Di conseguenza, definire un'architettura consiste nell'identificare i moduli e le politiche di comunicazione tra essi.

Per una decomposizione ottimale ed un livello di manutenibilità¹ elevato è stata scelta un'architettura bassata su *layer* (strati). Ogni layer rappresenta una decomposizione gerarchica in macro blocchi e ogni macro blocco rappresenta un sottosistema che offre dei servizi. L'architettura a tre strati adottata è chiusa: ciò significa che ogni layer potrà accedere solo al layer immediatamente al di sotto di esso; in altre parole, la modifica a un livello non influenzera il quello superiore (ad esempio sarà possibile modificare la tecnologia del database senza che i client vi siano influenzati), portando quindi a un livello di manutenibilità, modularità², riusabilità³ e portabilità⁴ più alto. Bisognerebbe sempre progettare per anticipare il cambiamento. In Figura 8.1 è rappresentata l'architettura a tre livelli adottata. L'utente impiega il proprio client mobile per effettuare richieste http ad un server Spring; il server, a questo punto, raccoglierà i dati richiesti da un Database noSQL, MongoDB; infine una loro rappresentazione verrà restituita tramite una risposta http ai client che avevano fatto richiesta. Lo stile architettonico adottato è quello REST (*REpresentational State Transfer*) che si basa sul funzionamento del protocollo http (in particolare sui relativi metodi GET, POST, PUT e DELETE) ed una struttura delle risorse univocamente determinate (e indirizzabili) da un identificatore universale (URI). Un'architettura REST consente di sfruttare al massimo le caratteristiche del www e del protocollo http e di incrementare il livello di scalabilità⁵ del sistema che si sta progettando; in particolare il fatto che http sia *stateless*⁶ consente di semplificare la comunicazione tra client e server poiché al client sarà sufficiente conoscere solo l'URI della risorsa cercata ed un metodo http con cui far richieste; non occorre conoscere il contesto del client - al più esso potrà essere salvato mediante altri espedienti di session tracking non nativamente contemplati dal protocollo http. Il formato di rappresentazione scelto è JSON perché rispetto a HTML o XML è più simile al linguaggio naturale e più facilmente leggibile dagli esseri umani, e quindi dal team di sviluppo.

¹Qualità interna di un Software che stabilisce quanto sia facile da parte degli sviluppatori la modifica di qualche funzionalità o l'aggiunta di nuove, sia in termini di requisiti del cliente che in termini di cambiamenti legislativi

²Scissione del Software in micro-componenti

³Qualità interna del Software che stabilisce la possibilità di riciclare classi o, in generale, brani di codice per altri progetti

⁴Qualità interna del Software che stabilisce su quante piattaforme è possibile eseguirlo

⁵Qualità esterna del Software che stabilisce quanto è adattabile ad un cambiamento in base alla mole (ad esempio gestire 1000 record e gestirne 10000)

⁶Non tiene traccia delle informazioni di sessione

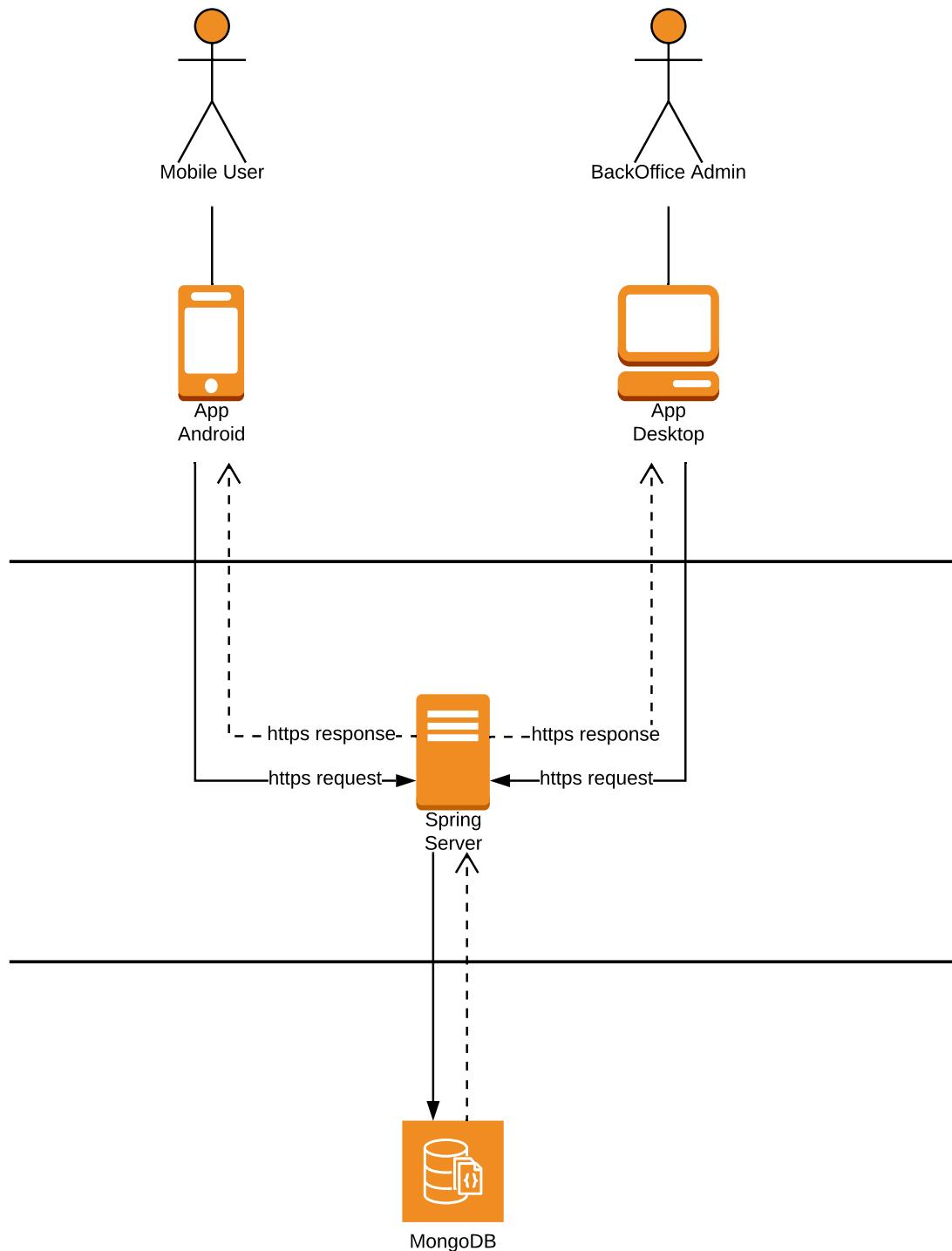


Figura 8.1: Architettura del Sistema

8.1 Il client Android

Il client è stato progettato usufruendo dei principali Design Pattern, in particolare:

- Data Access Object - mira a risolvere il problema dell'accesso ai dati il cui storage può cambiare nel tempo. L'idea è quella di avere una logica di funzionamento, un modello che rappresenta il dato ed una classe gemella che si occupa esclusivamente dell'accesso al dato (e in generale alle relative operazioni CRUD). L'idea è quindi quella di creare n classi entità ed altrettante classi DAO che si occupano *solo* di gestire l'accesso al dato. È importante questa modularizzazione per far sì che ogni classe abbia *un'unica responsabilità* (responsability-driven design, ovvero progettare pensando in primis alla responsabilità - unica - che una classe dovrebbe avere). Lo svantaggio principale dovuto a questo design pattern è quello di ritrovarsi, alla fine, un class diagram più complesso in termini di numero di classi presenti
- Factory Method - design pattern creazionale che permette di creare istanze di classi in un unico punto del codice in base ad una configurazione o logica a runtime. In particolare, nel nostro progetto DAO e Factory Method sono stati usati in maniera congiunta per istanziare la giusta classe DAO in base allo storage utilizzato (nel nostro caso mongodb per le strutture, ad esempio). Questo naturalmente porta a un livello di manutenibilità molto alto, pur aumentando leggermente la dipendenza tra la classe Factory e quelle che implementano l'interfaccia
- Singleton - design pattern creazionale che consente di avere un'unica istanza per una certa classe. In particolare nel nostro progetto una classe implementata tramite questo design pattern è la classe DAOFactory, cioè quella che consente di istanziare la corretta classe DAO in base ad una configurazione runtime, come discusso nel punto precedente

8.2 Il server

Il server è stato implementato attraverso il framework Spring Boot. Esso mette a disposizioni diverse caratteristiche native, come ad esempio la *dependency injection*: le variabili di istanze sono *iniettate* all'interno dell'oggetto tramite il suo costruttore e non sarà mai l'oggetto stesso a implementare gli altri oggetti che necessita; per questo motivo, le classi risulteranno maggiormente disaccoppiate (loose coupling) e avranno meno responsabilità (alta coesione), e quindi di conseguenza sarà anche più semplice fare testing (unit e integration) e sarà più semplice sostituire eventualmente un'implementazione, qualora ve ne sia bisogno in futuro (aumento di manutenibilità). Un'altra caratteristica nativa di Spring Boot è la presenza di *annotation*; citiamo alcune:

- @Service - indica che un'interfaccia fornisce dei servizi (ad esempio servizi CRUD)
- @Repository - indica che una classe astrae una repository
- @Controller - indica che una classe astrae la logica di business e può quindi gestire richieste http
- @GetMapping - indica che un metodo di un controller gestisce le richieste GET
- @PostMapping - indica che un metodo di un controller gestisce le richieste POST
- @PutMapping - indica che un metodo di un controller gestisce le richieste PUT
- @DeleteMapping - indica che un metodo di un controller gestisce le richieste DELETE

8.3 Le tecnologie impiegate

Durante la fase di System Design sono state scelte le tecnologie e i dettagli implementativi che potevano al meglio realizzare il software.

- Java - uno dei linguaggi di programmazione Object-Oriented più utilizzati, molte API disponibili e documentazione ufficiale molto ricca;
- Spring Boot - web framework open source per lo sviluppo di web app basate su Java;
- Atlas - sistema cloud per il deployment e la gestione di un database noSQL MongoDB su un provider cloud come AWS
- Google Maps API - fornisce un'API tramite cui ricercare le strutture ricettive su mappa;
- Servizi AWS - in particolare:
 - Cognito, strumento scalabile di accesso, registrazione e controllo degli accessi sicuro basato su autenticazione standard. Il suo funzionamento è illustrato in Figura 8.2;
 - Elastic Beanstalk, permette di distribuire e ridimensionare web app senza preoccuparsi dell'infrastruttura sottostante (PaaS: Platform as Service): sarà lui che gestirà provisioning, autoscaling e healthing. Il suo funzionamento è illustrato in Figura 8.3;
 - S3, servizio di object storage che offre scalabilità, sicurezza e prestazioni all'avanguardia mediante la creazione di bucket (cartelle) che consentono di accedere ai loro contenuti tramite un URL;
 - API Gateway, servizio che permette la creazione, la manutenzione e la protezione di API su qualsiasi scala. Il suo funzionamento è illustrato in Figura 8.4;

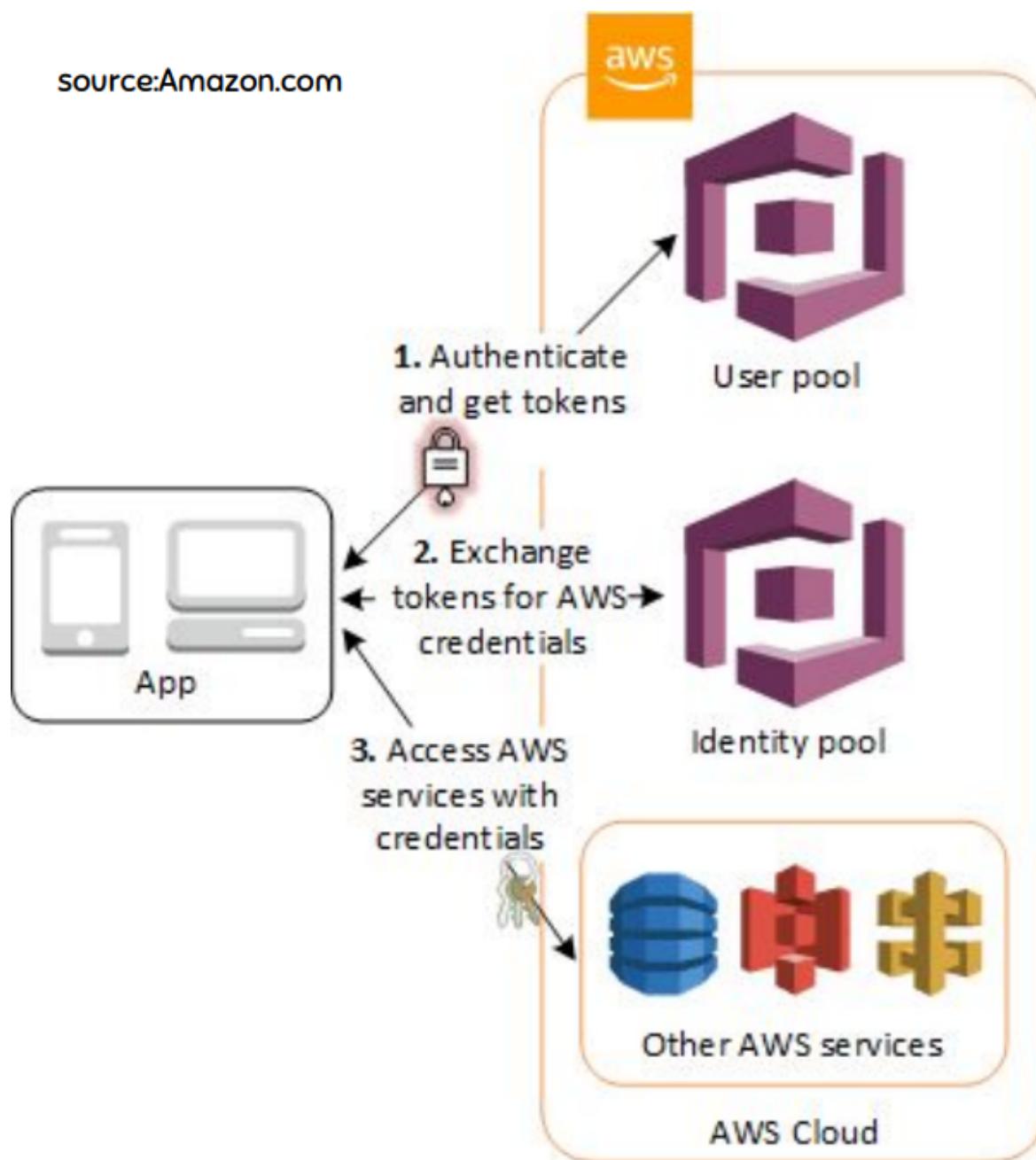


Figura 8.2: Funzionamento Cognito

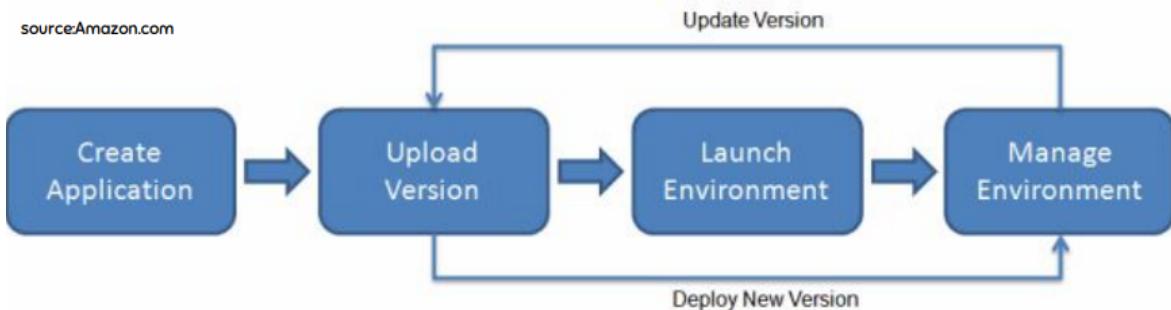


Figura 8.3: Funzionamento Elastic Beanstalk

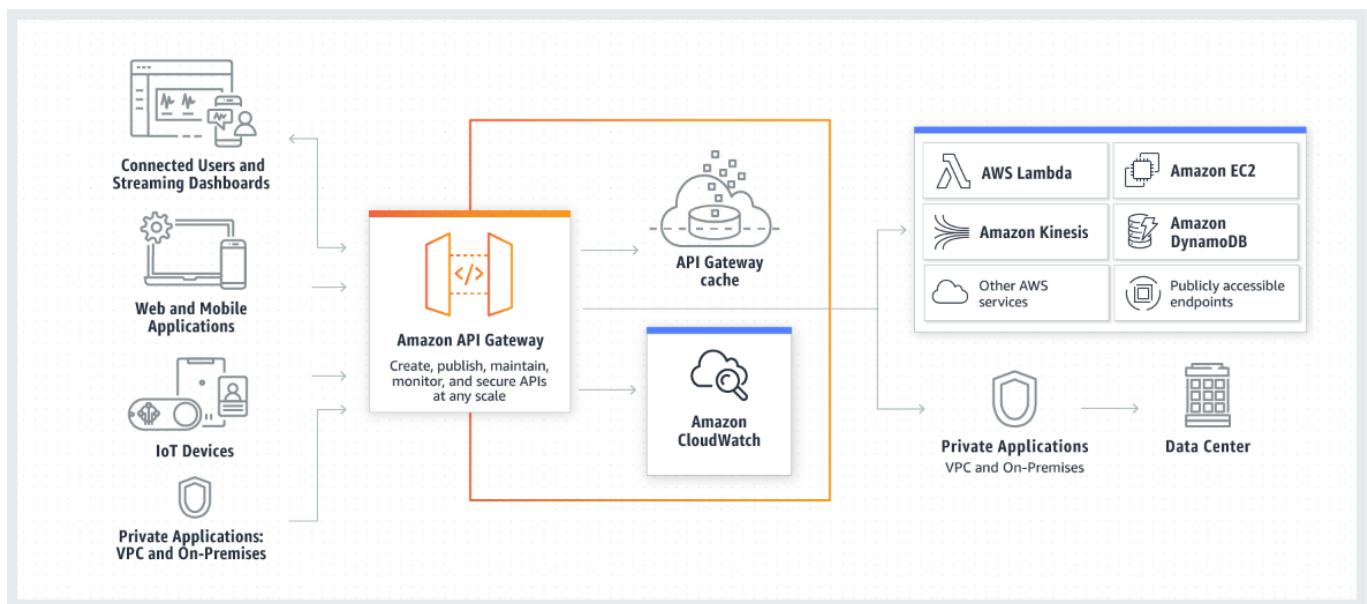


Figura 8.4: Funzionamento API Gateway

8.3.1 Perché Java?

Come già anticipato prima, Java è uno dei linguaggi Object-Oriented più utilizzati inoltre è oggetto di studio in numerosi corsi del Corso di Laurea. Da un punto di vista pratico, l'impiego di questo linguaggio di programmazione ha implicato numerosi vantaggi. In primis, Java è un metà compilato e metà interpretato: questo consente di attingere a caratteristiche e peculiarità sia dei compilatori che degli interpreti, come ad esempio il numero di controlli elevato - caratteristica di un compilato - oppure un'interezione più snella - caratteristica di un interpretato. Java, inoltre, consente di gestire in maniera più sicura la memoria allocata: attraverso un sistema di Garbage Collector, esso riesce in automatico a individuare aree di memoria dereferenziate e disallocarle, senza tuttavia richiedere un'operazione esplicita da parte del programmatore (ad esempio, il linguaggio C presupponeva l'uso di `free` per liberare memoria precedentemente allocata in maniera dinamica - e questo poteva essere pericoloso). Inoltre si tratta di un linguaggio (quasi) fortemente tipizzato, perché i controlli di tipo sono (quasi) tutti sempre effettuati. Java è detto anche indipendente dalla macchina e questo permette di elevare il livello di portabilità del sistema. Il motivo di ciò è dovuto al fatto che il bytecode viene eseguito sulla Java Virtual Machine che funge anche da interprete e da sandbox, ovvero un ambiente protetto in cui ciò che accade al suo interno non influenzerà l'ambiente esterno che lo ospita e quindi senza danneggiare il sistema (al peggio avviene un Denial of Service, consumando tutte le risorse del processore). La creazione del bytecode (`.class`) da parte del compilatore di Java consente di definire il fulcro per cui questo linguaggio risulta essere un ottimo compromesso tra i linguaggi compilati e quelli interpretati: è più portabile di un linguaggio compilato ma più efficiente di uno interpretato. È un linguaggio molto consigliato per le applicazioni web non solo per le caratteristiche appena descritte, ma anche perché risulta molto sicuro: come già anticipato, non ammette aritmetica dei puntatori (potrebbe essere pericoloso gestire esplicitamente le aree di memoria, oppure potrebbe portare ad uno stato indicibile e accessi non protetti alle aree di memoria portano ai più comuni problemi di sicurezza di un'applicazione⁷); quando un programma Java entra in esecuzione, la JVM costruisce per essa una sandbox in cui, come detto, ciò che accade al suo interno non influenza in modo critico il sistema che lo ospita. Oltre a ciò, all'interno della sandbox vengono fatti anche diversi controlli sulle classi che vi transitano, oppure controlli che mirano a gestire accessi all'esterno della sandbox. È importante notare che non è sufficiente implementare costrutti sicuri solo nel linguaggio utilizzato, ma anche nell'ambiente di esecuzione: il bytecode può essere generato a partire da codice `.java` ma anche da codice che deriva da altri linguaggi di programmazione meno sicuri, oppure addirittura potrebbe essere stato scritto a mano con intenzioni ostili; per fare ciò l'esecuzione di una classe Java prevede il caricamento da parte del ClassLoader del bytecode `.class`, poi l'esecuzione del metodo `main` ed infine il caricamento delle classi accessorie. Così facendo il bytecode viene verificato ogni volta che una classe viene caricata. Il controllo comunque può essere più o meno flessibile quando le classi sono create localmente (poichè ritenute affidabili di natura), mentre è più rigido se le classi provengono da altre fonti. Il Class Loader, dopo aver compiuto queste verifiche e solo quando è tutto ok, crea le classi vere e proprie, preparando un namespace diverso per ogni fonte da cui provengono le classi, evitando quindi problemi di collisione. Il Class Loader, inoltre, evita che classi provenienti da host diversi possano comunicare tra di loro, questo per escludere la possibilità che un programma poco affidabile possa attingere e ricavare informazioni da uno affidabile. Per aumentare maggiormente il livello di sicurezza del linguaggio, Java prevede che ogni JVM sia dotata (al più) di un Security Manager responsabile delle politiche di sicurezza. È possibile restringere oppure allentare i vincoli e i controlli sulle operazioni settando opportunamente le politiche del Security Manager e ogni qual volta venga tentato di fare un'azione vietata, verrà sollevata una `SecurityException`.

⁷Molti virus traggono vantaggio dalla possibilità di dirottare l'aritmetica dei puntatori e quindi fare accesso e modificare aree di memoria in modo incontrollato

8.3.2 Perché un web framework?

L'impiego di un framework, in generale, consente di alleggerire il lavoro legato allo sviluppo di una qualche applicazione. Nel nostro progetto è stato utilizzato il *web framework* Spring Boot, che è stato utile quindi per alleggerire lo sviluppo di una web app, in particolare del server, mettendo a disposizione strumenti per la programmazione lato server, dal momento che in generale la programmazione di un server ruota sempre intorno ad operazioni ricorrenti, come la connessione al data source, oppure alla specifiche del content-type e così via. Con Spring Boot, o in generale con un web framework, ci è stato possibile focalizzarci sul lato significativo della programmazione server. Oltre a quanto descritto, l'impiego di un web framework implica alcune caratteristiche di design come ad esempio minore quantità di codice, *DRY* (Don't Repeat Yourself), che sconsiglia di ripetere più volte brani di codice e di "reinventare la ruota", *Loose coupling*, concetto di basso accoppiamento già discusso.

8.3.3 Perché MongoDB?

L'impiego di un sistema di database noSQL non è casuale: MongoDB ci ha permesso diversi vantaggi:

- Facilitare lo sviluppo attraverso nuove metodologie di progettazione;
- Rappresentazione dei dati intuitiva - *schemaless* o *dynamic schema*;
- Polimorfismo tra dati - esattamente come nei linguaggi Object Oriented;
- Scalabilità - sfruttando un hardware evoluto e la scalabilità orizzontale;

I Database meno evoluti di MongoDB consentono la scalabilità *verticale*: inizialmente potrà essere sufficiente un mini server; ma quando esso non riuscirà più a soddisfare tutte le richieste di un pubblico crescente, dovrà essere necessario sostituire il mini server con uno più potente. Uno svantaggio di questo approccio è che, solitamente, server molto potenti sono anche molto costosi, inoltre non permetterebbe una gestione troppo elastica della scalabilità del sistema.

MongoDB, invece, supporta la scalabilità *orizzontale*: inizialmente sarà sufficiente un mini server; in questo caso, però, quando esso non riuscirà più a soddisfare tutte le richieste di un pubblico crescente, anziché sostituirlo con uno più potente, verrà aggiunto al mini server un altro mini server. In generale, collegando tra loro n mini server, sarà possibile soddisfare le richieste simulando un server medio o grande. In questo modo si avrà una reale scalabilità, perché in qualsiasi momento sarà possibile aggiungere o rimuovere i server nel cluster o rendere attivi solo quelli realmente necessari, senza sprechi in termini di costi. Altri vantaggi legati alla scalabilità orizzontale offerta da MongoDB è la gestione del fail-over (se un server smette di funzionare, le sue attività vengono indirizzate verso un'altra macchina attiva), alta disponibilità e gestione del disaster recovery (garantendo un'erogazione sempre attiva dei servizi anche in casi estremi). L'approccio è illustrato in Figura 8.5.

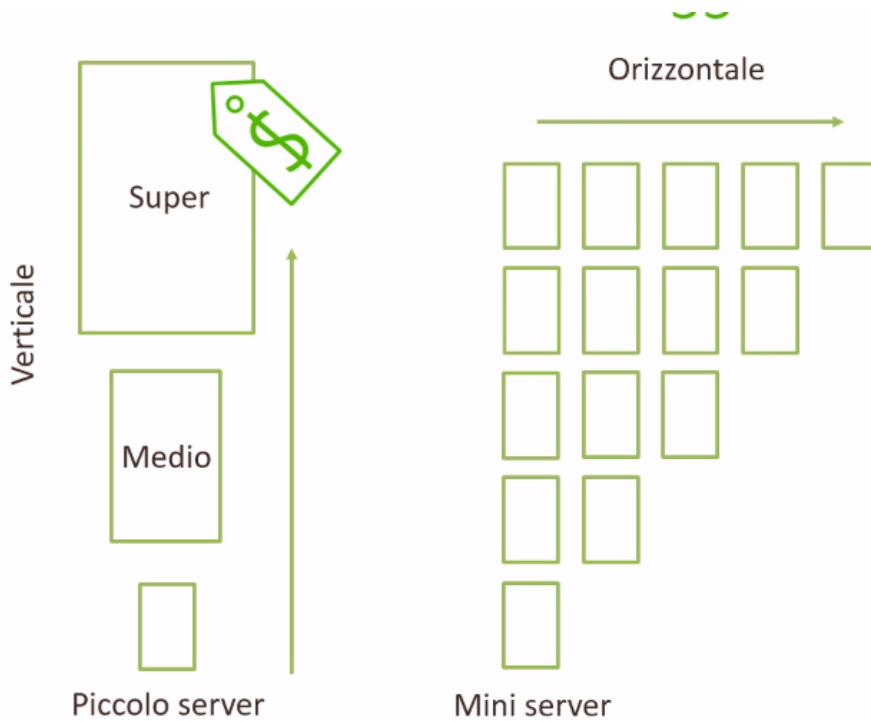


Figura 8.5: La scalabilità verticale e orizzontale a confronto

La scalabilità orizzontale è piuttosto interessante, il problema nasce da un punto di vista pratico quando tutti i server devono comunicare tra loro e farlo nel minor tempo possibile; MongoDB cerca di soddisfare questa esigenza posizionandosi in un punto *strategico* nel panorama dei Database moderni. Come mostra la Figura 8.6, MongoDB si posiziona in un punto medio tra la scalabilità e le performance fornite da un Database di tipo key:value e le funzionalità di un RDBMS come Oracle o MySQL poco prima che la curva inizi a decrescere vertiginosamente, in modo tale da garantire molte funzionalità senza rinunciare a scalabilità e performance. Ma cosa manca a MongoDB rispetto a un RDBMS tradizionale? Mancano le *join*; tuttavia non se ne sente per nulla la mancanza perché il concetto di *join* viene implementato attraverso un *nesting* di file JSON che, addirittura, rende più immediata e semplice la lettura ad un pubblico umano seguendo una sintassi simile al modello "key": "value" senza tuttavia risultare pesante al Parser come un formato XML. Inoltre il JSON viene rappresentato in formato binario dal BSON che viene usato soltanto internamente al driver di MongoDB in maniera del tutto automatica, per rendere un file JSON ancora più leggero; ciò comporta un aumento di prestazione anche relativamente alle operazioni CRUD: in questo modo la ricerca di una chiave non sarà più lineare, costringendo al driver di MongoDB a scorrere riga per riga *n* coppia "key": "value" ma casuale, riducendo il tempo di esecuzione dell'algoritmo di ricerca di un document all'interno della collection da parte del driver di MongoDB da lineare a costante.

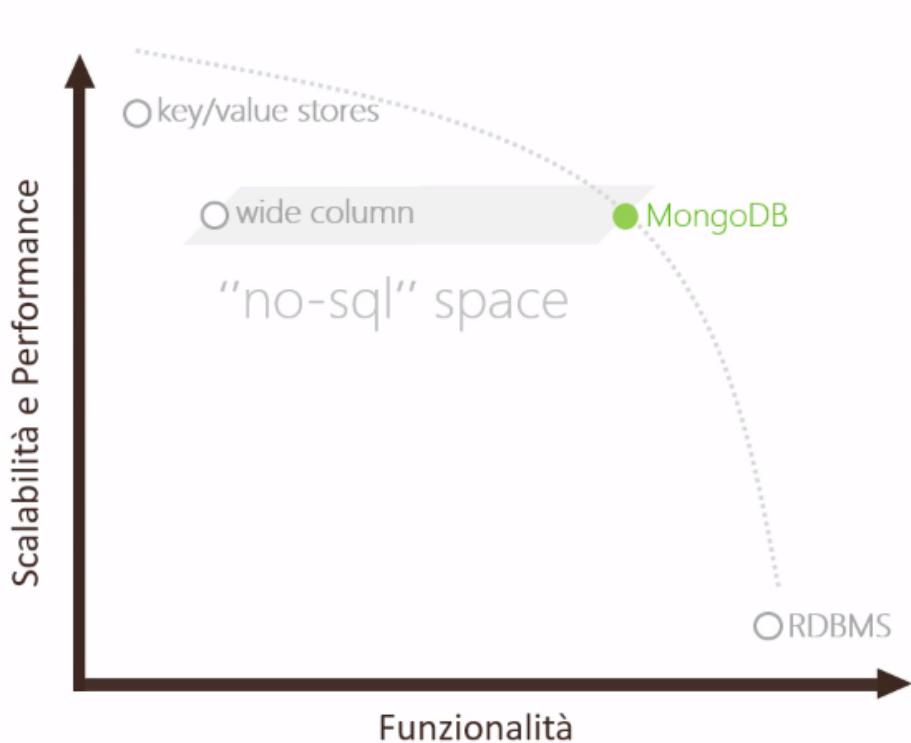


Figura 8.6: La posizione strategica di MongoDB nel panorama moderno

La caratteristica di essere *schemaless* o, per alcuni di avere un *dynamic schema*, garantisce il polimorfismo come i linguaggi Object Oriented, rendendo quindi il "linguaggio" di MongoDB molto più simile a quello di Java, ad esempio, rispetto a quanto non facevano gli altri Database, risolvendo quindi la mancata (o debole) corrispondenza tra oggetti O-O e le entità del Database.

Capitolo 9

Note finali

Abbiamo fortemente voluto progettare un prodotto che fosse anche socialmente utile ed impegnato, sia per il turismo locale e quindi per i commercianti e i responsabili delle attrazioni turistiche, ma anche per tutti coloro che stanno soffrendo psicologicamente: la pandemia Covid ha portato troppi problemi sul piano mentale e molti di essi ignobilmente liquidati con un "c'è chi sta peggio", lasciando chi è solo ancora più solo e come se la vita fosse una gara a chi soffre di più senza rendersi conto che ogni sofferenza deve essere rispettata.

Con l'augurio che finisca tutto presto e nel migliore dei modi,
il team di sviluppo.



Figura 9.1: Il team di sviluppo

Ogni persona che incontri sta combattendo una battaglia di cui non sai niente.

Capitolo 10

Glossario

Dato personale - qualsiasi informazione riguardante una persona fisica identificata o identificabile («interessato»); si considera identificabile la persona fisica che può essere identificata, direttamente o indirettamente, con particolare riferimento a un identificativo come il nome, un numero di identificazione, dati relativi all'ubicazione, un identificativo online o a uno o più elementi caratteristici della sua identità fisica, fisiologica, genetica, psichica, economica, culturale o sociale

CRUD - tipiche operazioni di Create-Retrieve-Update-Delete effettuate su un database

Flood - abuso di messaggi o testo in generale

Paginazione - proiezione parziale dei dati al fine di preservare performance ed evitare carichi alla rete

Comprendibilità - caratteristica di un software che quantifica quanto è facile interpretare il tutto

Visibilità - caratteristica di un software che quantifica a che punto si è con lo sviluppo dello stesso

Supportabilità - caratteristica di un software che quantifica quanti strumenti di supporto ci sono

Accettabilità - caratteristica di un software che quantifica quanto sono contente le persone nel team nel ruolo assegnatogli

Affidabilità - caratteristica di un software che quantifica la qualità dello stesso

Robustezza - caratteristica di un software che quantifica quanto è robusto il software a cambiamenti in corso d'opera

Manutenibilità - caratteristica di un software che quantifica quanto è robusto e adattabile il software a cambiamenti futuri

Rapidità - caratteristica di un software che quantifica il tempo necessario al deploy

Modularità - scissione del software in micro-componenti

Riusabilità - qualità interna del software che stabilisce la possibilità di riciclare classi o, in generale, brani di codice per altri progetti

Portabilità - qualità interna del software che stabilisce su quante piattaforma possa essere eseguito

Scalabilità - qualità esterna del software che stabilisce quanto è adattabile lo stesso ad un cambiamento rispetto alla mole (crescente o decrescente) verso cui è orientata

Stateless - caratteristica del protocollo http che stabilisce che non riesce a tenere traccia delle informazioni sulla sessione di comunicazione tra client e server

Approccio "Code-and-fix" - metodologia per cui, ciclicamente, si scrive del codice, lo compila e poi lo si riprova

Elenco delle figure

2.1	Sistemi user-centered	7
2.2	Diagramma di Eulero-Venn che rappresenta la popolazione affetta da sindrome della capanna	9
2.3	Diagramma di Eulero-Venn che rappresenta la popolazione affetta da sindrome della capanna appassionata di arte e attrazioni turistiche	10
2.4	Distribuzione delle variabili comportamentali degli elementi dell'insieme C	11
2.5	Personas card per Mariavittoria Ortona	13
2.6	Personas card per Salvatore Ascione	14
2.7	Personas card per Rebecca Giamundo	15
2.8	Personas card per Giovanni Arpa	16
2.9	Primary personas goal e Business goal	17
2.10	Primary e Secondary personas-type goal e Business goal	18
2.11	Necessità delle negative personas disgiunte rispetto agli obiettivi del prodotto	19
3.1	Spazio dei prototipi in relazione al loro scopo	21
3.2	Prototipazione iniziale "a schizzo" del prodotto	22
4.1	Project Management - Work Breakpoint Structure	26
4.2	Project Management - Diagramma di PERT	27
4.3	Project Management - Costruzione e Distruzione	28
4.4	Use Case Diagram	29
5.1	Prototipazione "a schizzo" del prodotto per la completezza orizzontale	31
5.2	Prototipo verticale per il login	33
5.3	Mockup per la homepage	36
5.4	Mockup per il caso d'uso Cerca Strutture	37
5.5	Mockup per il caso d'uso Visualizza Strutture su Mappa	38
5.6	Mockup per il caso d'uso Cerca Strutture su Mappa	39
5.7	Mockup per l'overview di un'attrazione	40
5.8	Mockup per il caso d'uso Lascia Recensione	41
5.9	Mockup per la lista di recensioni	42
5.10	Mockup per il profilo	43
5.11	Mockup per il caso d'uso Login	44
5.12	Mockup per il caso d'uso Crea Account	45
5.13	Mockup per la classifica	46
5.14	Mockup per la homepage	47
5.15	Mockup per la ricerca di strutture su mappa	48
5.16	Mockup per la mappa	49
5.17	Mockup per l'overview di una struttura	50
5.18	Mockup per la scrittura di una recensione	51
5.19	Mockup per la lettura di recensioni	52

5.20 Mockup per il profilo	53
5.21 Mockup per il login	54
5.22 Mockup per la registrazione	55
5.23 Mockup per la classifica	56
5.24 Statechart Diagram d'Analisi per il caso d'uso Crea Account	57
5.25 Statechart Diagram d'Analisi per il caso d'uso Login	58
5.26 Statechart Diagram d'Analisi per il caso d'uso Visualizza Strutture	59
5.27 Statechart Diagram d'Analisi per il caso d'uso Visualizza Strutture su Mappa	60
5.28 Statechart Diagram d'Analisi per il caso d'uso Legge Recensioni	61
5.29 Statechart Diagram d'Analisi per il caso d'uso Lascia Recensione	62
5.30 Mobile - Statechart Diagram di Design per il caso d'uso Crea Account	63
5.31 Mobile - Statechart Diagram di Design per il caso d'uso Login	64
5.32 Mobile - Statechart Diagram di Design per il caso d'uso Visualizza Strutture	65
5.33 Mobile - Statechart Diagram di Design per il caso d'uso Visualizza Strutture su Mappa	66
5.34 Mobile - Statechart Diagram di Design per il caso d'uso Legge Recensioni	67
5.35 Mobile - Statechart Diagram di Design per il caso d'uso Lascia Recensione	68
5.36 Mobile - Activity Diagram per il caso d'uso Crea Account	69
5.37 Mobile - Activity Diagram per il caso d'uso Login	70
5.38 Mobile - Activity Diagram per il caso d'uso Visualizza Strutture	71
5.39 Mobile - Activity Diagram per il caso d'uso Visualizza Strutture su Mappa	72
5.40 Mobile - Activity Diagram per il caso d'uso Legge Recensioni	73
5.41 Mobile - Activity Diagram per il caso d'uso Lascia Recensione	74
6.1 Confronto tra valutazione euristica di Nielsen e test di usabilità	78
6.2 Palette di colori di Vamos!	91
8.1 Architettura del Sistema	97
8.2 Funzionamento Cognito	100
8.3 Funzionamento Elastic Beanstalk	101
8.4 Funzionamento API Gateway	101
8.5 La scalabilità verticale e orizzontale a confronto	104
8.6 La posizione strategica di MongoDB nel panorama moderno	105
9.1 Il team di sviluppo	106

Elenco delle tabelle

5.1 Verifica della prototipazione orizzontale	32
6.1 Coerenza del sistema rispetto alle otto regole d'oro di Schneiderman	75
6.3 Valutazione mediante euristica di Nielsen	77
6.5 Compito - Crea Account	80
6.7 Compito - Login	81
6.9 Compito - Visualizza Strutture	82
6.11 Compito - Visualizza Strutture su Mappa	83
6.13 Compito - Legge Recensioni	84
6.15 Compito - Lascia Recensione (e acquisisce punti/titoli)	85
6.17 Analisi di coerenza delle icone	89
6.19 Analisi di coerenza dei colori	90
6.21 Analisi della legibility	92
6.23 Analisi della readability	93