

GamePlate: La piattaforma digitale di advergames per il rilancio della ristorazione

Relatore: *Prof.ssa Daniela Micucci*

Co-relatore: *Dott. Davide Ginelli*

Relazione della prova finale di:

Valerio Bonacina

Matricola 829840

Anno Accademico 2019-2020

A chi ha sempre creduto in me,
ma anche, soprattutto, a chi non l'ha mai fatto.

Indice

Sommario	3
Introduzione	4
1 Tecnologie	5
1.1 Ambienti di sviluppo	5
1.2 Framework e API	6
1.1.1 Apache Cordova	6
1.1.2 Firebase	7
1.1.3 Google Maps Platform	9
1.1.4 OpenCage API	10
2 Funzionalità e architettura	11
2.1 Funzionalità	11
2.2 Architettura	14
2.2.1 I motivi delle scelte	14
3 Progettazione	16
3.1 Firestore	16
3.1.1 Regole di lettura/scrittura	18
3.2 Realtime Database	19
3.2.1 Regole di lettura/scrittura	22
3.3 Applicazione	23
4 Sviluppo	25
4.1 Gestione utenze	25
4.1.1 Registrazione	25
4.1.2 Login	27
4.1.3 Stato account	28
4.1.4 Nickname	29
4.1.5 Logout	30
4.2 Ricerche	31
4.2.1 Geohashing	31
4.2.2 Ricerca nelle vicinanze	32
4.2.3 Cerca qui	34
4.2.4 Cerca tramite keyword	35
4.3 Giochi	38
4.3.1 Ricezione dati di dettaglio	38
4.3.2 Checksum crc32	39
4.3.3 Ricezione classifica da app	41
4.3.4 Apertura videogioco	41
4.3.5 Comunicazione CordovaWebView- Java nativo	43
4.4 Gestione premi	47
4.4.1 RSA	47
4.4.2 Erogazione ticket	48
4.4.3 Gestione scadenze	52
4.4.4 Riscatto	53
4.5 Salvataggio e ripristino stati	56

5	Interfaccia Grafica	58
5.1	Poncho: mascotte e aiutante	58
5.2	Registrazione e login	59
5.3	Schermata Home	60
5.3.1	Bottom Sheet	61
5.3.2	Ricerche	61
5.4	Dettaglio di un gioco	63
5.5	Premi	64
5.6	Profilo	65
6	Analisi di usabilità	66
6.1	Valutazione euristica	67
6.1.1	Ambiente di valutazione e composizione del campione	67
6.1.2	Euristiche	68
6.1.3	Prioritizzazione	69
6.1.4	Risultati	70
6.2	Release 2	71
6.3	Test Utente	75
6.3.1	Composizione del campione	76
6.3.2	Test statistici	76
6.3.3	Efficacia	77
6.3.4	Efficienza	78
6.4	Questionario	79
6.4.1	Test UEQ	79
6.4.2	Confronto statistico tra sistemi	80
6.4.3	Benchmark	83
6.4.4	Net Promoter Score	84
6.5	Considerazioni finali	84
	Conclusioni	85
	Appendice A	86
A.1	Task	86
A.2	Risultati della valutazione euristica	88
A.3	Risultati dei questionari	92
	Sitografia	93

Sommario

Il presente documento descrive il lavoro di stage svolto, in modalità smart-working per emergenza COVID-19, dai laureandi Bonacina Valerio e Cogo Luca, iniziato il 31/03/2020 e concluso il 15/06/2020.

Lo scopo dello stage è stato quello di sviluppare l'applicazione GamePlate, spiegata nel dettaglio nella pagina successiva di introduzione.

Gli obiettivi sono stati quelli di realizzare il maggior numero di funzionalità per la piattaforma nel modo più corretto e implementare le caratteristiche di contorno col tempo restante a disposizione.

È stato infine possibile svolgere un'analisi di usabilità sull'applicazione studiando statisticamente quanto fosse apprezzata dagli utenti reali attraverso l'interrogazione, secondo precise metodologie, di alcuni campioni di persone adeguatamente scelti tra la popolazione.

In particolare, il compito del tesista Bonacina Valerio è stato quello di realizzare il backend e la parte dell'applicazione che riguardava le pagine di 'home' e 'profilo', oltre che prendere parte allo studio di usabilità e creare il report della seconda metà di quella parte (test utenti e questionario).

Introduzione

Nell'era digitale la pubblicità è diventata talmente tanto parte delle nostre vite che evitarla è ormai un'azione che svolgiamo in automatico. L'unico effettivo scopo di queste campagne è divenuto quello di influenzare a livello subliminale l'utente, senza che quest'ultimo se ne accorga. La pubblicità in questo senso funziona, ma appare come un'imposizione dettata dal mercato e non certo una scelta personale.

Lo studio per una pubblicità più ‘pulita’ ci ha portato a pensare ad un modello di comunicazione prodotto/servizio-utente che sia dichiarato e chiaro fin dal principio e che sia stimolante e utile per tutte le parti.

L'industria videoludica ha ormai raggiunto e superato quota 120 miliardi di dollari e ha chiuso il 2019 con una crescita pari al 3% sui 12 mesi precedenti. A metterlo nero su bianco l'istituto SuperData Research^[1]. Questo dato fa chiaramente intuire quanto sia seguito questo ambiente, ormai non più solo dalla fascia d'età giovanile ma anche da quella più adulta.

In principio l'idea è stata quella di unire questi due mondi: uno malvisto e quasi sempre ignorato ma comunque influente (la pubblicità) e l'altro molto seguito da quasi tutte le fasce d'età ma che non ha alcuno scopo di influenzare le masse (i videogame).

Questa strategia non è nuova nel mondo del marketing ed ha il nome di ‘advergame’ (acronimo delle parole inglesi advertising e game), i primi videogiochi sviluppati con queste finalità risalgono addirittura al 1983^[2]. Un noto esempio è Pepsi Invaders, un clone di Space Invaders per Atari 2600, commissionato dalla The Coca-Cola Company.

L'evoluzione del mercato e del digitale non ha portato con sé questo business che però mostra ancora molto potenziale. Per rivoluzionare questo strumento abbiamo pensato all'aggiunta di una componente meritocratica verso gli utenti, offrendo premi, collegati al tipo di prodotto/servizio pubblicizzato, ottenibili attraverso i traguardi raggiunti nel gioco stesso.

GamePlate nasce con l'intento di fare pubblicità a locali e ristoranti in modo innovativo e originale attraverso i videogiochi. Ad ogni ristorante è affiancato un particolare videogioco basato su punteggi e gli utenti hanno la possibilità di vincere buoni e sconti, spendibili in quel locale, attraverso il superamento di determinati obiettivi nel gioco correlato.

Ogni settimana viene eseguito l'azzeramento di una classifica per ogni gioco e il vincitore riceverà in premio un buono gratuito per mangiare nel ristorante associato.

Questa nuova implementazione di advergame permette sia di incentivare il cliente a giocare (attraverso l'erogazione dei premi), sia di dare l'opportunità a quest'ultimo di poter visitare e testare il luogo sponsorizzato.

1 Tecnologie

1.1 Ambienti di sviluppo

Per sua natura GamePlate è costituita da una libreria di giochi hyper-casual e una piattaforma che consenta di accedervi.

Per realizzare quest'ultima si è optato per lo sviluppo di un'app Android nativa, in modo da ottenere efficienza e accesso a tutte le funzionalità del sistema operativo.

Per quanto riguarda i videogiochi invece si è data priorità alla portabilità, grazie all'utilizzo di HTML5 per la loro realizzazione. In questo modo tutti i giochi saranno compatibili con una eventuale versione di GamePlate per IOS.

Per sviluppare l'applicazione è stato utilizzato l'IDE Android Studio, mentre per i giochi ci si è serviti del software Construct.

Android Studio

Android Studio^[3] ^[4] è un ambiente di sviluppo integrato (IDE) basato sul software di JetBrains IntelliJ IDEA (un IDE per lo sviluppo Java) e progettato specificatamente per lo sviluppo di applicazioni Android.

Realizzato dalla stessa JetBrains e da Google, uscì in anteprima nel 2013 per poi essere rilasciato nella sua prima build stabile l'anno dopo.

Il software ha sostituito gli Android Development Tools (ADT) di Eclipse, diventando l'IDE primario di Google per lo sviluppo Android.

Tra le principali funzionalità introdotte si annoverano:

- Supporto ai linguaggi Java, C++ e Kotlin
- Sistema automatizzato di build basato su Gradle
- Editor visuale drag and drop per la realizzazione di UI
- Emulatore Android integrato (Android Virtual Device)
- Funzionalità di profiling e debugging

Construct

Construct^[5] è un game editor basato su HTML5 sviluppato da Scirra Ltd. e rilasciato nella sua prima versione nel 2011.

Caratterizzato dalla sua estrema semplicità d'uso, consente di realizzare rapidamente giochi grazie alla sua interfaccia drag and drop e all'approccio block-based.

Arrivato ormai alla sua terza versione, le sue funzionalità comprendono:

- Motore fisico integrato
- Algoritmi di pathfinding
- Editor di immagini integrato
- Funzionalità di profiling
- Supporto a JavaScript

Sono inoltre presenti numerosi plugin esterni per estenderne ulteriormente le possibilità.

Per queste ragioni, Construct è diventato uno dei game editor più utilizzati per la realizzazione di giochi per il web, al punto che oltre il 50% dei titoli presenti sul famoso portale Kongregate sono realizzati proprio con questo software.

1.2 Framework e API

Lo sviluppo di GamePlate ha portato a integrare diverse tecnologie e servizi esterni.

Per l'integrazione dei giochi HTML5 con Android è stato utilizzato il framework Apache Cordova.

Per attingere ai dati relativi a giochi, ristoranti e utenti, l'applicazione comunica con un database. In particolare, si è optato per la soluzione NoSQL fornita da Firebase.

Per visualizzare i ristoranti su una mappa si è scelto di utilizzare le Google Maps API e, infine, per implementare le ricerche basate su luoghi è stata utilizzata OpenCage, un'API di reverse geocoding.

Di seguito una spiegazione più dettagliata delle tecnologie appena elencate.

1.2.1 Apache Cordova

Apache Cordova^[6] è un framework di sviluppo mobile open source. Esso consente di utilizzare tecnologie standard web come CSS3, HTML5 e JavaScript per lo sviluppo di piattaforme, evitando il linguaggio di sviluppo nativo su sistemi operativi differenti.

In altre parole, consente di "tradurre" le applicazioni web in applicazioni mobili.

Creato originariamente da Nitobi, è stato poi acquisito nel 2011 da Adobe e ha visto la collaborazione di BlackBerry, Google, IBM, Intel, Microsoft, Mozilla e altri ancora.

Apache Cordova attualmente supporta i sistemi operativi Android, iOS, Bada, BlackBerry, Firefox OS, LG webOS, Microsoft Windows Phone, Symbian, Tizen e Ubuntu Touch.

Il software, come descritto nella figura 1, incapsula il codice CSS3, HTML5, JavaScript e consente di riutilizzarlo in contesto mobile. Le applicazioni ottenute sono definibili ibride, poiché non possono considerarsi né puramente native né basate completamente sul web.

A partire dalla versione 1.9 è possibile mischiare parti di codice ibrido (realizzato come descritto sopra) e parti di codice nativo.

Sono poi presenti numerosi plugin scritti in Java che utilizzano le API Android per offrire facilmente accesso alle funzionalità del sistema operativo.

Apache Cordova si rivela uno strumento molto utile nel caso in cui si voglia distribuire una web app su più piattaforme riutilizzando la stessa code base.

Il framework può però essere utilizzato anche nel contesto di uno sviluppo nativo, che necessiti di integrare una WebView in grado di accedere alle API del dispositivo.

1.2.2 Firebase

Firebase^[7] [8] è una piattaforma per lo sviluppo di applicazioni web e mobili, rilasciata nel 2011 ed acquisita da Google nel 2014.

Il primo servizio fornito è stato Firebase Real-time Database, un cloud database NoSQL.

Nel corso degli anni le funzionalità rilasciate sono sempre aumentate, tra cui Authentication, Hosting, Cloud Messaging, Firestore e altre ancora.

Ad oggi la piattaforma dispone di 19 prodotti, utilizzati di più di 1.5 milioni di applicazioni.

Di seguito una lista dei principali servizi offerti:

Firebase Authentication

Firebase Authentication è un servizio di autenticazione token-based, che consente di implementare registrazione e accesso degli utenti utilizzando solo codice lato client. Supporta i principali servizi di social login, come Facebook, Google e Twitter.

Viene integrato anche un sistema automatico per l'autenticazione degli utenti tramite e-mail e per il recupero delle password.

Di un utente autenticato è possibile ottenere diversi dati tra cui nome, foto profilo ed e-mail.

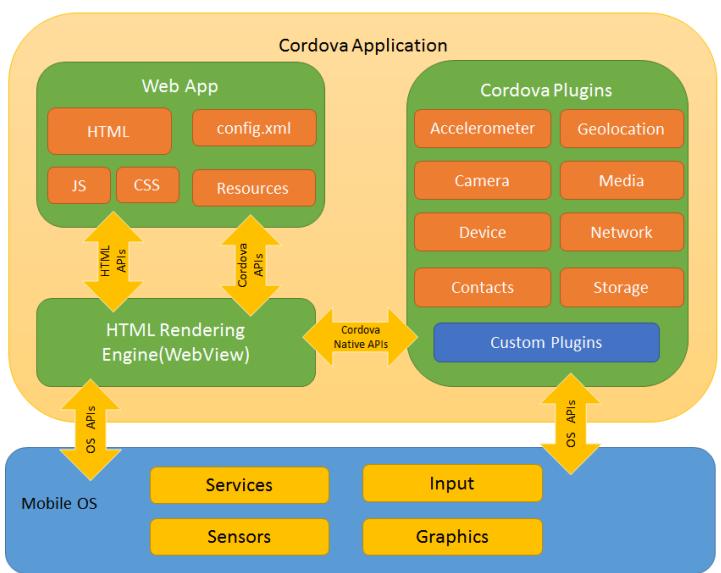


Figura 1
Schema di funzionamento di un'applicazione Cordova
(fonte: <https://cordova.apache.org/docs/en/latest/guide/overview>)

Firebase Realtime Database

Firebase Realtime Database è il primo servizio ad essere stato realizzato. Si tratta di un cloud database NoSQL, i cui dati vengono salvati sotto forma di file JSON e sincronizzati in tempo reale con tutti i client connessi. Tra le principali funzionalità si annoverano:

- **Realtime:** al contrario delle classiche richieste http, i dati vengono sincronizzati in pochi millisecondi con tutti i client connessi
- **Offline:** se il dispositivo è privo di connessione l'applicazione mantiene i dati già scaricati e li aggiorna una volta che torna online
- **Regole:** l'accesso ai dati può essere limitato da delle regole di sicurezza che vengono eseguite in lettura e in scrittura

Cloud Firestore

Rilasciato ufficialmente nel 2019, Cloud Firestore rappresenta un'alternativa a Firebase Realtime Database. Anch'esso è un cloud database NoSQL, che però abbandona la struttura ad albero e abbraccia una struttura a documenti.

Un'altra caratteristica che lo differenzia è la compatibilità con Cloud Functions e il supporto a un maggior numero di tipi di dati.

Cloud Functions

Cloud Functions è un framework che consente di eseguire automaticamente codice backend.

Lo sviluppatore scrive le funzioni specificando il tipo di evento e le condizioni necessarie perché esse vengano chiamate. Ad esempio, è possibile far eseguire il codice solamente quando viene creato un nuovo utente, oppure quando viene fatta una modifica al database.

Quando lo sviluppatore carica una nuova versione delle funzioni, le istanze precedenti vengono cancellate e sostituite da nuove.

Inoltre, il codice può essere avviato anche direttamente tramite una chiamata HTTP o una richiesta del client.

In quest'ultimo caso vi è anche la possibilità di recuperare tutte le informazioni relative all'utente che effettua la chiamata, loggato attraverso il modulo Firebase Authentication.

Firebase Cloud Storage

Firebase Cloud Storage consente alle applicazioni Firebase l'accesso al servizio di Cloud Storage di Google. In questo modo è possibile effettuare upload e download di file in modo sicuro, indipendentemente dalla qualità della rete.

Il sistema rimane robusto anche in caso di perdite di connessione: semplicemente le operazioni vengono messe in pausa e ricominciano appena il dispositivo torna online.

Firebase Hosting

Firebase Hosting è un servizio di web hosting utilizzato per l'allocazione di file come CSS, HTML, JavaScript e altro. I dati vengono inviati tramite una CDN (Content Delivery Network) utilizzando HTTPS e SSL.

È inoltre possibile specificare delle regole per la gestione della memoria cache nel dispositivo che scarica il sito.

Firebase Cloud Messaging

Firebase Cloud Messaging è un servizio che consente di inviare notifiche ai client connessi.

I messaggi possono essere indirizzati a un singolo dispositivo o a gruppi.

1.2.3 Google Maps Platform

Google Maps Platform^[9] è un insieme di API e SDK che consentono agli sviluppatori di integrare le funzionalità di Google Maps in applicazioni e pagine web.

Sono disponibili diversi servizi, combinabili tra loro a seconda delle necessità degli sviluppatori.

Maps SDK, ad esempio, è il toolkit che consente di inserire delle MapView all'interno di applicazioni e siti web.

Il servizio si occupa in modo automatico di accedere ai server di Google Maps, scaricare i dati delle mappe e gestire le gestures.

È inoltre possibile personalizzare l'estetica della mappa (tramite un file JSON) e inserire degli elementi grafici come indicatori, linee, poligoni e immagini.

Infine, vengono incluse le funzionalità di geolocalizzazione tramite GPS, WiFi e celle telefoniche.

A tutto ciò si aggiungono altri servizi come Directions API per il calcolo di itinerari oppure Places API per implementare un sistema di ricerche.

1.2.4 OpenCage API

OpenCage API^[10] è un servizio RESTful di forward e reverse geocoding, nato nel 2013.

Un forward geocoder è in grado di individuare una coppia di coordinate geografiche a partire da un input testuale, come un indirizzo o un luogo.

Al contrario, un reverse geocoder converte le coordinate ricevute in input nel nome o nella descrizione di una località.

OpenCage in realtà è una API aggregativa, quindi, come mostra la figura 2, si appoggia a diversi altri servizi, in grado di svolgere la funzione appena descritta, e ne unisce gli output.

Quando viene effettuata una ricerca, delle query sono inviate a vari servizi backend (Yahoo GeoPlanet e Flickr Shapefiles tra gli altri) per poi fornire i risultati migliori tra quelli trovati.

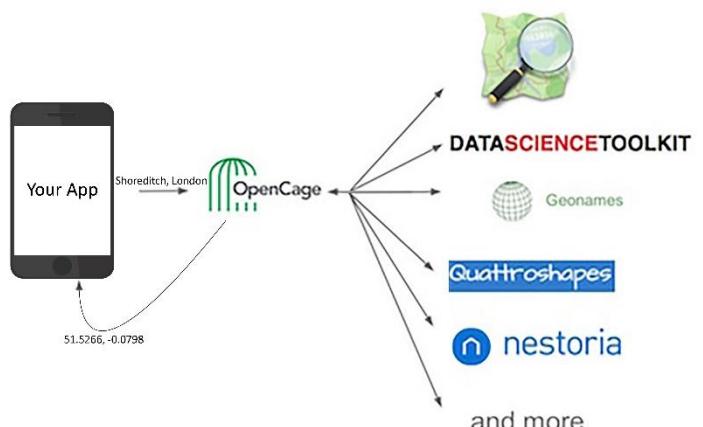


Figura 2
Schema di funzionamento del forward geocoding di OpenCage

Le richieste all'API vengono effettuate tramite il metodo GET di HTTP. Il formato di risposta della chiamata può essere scelto tra JSON o XML.

Nella chiamata, è possibile specificare se si vuole che i risultati non vengano deduplicati (parametro ‘no_dedupe’) oppure se si desiderano informazioni aggiuntive sui luoghi (parametro ‘no_annotation’).

Si può poi impostare la lingua (parametro ‘language’) e settare un numero massimo di luoghi da restituire (parametro ‘limit’).

Nel caso di una richiesta di forward geocoding, l'output potrebbe riportare più di un risultato. In questo caso le località sono ordinate in ordine di rilevanza.

2 Funzionalità e architettura

2.1 Funzionalità

L'applicazione presenta varie funzionalità e alcune di queste sono accessibili solo da un account amministratore, ovvero che identifica un ristoratore proprietario di un gioco.

Di seguito una lista delle principali funzionalità realizzate:

Registrazione e Login

È possibile registrarsi sia tramite e-mail, sia tramite account Facebook. Nella schermata di registrazione tramite e-mail è richiesto l'inserimento di una e-mail valida, che rispetti quindi il pattern di un indirizzo di posta elettronica generico, e l'inserimento di una password superiore a 6 caratteri da scrivere allo stesso modo in due campi distinti.

È presente anche un sistema di validazione dell'indirizzo e-mail dove l'utente, per completare la registrazione, deve premere il link presente nel corpo della mail che gli è stata spedita.

Nel caso in cui un account non sia stato ancora validato viene mostrato, nella pagina del profilo, questa informazione e compare in evidenza un tasto che permette all'utente di rimandare l'e-mail di verifica.

Nella schermata di login è possibile richiedere il reset della password del proprio account, confermando la propria e-mail nel campo indicato e seguendo la procedura raggiungibile dal link presente nel corpo della mail mandata all'utente in questione.

L'utente non è obbligato a registrarsi, inizialmente può saltare questa fase e visitare l'app in modo preliminare, senza però avere accesso ad alcune delle principali funzionalità, come entrare nelle sezioni 'premi' o 'profilo' oppure giocare ad un qualsiasi gioco. Il tentativo di accedere a queste funzionalità porterà il sistema a chiedere all'utente di loggarsi.

Ricerca tramite keyword

Si può effettuare una ricerca nell'apposita barra inserendo il nome di un gioco o di un ristorante oppure specificando un luogo.

Tale ricerca è ordinata sulla base un filtro apposito (rilevanza, premi) che viene salvato ogni volta che l'utente ci interagisce e in principio viene impostato con un valore di default.

I giochi individuati sono poi mostrati sulla mappa e nella apposita scheda dei risultati.

Trova intorno a me

Si può effettuare una ricerca relativa ai ristoranti nei dintorni, senza precisare nessuna keyword, basata sui filtri specificati dall'utente.

I filtri vengono salvati ogni volta che l'utente ci interagisce e in principio vengono impostati con dei valori di default. In particolare, è possibile specificare la distanza (da 10 a 50 chilometri) e l'ordinamento dei risultati (distanza, rilevanza, premi).

I giochi individuati sono poi mostrati sulla mappa e nella apposita scheda dei risultati.

Cerca qui

Si può effettuare una ricerca relativa ai soli ristoranti situati nella zona che l'utente sta guardando nella mappa.

I giochi individuati sono poi mostrati sulla mappa e nella apposita scheda dei risultati.

Mostra dettaglio gioco

Premendo sui risultati di una ricerca o ad un elemento nella lista dei giochi recenti è possibile accedere alla pagina dettagliata relativa al gioco selezionato. Vengono mostrati nome e breve descrizione del gioco, i dettagli sul ristorante e i premi in palio (inclusi di dettaglio e scadenza).

I dettagli vengono scaricati dal database in fase di ricerca oppure quando aperti da un gioco recente.

Mostra classifica gioco

Premendo l'apposito pulsante nella schermata di dettaglio di un gioco, oppure nella sezione delle classifiche utente presente nella pagina dedicata al profilo, è possibile visionare la classifica relativa a un gioco.

Quest'ultima è divisa in due categorie: la classifica settimanale (sono mostrati i migliori 7 giocatori della settimana) e quella globale (sono mostrati i migliori 3 giocatori di sempre). Viene infine indicato, per ciascuna delle due classifiche, la posizione occupata dall'utente.

Visita sito

Premendo l'apposito pulsante nella schermata di dettaglio di un gioco è possibile visitare il sito internet del ristorante.

Mostra marker sulla mappa

Premendo l'apposito pulsante nella schermata di dettaglio di un gioco è possibile mostrare il marker del ristorante associato sulla mappa.

Accesso al gioco

Premendo il pulsante 'Gioca' viene avviato il gioco selezionato.

Se l'utente non ha verificato l'indirizzo e-mail viene invitato a farlo prima di poter giocare.
Se l'utente non ha effettuato il login all'applicazione, gli viene richiesto di farlo.

Se l'utente non ha ancora un nickname gli viene richiesto di inserirne uno.

Mostra premi vinti

Accedendo alla pagina dedicata ai premi vinti dall'utente, è possibile prenderne visione e controllarne i dettagli (tipologia di premio, codice QR, scadenza).

Giochi recenti

I giochi giocati dall’utente vengono memorizzati: se l’utente ha già avviato un gioco, questo sarà mostrato nell’apposita scheda nella home e avrà la possibilità di iniziare una sessione di gioco rapidamente senza avviare una ricerca o passare dalla schermata di dettaglio.

I giochi vengono memorizzati fino ad un massimo di 10 elementi, oltre questo limite la lista si comporterà come una FIFO (first in first out).

Modifica nickname

Nella pagina dedicata al profilo utente è possibile modificare il nickname che si era impostato precedentemente o inserirne uno se questo non era presente.

Nel caso di login tramite Facebook verrà impostato in automatico il nome completo del profilo come nickname. Inoltre, il sistema impedisce all’utente di utilizzare un nickname già usato da qualcun altro.

Notifiche push

Quando l’utente vince un nuovo premio viene avvisato tramite una notifica push che, premuta, porta direttamente alla pagina dei premi vinti dall’utente.

Tale funzionalità può essere disattivata dalla scheda di impostazioni, accessibile dalla sezione dedicata al profilo utente.

Contatta il team

Nella scheda delle impostazioni, accessibile dalla sezione dedicata al profilo utente, è possibile contattare, per e-mail, il team degli sviluppatori premendo sull’apposito tasto.

Esci dall’account

Nella scheda delle impostazioni, accessibile dalla sezione dedicata al profilo utente, è possibile uscire dal proprio account premendo sull’apposito tasto.

Quando il processo di logout è completato l’utente viene rimandato sulla pagina principale senza però avere accesso ad alcune delle principali funzionalità, come entrare nelle sezioni ‘premi’ o ‘profilo’ oppure giocare ad un qualsiasi gioco. Il tentativo di accedere a queste funzionalità porterà il sistema a chiedere all’utente di loggarsi.

Riscatta codice (Solo account amministratore)

Nella sezione dedicata al gioco di cui si è proprietari, presente nella pagina del profilo utente, è possibile scannerizzare il codice QR mostrato da un cliente premendo sull’apposito tasto.

Al termine della scansione viene mostrato lo stato della richiesta, distinguibile in: scaduto, già usato, valido. Nei primi due casi la richiesta restituisce anche le date interessate, mentre nello stato ‘valido’ viene incluso il dettaglio del premio appena riscattato.

Mostra partite giocate (Solo account amministratore)

Nella sezione dedicata al gioco di cui si è proprietari, presente nella pagina del profilo utente, è possibile visionare il numero totale di partite giocate al relativo gioco.

2.2 Architettura

Al fine di poter realizzare una piattaforma dinamica e performante è stato scelto di sviluppare tutti i videogiochi in HTML5 e Javascript, tramite il programma Construct 2, e integrarli nell'app come Webview attraverso Apache Cordova. L'integrazione con questo framework permette, mediante l'uso di plugin personalizzati, la comunicazione tra le pagine web e l'ambiente nativo Java.

L'applicazione basa gran parte del suo funzionamento sull'interazione col servizio esterno Firebase, offerto da Google. In particolare, per attingere ai dati sono stati utilizzati i database Firestore e Realtime Database, soluzioni NoSQL che possiedono diversi scopi di utilizzo, e per l'archiviazione dei contenuti multimediali è stato usato Firebase Cloud Storage.

La manipolazione dei dati e in generale le operazioni più dispendiose a livello computazionale sono state messe a carico del modulo Firebase Functions.

I videogame sono stati caricati sul servizio di hosting per pagine statiche Firebase Hosting che dispone di particolari opzioni per la gestione della cache nei dispositivi, tra cui la possibilità di scaricare le pagine web soltanto quando queste non sono già presenti nella memoria dell'app.

La gestione delle utenze è affidata al modulo Firebase Authentication che integra anche il login tramite Facebook, mentre il sistema di notifiche push, inviate quando un utente vince un premio, è implementato tramite Firebase Cloud Messaging.

Per quanto riguarda le mappe, anche qui si è optato per un prodotto Google: attraverso le Google Maps Api è stato possibile usufruire sia delle mappe, sia del servizio di geolocalizzazione. Infine, si è scelto di utilizzare OpenCage, una Api di geocoding, per implementare nell'app ricerche tramite l'inserimento di una specifica località.

2.2.1 I motivi delle scelte

La scelta di utilizzare Firebase come principale servizio di backend è stata dettata da numerosi fattori, molti dei quali determinanti per un mantenimento e uno sviluppo longevo nel tempo.

Di seguito un elenco dei principali motivi:

- La **quantità di moduli** offerti dal servizio, necessari allo sviluppo dell'applicazione.
- Disponibilità di **librerie client** per integrarsi facilmente sia con Android che con IOS, semplificando la futura operazione di porting sul sistema operativo di Apple.
- Capacità di **sincronizzazione dei dati** oltre che di storage. L'app può recuperare i dati e restare in ascolto delle modifiche in tempo reale.
- I dati immagazzinati possono essere soggetti a delle particolari regole di **sicurezza** progettate dallo sviluppatore. La comunicazione con i client avviene sempre in modalità crittografata tramite SSL con certificati a 2048-bit.
- **Costi differenziati** in base all'uso e alle capacità richiesti. Il servizio di base, con alcune limitazioni, è totalmente gratuito così da poter permettere l'intera fase di sviluppo senza spendere nulla.

L'utilizzo del framework Cordova per eseguire i giochi permette non solo un'integrazione dinamica dei giochi, che quindi vengono scaricati soltanto quando richiesto, ma anche una precisa gestione della sicurezza. Cordova offre la possibilità di implementare regole di whitelist sugli indirizzi web che richiedono il servizio, così da bloccare i siti non autorizzati che avrebbero la possibilità di compromettere il sistema.

La scelta di utilizzare entrambe le tipologie di database offerte da Firebase (Firestore e Realtime Database) è motivata dalla necessità di usufruire di alcuni aspetti specifici di ognuna delle due basi di dati. Firestore possiede il tipo di dato geopoint che, attraverso una libreria apposita, consente l'uso del geohashing per effettuare ricerche performanti e veloci. Realtime Database, invece, permette di ricevere specifici dati in maniera istantanea, semplicemente specificando il percorso, e di restare in ascolto delle modifiche su quel nodo.

Si è usufruito di OpenCage in quanto predispone la possibilità di avere una chiave gratuita sfruttabile per usare i servizi di geocoding senza impegno di costi. Chiaramente vengono imposte le dovute limitazioni all'uso, ma non troppo strette per una versione di test.

Viene mostrato di seguito nella figura 3 lo schema riassuntivo di tutta l'architettura dell'applicazione:

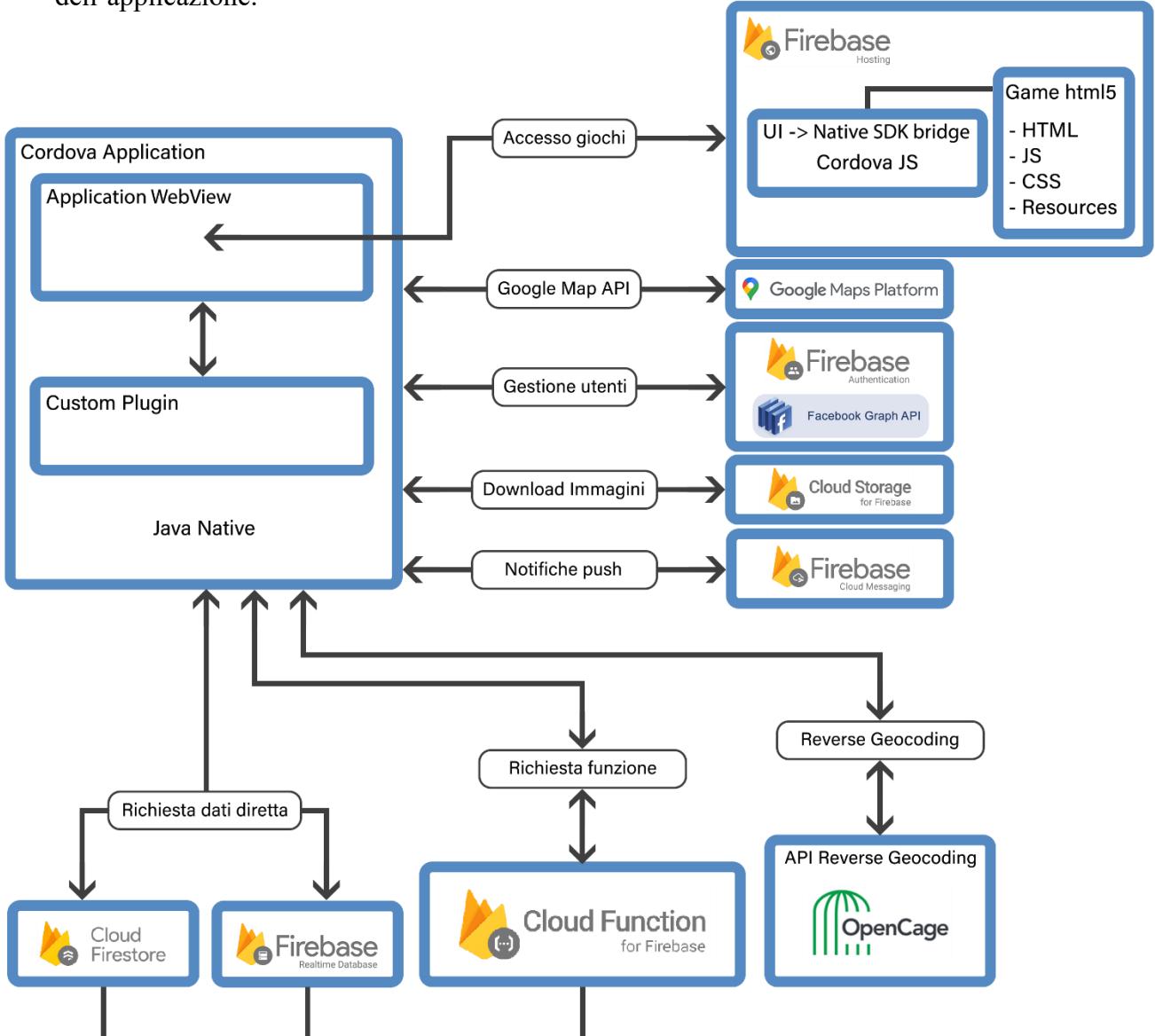


Figura 3
Schema dell'architettura dell'applicazione

3 Progettazione

3.1 Firestore

Come sappiamo Firestore è un database NoSQL document-oriented, abbiamo quindi un insieme di raccolte di documenti, all'interno dei quali si possono inserire vari tipi di dati. In questo database sono state archiviate le sole informazioni relative ai ristoranti e i premi erogati per ogni ristorante.

Non potendo rappresentare il modello di database attraverso lo schema entità-relazione si è scelto di utilizzare il tool Hackolade per costruire i diagrammi e spiegarli nel dettaglio in seguito. Le raccolte sono state strutturate secondo quanto mostrato dalla figura 4:

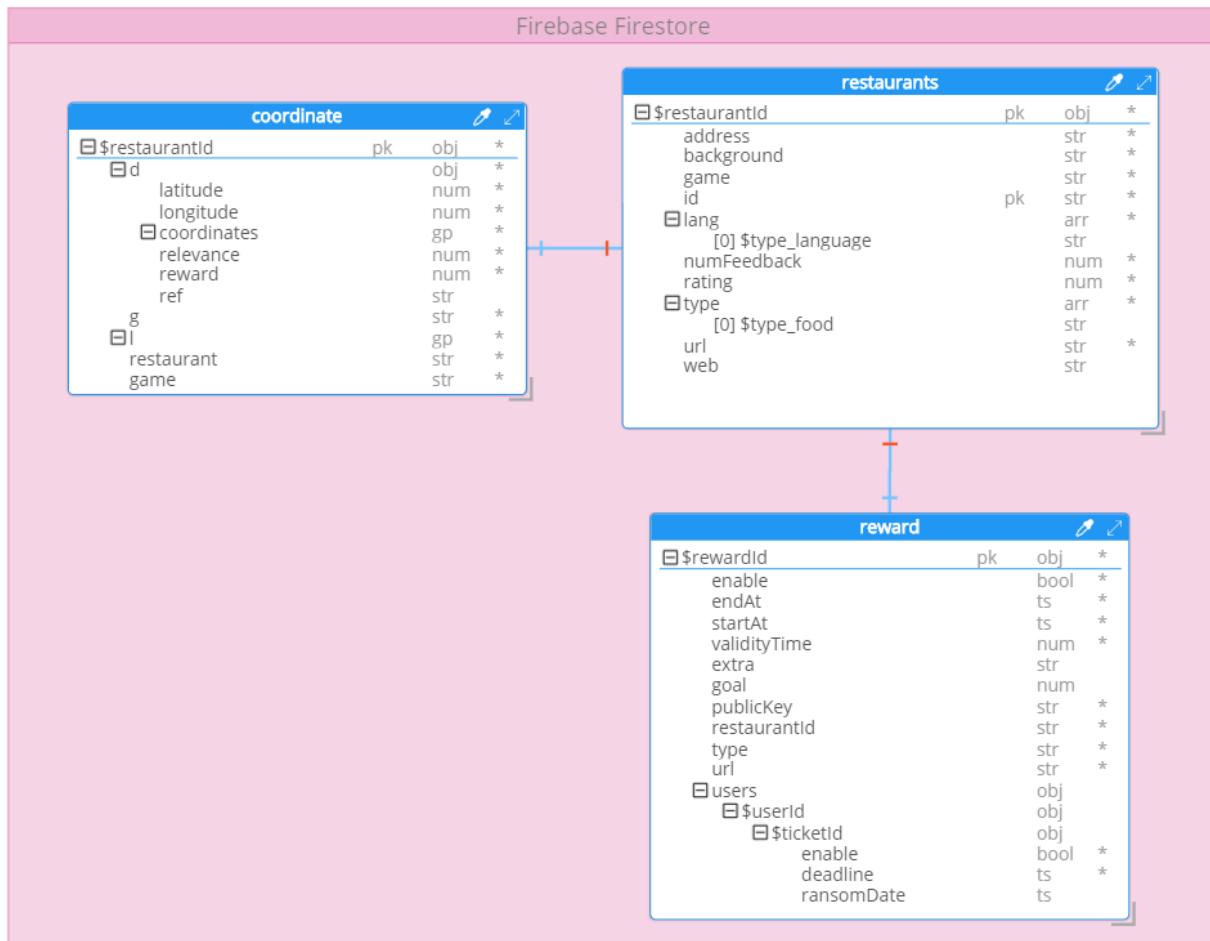


Figura 4
Diagramma strutturale dei dati in Firestore

Ogni raccolta principale è figurata da una tabella e le relazioni tra le tabelle indicano gli accoppiamenti tra i nomi di specifici nodi o del loro contenuto. Gli oggetti principali presenti in ogni tabella figurano la struttura di un qualsiasi documento contenuto nella relativa collezione.

Coordinate

La raccolta delle coordinate presenta una lista di documenti, i quali rappresentano ogni ristorante presente nell'app. I nomi di questi documenti sono infatti gli identificatori dei ristoranti e possiedono vari attributi. I campi ‘restaurant’ e ‘game’ servono per effettuare i match diretti quando si effettua una ricerca dalla barra di ricerca inserendo il nome del ristorante o del gioco. Gli altri campi (g, l, d) sono necessari alla libreria GeoFirestore per effettuare la ricerca tramite gli algoritmi di geohashing. In particolare:

- g è la stringa che contiene l'hash di 10 cifre delle coordinate del luogo
- l è l'oggetto geopoint del posto in questione, formato da latitudine e longitudine
- d sono tutti i dati a cui si può attingere nella ricerca

Nel nodo d sono presenti varie informazioni importanti per il funzionamento di alcune funzionalità. L'ordinamento dei risultati durante una query è possibile grazie ai campi di ‘reward’ e ‘relevance’, che identificano relativamente il numero di premi disponibili per quel ristorante e il numero di partite giocate al gioco correlato.

Infine, il campo ref contiene la referenza al documento nella raccolta ‘restaurants’ che identifica quel luogo.

Restaurants

La raccolta dei ristoranti incorpora tutte le informazioni più dettagliate per ogni ristorante ed ogni documento ha come nome l'identificatore del locale. La relazione con le due tabelle presuppone che tale id sia utilizzato nei documenti della raccolta delle coordinate, per rappresentare il luogo in questione, e che sia contenuto nel campo ‘restaurantId’ presente in un documento della raccolta dei reward.

Il campo background è una stringa che rappresenta il codice esadecimale di 6 cifre del colore primario presente nel logo del ristorante.

L'array ‘lang’ contiene una lista di lingue nel formato da 2 lettere (es. IT). La lingua garantita per ogni ristorante è l'inglese (EN), che quindi è inserita di default. Invece, l'array ‘type’ contiene tutti i tag del tipo di cucina del ristorante, mostrati poi nella scheda di dettaglio nell'app.

La stringa ‘url’ riporta l'id del gioco correlato, in quanto viene utilizzato come percorso, in seguito all'origine del sito in cui vengono depositati tutti i giochi, per accedere al gioco web. Anche in questo caso la relazione certifica che questo dato sia lo stesso usato nell'attributo ‘url’, citato nei documenti della collezione dei premi.

Il campo ‘web’ indica il sito internet del locale ed è impostato opzionale, in quanto è possibile che un locale non disponga di tale informazione.

Reward

L'ultima raccolta da discutere è quella dei reward, la quale raccoglie tutti i premi di ogni ristorante. Ogni documento in questa collezione è rappresentativo quindi di un particolare premio e il suo nome ne indica l'identificativo.

Il reward può essere attivo o disattivo sulla base del campo ‘enable’ e possiede i timestamp di inizio (startAt) e fine (endAt). Il campo ‘validityTime’ è il numero di millisecondi di validità del ticket erogato quando un utente vince il relativo premio, sulla quale si calcola la rispettiva scadenza.

Il campo ‘type’ spiega il tipo di premio associato (es. Weekly) e la stringa extra dà la possibilità di creare varie versioni dello stesso tipo di reward, andando a scaricare diversi termini di utilizzo (reperibili nella sezione stringhe nel Realtime Database) sulla base di questo dato. Ad esempio, se extra vale “versione1” e type vale “Weekly” nel momento di ricezione dei termini di utilizzo del premio il sistema andrà a reperire i dati relativi a “reward_Weekly_versione1”, senza però andare a cambiare la dicitura della tipologia mostrata all’utente, che resterà “Weekly”.

Il premio viene vinto in caso di superamento del punteggio indicato dal numero presente in ‘goal’ se si tratta di un premio non di tipo ‘Weekly’, altrimenti questo numero non viene inserito.

I vincitori dei rispettivi premi sono contenuti nell’oggetto users, seguendo la regola che per un premio di tipo ‘Weekly’ sia possibile per un utente vincere più volte la stessa promozione, quindi avere più ticket, mentre per un premio di tipo diverso non sia possibile la rivincita dello stesso. Ogni ticket erogato è caratterizzato da una data di scadenza (deadline) e un valore booleano di validità (enable). Inoltre, nel momento del riscatto si aggiunge in questo percorso la data di utilizzo (ransomDate).

Infine, il valore ‘PublicKey’ riporta la chiave pubblica del ristorante, utilizzata per la criptazione del codice sconto da associare ad un ticket nel processo di creazione.

3.1.1 Regole di lettura/scrittura

Le regole programmabili in Firestore sono state scritte in modo che la scrittura e la lettura rispettino i vincoli, relativi a specifici percorsi, spiegati nella tabella 1.

	PERCORSO	VINCOLI
SCRITTURA	/*	False
LETTURA	/restaurants	if request.auth.uid != null (l’utente è loggato)
	/coordinates	if request.auth.uid != null (l’utente è loggato)

Tabella 1
Descrizione delle regole di lettura e scrittura in Firestore

3.2 Realtime Database

Realtime Database è stato utilizzato per archiviare tutti i dati relativi ai giochi (classifiche, informazioni), agli utenti e alle stringhe nelle diverse lingue supportate. Il database NoSQL è stato rappresentato, come per la tipologia precedente, attraverso il tool Hackolade.

Il database è quindi stato strutturato secondo quanto mostrato dalla figura 5:

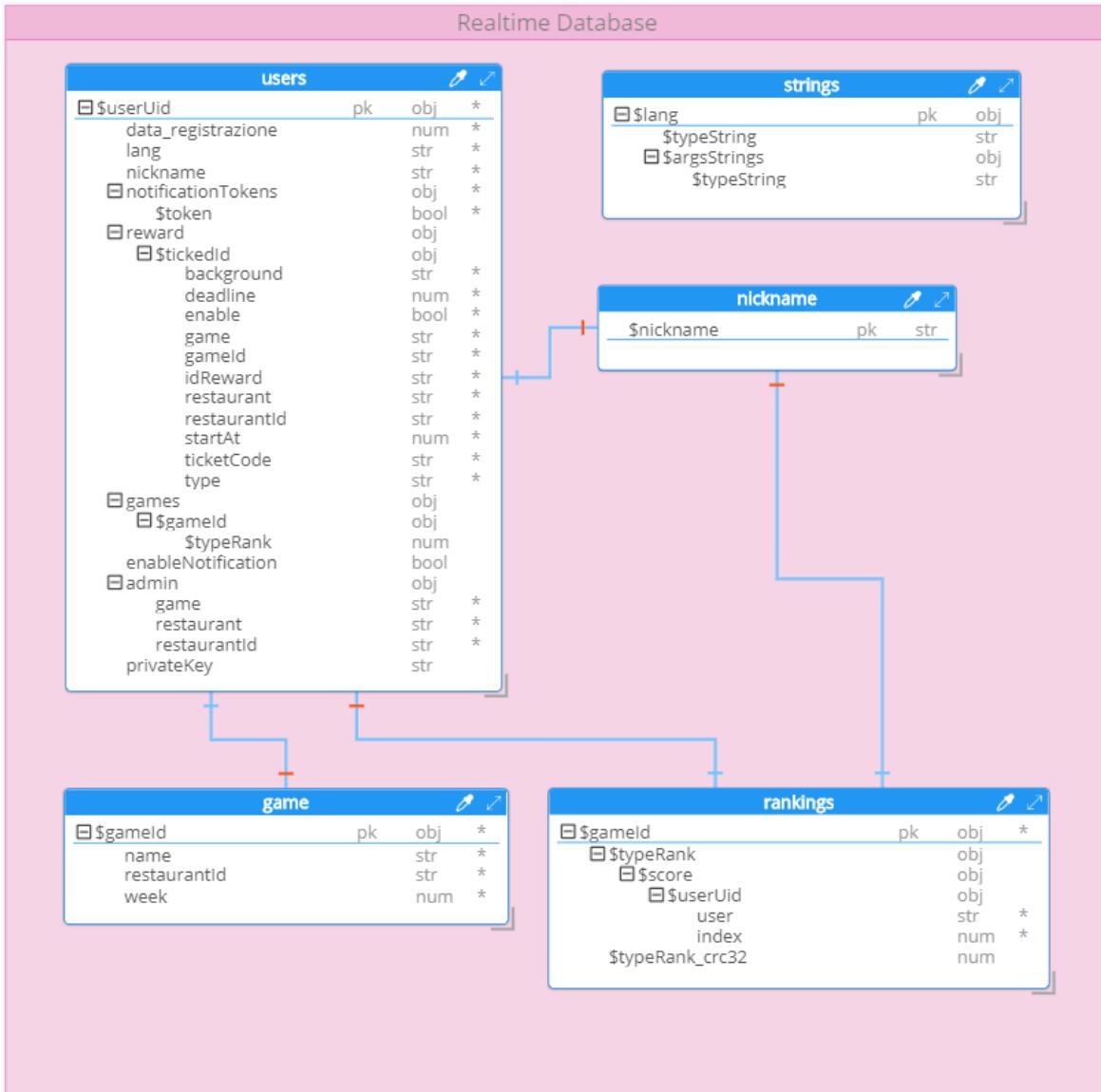


Figura 5
Diagramma strutturale dei dati in Realtime Database

In questo caso ogni nodo principale è figurato da una tabella e i collegamenti tra i nomi di particolari nodi o dei loro contenuti, spiegati in seguito, sono rappresentati da relazioni. Gli oggetti presenti in ogni tabella coincidono con gli eventuali sottonodi inglobati nel genitore.

Users

Il nodo users contiene tutti gli utenti registrati alla piattaforma, ogni profilo è caratterizzato da un nodo nominato con l'UID dell'utente in questione.

Il sistema salva la data di registrazione degli utenti e aggiorna dinamicamente, nel campo 'lang', la lingua del dispositivo sulla quale sta girando l'app, in modo da poter inviare le notifiche all'utente tradotte con l'ultima lingua segnalata all'apertura della piattaforma.

Le notifiche sono rese possibili dalla creazione dei notification token inseriti dinamicamente nell'apposito attributo. L'invio delle notifiche viene abilitato a seconda del valore booleano nel campo ‘enableNotification’. Tale attributo non è presente di default e questo aspetto è equivalente alla sua impostazione su ‘true’.

L'oggetto ‘reward’ contiene tutti i ticket vinti dall'utente relativo al nodo padre. Ogni ticket è caratterizzato da un id, usato come nome dell'elemento in questione, e da vari attributi:

- ‘background’ è la copia del dato presente nel documento del reward corrispondente in Firestore. Questo dato è utilizzato in caso di visualizzazione di un premio vinto, nella pagina dei premi, per impostare il colore di sfondo della schermata sulla base del ticket che si sta osservando.
- ‘enable’ indica lo stato del ticket, in caso di utilizzo o scadenza questo valore booleano si imposta falso.
- ‘startAt’ e ‘deadline’ contengono rispettivamente i timestamp in millisecondi della data di erogazione e di scadenza del ticket
- ‘game’, ‘gameId’, ‘restaurant’ e ‘restaurantId’ sono le relative informazioni del gioco e del ristorante al quale il premio si riferisce.
- ‘type’ presenta la dicitura del tipo di premio (es. Weekly) che si è vinto.
- ‘ticketCode’ contiene il codice necessario per il riscatto del buono.

Il campo ‘admin’ indica il diritto di un account ad avere le funzionalità aggiuntive necessarie ad un ristoratore proprietario di un gioco, infatti questa impostazione può essere inserita soltanto manualmente. All'interno di questo nodo ci sono tutte le informazioni del gioco e del ristorante di cui l'utente rivendica la proprietà. Inoltre, per poter riscattare i codici sconto, un account amministratore necessita anche della stringa ‘privateKey’, che indica una chiave unica richiesta per la decriptazione dei codici esposti dai suoi clienti.

Infine, il campo ‘games’ salva tutte le informazioni dei record raggiunti dall'utente nei giochi alla quale ha interagito. Viene salvato per ogni gioco un oggetto, nominato col rispettivo id, che possiede tutti i massimi punteggi raggiunti nelle specifiche categorie. In particolare, vi sono dei dati numerici contenenti il punteggio record, nominati con la categoria della classifica associata. Nel caso di classifica globale questa sarà chiamata ‘global’, mentre per le classifiche settimanali varrà ‘weekly’ seguito dal numero della settimana a cui si riferisce. Gli indici contenuti in questo percorso sono appaiati ad alcuni tra quelli contenuti nel nodo ‘ranking’. Nel dettaglio, i nomi interessati sono ‘\$gameId’, ‘\$typeRank’ e ‘\$userUid’, inoltre il nome del nodo ‘\$score’ in ‘rankings’ prende il valore del nodo ‘\$typeRank’ relativo presente nelle informazioni utente.

Rankings

Il nodo ‘rankings’ tiene memoria di tutte le classifiche per ogni gioco presente sulla piattaforma. Tutti gli oggetti principali in questo percorso possiedono come nominativo l'id del gioco a cui si riferiscono le classifiche contenute.

Tali classifiche sono costruite con un modello nidificato che segue i seguenti argomenti dall'alto verso il basso:

- Tipo di classifica interessata.
- Punteggio raggiunto per quella determinata classifica.
- Utente che ha raggiunto quel punteggio in quella determinata classifica.

L’utente che si trova nell’ultimo punto della lista appena stilata è rappresentato da un oggetto, nominato con l’UID di proprietà del profilo in questione, contenente un indice numerico e il nickname dell’utente, contenuto nell’attributo ‘user’. L’indice in questione ha lo scopo di tenere traccia della tempistica di inserimento in questa particolare posizione di punteggio (maggiori informazioni al capitolo 4.3.5).

Per ogni tipo di classifica c’è in parallelo, nello stesso percorso, un corrispettivo attributo numerico contenente il checksum, calcolato con l’algoritmo crc32, delle prime posizioni di quella particolare graduatoria (maggiori informazioni al capitolo 4.3.2).

Nickname

Ogni utente ha l’obbligo di avere un nickname unico per poter avviare una partita di gioco e quindi questi attributi sono elencati nel nodo ‘nickname’, così da poter verificare l’unicità degli stessi in modo rapido.

In questo percorso sono presenti solo stringhe, strutturate in modo da avere come nome il nickname di un account e come contenuto l’UID dell’utente possessore dello pseudonimo.

Tale nickname deve corrispondere sia all’attributo presente nel nodo ‘users’ per l’account a cui si riferisce, sia al valore del campo ‘user’ dell’elemento utente presente in ogni classifica se riferito al profilo in oggetto.

Game

Il nodo ‘game’ raccoglie le informazioni di tutti giochi, caratterizzati da oggetti aventi come nome l’identificatore associato e vari attributi. Tale id viene replicato nel nodo utenze nella sezione dedicata ai punteggi raggiunti in quel determinato gioco, per chiunque ne abbia collezionato almeno una partita.

L’attributo ‘name’ indica la dicitura mostrata all’utente per rilevare quel gioco, mentre il campo ‘restaurantId’ contiene l’id del ristorante a cui è correlato.

Infine, il dato numerico ‘week’ tiene il conteggio delle settimane che contraddistinguono tutte le relative classifiche. Il dato ‘\$typeRank’ presente nella tabella ‘users’, quando si riferisce alla tipologia settimanale, prende il valore ‘weekly’ seguito dal numero di settimana reperito in questo contesto, quando è riferito al medesimo gioco ed è appena stato creato o aggiornato, in seguito al miglioramento del record personale.

Strings

In questa particolare sezione sono contenute tutte le stringhe che necessitano il deposito sul database. L’oggetto principale prende il nome della lingua disponibile nel formato 2 lettere (es. EN) e può contenere essenzialmente due tipologie di sottostrutture:

- Un solo dato di tipo stringa avente il nome come identificativo e contenente il valore della stringa ricercata.
- Un oggetto avente come nome un macro-argomento che accomuna tutte le stringhe contenute. Gli elementi inseriti in questo oggetto sono strutturati nel modo spiegato nel punto precedente.

3.2.1 Regole di lettura/scrittura

Le regole programmabili in Realtime Database sono state scritte in modo che la scrittura e la lettura rispettino i vincoli, relativi a specifici percorsi, spiegati nella tabella 2.

	PERCORSO	VINCOLI
SCRITTURA	/*	false
LETTURA	/users/\$uid/notificationTokens	auth !== null && \$uid == auth.uid (l'utente è loggato e la variabile 'uid' contiene l'identificativo relativo al proprio account)
	/users/\$uid/enableNotification	
	/users/\$uid/lang	
LETTURA	/users/\$uid	auth !== null && \$uid == auth.uid (l'utente è loggato e la variabile 'uid' contiene l'identificativo relativo al proprio account)
	/game/\$game	
	/rankings/\$game	if request.auth.uid != null (l'utente è loggato)
	/strings	

Tabella 2
Descrizione delle regole di lettura e scrittura in Realtime Database

I valori definiti nei percorsi della tabella 2 che iniziano col carattere ‘\$’ indicano delle varabili jolly, che possono contenere qualsiasi valore. Sulla base del percorso specificato in fase di richiesta, dalla parte del client, è possibile congiungersi con questi tragitti variabili e applicare le relative regole. I valori speciali possono poi essere ripresi nei vincoli semplicemente riutilizzando la medesima dicitura.

3.3 Applicazione

Conoscere i principi di buona progettazione è importante per creare un'applicazione fatta da codice pulito e modulare, che consenta l'estensibilità, la riusabilità dei componenti e la facilità di manutenzione.

L'applicazione è stata realizzata in modo da evitare codice duplicato, per renderlo più leggibile e facilmente aggiornabile.

L'implementazione delle classi tenta di avere il codice con:

- Elevata coesione, ovvero ogni modulo esegue un singolo compito.

Ogni componente è stato creato con un preciso scopo e sono state separate le responsabilità in base alla struttura dei vari package, visibile in figura 6.

Tutte le activity sono state raccolte nella directory principale ‘gameplate’, così da essere facilmente reperibili.

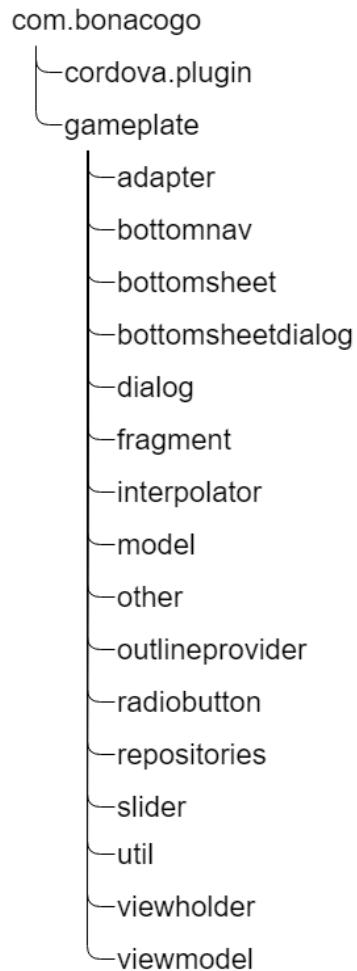
La raccolta ‘util’ raccoglie varie classi di uso generico e frequente in vari punti del progetto. Ad esempio, sono state inserite le classi contenenti metodi statici, molto utilizzati e con vari scopi, e classi con il fine di archiviare stringhe comunemente usate.

Il package ‘other’ ingloba tutti i componenti minori che non necessitavano di un contenitore proprio.

‘viewmodel’ e ‘repositories’ sono i contenitori delle classi incaricate al recupero e salvataggio dei dati che possono necessitare di un ripristino.

Tutti gli oggetti instanziabili che rappresentano particolari strutture dati sono raggruppati nella cartella ‘model’.

Le altre raccolte sono auto esplicative dei componenti che collezionano, per lo più grafici.



- Basso accoppiamento, basato sul grado di relazione tra i vari moduli.

Le comunicazioni tra i vari fragment non sono mai state effettuate direttamente, in quanto il principio in esame vieta questo tipo di implementazione.

Per poter far interagire i vari componenti tra loro sono stati utilizzati dei callback verso l'activity genitore. Una volta che l'activity viene associata al fragment come callback è possibile chiamare al suo interno un metodo, integrato come interfaccia, che ha quindi libero accesso a tutti i componenti di sua padronanza.

Viene illustrato nella figura 7 un esempio della procedura appena descritta:

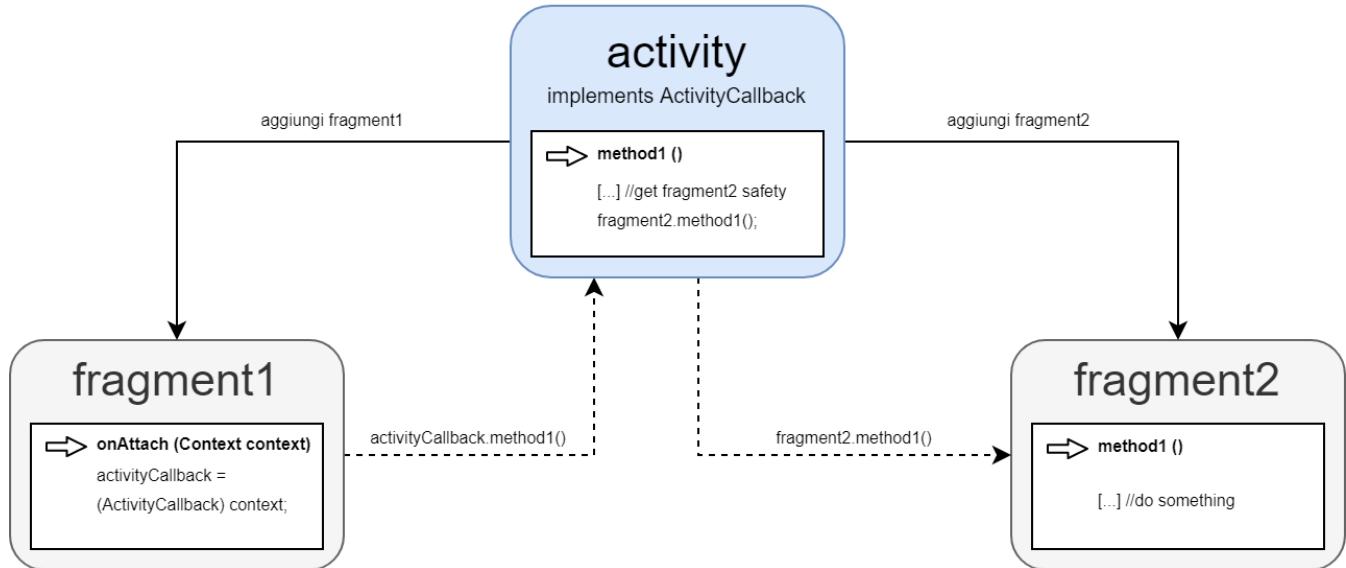


Figura 7
Diagramma di comunicazione tra due fragment

Andiamo quindi ora ad esporre per intero, nella figura 8, lo schema che rappresenta i collegamenti fra i vari fragment e activity nell'applicazione:

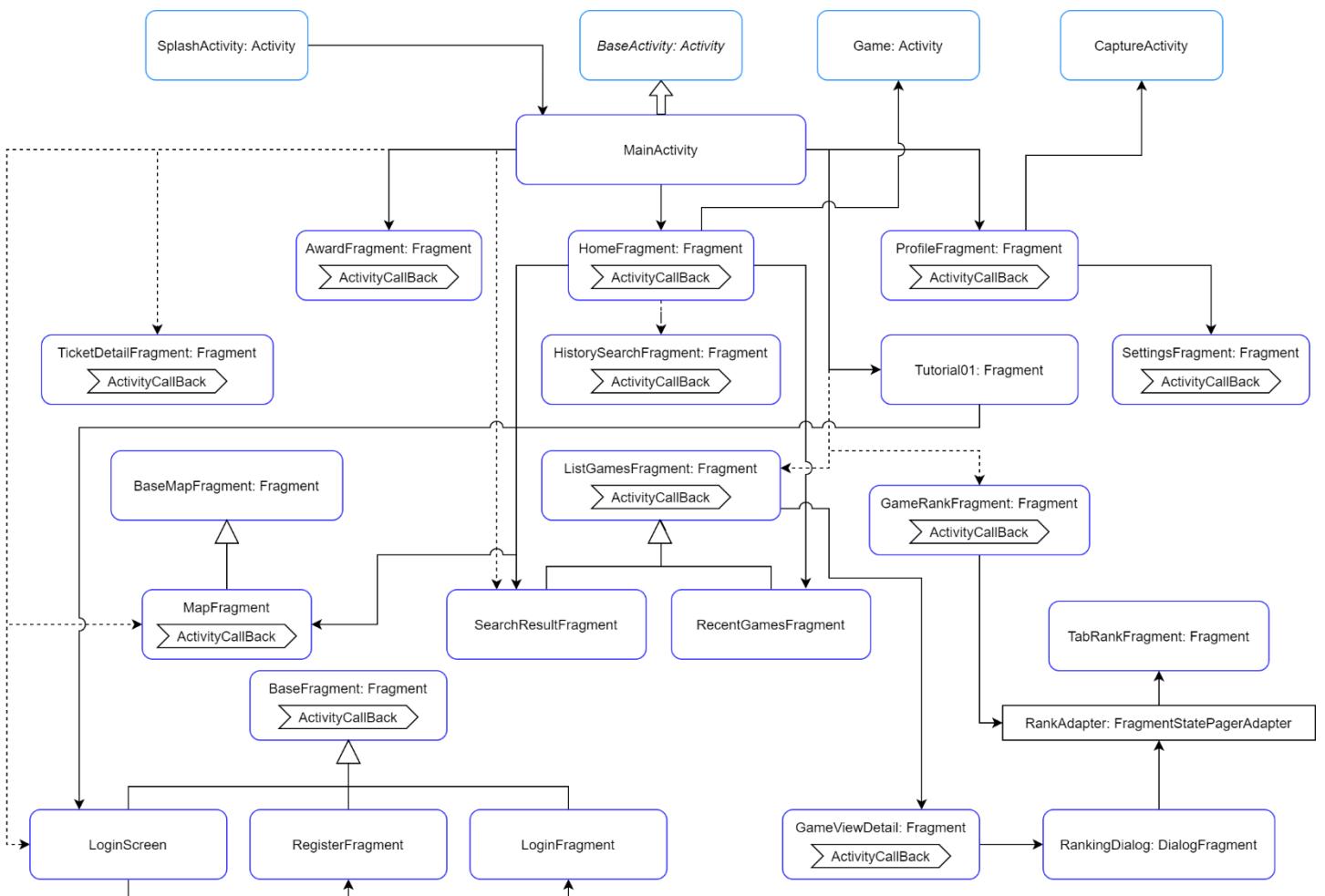


Figura 8
Schermata di comunicazione tra fragment e activity

4 Sviluppo

4.1 Gestione utenze

Come abbiamo già accennato in precedenza, le utenze vengono gestite da Firebase Authentication.

In questo capitolo andiamo ad analizzare, nel dettaglio, tutto quello che il sistema svolge per gestire gli aspetti più importanti degli account utente.

4.1.1 Registrazione

La prima fase di registrazione è possibile sia tramite e-mail, dall'apposita schermata, sia tramite provider Facebook.

Tramite e-mail

L'utente accede alla pagina dedicata alla registrazione, mostrata nella figura 9, e compila correttamente i campi che gli vengono richiesti. Viene effettuato un controllo preliminare, eseguito nel lato client, che certifica i seguenti punti:

- Tutti e tre i campi non devono essere lasciati vuoti.
- La e-mail inserita rispetta l'espressione regolare di un generico indirizzo di posta elettronica.
- Le due password richieste coincidono.
- La password scelta dall'utente è lunga almeno 6 caratteri.

Tutti i controlli vengono eseguiti uno dopo l'altro dall'alto verso il basso, segnalando con i rispettivi avvisi gli errori trovati. L'unica peculiarità di questa procedura è che nel caso in cui il primo punto non venisse rispettato il controllo non prosegue e quindi viene segnalato il problema all'utente.



Figura 9
Schermata di registrazione

Al passaggio del controllo preliminare entra in causa Firebase Authentication che si occupa della creazione dell'account. Tale processo può avere vari riscontri negativi, segnalati dal tipo di eccezione lanciata dal metodo in causa. Le eccezioni^[11] possibili sono le seguenti:

- **FirebaseAuthWeakPasswordException**, generato se la password non è abbastanza forte (minore di 6 caratteri).
- **FirebaseAuthInvalidCredentialsException**, generato se l'indirizzo e-mail non è valido.
- **FirebaseAuthUserCollisionException**, generato se esiste già un account con l'indirizzo e-mail indicato.

Se la procedura conclude con successo si viene a creare un account che predispone la proprietà relativa alla verifica dell'e-mail su ‘false’. Di conseguenza, il sistema invia una mail all'indirizzo di posta usato per la registrazione dall'utente, al cui interno è presente il link da premere per verificare il proprio indirizzo e-mail.

Tramite Facebook

La registrazione tramite Facebook è totalmente eseguita in automatico dall'implementazione del tasto di login del Facebook SDK collegato al sistema Firebase.

Per procedere con la creazione dell'account tramite Facebook si deve premere sul tasto ‘continua con Facebook’ visibile nella schermata dedicata nella figura 10.

Se la registrazione avviene con successo si viene a creare un account che eredita alcune informazioni dal profilo Facebook collegato. In particolare, viene impostato l'attributo ‘DisplayName’ con il nome e cognome dell'utente Facebook rilevato.

Infine, un aspetto importante di questa operazione è che Firebase non verifica la mail per gli utenti connessi con questo metodo, di conseguenza il relativo attributo viene impostato su ‘false’.

In questo contesto, a differenza del sistema precedente, che inviava una mail per verificare la veridicità dell'indirizzo, il problema viene trattato automaticamente dal backend, spiegato di seguito.

Backend

Sul lato server Firebase Function vi è una particolare funzione che viene lanciata all'evento di creazione di un utente. Questa implementazione permette di gestire le complicatezze che potrebbero sorgere dagli account registrati tramite provider Facebook.

In particolare, vengono gestite le seguenti tematiche:

- Inserimento nickname e gestione conflitti con quelli già presenti nella lista di utenti
- Impostazione della mail verificata e della data di registrazione



Figura 10
Schermata di accesso

Illustriamo nella figura 11 il diagramma che spiega la procedura della funzione, in cui la variabile ‘user’ coincide con l’oggetto contenente le informazioni dell’utente creato.

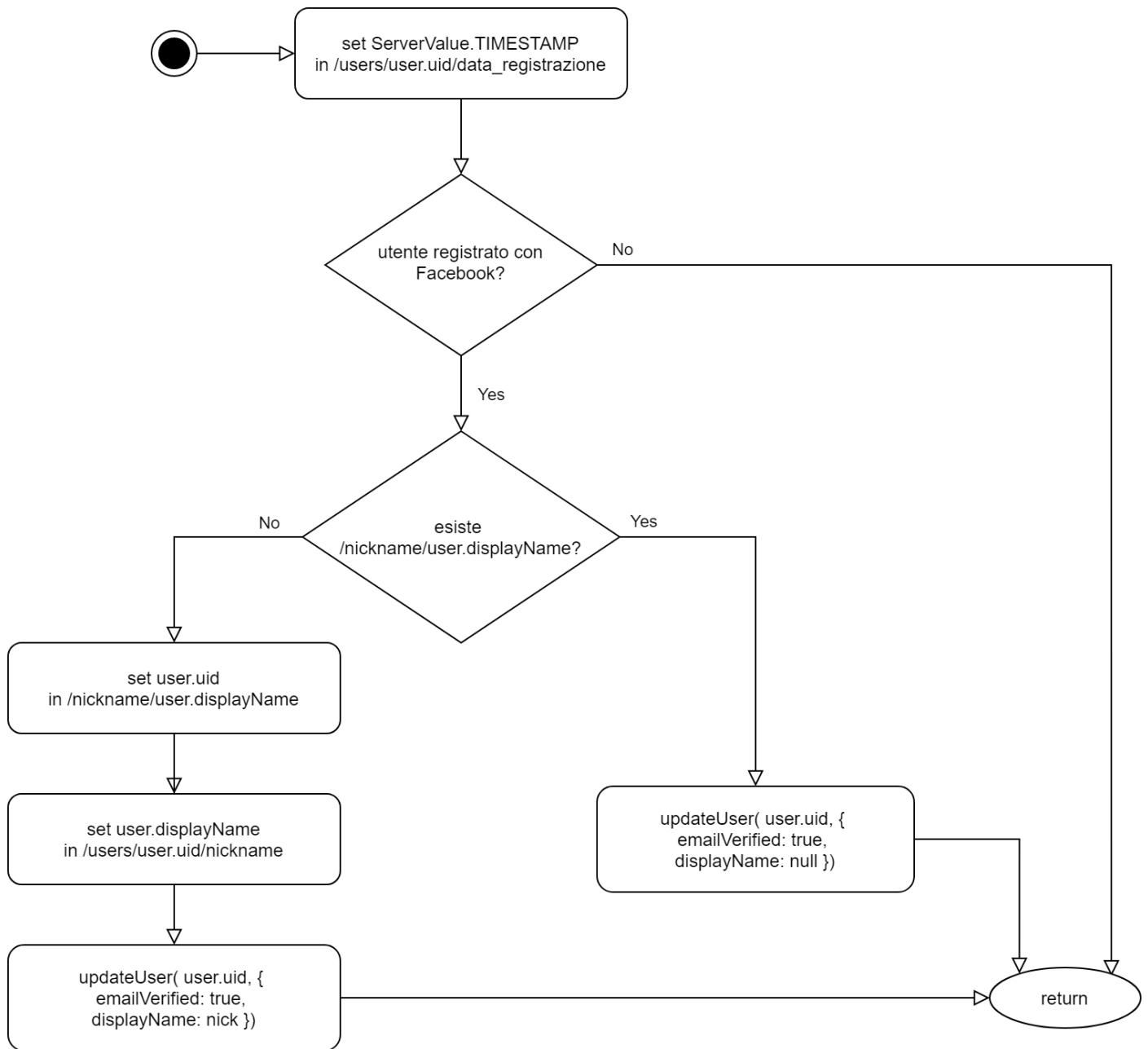


Figura 11
Diagramma della funzione backend lanciata alla creazione di un utente

4.1.2 Login

La funzionalità di login spiegata riguarda soltanto gli account registrati tramite e-mail, in quanto il processo tramite Facebook è integrato in automatico dal tasto di login.

Tramite e-mail

L’utente accede alla pagina dedicata al login, mostrata nella figura 12, e compila correttamente i campi che gli vengono richiesti.

Proprio come nello scenario di registrazione, anche qua viene effettuato un controllo preliminare, eseguito nel lato client, che certifica i seguenti punti:

- Tutti e due i campi non devono essere lasciati vuoti.
- La e-mail inserita rispetta l'espressione regolare di un generico indirizzo di posta elettronica.

Nel caso il primo controllo non passasse la procedura si interrompe e sagnala il problema all'utente.

Al passaggio del controllo preliminare entra in causa Firebase Authentication che si occupa del login dell'account. Tale processo può avere vari riscontri negativi, segnalati dal tipo di eccezione lanciata dal metodo in causa. Le eccezioni^[12] possibili sono le seguenti:

- **FirebaseAuthInvalidUserException**, generato se l'account utente corrisponde a un'e-mail non esistente o è stato disabilitato.
- **FirebaseAuthInvalidCredentialsException**, lanciato se la password è sbagliata.

Se il metodo di login conclude con successo viene aggiornata l'interfaccia in modo da avvertire l'utente del riscontro.

In caso l'utente non ricordi la propria password è possibile premere il tasto per il suo recupero, visibile nella figura 12. Verrà avviata una Dialog in cui si richiede all'utente la email correlata al proprio account che, una volta confermata, riceverà un messaggio di posta contenente il link per accedere alla procedura di recupero. Tutta l'implementazione del reset della password utente è gestita automaticamente da Firebase Authentication.

4.1.3 Stato account

Un account utente può assumere vari stati sulla base della completezza del processo di registrazione. Gli stati in questione sono:

- **Loading**, assunto quando un utente è registrato tramite Facebook e possiede l'attributo di verifica dell'e-mail impostato su ‘false’. Questo accade quando la funzione, illustrata nella figura 11, non ha ancora concluso il suo compito.
- **Not verified**, assunto quando un utente è registrato attraverso e-mail e non ha ancora completato la verifica, premendo il link inviatogli per posta.
- **Verified**, assunto in ogni altra casistica.

Nel caso in cui un account sia nello stato ‘Not verified’ si mostra una sezione aggiuntiva, nella pagina dedicata al profilo utente, dove viene notificata la situazione. Inoltre, in questa particolare sezione, vi è un tasto apposito per poter rimandare la mail necessaria alla soluzione del problema.

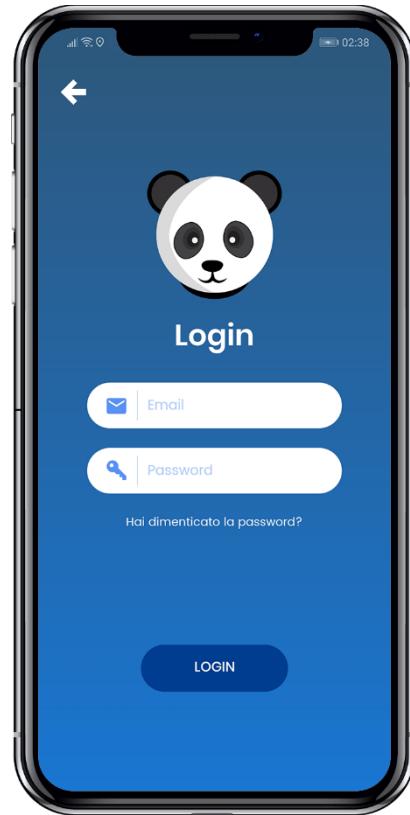


Figura 12
Schermata di login

4.1.4 Nickname

La gestione dei nickname viene presa in considerazione molto spesso nella piattaforma, in quanto devono essere visualizzati sempre correttamente nelle classifiche. Ogni pseudonimo ha inoltre la caratteristica di essere unico ma anche alterabile.

Nella figura 11 è già stato mostrato come si imposta il nickname quando un'utente effettua una registrazione attraverso provider Facebook. In questa sezione, quindi, viene spiegato il suo procedimento di inserimento o modifica.

Gli utenti hanno la possibilità di impostare il proprio nickname in qualsiasi momento, recandosi nella pagina dedicata al profilo utente, mostrata nella figura 13, e premendo il relativo tasto di modifica.

Il nickname viene settato nell'attributo 'displayName' dell'oggetto FirebaseAuther, in modo che il suo recupero sia istantaneo, senza dover connettersi quindi al database. Questo dato, se impostato, viene esposto insieme alle altre informazioni rilevanti del profilo nella schermata mostrata in figura 13.

Alla pressione del tasto di modifica entra in azione una Dialog, contenente un EditText, che viene mostrata con diversi comportamenti illustrati nella figura 14:

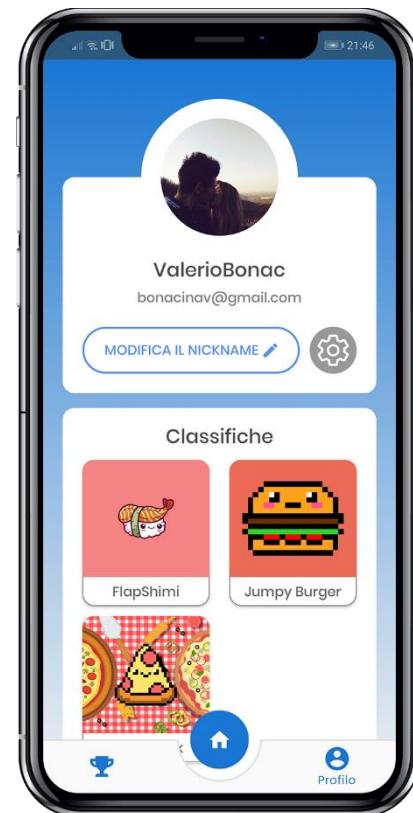


Figura 13
Schermata del profilo utente

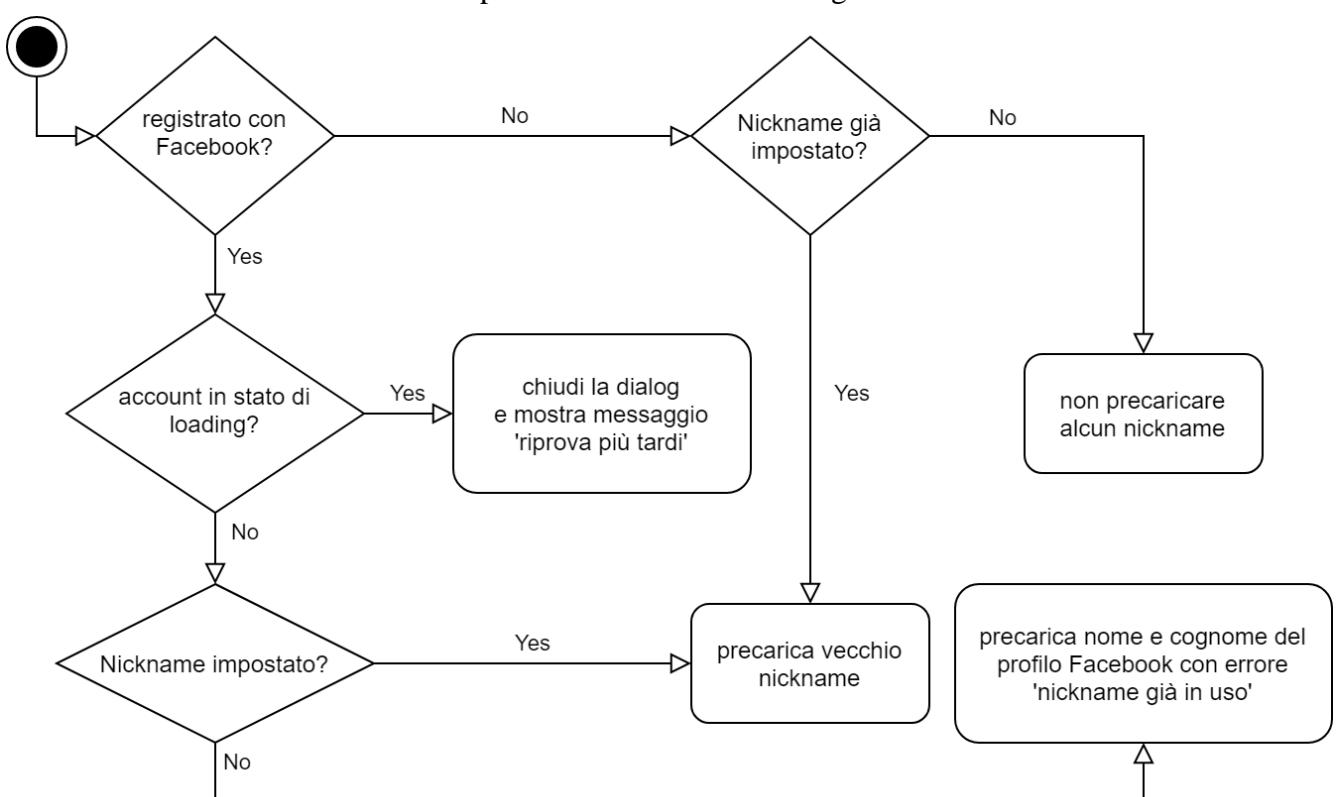


Figura 14
Diagramma dei comportamenti all'apertura della dialog di modifica del nickname

Backend

Nel momento di conferma della modifica del nickname da parte dell'utente viene avviata una richiesta al backend Firebase Functions, che riceve in input il nuovo nickname e risponde alla chiamata con un valore booleano indicante l'effettivo riscontro del cambiamento. Se il riscontro è negativo allora significa che il nickname è già presente sul database.

La funzione in questione viene illustrata in maniera approssimativa nella figura 15, dove i parametri passati vengono segnati tra parentesi graffe.

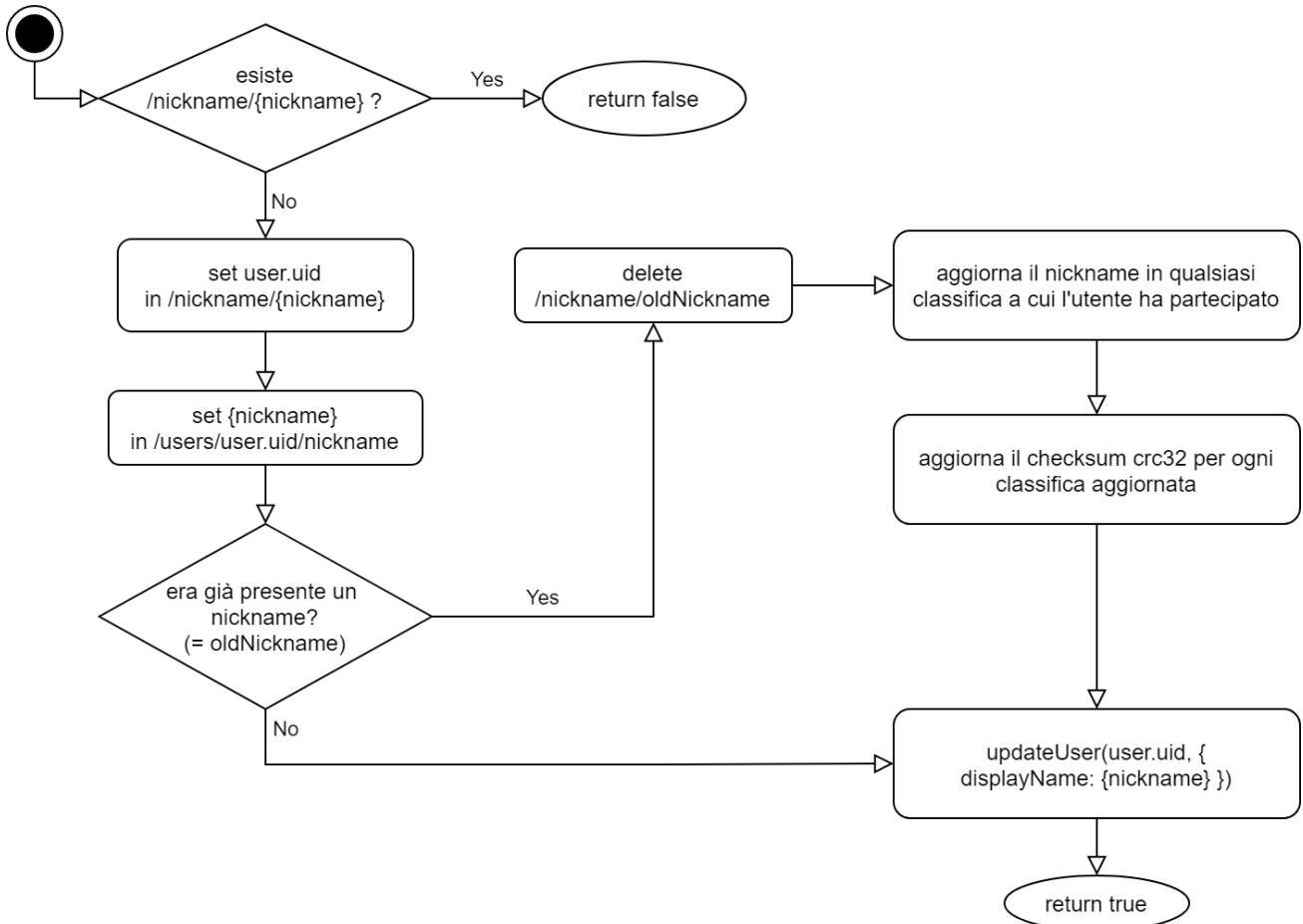


Figura 15
Diagramma riassuntivo della funzione backend della richiesta di modifica del nickname

Sulla base della risposta da parte del server viene aggiornata l'interfaccia e quindi notificato all'utente lo stato corrente. Se il nickname non è stato impostato correttamente verrà mostrato un errore, che comunica l'uso prioritario di quel particolare pseudonimo da parte di un altro utente.

4.1.5 Logout

All'interno della scheda delle impostazioni è presente il tasto per poter uscire dal proprio account. La disconnessione viene eseguita tramite Firebase Authentication che cancella i token di accesso in memoria. Inoltre, per gli account loggati tramite provider Facebook, si effettua un'ulteriore operazione di logout tramite le Api di Facebook.

In seguito al logout, l'applicazione rimanda l'utente alla schermata di 'home' e lo limita nell'uso. Al tentativo di accesso al gioco, oppure alle schermate 'premi' e 'profilo' viene richiesto il login, mostrando sullo schermo la schermata in figura 10.

4.2 Ricerche

Le ricerche dei ristoranti sono fondamentali per usufruire del servizio offerto dalla piattaforma. Le tipologie di ricerca a disposizione sono 3:

- **Cerca nelle vicinanze**, basata su filtri specifici di distanza e ordinamento.
- **Cerca qui**, ovvero all'interno della finestra visualizzata sulla mappa.
- **Cerca tramite keyword** dalla barra di ricerca, basata su filtro di ordinamento.

Ogni categoria appena stilata usa metodi di ricerca che sfruttano il geohashing per performare le operazioni e renderle scalabili.

4.2.1 Geohashing

Il geohashing^[13] è un metodo di geocodifica utilizzato per codificare le coordinate geografiche (latitudine e longitudine) in una breve stringa di cifre e lettere che delineano un'area su una mappa, chiamata ‘cella’, con risoluzioni diverse. La precisione della posizione dipende dalla quantità di caratteri nella stringa, più la stringa è lunga, più risulta precisa la locazione.

I geohash usano la codifica alfabetica Base-32 (i caratteri possono essere da 0 a 9 e dalla A alla Z, escluso "A", "I", "L" e "O").

Il mondo viene suddiviso in una griglia con 32 celle (figura 16). Il primo carattere in un geohash identifica la posizione iniziale come una delle 32 celle. Questa cella conterrà a sua volta 32 celle e ognuna di esse conterrà 32 celle (e così via ricorsivamente). L'aggiunta di caratteri al geohash suddivide una cella, descrivendo un'area più dettagliata.

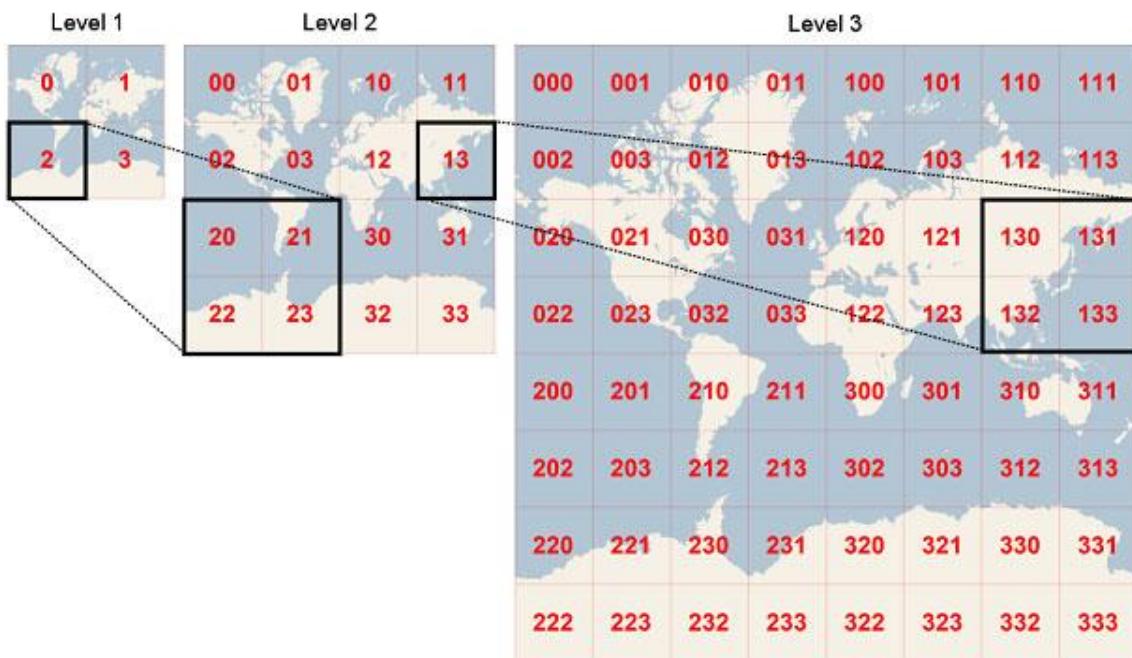


Figura 16
Suddivisione del mondo in celle secondo il metodo del geohashing

Il fattore di precisione determina la dimensione della cella. Ad esempio, un fattore di precisione di uno crea una cella di 5.000 km di altezza e 5.000 km di larghezza, un fattore di precisione di sei, invece, crea una cella di 0,61 km di altezza e 1,22 km di larghezza. Va infine sottolineato che le celle non sono sempre quadrate.

4.2.2 Ricerca nelle vicinanze

Il processo di ricerca nelle vicinanze si basa su due punti distinti:

- (**Facoltativo**) impostazione dei filtri da applicare alla ricerca.

I filtri sono visti come passaggio facoltativo in quanto l'utente può ignorare questa fase e la ricerca userà i vincoli salvati in memoria dopo l'ultima modifica. Nel caso in cui i valori di filtraggio non siano mai stati impostati questi assumeranno dei contenuti di default.

I filtri disponibili sono i seguenti:

- **Distanza** massima dalla posizione dell'utente, che può variare dai 10km ai 50km. Il suo valore di default iniziale è 20km.
- **Ordinamento** dei risultati di ricerca sulla base del criterio scelto, che può essere distanza, rilevanza (equivalente al numero di partite giocate nei rispettivi giochi) o premi (ovvero il numero di premi che offre quel ristorante). Il suo valore di default iniziale è la distanza.
- Conferma della ricerca tramite l'apposito tasto.

Prima di procedere con l'effettiva operazione di ricerca, però, vengono controllati i permessi necessari per accedere alla posizione GPS del dispositivo e si richiede l'autorizzazione per poter attivare il servizio di geolocalizzazione, nel caso in cui sia disattivo.

In seguito alle operazioni preliminari sopracitate vengono eseguite a lato client una serie di istruzioni che predispongono l'interfaccia in modo da far capire all'utente lo stato di caricamento. La mappa viene svuotata da ogni marker e viene sostituito il fragment all'interno della BottomSheet con uno nuovo dedicato alla ricerca corrente. Nel caso in cui nello stack dei fragment ce n'era già presente uno, appartenente ad una ricerca precedente, questo viene rimosso anche dallo stack e l'interfaccia aggiorna soltanto i componenti necessari.

A questo punto viene individuata la posizione GPS dell'utente e la camera della mappa effettua una prima animazione su di essa con un livello di zoom definito.

Backend

La ricerca dei ristoranti avviene sul lato backend attraverso una chiamata http ad una funzione caricata nel servizio Firebase Functions. Tale funzione prende in input vari parametri, elencati di seguito:

- **Latitudine** (nominata 'lat') e **longitudine** (nominata 'lng'), riguardante la posizione GPS dell'utente rilevata.
- **Distanza** (nominata 'distance') e **ordinamento** (nominato 'orderBy'), relativi ai filtri discussi precedentemente.
- **Lingua** (nominata 'lang'), relativa alla lingua di sistema impostata sul dispositivo che effettua la richiesta. Questa variabile è necessaria al fine di mostrare i risultati della ricerca tradotti nella lingua più familiare all'utente che li vede.

Per poter sfruttare i metodi del geohashing nella ricerca è stata utilizzata la libreria open-source GeoFirestore (v.3.4.3)^[14], che estende l'API ufficiale Firestore al fine di interrogare i documenti in base alla posizione geografica associata, specificando soltanto un punto e una distanza in km da quel punto.

Viene mostrato brevemente, nella figura 17, il lavoro svolto lato server dalla funzione in questione.

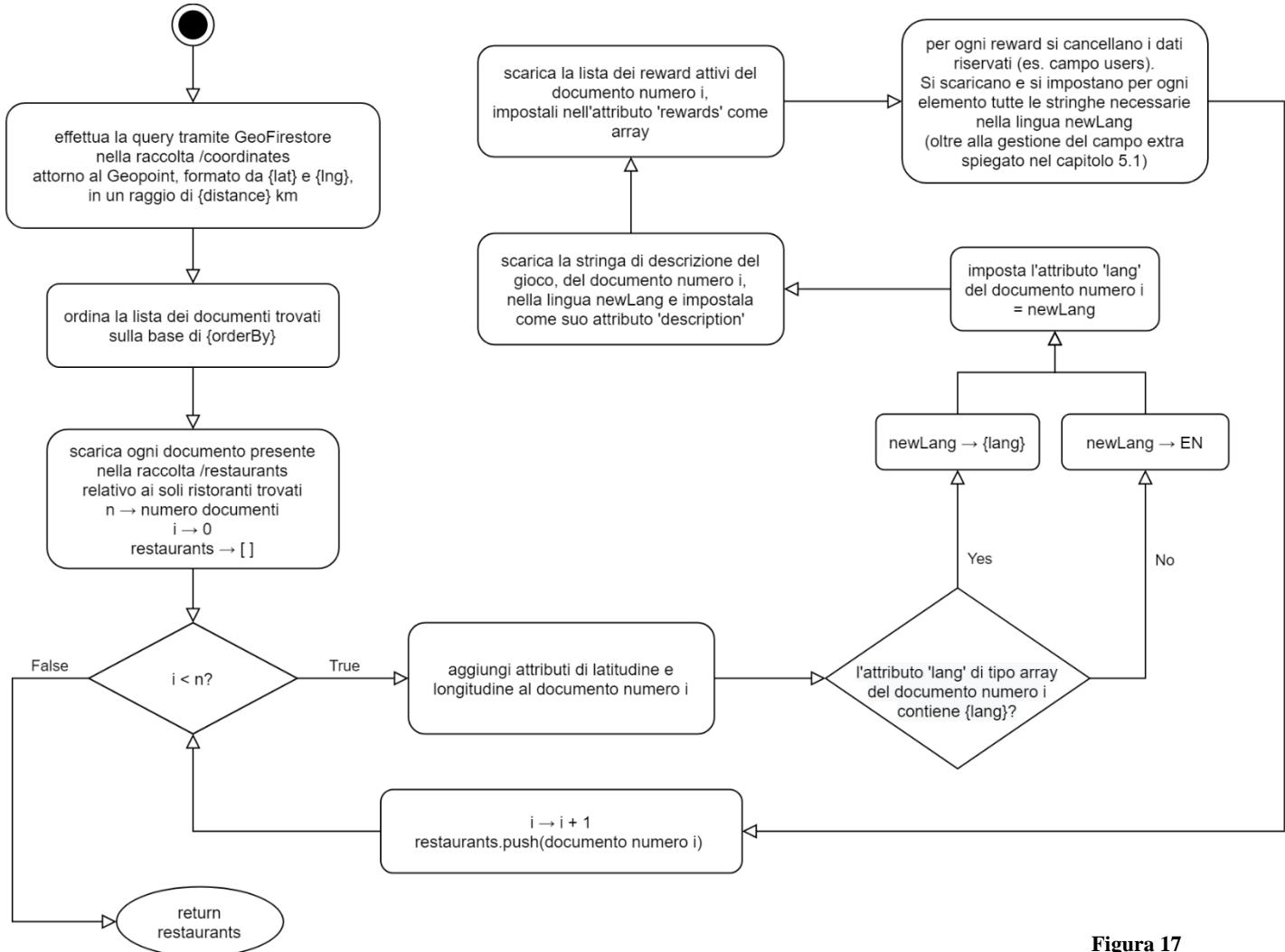


Figura 17
Diagramma riassuntivo della funzione backend di 'ricerca nelle vicinanze'

Questa funzione, ma anche quelle successive, in realtà risulta ovviamente più complessa di quanto descritto, si sono mostrate solo le funzionalità chiave per la sua corretta esecuzione.

Si è fatto uso anche di programmazione dinamica allo scopo di evitare il download di dati già reperiti in precedenza, così da avere maggiori prestazioni.

Nel momento di ricezione, da parte del client, della risposta si effettua una scomposizione dell'oggetto restituito. Nel caso in cui l'oggetto sia vuoto allora viene aggiornata l'interfaccia, in maniera da comunicare all'utente l'assenza di risultati per quella determinata ricerca. Se, invece, vi sono dei riscontri positivi dalla chiamata allora l'applicazione provvede a mostrare tutti i marker trovati, con il relativo effetto di zoom nella mappa, e ad aggiungere le informazioni recepite nella lista dei risultati, posizionata all'interno della BottomSheet.

4.2.3 Cerca qui

Questo tipo di ricerca viene effettuata, come detto precedentemente, per mostrare i ristoranti nella finestra visualizzata sulla mappa.

Il processo in questo caso avviene semplicemente con la pressione del tasto ‘cerca qui’, senza la necessità di particolari controlli sui permessi GPS.

Ad avvio procedura, così come nella tipologia precedente, viene effettuato un aggiornamento dell’interfaccia in stato di caricamento e una serie di operazioni su componenti e fragment per predisporre la corretta esecuzione della funzionalità.

Backend

La chiamata al backend Firebase Functions in questo contesto richiede molti più parametri, in quanto la finestra visualizzata, oggetto della ricerca, è di forma rettangolare e quindi delimitata da due punti estremi principali.

Vengono elencati di seguito i parametri da passare alla funzione con la relativa spiegazione:

- **Latitudine e Longitudine** di vari punti presenti nell’area rilevata, ovvero:
 - **Nord-est** (formato dal parametro ‘maxLat’ e ‘maxLng’)
 - **Sud-ovest** (formato dal parametro ‘minLat’ e ‘minLng’)
 - **Centro** (formato dal parametro ‘centerLat’ e ‘centerLng’)
- **Distanza** (nominata ‘distance’) in km che separa il punto geografico al centro della finestra visiva e quello situato ad uno dei due estremi.
Anche in questa situazione è stato usato uno strumento messo a disposizione da GoogleMap, in una libreria dedicata^[15]. In particolare, viene utilizzata la classe *SphericalUtil* che contiene tutta una serie di metodi incentrati su operazioni con punti geografici, tra cui, appunto, il calcolo della distanza in km tra due posizioni.
- **Lingua** (nominata ‘lang’).

La libreria GeoFirestore, in questa ricerca, non restituirà però i soli ristoranti contenuti nella finestra di interesse, avendo l’unica funzione di trovare i risultati intorno a un punto e data una determinata distanza.

Come si può vedere nella figura 18, attraverso i punti di minimo e massimo, passati nei parametri, verrà effettuata un’operazione di esclusione dei risultati che non rientrano in questi bordi (area rossa), così da avere tutti e soli i dati richiesti (area verde).

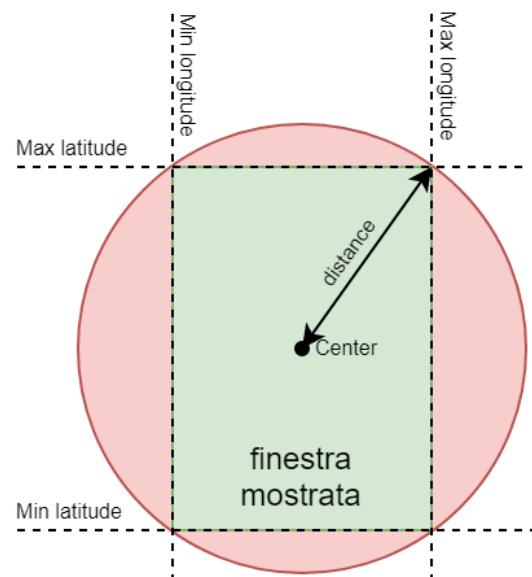


Figura 18
Illustrazione della esclusione dei risultati non pertinenti alla ricerca

Viene mostrata, nella figura 19, una versione sintetica della funzione backend.

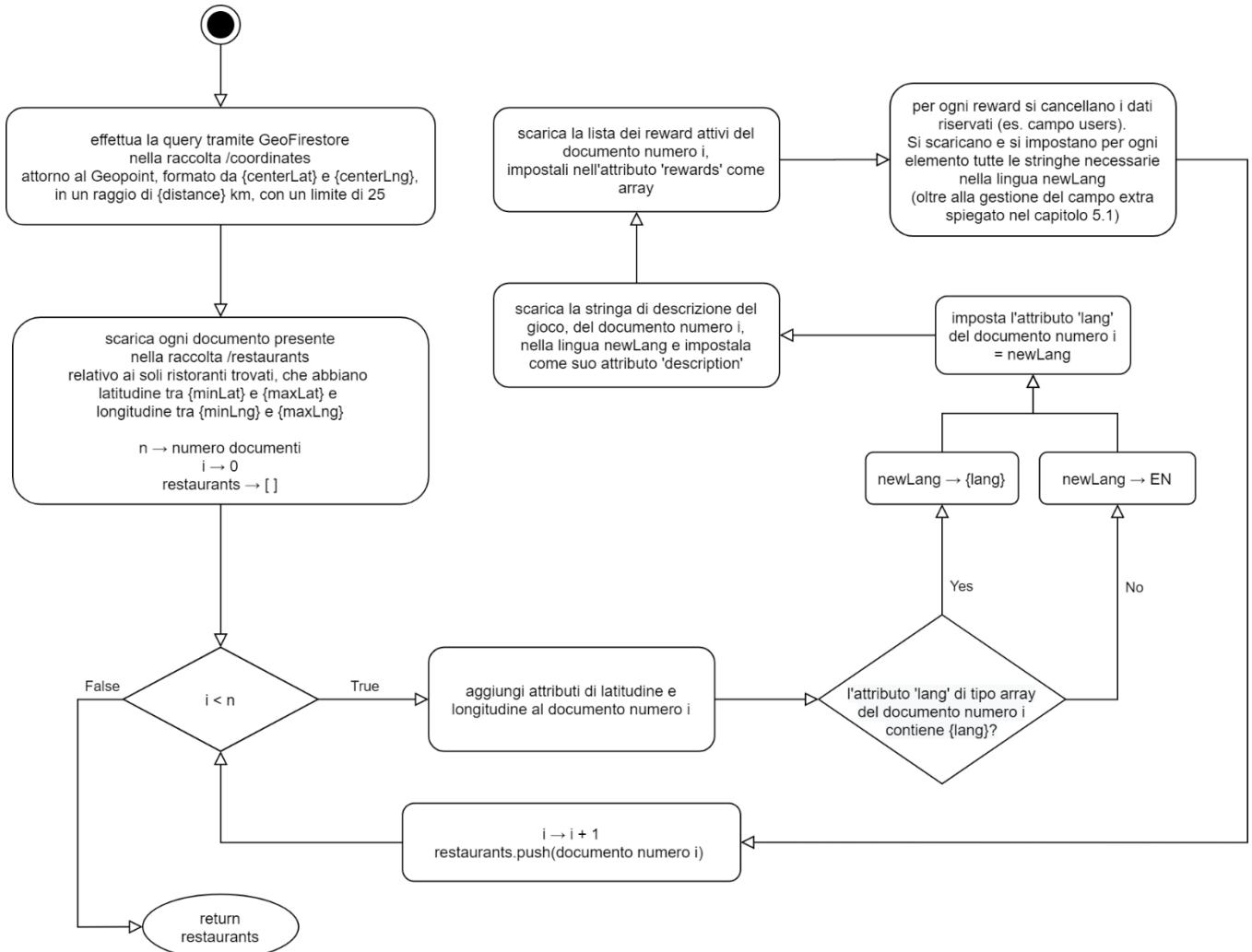


Figura 19
Diagramma riassuntivo della funzione backend 'cerca qui'

Al completamento di questa funzione il lavoro svolto dal client è il medesimo eseguito nella tipologia di ricerca precedente.

4.2.4 Cerca tramite keyword

Per poter effettuare questo tipo di ricerca viene richiesto l'inserimento di una specifica keyword che può essere riferita alle seguenti categorie:

- **Nomi di ristoranti**, rispettando la regola del ‘case-insensitive’.
- **Nomi di giochi**, rispettando la regola del ‘case-insensitive’.
- **Nomi di luoghi**, che possono rappresentare qualsiasi locazione geografica esistente.

Per poter ricavare delle particolari aree, in cui effettuare la ricerca, attraverso la specifica di luoghi viene utilizzato OpenCage, un servizio esterno di geocoding.

Lo strumento utilizzato nello specifico è stato quello di forward geocoding, ovvero quello di recuperare dati geografici sulla stringa di testo passata.

L'accesso a questa funzionalità avviene premendo sulla SearchView, in cui l'interfaccia mostra all'utente la possibilità di digitare del testo, oppure di scegliere un elemento dalla cronologia salvata, per effettuare la ricerca, come mostrato in figura 20.

Inoltre, si offre la possibilità di scegliere l'ordinamento dei risultati restituiti. In questo contesto i filtri di ordinamento selezionabili sono quelli di rilevanza e premi, con gli stessi significati applicati per i filtri della prima tipologia di ricerca.

Alla conferma della stringa da ricercare viene impostata l'interfaccia e i vari componenti come nelle tipologie di ricerca precedenti. Successivamente, il concreto meccanismo della funzionalità viene diviso in due fasi:

- **Forward geocoding**, eseguito lato client tramite la libreria Volley^[16] per via di limitazioni dovute alla versione gratuita del backend Firebase Functions, il quale blocca tutte le comunicazioni con servizi esterni a Google.

La richiesta al servizio viene effettuata con le seguenti opzioni aggiuntive:

- **Limit = 1** (un singolo risultato restituito)
- **No_dedupe = 1** (non cercare risultati doppi)
- **No_annotation = 1** (non ritornare annotazioni)

Questa fase può avere riscontro sia positivo che negativo, nel primo caso vengono estratti dal risultato i punti geografici estremi, che delimitano l'area d'interesse, e si calcolano i parametri per la fase successiva. Nel secondo caso, invece, si impostano i parametri con dei valori default di errore, così da far capire al backend il riscontro di questa parte.

- **Chiamata al backend** del servizio Firebase Functions.

Backend

La funzione chiamata prende in ingresso i seguenti parametri:

- **Latitudine e Longitudine** di vari punti presenti nell'area rilevata, ovvero:
 - **Nord-est** (formato dal parametro ‘maxLat’ e ‘maxLng’, default = -1).
 - **Sud-ovest** (formato dal parametro ‘minLat’ e ‘minLng’, default = -1).
 - **Centro** (formato dal parametro ‘centerLat’ e ‘centerLng’, default = -1).

Questo parametro è calcolato individuando il punto intermedio tra i due punti estremi sopraelencati, sfruttando sempre la classe *SphericalUtil*, della libreria citata in precedenza al capitolo 4.2.3.

- **Distanza** (nominato ‘distance’, default = -1) in km che separa il centro dell'area di interesse e dai uno dei due punti estremi (approfondimento al capitolo 4.2.3).
- **Ordinamento** (nominato ‘orderBy’), relativo ai filtri discussi sopra.

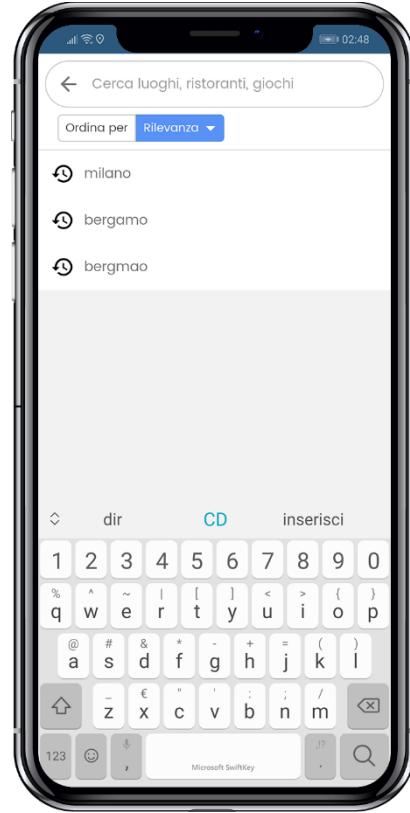


Figura 20
Schermata di ricerca tramite keyword

- **Query** (nominato ‘textQuery’), contenente la stringa di testo cercata.
- **Lingua** (nominato ‘lang’).

In questa funzione viene innanzitutto effettuato un confronto diretto con i nomi dei giochi e ristoranti presenti nel database. In caso di riscontro negativo verrà effettuata l’analisi da parte di GeoFirestore dei parametri passati. A seguito di questa operazione, anche in questa procedura avviene l’esclusione dei punti non rientranti nei limiti definiti (maggiori informazioni al capitolo 4.2.3). Viene mostrata, nella figura 21, una versione sintetica della funzione backend.

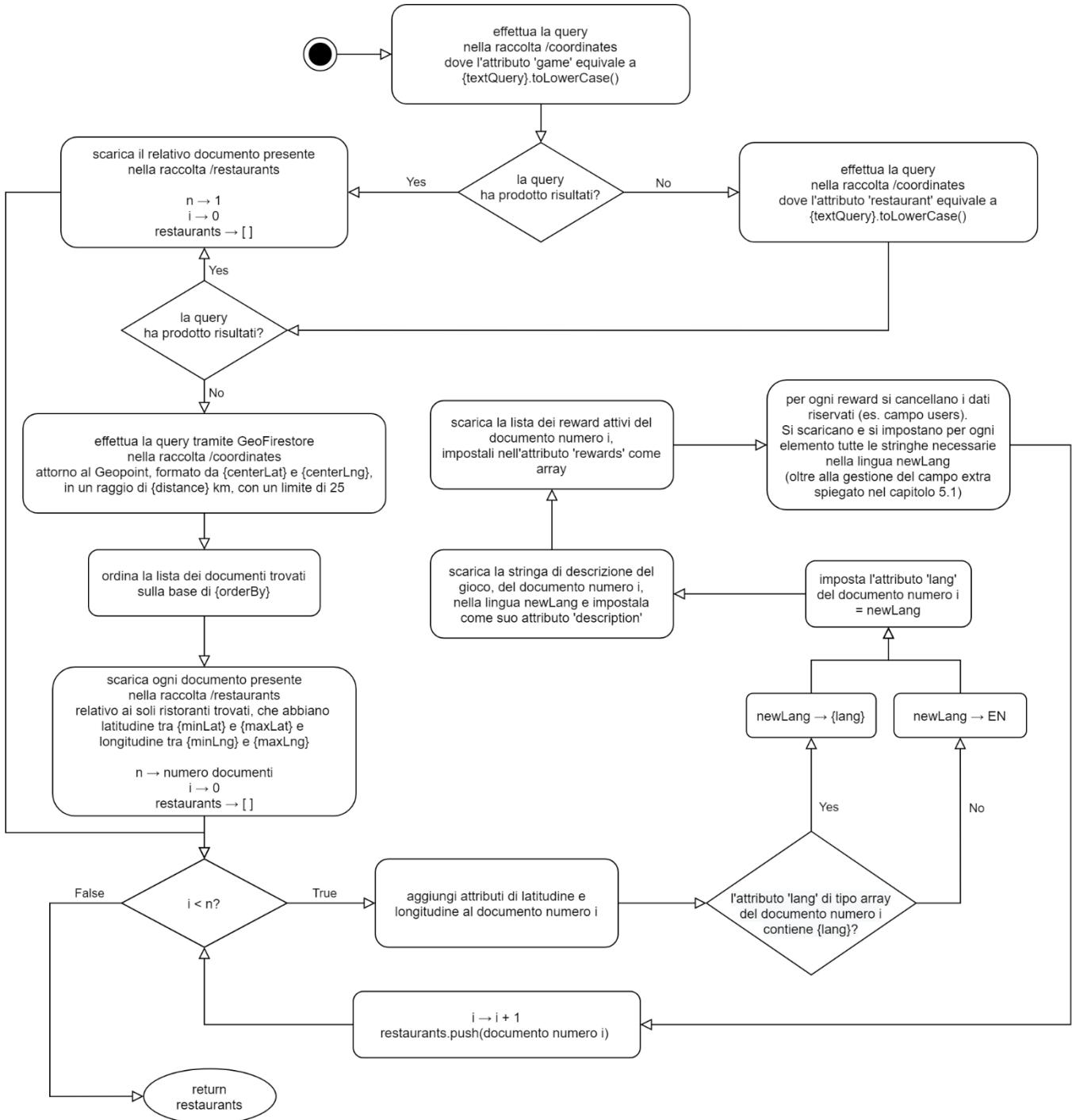


Figura 21
Diagramma riassuntivo della funzione backend ‘cerca tramite keyword’

Al completamento di questa funzione il lavoro svolto dal client è il medesimo eseguito nelle tipologie di ricerca precedente.

4.3 Giochi

Il fulcro intorno al quale la piattaforma basa il suo servizio sono i videogiochi offerti agli utenti. Questi vengono presentati con una schermata di dettaglio precisa, mostrata nella figura 22, la quale permette all'utente di poter giocare e espone tutte le informazioni relative a quel gioco (descrizione, ristorante, premi). In questa pagina sono anche accessibili ulteriori funzionalità:

- **Mostra sulla mappa** il marker del ristorante associato, evidenziandolo come se venisse selezionato dall'utente in seguito ad una ricerca.
- **Visita il sito** del ristorante, aprendolo sul browser predefinito del dispositivo.
- **Vedi la classifica** settimanale (prime 7 posizioni) e globale (prime 3 posizioni) all'interno di una FragmentDialog, apribile premendo sull'icona della coppa in alto a destra. Viene anche comunicata la posizione dell'utente, se presente, in ognuna delle due tipologie.
- **Leggi i termini e condizioni di un premio** all'interno di una Dialog, apribile premendo sull'icona informativa presente in ogni Card relativa ad un premio.

4.3.1 Ricezione dati di dettaglio

Come si è visto in tutto il capitolo 4.2 tali informazioni vengono scaricate in fase di ricerca e quindi impostate nel fragment incaricato all'apertura di un risultato. L'applicazione predispone però una funzionalità che salva una cronologia dei giochi con cui si interagisce, mostrata nella schermata di 'home' senza effettuare alcuna ricerca. In questa sezione le informazioni di dettaglio sono assenti ed è quindi necessario un collegamento al database se si vuole aprire la scheda informativa. La chiamata al backend viene eseguita dal fragment sopracitato nel caso in cui veda la mancaza di dati nell'oggetto passatogli, contenente le informazioni necessarie per la corretta visione della schermata.

Backend

La chiamata alla funzione di Firebase Functions richiede i seguenti parametri:

- **Id del ristorante** (nominato 'id'), necessario alla ricezione dei dati richiesti.
- **Lingua** (nominato 'lang').

In questa funzione viene fondamentalmente ripetuto quanto accade nelle funzioni di ricerca, con la semplificazione che già si è a conoscenza del documento da dover scaricare, in quanto si è in possesso del suo identificativo. Quindi, si passa direttamente all'aggiunta dei parametri rilevanti e alla gestione della lingua. Successivamente, i dati vengono restituiti e il client li predisponde alla corretta visualizzazione.



Figura 22
Schermata di dettaglio di un gioco

4.3.2 Checksum crc32

Il cyclic redundancy check^[17] ^[18] (ovvero controllo di ridondanza ciclico, il cui acronimo CRC è più diffuso) è un metodo per il calcolo di somme di controllo (checksum in inglese). Il nome deriva dal fatto che i dati d'uscita sono ottenuti elaborando i dati di ingresso i quali vengono fatti scorrere ciclicamente in una rete logica. Il controllo CRC è molto diffuso perché la sua implementazione binaria è semplice da realizzare, richiede conoscenze matematiche modeste per la stima degli errori e si presta bene a rilevare errori di trasmissione su linee affette da elevato rumore di fondo.

L'idea basilare che sta dietro ogni algoritmo per il calcolo del CRC è abbastanza semplice: trattare il flusso di dati come un enorme numero binario, dividerlo per un altro numero binario fisso (costante), ed utilizzare il resto di questa divisione come valore di CRC.

In realtà, il divisore, il dividendo (il flusso dati), il quoziente, ed il resto non vanno considerati come interi positivi, ma piuttosto come polinomi con coefficienti binari. Questo avviene considerando ciascuno di questi numeri come una sequenza di bit che non sono altro che i coefficienti di un polinomio. Vediamo come ciò avviene con un esempio: il numero decimale 35 viene rappresentato come $0x23H$ in esadecimale, e 100011 in formato binario. Se ai bit vengono fatti corrispondere i coefficienti di un polinomio, il numero rappresenterebbe anche il seguente polinomio: $1 * x^5 + 0 * x^4 + 0 * x^3 + 0 * x^2 + 1 * x^1 + 1 * x^0$, ovvero $x^5 + x^1 + 1$.

Al fine di eseguire il calcolo del CRC, la prima cosa da fare è quella di scegliere un divisore, che tecnicamente viene definito come il “polinomio generatore”. Un'importante proprietà del polinomio generatore è l'ampiezza (width), definite come la posizione del suo bit più significativo con valore 1. Valori tipici per l'ampiezza sono 16 e 32, perché essi semplificano l'implementazione dell'algoritmo sui normali computer.

Utilizzo nell'applicazione

L'uso dell'algoritmo crc32 nel sistema è sfruttato per velocizzare la ricezione delle classifiche, in quanto durante una sessione di gioco può essere richiesta varie volte al termine di una partita.

Di conseguenza tali dati, una volta scaricati, vengono memorizzati localmente nella webview in una variabile. Alla successiva richiesta di ricezione si effettua un controllo tra il checksum dei dati salvati e quello sul database, rappresentante la stessa tipologia di dati. Se il checksum risulta equivalente si provvede a rimostrare i dati in memoria, altrimenti viene scaricata nuovamente la porzione della graduatoria dal database.

Ricordando come è stata progettata la struttura del database per le classifiche (capitolo 3.2), è stato scelto di comprimere con l'algoritmo una stringa, che contenga tutte le informazioni importanti al fine di avere un'unica corrispondenza con la reale classifica richiesta. Tale stringa viene così composta:

$$\sum_{k=1}^n punteggio_k + userUID_k + nickname_k + indice_k$$

Dove n indica le posizioni rilevanti da mostrare all'utente, ovvero 7 posizioni per la classifica settimanale e 3 posizioni per quella globale, e k prende quindi il valore dell'indicatore di posizione di un utente nella relativa classifica.

Al fine di spiegare meglio quanto detto viene mostrato di seguito un metodo, presente nel backend, che verrà spesso anche richiamato in seguito col nome di ‘returnRank’.

Tale metodo ha lo scopo di ritornare le due tipologie di classifica di un dato gioco, valutando il checksum e altre opzioni. I parametri in ingresso sono:

- **Id del gioco** (nominato ‘game’)
- **Numero della classifica settimanale corrente** (nominato ‘numberWeek’), relativo al gioco in questione.
- **Checksum crc32 salvati in locale** delle due classifiche (nominati ‘crc32GlobalSaved’ e ‘crc32WeeklySaved’).
- **Parametro di forzatura** (nominato ‘force’), per forzare il download senza controlli sul checksum.
- **Richiesta delle foto profilo** di tutti gli utenti in classifica (nominato ‘getPhoto’), indica la volontà di aggiungere alla classifica richiesta il link delle foto profilo Facebook degli utenti trovati.

Il diagramma schematico del metodo viene illustrato nella figura 23.

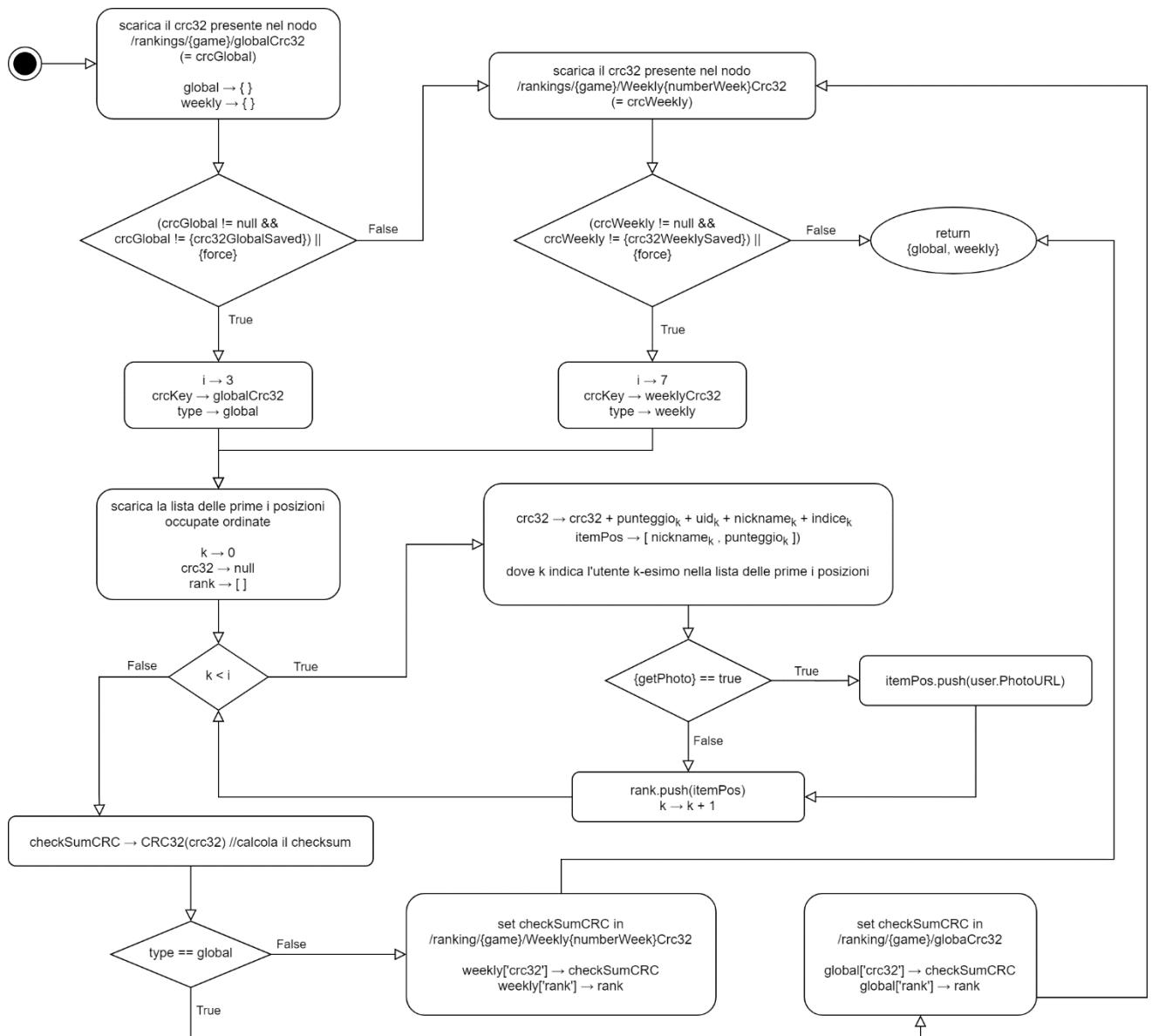


Figura 23
Diagramma riassuntivo del metodo ‘returnRank’ situato nel backend

4.3.3 Ricezione classifica da App

È stato spiegato, nella sezione 4.3, dove viene richiesta la ricezione della classifica all'interno dell'applicazione. Oltre a quanto detto, si aggiunge che tale classifica è reperibile anche nella pagina del profilo, dove è presente una sezione che offre, per ogni gioco in cui l'utente occupa una posizione in classifica, l'accesso ad ogni graduatoria (vedi capitolo 5.6).

I relativi dati vengono ricavati dalla chiamata ad una funzione del backend e impostati in una FragmentDialog, nel caso di richiesta nella schermata di dettaglio, oppure in un Fragment, in cui la richiesta sia effettuata dalla pagina del profilo.

Backend

La funzione interessata chiede soltanto il seguente parametro:

- **Id del gioco** (nominato ‘game’) relativo alle classifiche che si vuole reperire.

Nella seguente procedura viene eseguito il metodo ‘returnRank’ con l'aggiunta della funzionalità di ricavare la posizione dell'utente nelle classifiche. Inoltre, è importante ricordare che l'utente potrebbe non essere loggato quanto effettua questa richiesta, quindi viene gestita anche questa casistica.

Il diagramma schematico della funzione viene illustrato nella figura 24.

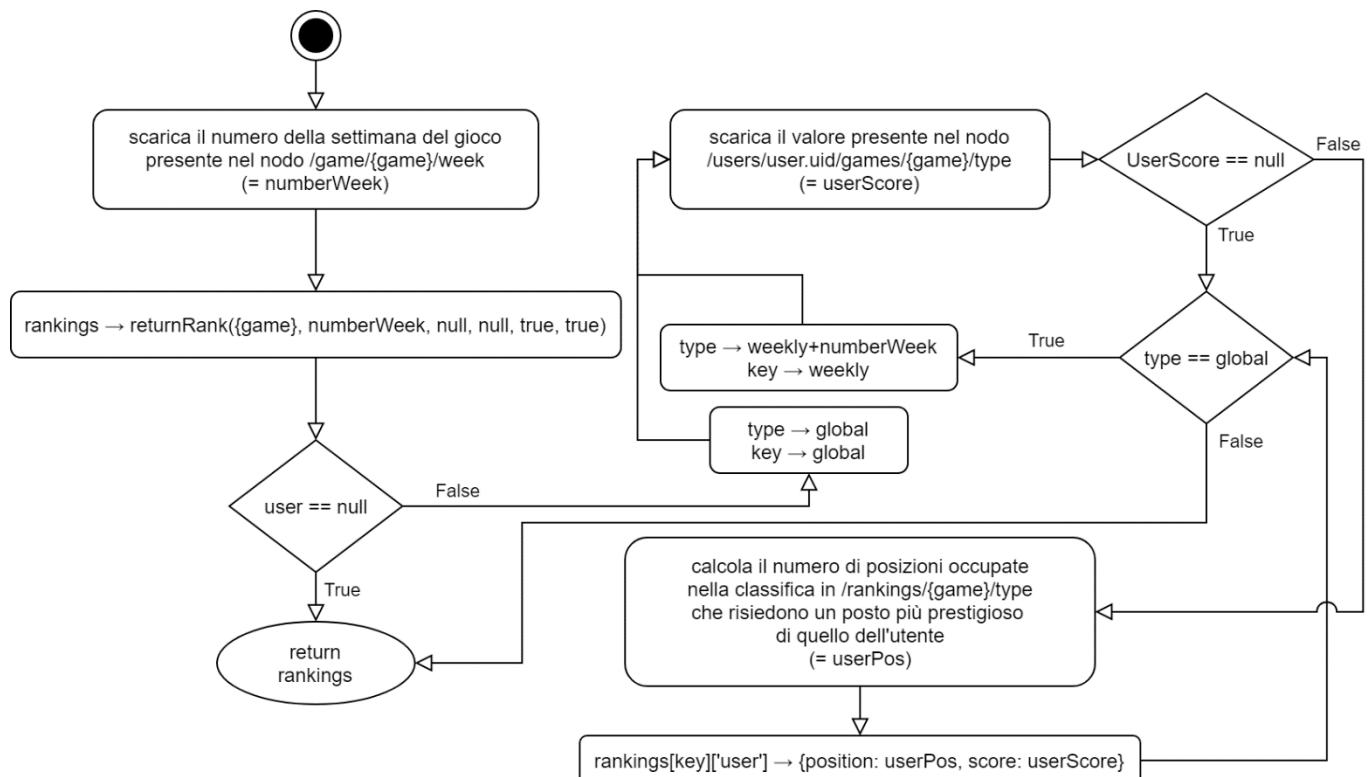


Figura 24

Diagramma riassuntivo della funzione backend
che ritorna le classifiche e la posizione utente di un gioco

4.3.4 Apertura videogioco

Nel momento della richiesta di apertura di un videogioco vengono effettuati dei controlli sulla condizione dell'account loggato, che possono negare la funzionalità. Se l'activity viene lanciata, gli si passa l'id del gioco così che possa comporre l'url corretto.

Tali controlli verificano, in particolare:

- **Lo stato dell'account**, che deve essere impostato su ‘verified’.
- **La presenza di un nickname**, per poter comparire nelle classifiche con il relativo pseudonimo.
- **Lo stato coerente del nickname nel database**, ovvero si effettua un paragone tra il nickname salvato nel modulo Firebase Authentication e l'attributo ‘nickname’ scritto sul database nel nodo utente.

Questo confronto avviene perché, a fronte di una successiva modifica del nickname, la funzione relativa in Firebase Functions potrebbe non aver concluso il suo compito. Quindi, sapendo che nella procedura si esegue tra le prime istruzioni la modifica del nodo ‘nickname’ nel database e come ultima istruzione la modifica del nickname nel modulo Firebase Authentication, se il paragone è verificato la funzione è quindi conclusa altrimenti si segnala il conseguente errore.

Tale operazione, per semplicità, è chiamata in seguito col nome ‘checkNick()’.

Viene mostrato, nella figura 25, lo schema dei diversi comportamenti relativi all’azione di apertura di un gioco.

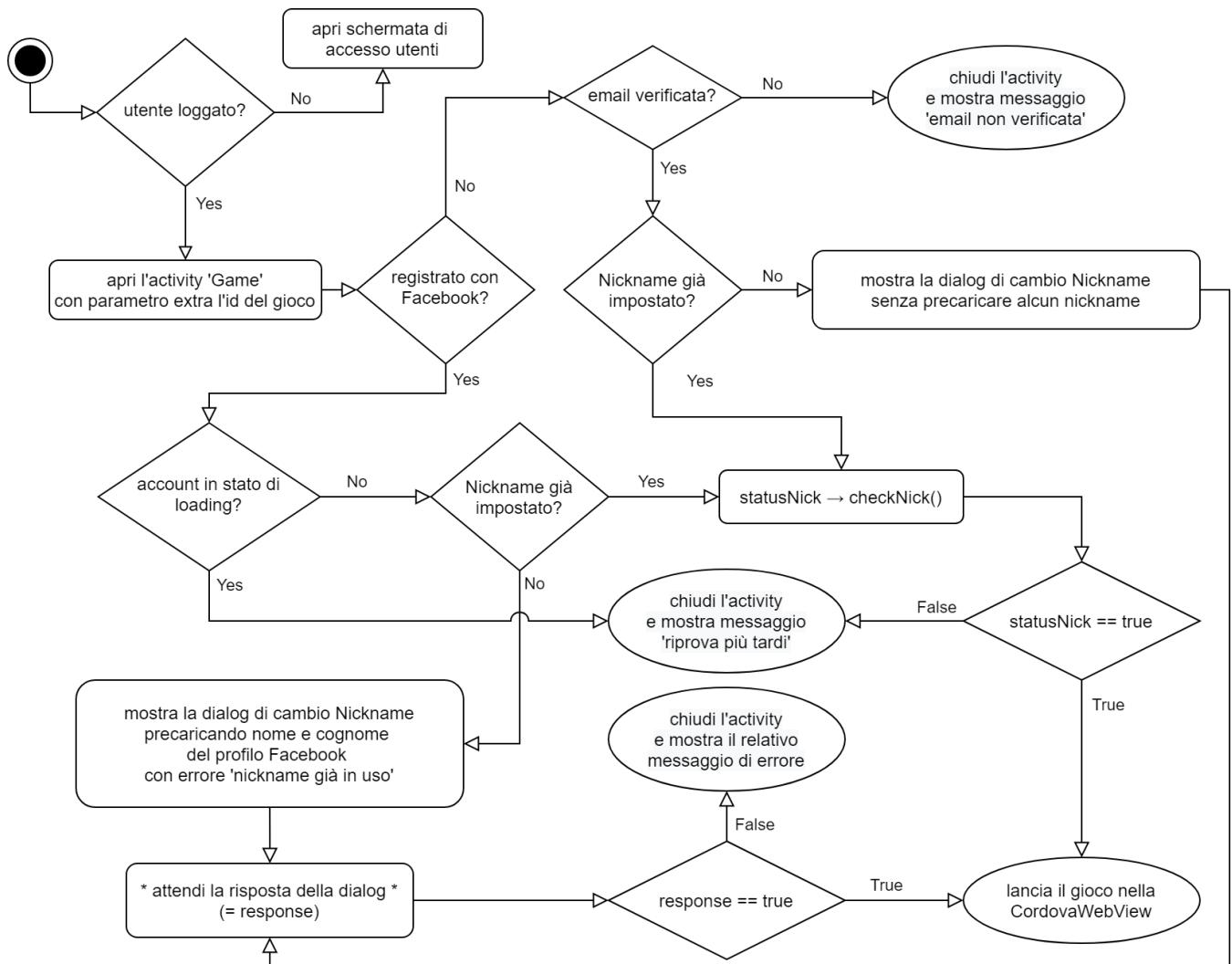


Figura 25
Diagramma riassuntivo comportamenti relativi all’azione di apertura di un gioco

4.3.5 Comunicazione CordovaWebView – Java nativo

Una volta lanciata la CordovaWebView, nell'activity contenitrice, la comunicazione con l'ambiente nativo Java avviene tramite un apposito plugin realizzato ad hoc.

Tale plugin ha lo scopo di reperire e salvare i dati nel database, in quanto l'ambiente nativo, al contrario della pagina web, dispone dell'accesso ai servizi Firebase. Le principali funzioni del plugin sono 3:

- **Richiesta dei record utente** (nominato ‘getBest’) per ogni classifica relativa al gioco.
- **Richiesta delle classifiche** (nominato ‘getRank’) relative al gioco con il confronto di checksum.
- **Salva il punteggio** (nominato ‘saveScore’), ovvero imposta nel database il nuovo record relativo alle classifiche del gioco, oltre che aggiornarle e restituirle al chiamante.

le funzioni ‘saveScore’ e ‘getRank’ (quest’ultima soltanto se chiamata a seguito di una sessione di gioco) si occupano anche del controllo relativo alla vincita di premi erogati al superamento di un certo numero di livelli.

Comportamento pagina web

La pagina web contenente il gioco HTML5 non si limita a comunicare col plugin, ma dispone di alcune funzionalità, inserite in uno script Javascript implementato, che riducono il numero di comunicazioni con l'ambiente nativo, aumentando l'efficienza del servizio.

Un generico gioco viene programmato con delle situazioni standard:

- **Ad avvio gioco**, come prima operazione, vengono scaricati e memorizzati nello script i punteggi record dell'utente, relativi alle tipologie di classifica settimanale corrente e globale. In seguito, si rende visibile la schermata del menu principale (mostrata in figura 26).
- Nel momento di **richiesta della classifica dal menu** (accessibile dal relativo tasto mostrato nella figura 26) si richiama la funzione ‘getRank’, passandogli i valori di checksum e un valore nullo, come rappresentante del punteggio raggiunto, per segnalare al sistema il contesto di applicazione della chiamata. Se la risposta, da parte del plugin, contiene dei valori nulli allora il controllo checksum ha avuto esito positivo e viene risposta la relativa classifica salvata in memoria. Altrimenti, vengono salvati i dati recuperati, inclusi i nuovi valori di checksum, e mostrata la nuova graduatoria.



Figura 26
Schermata di menu di un generico gioco caricato nella CordovaWebView

- Nella **situazione di game over** (mostrata nella figura 27), in una partita di gioco, si effettua un confronto tra il punteggio raggiunto nella sessione appena conclusa e i record in memoria reperiti in precedenza. Se il numero di livelli appena raggiunti è superiore ad uno dei record dell’utente, allora si chiama al plugin la funzione ‘saveScore’ con il nuovo punteggio. In caso contrario, si chiama ‘getRank’, specificando comunque il nuovo punteggio insieme ai valori di checksum. Il valore del punteggio viene comunicato in quanto il sistema deve verificare anche l’eventuale conquista di premi legati al superamento di determinati livelli. La chiamata a ‘saveScore’ restituisce sempre le classifiche aggiornate, di conseguenza i dati recuperati vengono poi salvati in memoria, inclusi i valori di checksum. Nel caso della chiamata di ‘getRank’ vale quanto detto al punto precedente relativamente alla gestione dei valori di ritorno del metodo.



Figura 27

Schermata di game over di un generico gioco caricato nella CordovaWebView

Funzione ‘getBest’

L’implementazione di questa funzione all’interno del plugin è realizzata con le sole comunicazioni dirette con Realtime Database, in quanto non ci sono particolari operazioni in aree riservate.

Essenzialmente, per il punteggio settimanale, viene recuperato il numero della settimana relativa al gioco in questione (chiamiamo *numberWeek* il numero trovato) e, con questo dato, si accede al valore del punteggio record dell’utente nella correlata classifica settimanale, presente nel nodo al percorso *users/userUID/game/gameId/weekly+numberWeek* (*gameId* indica l’identificativo del gioco e *userUID* contiene l’uid dell’utente loggato).

Per quanto riguarda il record globale, invece, si accede direttamente nel nodo al percorso *users/userUID/game/gameId/global*.

Nel caso in cui uno di questi due dati non dovesse essere presente, ovvero l’utente non occupa alcuna posizione in quella determinata classifica, viene impostato un valore di default gestito poi dallo script nella pagina web.

Funzione ‘getRank’

Questo metodo, nel plugin, va a chiamare una procedura in Firebase Functions che prende in input i seguenti parametri:

- **Id del gioco** (nominato ‘game’).
- **(Facoltativo) punteggio raggiunto** (nominato ‘score’), impostato solo se dalla WebView si riceve un valore valido.

- **Checksum crc32** delle due classifiche del gioco (nominati ‘crc32Global’ e ‘crc32Weekly’).

Questa funzione esegue soltanto due operazioni fondamentali, eseguite in due task differenti:

- **Controlla la vincita di un premio**, eseguito quando è rilevata la presenza del parametro ‘score’. Tale metodo, chiamato ‘checkRewardEarned’, viene spiegato nel capitolo 4.4.2.
- **Ritorna la classifica**, confrontando i valori di checksum passati. Questa fase prevede di recuperare il numero della settimana corrente del gioco in oggetto e successivamente ritornare quanto restituito dal metodo *returnRank({game}, {numberWeek}, {crc32Global}, {crc32Weekly}, false, false)*.

Funzione ‘saveScore’

La procedura in esame viene chiamata quando viene battuto un record dell’utente. Nonostante ciò, si effettuano comunque tutti i controlli del caso, in quanto, per ragioni di sicurezza, è meglio non affidarsi troppo al client. I parametri richiesti sono i seguenti:

- **Id del gioco** (nominato ‘game’).
- **Punteggio raggiunto** (nominato ‘score’).

La funzione, innanzi tutto, esegue, come in ‘getRank’, il controllo di eventuali vincite di premi, tramite la chiamata a ‘checkRewardEarned’, in un task separato. Successivamente, entra in azione l’elaborazione del punteggio raggiunto, con l’aggiornamento della classifica.

L’inserimento di una nuova posizione in classifica prevede sia l’aggiunta di un nodo all’interno della sezione dedicata nel RealTime Database, sia l’eliminazione del vecchio nodo rappresentante la propria vecchia posizione, se presente. Infatti, quando un utente batte un record il suo punteggio in classifica viene migliorato, e quello precedente scompare.

È stata inoltre aggiunta la funzione di imporre la priorità nelle graduatorie agli utenti che raggiungono per primi un determinato punteggio. Ad esempio, se *utente1* raggiunge il livello 5 e in secondo momento anche *utente2* arriva a totalizzare 5 punti, la classifica mostrerà sempre *utente1* in una posizione più prestigiosa di quella di *utente2*. Ovviamente questo viene gestito anche per le casistiche con più di 2 utenti in concorrenza.

La prioritizzazione viene implementata aggiungendo un valore progressivo che conteggia il numero di ingressi effettuati in un determinato punteggio. Quando si inserisce un nuovo oggetto utente nel nodo di un risultato, si reperisce, da quel percorso, l’oggetto utente già presente con l’indice progressivo più alto.

Se non è presente alcun dato allora significa che si dispone della massima priorità in quel punteggio e quindi l’indice, per l’oggetto utente che si vuole inserire, viene impostato a 1.

Altrimenti, viene recuperato l’indice in questione, rappresentante il più elevato di quella sezione, e si imposta per l’oggetto utente interessato un indice che vale quanto quello appena reperito incrementato di 1.

Si fa notare che quando un nodo torna a non contenere più alcun utente, a seguito di vari incrementi di record, allora il contatore progressivo viene azzerato. Infatti, al successivo ingresso di un nuovo utente il suo indice tornerà a valere 1.

Il diagramma schematico della parte di funzione incentrata alla elaborazione del nuovo punteggio viene illustrato nella figura 28.

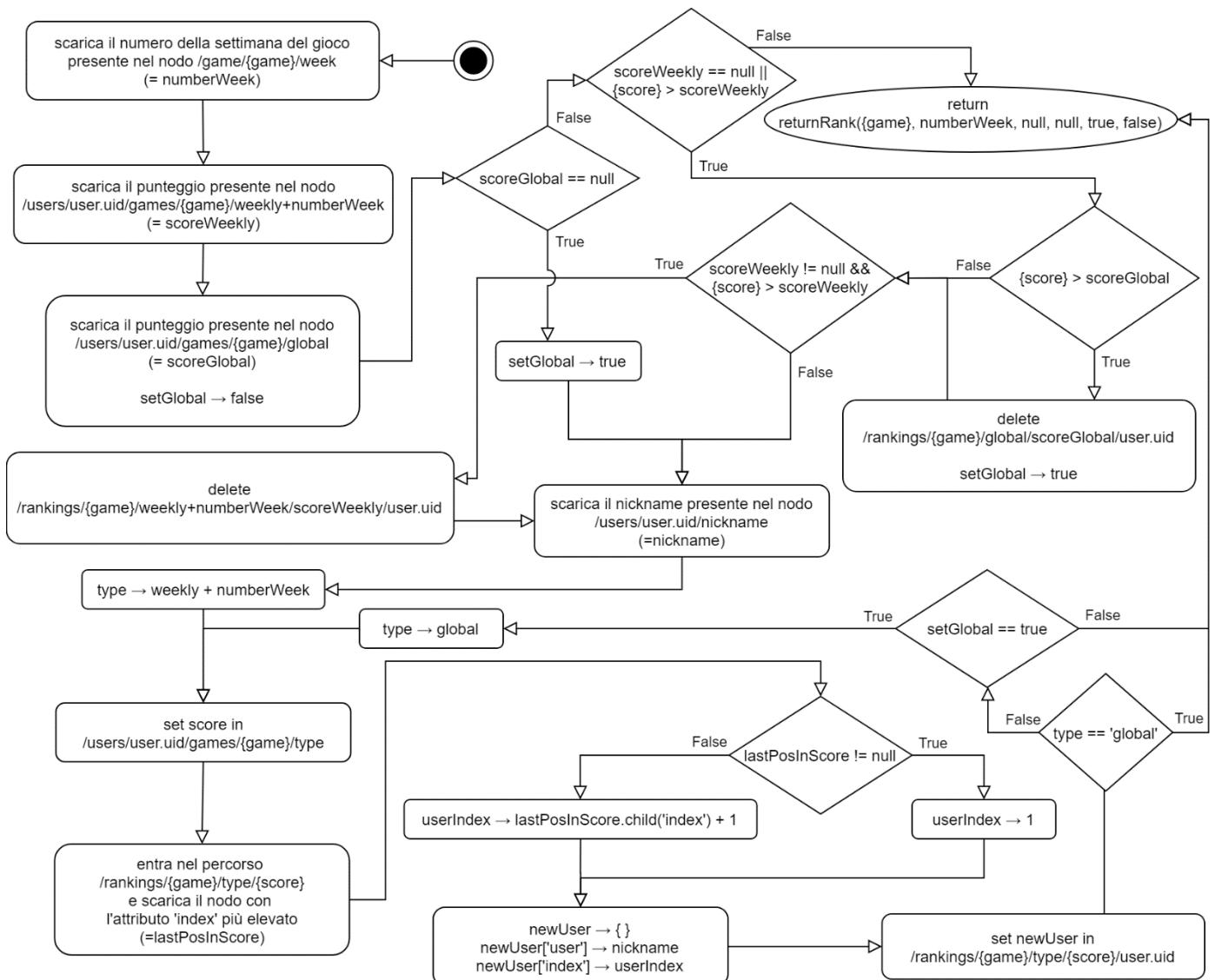


Figura 28
Diagramma riassuntivo della elaborazione di un nuovo punteggio nella funzione backend caricata

Una importante operazione non menzionata nello schema è l'incremento del contatore di partite giocate, situato nel relativo attributo ‘relevance’ del documento del ristorante, proprietario del gioco, contenuto nella raccolta /coordinates. Questo dato serve ad ordinare per rilevanza le ricerche dei locali, quindi ad ogni partita effettuata viene incrementato di 1.

L’incremento non avviene quindi soltanto nella funzione ‘saveScore’, ma anche quando si richiama ‘getRank’ a seguito di una partita.

4.4 Gestione premi

I premi sono l'aspetto più innovativo e coinvolgente che la piattaforma va a creare, per questo sono stati implementati con grande cura nella affidabilità e nella sicurezza.

4.4.1 RSA

In crittografia la sigla RSA^[19]^[20] indica un algoritmo di crittografia asimmetrica, inventato nel 1977 da Ronald Rivest, Adi Shamir e Leonard Adleman utilizzabile per cifrare o firmare informazioni.

Il sistema di crittografia si basa sull'esistenza di due chiavi distinte, che vengono usate per cifrare e decifrare, come illustrato nella figura 29. Se la prima chiave viene usata per la cifratura, la seconda deve necessariamente essere utilizzata per la decifratura e viceversa. La questione fondamentale è che, nonostante le due chiavi siano fra loro dipendenti, non è possibile risalire dall'una all'altra, garantendo in questo modo l'integrità della crittografia.

Il codice RSA si basa su un procedimento che utilizza i numeri primi e funzioni matematiche che è quasi impossibile invertire. Dati due numeri primi, è molto facile stabilire il loro prodotto, mentre è molto più difficile determinare, a partire da un determinato numero, quali numeri primi, dopo essere stati moltiplicati tra loro, hanno creato quel risultato. In questo modo si garantisce quel principio di sicurezza alla base della crittografia a chiave pubblica. Infatti, l'operazione di derivare la chiave segreta da quella pubblica è troppo complessa per venire eseguita in pratica. Vediamo di seguito l'algoritmo e le varie operazioni matematiche:

1. Calcolare il valore di n , prodotto di p e q , due numeri primi molto elevati (sono consigliati valori maggiori di 10^{100}). Negli esempi, in seguito, vengono usati numeri più piccoli per semplicità. Esempio: $n = p * q = 7 * 5 = 35$
2. Calcolare il valore di $z = (p - 1) * (q - 1) = (7 - 1) * (5 - 1) = 24$
3. Si sceglie un numero E tale che il massimo comun divisore di E e z sia 1. In formula matematica: $M.C.D.(E, z) = 1$. Quindi $E = 7$, in quanto $M.C.D.(7, 24) = 1$.
4. Trovare un numero D tale che $E * D \text{ mod } z = 1$, cioè che il resto della divisione tra $E * D$ e z sia 1. Di conseguenza, $D = 7$. Infatti, $49 \text{ mod } 24 = 1$.

A questo punto si procede con la cifratura calcolando $\mathbf{Mc} = M^E \text{ mod } n$. In ricezione, invece, per decifrare \mathbf{Mc} si calcola $\mathbf{Mc}^D \text{ mod } n$. Come si può capire da quanto appena detto per la cifratura si devono conoscere E ed n che quindi costituiranno la chiave pubblica, mentre per decifrare è necessario conoscere D ed n che quindi faranno parte della chiave segreta.



Figura 29
Schema crittografia asimmetrica

Utilizzo nell'applicazione

L'utilizzo di RSA è stato impiegato allo scopo di creare i codici QR, associati ad un particolare biglietto di utilizzo generato relativo ad un premio, al fine di permettere al ristoratore di verificare la validità del buono e riscattarlo.

Quindi ogni volta che un utente vince un determinato premio, il sistema genera ed elabora un oggetto particolare (chiamato 'ticket' da ora in avanti) che dispone di un suo identificativo e vari attributi legati ad esso, come la scadenza e, appunto, il codice QR, detto anche 'codice di riscatto'.

Nello scenario proposto vengono create tante coppie di chiavi quanti sono i ristoratori presenti sulla piattaforma. Ogni premio possiederà la chiave pubblica relativa al locale con cui si relaziona e, attraverso tale chiave, viene effettuata la criptazione di un particolare messaggio. Il testo cifrato corrisponde ad un codice sconto, relativo al ticket generato, trasformato in QR code nel dispositivo client che lo possiede.

Per ottenere una discreta sicurezza sono state utilizzate chiavi binarie di 2048 bit.

Il codice di riscatto, prima della sua criptazione, viene costruito come un oggetto JSON che contiene tutte le informazioni rilevanti per verificare la correttezza del ticket. La sua struttura viene composta da 3 valori:

- **UID dell'utente** che ha vinto il premio e quindi proprietario del codice sconto.
- **Identificativo del premio vinto**, con la quale si può reperire le informazioni sui termini e condizioni dello sconto guadagnato.
- **Identificativo del ticket generato**, relativo all'oggetto che possiede le informazioni uniche per il buono di proprietà dell'utente che provoca la sua creazione.

Il ristoratore, in seguito alla scansione di un codice QR esposto da un suo cliente, scatena la chiamata di una funzione in Firebase Functions che decripta il messaggio letto dalla scansione utilizzando la chiave privata presente nel suo specifico nodo utente.

È chiaro che la decriptazione ha successo solo se il messaggio è stato cifrato con la chiave pubblica corretta, questo evita il caso in cui un ristoratore tenti il riscatto di un codice sconto non di sua competenza.

Inoltre, si fa notare che se il processo avviene correttamente, allora, attraverso i dati ricavati, si può facilmente risalire alla posizione del ticket nel database per verificarne l'effettiva esistenza.

4.4.2 Erogazione ticket

I ticket vengono generati quando un utente vince un certo premio e questo può accadere in due situazioni distinte:

- In seguito alla **conquista di un obiettivo**. I premi che permettono ciò, per essere vinti, richiedono di arrivare a totalizzare o superare un determinato numero di livelli.
- Nel momento di **vincita di una classifica settimanale**, ovvero quando la settimana di applicazione della classifica settimanale termina e quindi si proclama il vincitore.

Oltre a ciò, viene gestito anche un sistema di notifiche che avverte l'utente della relativa vincita, così da poter mostrare l'informazione in tempo reale.

Conquista di un obiettivo

Questa tipologia di situazione si verifica alla presenza di un premio che permette la sua vincita attraverso questo metodo. Infatti, se il premio a cadenza settimanale esiste sempre per ogni ristorante, la presenza di questa categoria di premi è a discrezione del ristoratore, che può decidere quindi di aggiungerne quanti ne vuole con diversi obiettivi e, di conseguenza, diverse ricompense.

Questi premi detengono una data di inizio e di fine, in cui vengono effettuati i controlli sulla vincita ad ogni partita di gioco, e ogni utente ha la possibilità vincerli una sola volta. Se avviene l'erogazione di un ticket, a seguito della vincita di questo tipo di premio, l'utente che lo detiene non è abilitato a rivincere quel reward, anche nel caso in cui superasse nuovamente quel determinato obiettivo.

Il controllo quindi deve essere lanciato sempre ad ogni partita di gioco e interrotto nel caso in cui non si rilevano i premi interessati. Il metodo che effettua il controllo sulla vincita è stato già citato in precedenza nel capitolo 4.3.5 con il nome di ‘checkRewardEarned’. Questa procedura prende in ingresso i seguenti parametri:

- **UID dell'utente** che effettua la partita di gioco (nominato ‘user’).
- **Id del gioco** (nominato ‘game’) su cui effettuare i controlli.
- **Punteggio raggiunto** (nominato ‘score’), relativo alla partita effettuata.

Il diagramma schematico del metodo viene illustrato nella figura 30.

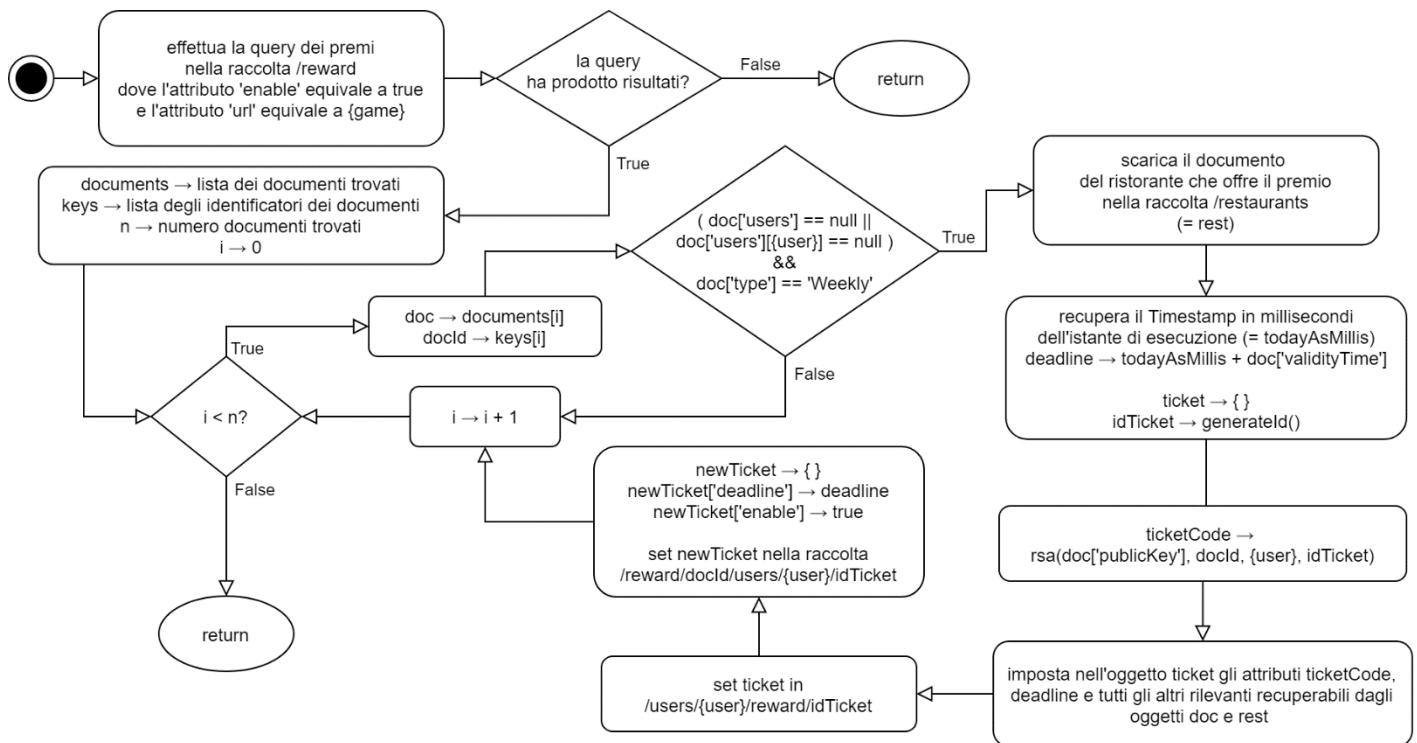


Figura 30
Diagramma riassuntivo del metodo ‘checkRewardEarned’ situato nel backend

La funzione essenzialmente effettua tutti i controlli di verifica e quando si deve generare un ticket calcola il codice di riscatto con l'algoritmo RSA prendendo in ingresso i parametri necessari. Inoltre, viene anche calcolata la data di scadenza per il ticket creato, sommando il Timestamp dell'ora rilevata e l'attributo di durata impostato nel documento del premio.

I dati necessari alla corretta implementazione del ticket sono poi impostati sia nel nodo dell'utente vincitore, sia nel nodo del premio, al fine di tenere traccia di tutti i possessori dello sconto.

Vincita di una classifica settimanale

La procedura che premia il vincitore di una classifica settimanale avviene in maniera completamente diversa da quella appena spiegata. Infatti, questa operazione viene eseguita autonomamente dal sistema, ovvero quando il timer del premio a cadenza settimanale conclude.

Firebase Functions permette di programmare funzioni temporeggiate ma la sua versione gratuita ne vieta l'utilizzo. Quindi, per fare ciò sono state utilizzate delle altre funzioni che hanno la particolarità di poter essere avviate attraverso una richiesta get ad un endpoint http. È stato utilizzato il servizio CronJob.org^[21] per poter programmare le richieste get a quel particolare endpoint con una cadenza giornaliera alle ore 00.00 (timezone: Europe/Rome).

Si illustra nella figura 31 il diagramma schematico della funzione.

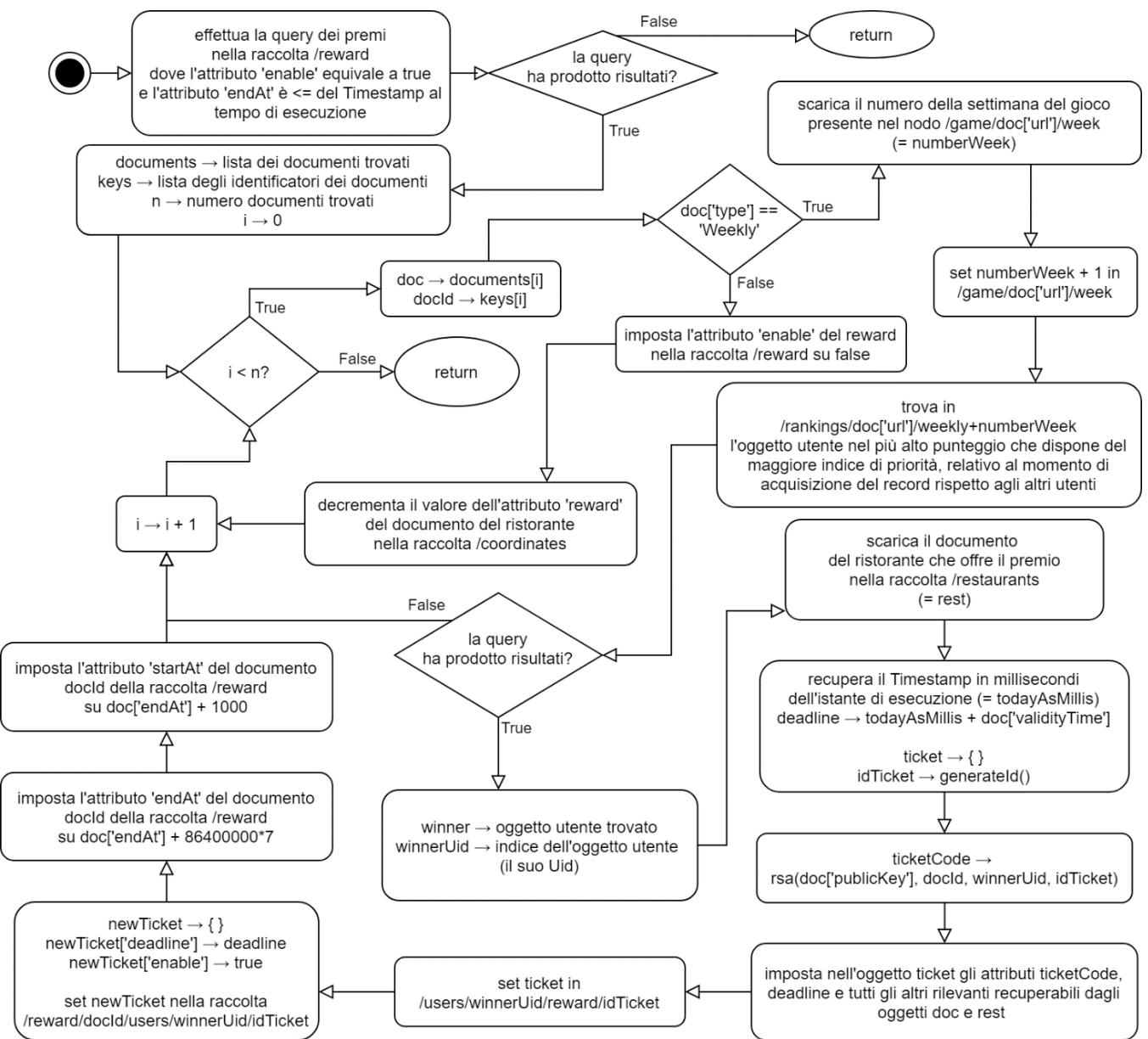


Figura 31
Diagramma riassuntivo dei controlli periodici sui premi situato nel backend

Con questo metodo è possibile controllare periodicamente lo stato dei premi per poterli disabilitare, se scaduti, e premiare i vincitori delle classifiche settimanali.

Essendo tale funzione chiamata soltanto in una determinata ora giornalmente, i premi sono costretti ad assumere quell'ora come unica opzione per la data che determina il passaggio dello stato di validità, in quanto il sistema deve intervenire subito per effettuare l'operazione.

Un dato importante è che i premi settimanali non vengono mai disabilitati, infatti si esegue un aggiornamento dei dati relativi alle date di scadenza e inizio, oltre all'incremento del contatore della relativa settimana, per poter permettere l'inizio di una nuova graduatoria.

I nuovi dati temporali sono impostati attraverso precisi calcoli:

- **Data di fine validità:** viene calcolata sommando al suo vecchio valore il numero di millisecondi equivalenti a 7 giorni, ovvero 86400000*7.
- **Data di inizio validità:** viene calcolata sommando al valore della obsoleta data di fine validità il numero di millisecondi equivalenti a 1 secondo, ovvero 1000. Questo viene fatto perché tutti i premi iniziano a valere alle ore 00.00.00 e scadono alle ore 23.59.59 di un qualsiasi giorno.

I premi che non rientrano nella categoria ‘Weekly’, invece, vengono disabilitati quando il loro periodo di validità termina e il loro stato non verrà mai resettato.

Alla disabilitazione di un premio si provvede anche a decrementare il relativo contatore nella raccolta ‘/coordinates’ che serve per poter fare ricerche e offrire il filtro di ordinamento sulla base della quantità di premi presenti.

L’effettiva erogazione del premio avviene determinando il vincitore, che quindi ha raggiunto il punteggio più alto della classifica prima di tutti gli altri, ed elaborando i dati al fine di creare e assegnare il relativo ticket.

Notifiche

Ogni qualvolta che avviene l’erogazione di un ticket per un utente, il sistema provvede anche a mandare notifiche personalizzate al vincitore, così da comunicare l’informazione in tempo reale.

Il sistema di notifiche è supportato da Firebase Cloud Messaging, che ne gestisce il funzionamento. L’implementazione prevede una funzione in Firebase Functions che viene lanciata tramite ‘trigger’ di creazione nel Realtime Database, ovvero alla creazione di nuovi nodi in determinati percorsi. In particolare, sono stati usati i valori jolly per poter indicare ogni nuovo ticket nel campo ‘reward’ di un qualsiasi nodo utente, quindi il percorso completo risulta ‘/users/\$uid/reward/\$idTicket’, dove si indica con \$ un elemento variabile.

Per poter usufruire del servizio di notifiche è stato necessario modificare il codice in modo che lo smartphone comunicasse al database i suoi notificationToken, identificativi del dispositivo e dell’applicazione che riceve la notifica.

I token possono cambiare in ogni momento e possono anche essere più di uno per account, nel caso in cui un utente faccia l’accesso su più dispositivi. Quindi l’implementazione, suggerita dalla documentazione di Firebase, imposta la scrittura di tutti i token in una sezione dedicata nel nodo utente.

Al fine di poter fornire un’esperienza migliore è stata aggiunta la funzione di mandare le notifiche tradotte all’ultima lingua rilevata dall’applicazione. Infatti, all’interno di ogni nodo

utente viene scritta, in uno specifico campo, la lingua in formato 2 lettere rilevata dai vari sistemi android e questa informazione viene aggiornata ad ogni accesso. In questo modo questo dato diventa facilmente reperibile e comunque abbastanza affidabile. Chiaramente, se una lingua non è supportata dal sistema viene impostato l'inglese come default, che resta sempre garantito.

La funzionalità può essere disabilitata per scelta dell'utente attraverso il comando presente nella scheda delle impostazioni dell'applicazione. Tale meccanismo impone la scrittura di un valore booleano nel nodo utente, che indica la volontà della ricezione delle notifiche verso i dispositivi identificati dai token presenti. Questo valore viene quindi controllato e valutato dalla funzione per agire nel modo corretto.

Se la procedura decide di inviare la notifica allora viene composta da:

- **Titolo**, ovvero la scritta che compare più in alto con un font maggiore. In questo campo viene inserito un messaggio di congratulazioni.
- **Corpo**, ovvero il testo che informa l'utente a cosa la notifica si riferisce nel dettaglio. Questa stringa viene formata in modo che comunichi il ristorante presso cui si è vinto il premio.
- **Immagine**, cioè il personaggio caratteristico della piattaforma. Così che possa sembrare essere lui l'interprete in prima persona delle comunicazioni.

Il metodo tenta di mandare la notifica a tutti i notificationToken presenti nel nodo utente analizzato e cancella quelli che portano ad un riscontro negativo.

Se la notifica si riceve con la piattaforma già avviata allora è stato implementato una funzionalità che permette di portare direttamente alla sezione dei premi vinti, semplicemente premendo sulla notifica stessa.

4.4.3 Gestione scadenze

In precedenza, è stato spiegato come vengono gestite le scadenze dei premi, ma oltre a queste anche i singoli ticket erogati dispongono di una scadenza propria che deve essere gestita più approfonditamente.

Infatti, sappiamo che la data di scadenza di ogni ticket viene calcolata nel proprio momento di generazione e non è possibile avere il controllo costante da parte del sistema sulle scadenze. Per questo motivo c'è il bisogno dell'aiuto del client e anche, soprattutto, del controllo nel momento dell'effettivo riscatto.

Nel database queste scadenze vengono gestite in maniera uguale a quanto accade per i premi, ovvero si utilizza una funzione su Firebase Functions che viene lanciata tramite richiesta get http dal servizio CronJob.org, ogni giorno alle ore 00.00.

Questa funzione si limita a fare la query di tutti i documenti nella raccolta reward per iterare ogni ticket erogato per quel premio e controllarne la validità.

Gli elementi non più validi sono trovati controllando il campo di validità abilitato e la data di scadenza minore dell'ora corrente di esecuzione. In caso un ticket fosse scaduto allora si disabilita modificando i suoi campi rilevanti sia su Firestore, sia su Realtime Database nel nodo utente del suo possessore.

Il diagramma schematico della funzione viene illustrato nella figura 32.

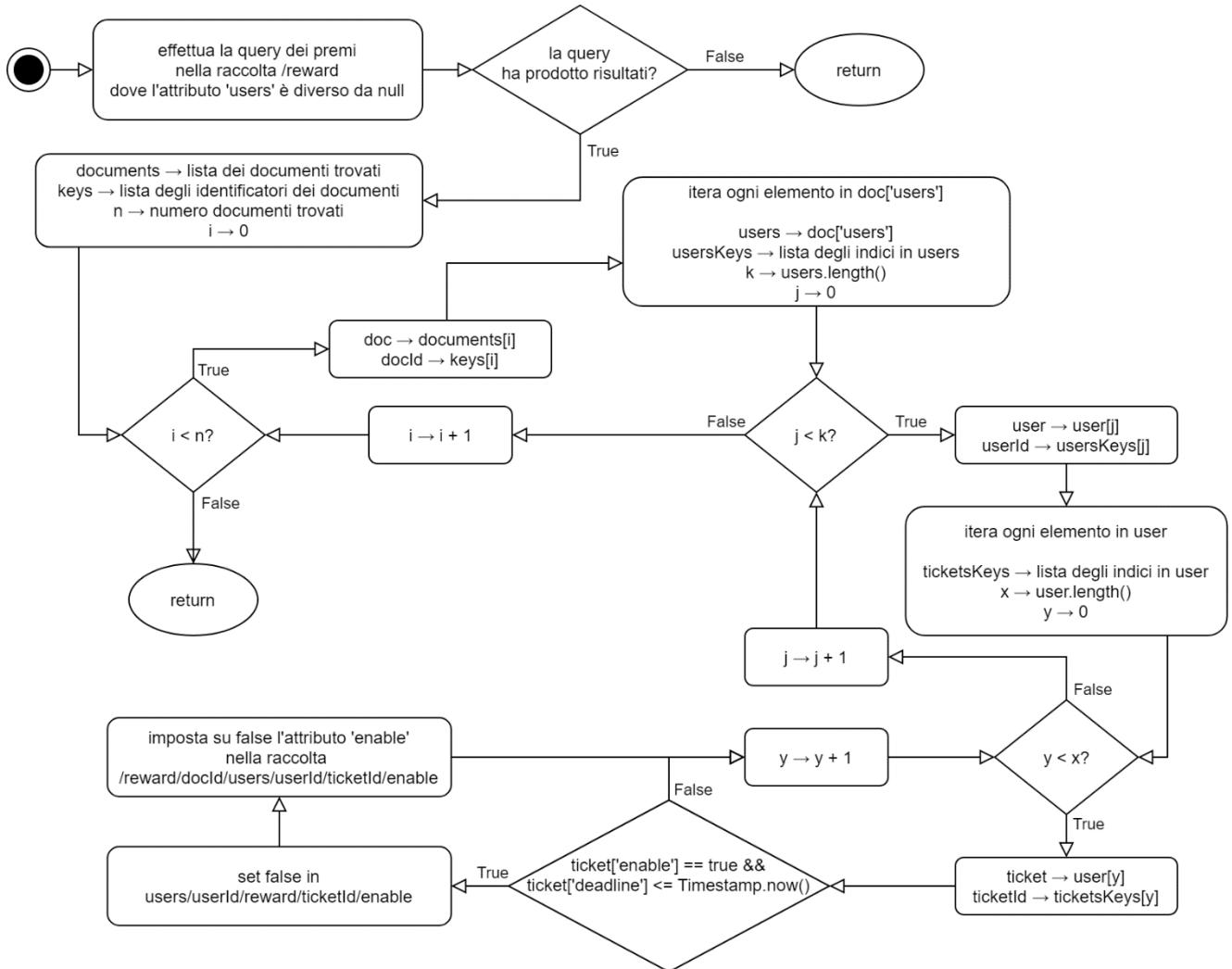


Figura 32
Diagramma riassuntivo dei controlli periodici sulle scadenze dei ticket situato nel backend

Il lato client però deve gestire a sua volta i ticket scaduti in modo che non li visualizzi all'utente in modo erroneo.

Di conseguenza la query effettuata al database, nel momento di ricezione dei premi da mostrare, chiede di ritornare i soli ticket che abbiano la data di scadenza maggiore del Timestamp dell'istante di esecuzione, oltre che essere validi. Quindi ogni ticket già scaduto non verrà preso in considerazione per il download. Inoltre, essendo queste informazioni salvate in un ViewModel, sono stati implementati dei controlli fatti dopo la prima ricezione che eliminano i ticket scaduti dalla lista salvata, nel caso in cui giunga la data limite mentre l'applicazione è in uso.

4.4.4 Riscatto

L'operazione di riscatto di un premio può avvenire soltanto da parte di un utente che dispone di particolari permessi, rilevati dal sistema. L'utente in questione viene riconosciuto come ristoratore e quindi proprietario di un videogioco, permettendogli di avere delle funzionalità aggiuntive che riguardano gli elementi di sua competenza.

In particolare, come visto nella figura 33, nella sezione del profilo utente gli sarà permesso di visualizzare il numero di partite giocate al proprio gioco e di poter riscattare i codici QR che gli vengono proposti dai suoi clienti.

Le partite giocate sono facilmente reperibili accedendo all’attributo ‘relevance’ nel documento, presente nella raccolta ‘/coordinates’ in Firestore, che identifica il suo ristorante, come spiegato nel capitolo 4.3.5.

Il dato che rileva lo stato di amministratore di un account è la presenza dell’attributo ‘admin’ all’interno del proprio nodo utente. Infatti, tramite i dati contenuti al suo interno è facile reperire le informazioni sopra citate. Queste operazioni, inoltre, sono eseguite tramite connessione diretta ai database Firestore e Realtime Database.

L’implementazione dei codici QR è stata effettuata con l’uso della libreria Zxing^[22] che permette sia di ricavare i dati tramite una scansione, sia di creare l’immagine rappresentante il codice soggetto all’analisi.

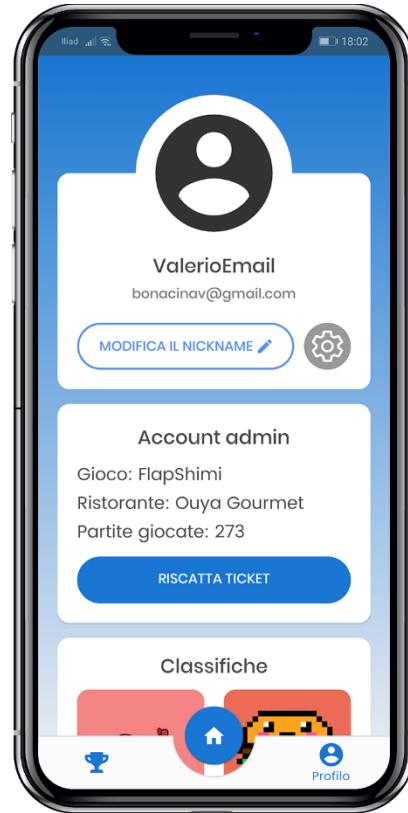


Figura 33
Schermata del profilo di un utente amministratore

Il riscatto di un codice prevede che l’utente beneficiario esponga il suo QR code, accessibile nella schermata di dettaglio del ticket dalla sezione dei premi, e il ristoratore acceda alla funzionalità dedicata alla scansione. Quando il codice viene scansionato, dall’account amministratore in questione, allora viene ricavato il codice generato tramite RSA, spiegato nel capitolo 4.4.1, del ticket in esame. A questo punto viene chiamata una funzione in Firebase Function che elabora il dato e restituisce il riscontro dell’operazione. La funzione richiede i seguenti parametri:

- **Codice scansionato** (nominato ‘ticketCode’), relativo al codice sconto del ticket che si sta riscattando.
- **Lingua** (nominato ‘lang’).

La procedura, come prima cosa, effettua il controllo della presenza dell’attributo ‘privateKey’ all’interno del nodo utente relativo all’account che effettua la chiamata. Tale valore serve per effettuare la decriptazione del codice e ricavare i risultanti dati.

Se la chiave è stata trovata e la decriptazione ha avuto successo allora si procede con il controllo effettivo dello sconto. In fase preliminare viene trovata la lingua più pertinente supportata, per mostrare al ristoratore il messaggio di risposta in modo comprensibile.

Successivamente si reperiscono i dati del ticket dell’utente che lo ha vinto, presenti nel documento della raccolta ‘/reward’. Attraverso lo studio di questi dati si scandiscono tre tipi di stati:

- **Ticket già usato** (nominato ‘TICKET_USED’), rilevato quando si segnala l’esistenza dell’attributo ‘randomDate’, ovvero la data in cui è stato già utilizzato lo sconto.

- **Ticket scaduto** (nominato ‘TICKET_EXPIRED’), rilevato quando la sua data di scadenza è precedente a quella attuale e non è presente la data di riscatto.
- **Ticket valido** (nominato ‘TICKET_VALID’), rilevato quando dispone di una scadenza non ancora sopraggiunta e non è presente la data di riscatto.

In tutti i casi nella risposta viene segnalato lo stato rilevato, ma con diversi allegati. Quando il ticket risulta già utilizzato in precedenza viene ritornata anche tale data, mentre quando questo risulta essere scaduto si riporta la scadenza.

Nel caso invece il codice sconto sia utilizzabile allora si procede con la sua disabilitazione, modificando i dati rilevanti nei database, e aggiungendo la conseguente data di riscatto. La chiamata, in questo scenario, comunica anche il nome del premio in oggetto e la stringa che descrive i suoi termini e condizioni.

Il diagramma schematico della funzione viene illustrato nella figura 34.

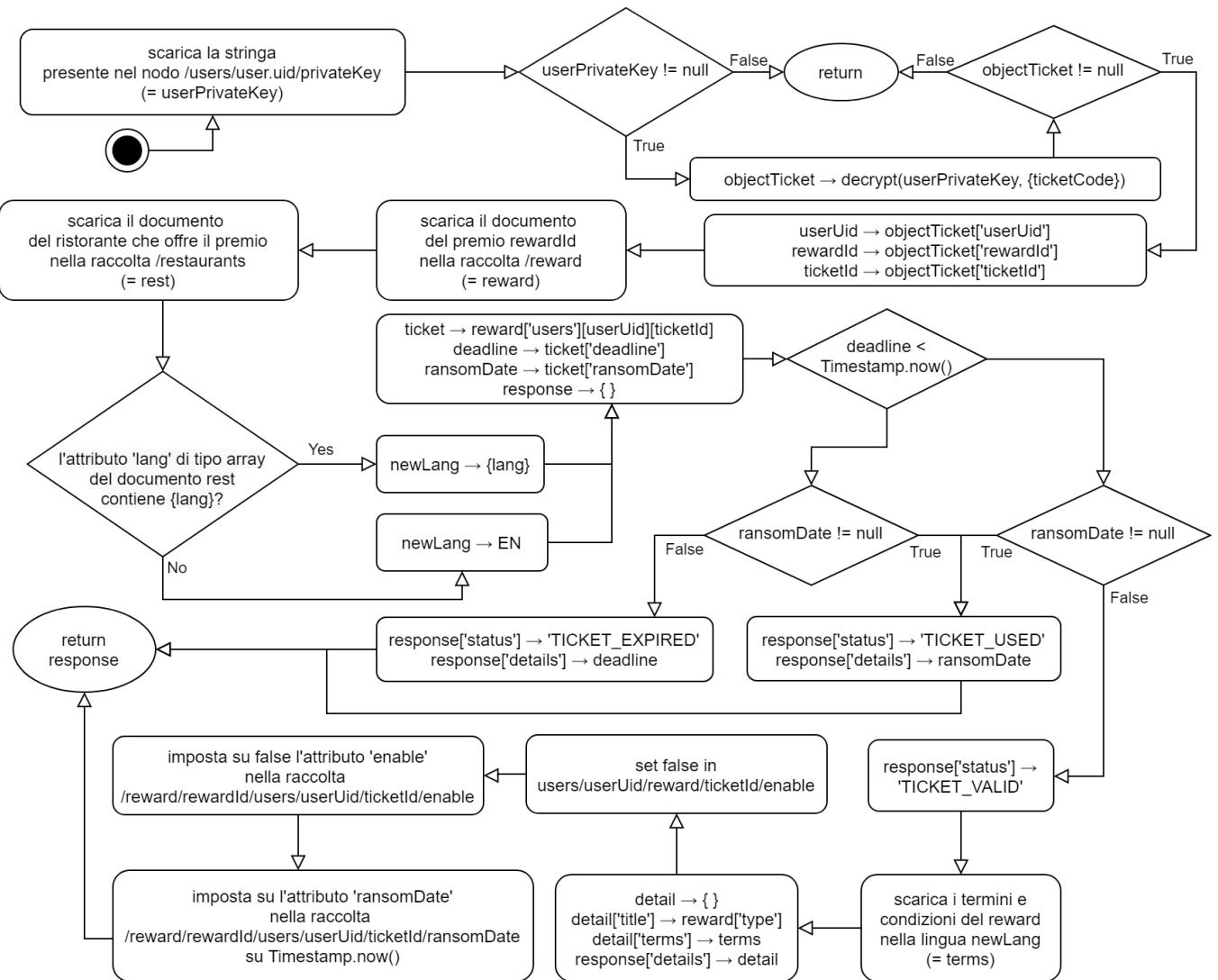


Figura 34
Diagramma riassuntivo della funzione backend che riscatta un ticket

Il client una volta ricevuto il riscontro dell'operazione di riscatto mostra al ristoratore tutte le informazioni ricevute attraverso una Snackbar, così da potergli consentire o negare la prestazione del suo servizio.

4.5 Salvataggio e ripristino stati

Gli stati vengono salvati e ripristinati tramite ViewModel all'interno di:

- BaseMapFragment (ripristino della visuale della mappa di GoogleMaps, lista dei Marker, Marker selezionato).
- HomeFragment (Ripristino contenuto e stato della BottomSheet).
- ListGameFragment (Ripristino lista dei risultati).
- ProfileFragment (Ripristino lista dei giochi utente nella sezione delle classifiche).

Questo dato viene scaricato accedendo direttamente all'oggetto 'games' in Realtime Database nel nodo dell'utente loggato. Infatti, in questa posizione sono presenti gli identificatori di tutti i giochi a cui l'utente abbia interagito almeno una volta, occupando una posizione in qualche classifica.

- AwardFragment (Ripristino lista dei ticket vinti dall'utente, incluse le informazioni relative ai singoli elementi).

La ricezione di queste informazioni è effettuata tramite connessione diretta all'oggetto 'reward' nel nodo relativo all'utente che effettua la richiesta. In questo contesto si presta particolare attenzione alle scadenze dei ticket (spiegato nel capitolo 4.4.3), al loro stato di validità e alla corretta versione linguistica delle stringhe da reperire.

La lista mostrata viene ordinata per data di inizio, quindi i premi appena vinti verranno mostrati per primi nella schermata e quelli più vecchi saranno posizionati più in basso.

Inoltre, è stata impostato il repository del ViewModel in modo che resti in ascolto di quanto accade nel percorso interessato all'interno del database, così che la lista dei ticket vista dall'utente sia sempre sincrona a quella online, con le dovute condizioni di validità.

Infatti, nel momento di riscatto di un ticket, questo viene disabilitato e quindi se il beneficiario sta visualizzando il buono, in quell'istante viene aggiornata la sua schermata, chiudendo la relativa pagina di dettaglio (se aperta) e rimuovendo dalla lista dei premi il ticket appena usato.

- SettingsFragment (Ripristino impostazione delle notifiche, impostata su 'true' di default).

Anche questo dato viene reperito tramite collegamento diretto al database, semplicemente con la lettura del relativo attributo nel nodo dell'utente loggato. Se questo dato non risulta presente allora per impostazione predefinita il servizio viene considerato come attivo.

In ListGameFragment viene inoltre salvato in una variabile l'ultimo oggetto scaricato che contiene i dati di dettaglio di un gioco, quando si è nella lista dei giochi recenti (RecentGamesFragment). Questo riduce il numero di download inutili ma, per evitare di mantenere valide informazioni obsolete, questa variabile non viene ripristinata a seguito della ricreaione del fragment come gli altri elementi.

I fragment relativi alle ricerche e quello della pagina di home sono preservati nello stack in modo che possano essere recuperati invece che ricreati. Infatti, nel momento di passaggio da una sezione all'altra nell'applicazione, i fragment principali vengono sostituiti ma quello di home viene semplicemente recuperato dallo stack, con i dovuti ripristini delle variabili. Lo stesso discorso vale per quanto riguarda il fragment di ricerca, che viene recuperato quando si preme il tasto indietro dopo aver aperto la SearchView con una ricerca già impostata nella BottomSheet.

Le schermate che si mostrano sopra la Bottom Navigation nell'applicazione non vengono mai ripristinate ma distrutte, il motivo è relativo al bisogno di salvare l'elemento View che necessita MaterialDesign per poter eseguire le animazioni inverse.

Gli elementi che necessitano di un ripristino anche dopo la chiusura dell'app vengono salvati nelle SharedPreferences. Rispettivamente gli elementi interessati sono:

- Cronologia di ricerca (Fino a 10 elementi, dopo il decimo la lista si comporta seguendo il metodo FIFO)
- Lista dei giochi recenti (Fino a 10 elementi, dopo il decimo la lista si comporta seguendo il metodo FIFO)
- Filtro di ricerca nell'area circostante all'utente (Ordinamento, distanza)
- Filtro di ricerca dalla SearchView (Ordinamento)
- Ultima area visualizzata sulla mappa (Oggetto che comprende latitudine, longitudine, bearing, tilt e zoom)

5 Interfaccia Grafica

Il successo di un'applicazione non passa solamente dalle sue funzionalità, ma anche da come queste vengono presentate a livello visuale.

Avere un'interfaccia intuitiva e piacevole alla vista è dunque determinante per garantire all'utente un'esperienza d'uso appagante.

In tal senso ci si è concentrati molto sugli aspetti visivi, lavorando anche sui piccoli dettagli e sulle animazioni.

Questo capitolo verte intorno alle soluzioni grafiche adottate nel progetto e alle motivazioni di tali scelte.

5.1 Poncho: mascotte e aiutante

Trattandosi GamePlate di un prodotto nuovo, si è ritenuto importante che all'utente fosse fornita fin da subito assistenza per comprendere il funzionamento dell'app.

Per fare ciò si è scelto di inserire tra le varie schermate una mascotte, con il compito di fornire informazioni chiave nei momenti principali dell'esperienza d'utilizzo.

Questa scelta non è di certo una novità nel mondo ICT, pensiamo ad esempio a Clippit: la celebre graffetta che fungeva da assistente in Microsoft Office 97, poi sparita nelle versioni successive per dare spazio a una più minimale barra di ricerca.



Figura 35
Logo di GamePlate

Nel contesto di GamePlate la figura di una mascotte/aiutante non stona, fornisce anzi all'utente un'interazione più personale e umana rispetto a dei semplici output testuali.

Dopo aver ponderato varie alternative, alla fine è nato Poncho: un assistente dalle fattezze di un panda antropomorfo.

La sua figura è stata scelta come simbolo dell'app, compare dunque nel logo (figura 35) e anche in molteplici occasioni durante l'esperienza d'uso (figura 36). Ciò contribuisce anche a rendere il prodotto GamePlate più riconoscibile e impresso nella memoria degli utenti.

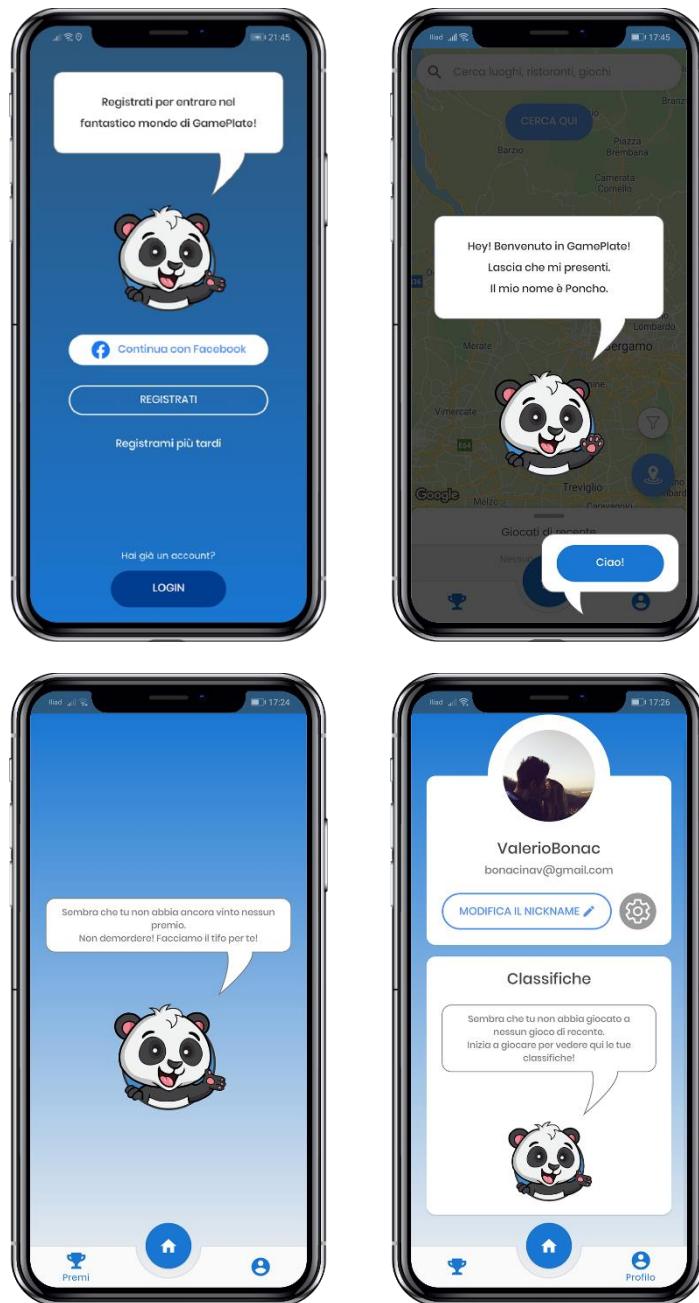


Figura 36
Occasioni in cui subentra la figura di Poncho

5.2 Registrazione e login

La prima schermata che appare quando si installa l'applicazione è quella legata alla registrazione o login del proprio account, mostrata nella figura 37.

Ad accogliere l'utente è Poncho, che lo invita a scegliere una delle opzioni disponibili.

L'interfaccia, in ogni caso, è piuttosto intuitiva e si rifà allo stile utilizzato da numerose altre applicazioni.

L'utente può rapidamente scegliere se registrare un nuovo profilo, accedere a uno già esistente oppure proseguire utilizzando il proprio account di Facebook. Viene anche lasciata la possibilità di saltare questa fase e procedere con l'esplorazione dell'app.

Le schermate che si aprono alla pressione dei pulsanti ‘Registrati’ e ‘Login’ sono entrambe molto semplici e intuitive.

In entrambi i casi i parametri richiesti sono l’indirizzo e-mail e la password (da inserire due volte nel caso di registrazione). È comunque sempre possibile tornare alla schermata precedente tramite il pulsante ‘Indietro’ in alto a sinistra.

Nel caso del login poi, è presente anche l’opzione di recuperare la password nel caso fosse stata dimenticata.



Figura 37
Schema di funzionamento della fase di registrazione

5.3 Schermata Home

Una volta superata la fase di registrazione/login si accede alla Home (figura 38), schermata tra le più ricche di elementi grafici.

Gran parte dello schermo viene occupato da una mappa, sulla quale verranno visualizzati i giochi trovati.

Nella parte alta è presente una barra di ricerca, lungo il lato destro sono invece presenti i pulsanti ‘Filtri’ e ‘Trova intorno a me’.

Nella zona bassa infine, è presente una Bottom Sheet dove è possibile prendere visione dei giochi recenti o dei risultati di una ricerca.



Figura 38
Schermata Home

5.3.1 Bottom Sheet

Nella parte bassa della schermata si è scelto di introdurre una Bottom Sheet contenente una RecyclerView con all'interno una lista di CardView, la cui forma varia a seconda che l'utente abbia effettuato una ricerca o meno.

Nel primo caso, viene mostrata una carta per ogni gioco trovato, essa contiene informazioni sul ristorante associato.

Altrimenti, se non è stata effettuata alcuna ricerca, vengono mostrati i giochi recenti dell'utente. La CardView in questo caso cambia aspetto: spariscono alcuni dettagli sul ristorante e appare il pulsante Gioca, così da lasciar entrare rapidamente in partita l'utente. La Bottom Sheet può essere posizionata in tre differenti modalità (mostrate nella figura 39): contratta, estesa a metà schermo oppure a tutto schermo. A seconda di ciò si spostano anche i vari pulsanti sulla mappa.



Figura 39
Le varie posizioni che la bottom sheet può assumere

In particolare, quando viene estesa alla sua massima lunghezza, si fonde con la barra di ricerca in alto tramite un'animazione che nasconde completamente la mappa retrostante.

5.3.2 Ricerche

Dalla Home è possibile effettuare ricerche tramite tre diverse modalità: tramite parola chiave, tramite il pulsante ‘Trova intorno a me’ e tramite il pulsante ‘Cerca qui’.

Ricerca tramite keyword:

Questa modalità può essere utilizzata tramite l'apposita barra in alto, realizzata per mezzo di una SearchView.

Quando viene premuta, la Bottom Sheet si estende a tutto schermo bloccandosi e mostrando la cronologia di ricerca (figura 40).

Inserendo una parola chiave (come un ristorante, un gioco o un luogo) e premendo invio si avvia la ricerca.

È inoltre possibile selezionare l'ordine con cui mostrare i risultati (rilevanza o numero di premi) attraverso un radio button custom.

Barra di ricerca testuale

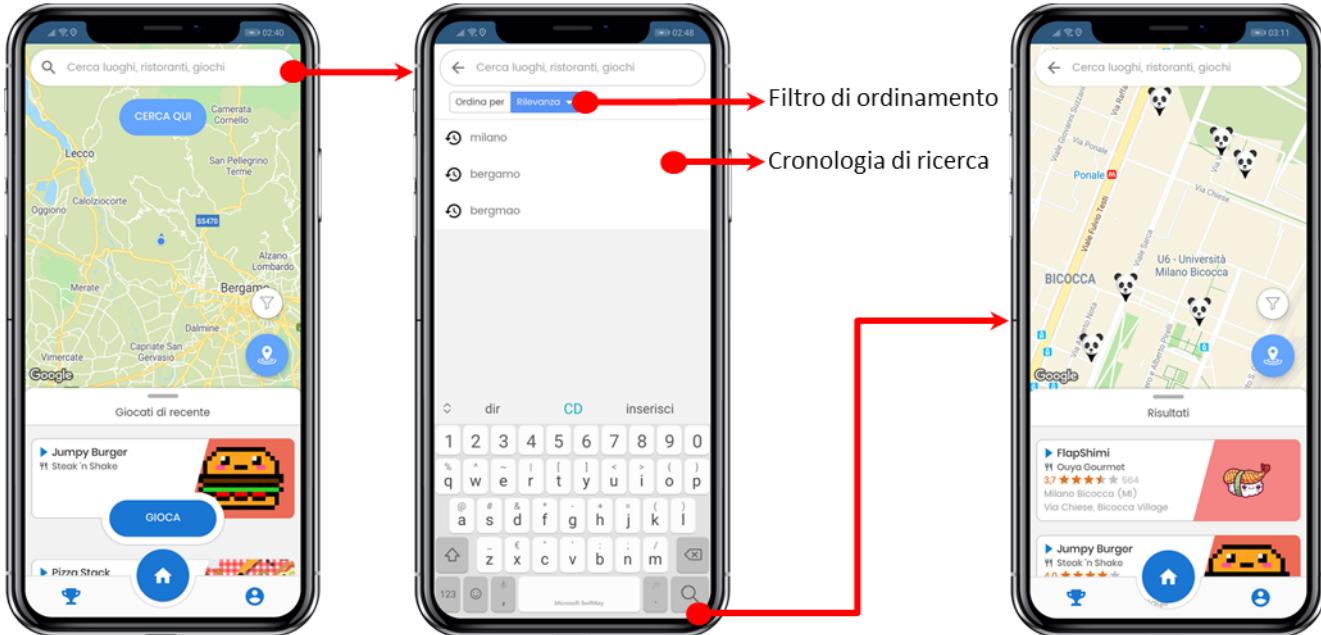


Figura 40

Funzionamento delle ricerche tramite keyword

Ricerca nei dintorni:

Questa modalità è accessibile tramite l'utilizzo dei due floating action button (FAB) presenti sul lato destro della schermata (mostrata nella figura 41): ‘Trova intorno a me’ e ‘Filtri’.

Filtro di ricerca

Risultati di ricerca nell'area circostante l'utente

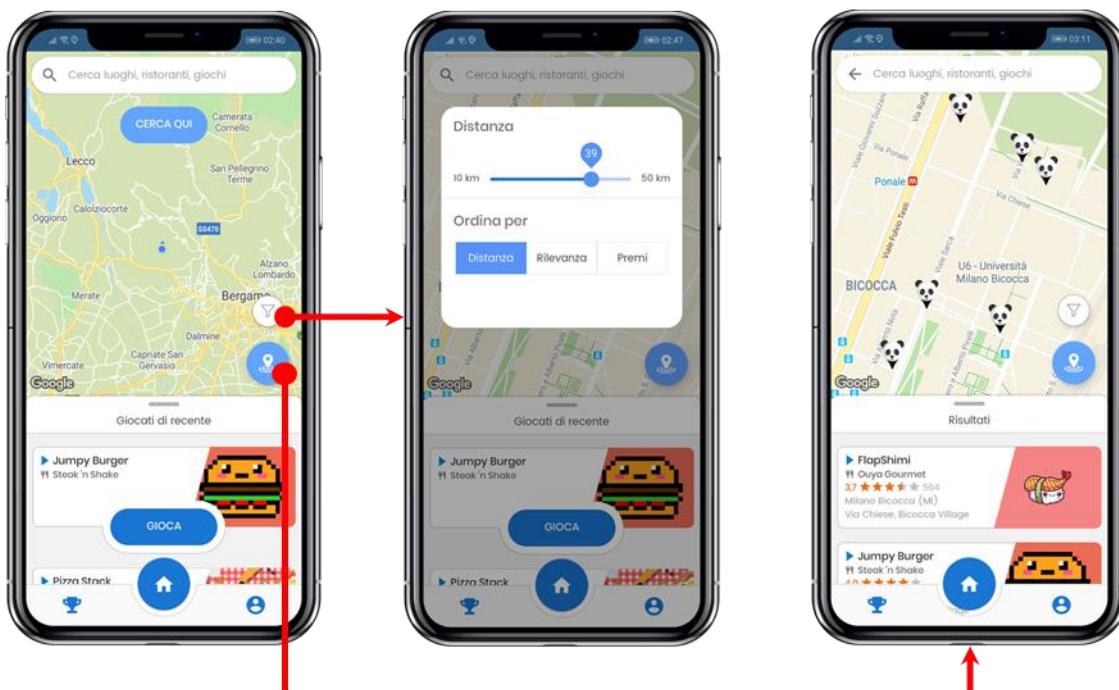


Figura 41

Funzionamento delle ricerche dei ristoranti/giochi nei dintorni

Tramite il primo è possibile effettuare una ricerca nei dintorni dell'utente, il quale viene geolocalizzato tramite GPS.

Premendo il secondo invece, viene mostrata una Cardview tramite cui impostare la distanza massima e un criterio di ordinamento (distanza, rilevanza, numero di premi).

Entrambi gli elementi, realizzati rispettivamente con uno slider e con un radio button, sono stati customizzati per adattarli all'estetica del resto dell'app.

Cerca qui:

Tramite il pulsante ‘Cerca qui’, posizionato nella parte alta dello schermo, si avvia una ricerca nei dintorni del luogo che si sta guardando sulla mappa. Il tasto scompare quando viene premuto (dopo un’animazione di caricamento) poiché non avrebbe senso premerlo nuovamente senza spostare la visuale sulla mappa (verrebbero caricati di nuovo gli stessi risultati). Appena l’utente muove la mappa, il pulsante torna visibile e pronto alla pressione.

In tutti i casi, i risultati vengono mostrati sia sotto forma di marker posizionati sulla mappa, sia in una lista nella Bottom Sheet.

5.4 Dettaglio di un gioco

Quando si preme su un gioco nella Bottom Sheet viene aperta una nuova schermata contenente maggiori dettagli a riguardo, mostrata nella figura 42.

L’elemento principale che si presenta davanti all’utente è il tasto ‘Gioca’, in modo tale da farlo entrare in partita il più agilmente possibile.

Tra le informazioni disponibili ci sono poi una breve descrizione del gioco e le informazioni sul ristorante correlato.

Due pulsanti consentono di visitare il sito del ristorante (il primo) e di visualizzare nuovamente il marker sulla mappa (il secondo).

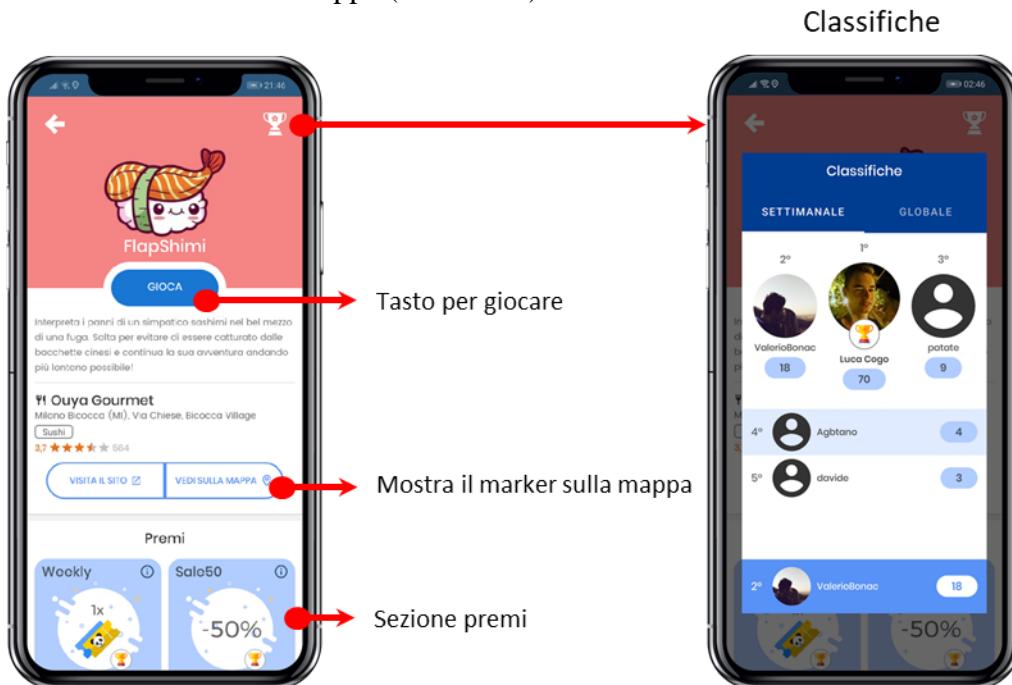


Figura 42
Schermata contenente i dettagli di un gioco

È presente poi una sezione dedicata ai premi attualmente in palio. Ciascuno è rappresentato da una CardView con indicato il tipo di ricompensa, l'obiettivo da raggiungere per vincere e un conto alla rovescia che indica per quanto ancora sarà disponibile.

Premendo sul pulsante ‘Info’ si apre una finestra di dialogo contenente i termini e le condizioni di utilizzo.

Nell'angolo in alto a destra infine, è presente un pulsante a forma di coppa per consultare le classifiche. Premendolo si apre un'altra finestra di dialogo contenente due liste separate: la prima consente di vedere i migliori sette giocatori della settimana, la seconda invece mostra i migliori punteggi di sempre. In entrambi i casi viene mostrata anche la posizione dell'utente.

5.5 Premi

Attraverso la barra di navigazione posta nella parte bassa dello schermo, premendo il pulsante a forma di trofeo, è possibile spostarsi alla sezione relativa ai premi vinti, mostrata nella figura 43.

In questa schermata si è optato per l'utilizzo di un ViewPager2, composto da una serie di Cardview (una per premio). In questo modo l'effetto ottenuto è quello di una lista scorribile verticalmente. Lo sfondo, invece, è realizzato tramite un gradiente che varia a seconda del colore principale dell'immagine nella carta.

Per ogni premio vengono indicati il ristorante correlato, il tipo di ricompensa e un conto alla rovescia entro il quale deve essere riscattato.

Premendo sulla card viene aperta una nuova schermata contenente, oltre alle informazioni già presenti nella carta, un codice QR (di cui si può fare lo zoom) e un pulsante che apre una finestra di dialogo riportante maggiori dettagli sull'offerta.



Figura 43
Sezione relativa ai premi vinti

5.6 Profilo

Sempre attraverso la barra di navigazione, premendo il pulsante sulla destra, è possibile anche accedere alla sezione relativa al proprio profilo, mostrata nella figura 45.

La schermata si presenta con una Cardview contenente una foto profilo (ottenuta da Facebook), il nome dell'utente e i pulsanti ‘Modifica nickname’ e ‘Impostazioni’.

Sulla sezione dello schermo immediatamente sottostante, è posizionata un'altra Cardview con all'interno una RecyclerView che contiene a sua volta una carta per ogni gioco a cui l'utente ha giocato. Anche da qui infatti è possibile accedere alle classifiche, questa volta visualizzate a schermo intero e non più sotto forma di finestra di dialogo.

Alla pressione di ‘Modifica nickname’ viene aperta un'altra scheda pop-up che consente di inserire un nuovo nome utente.

Tramite il pulsante ‘Impostazioni’ (simbolo dell'ingranaggio) si passa invece a una nuova schermata contenente nell'ordine:

- Un toggle button per attivare/disattivare le notifiche
- Il pulsante ‘Contattaci’ che rimanda, tramite un Intent, all'app di posta elettronica
- Il pulsante ‘Esci’ che consente di effettuare il logout

Se l'account dell'utente è di tipologia ‘Admin’, ovvero il profilo di un ristoratore, allora è presente un'ulteriore sezione (sempre contenuta in una Cardview) che riporta alcuni dettagli legati al proprio gioco e un pulsante per riscattare i ticket dei clienti (figura 44).

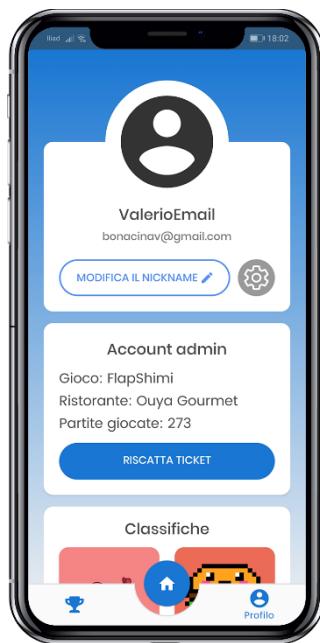


Figura 44
Cardview aggiuntiva per l'account admin

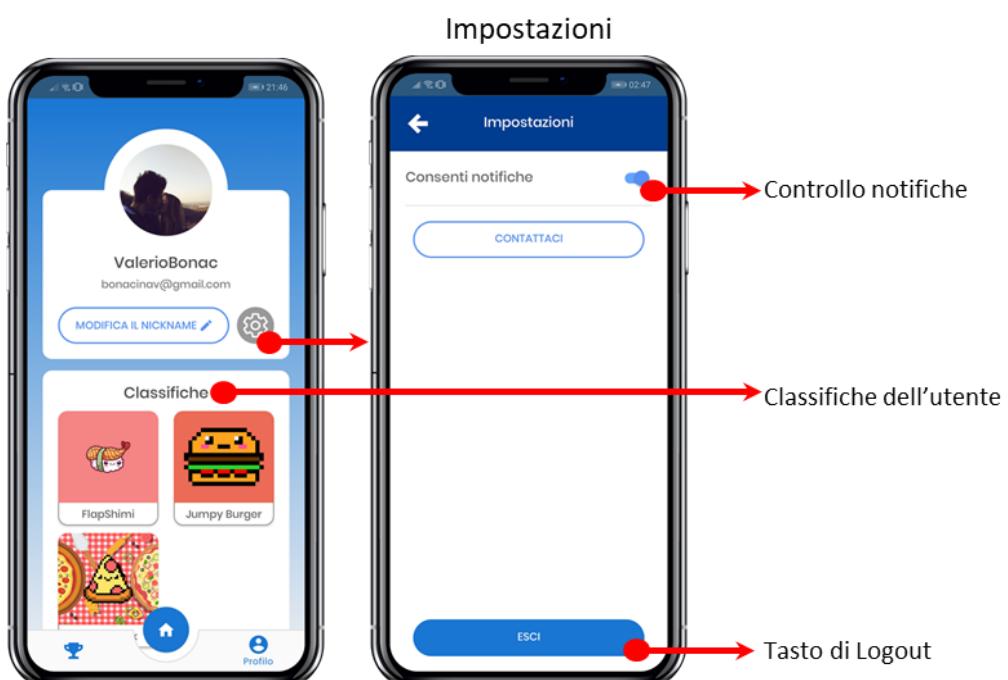


Figura 45
Sezione relativa al profilo dell'utente

6 Analisi di usabilità

Secondo lo standard ISO 9241^[23], per usabilità si intende “*il grado con cui un prodotto può essere usato da determinati utenti per raggiungere determinati obiettivi con efficacia, efficienza e soddisfazione in un determinato contesto d'uso*” intendendo:

- Efficacia come precisione e completezza con cui gli utenti raggiungono specifici obiettivi
- Efficienza come risorse impiegate in relazione alla precisione e completezza con cui gli utenti raggiungono specifici obiettivi
- Soddisfazione come libertà dal disagio e attitudine positiva con cui gli utenti raggiungono specifici obiettivi attraverso l'uso del prodotto.

Un’analisi di usabilità consiste in una serie di test effettuati su un campione di utenti con l’intento di valutare la loro esperienza d’utilizzo. Questo tipo di studio, a seconda del contesto, può svilupparsi con due modalità distinte: confronto trasversale e confronto longitudinale.

Nel caso di un confronto trasversale si prendono in esame due prodotti concorrenti e si cerca di stabilire in quali aspetti uno è migliore dell’altro.

Se invece si effettua un confronto longitudinale, illustrato nella figura 46, si ha a che fare con due versioni dello stesso prodotto e se ne effettua una comparazione.

Nello specifico caso di GamePlate, non esistendo un vero e proprio competitor, si è scelto di procedere con questa seconda modalità. Le fasi dell’analisi si possono ricondurre alle tre attività seguenti, che verranno poi spiegate in dettaglio nei prossimi paragrafi:

- Valutazione euristica: un’analisi qualitativa sull’usabilità dell’app, effettuata da un gruppo poco numeroso
- Test Utente: una serie di interviste agli utenti per quantificare efficienza ed efficacia
- Questionario: una serie di domande sottoposte agli intervistati per quantificare il loro grado di soddisfazione

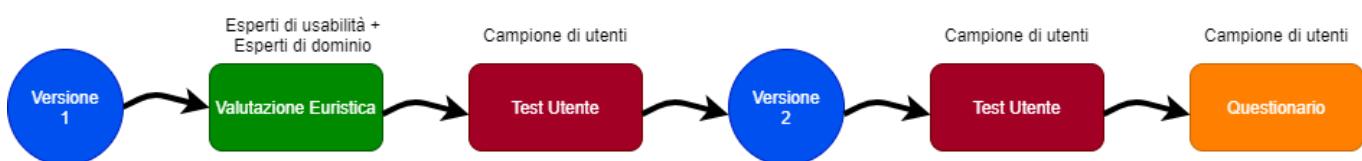


Figura 46
Schema dei passaggi seguiti in un confronto longitudinale

6.1 Valutazione euristica

Come accennato precedentemente, una valutazione euristica è un'analisi di tipo qualitativo orientata a verificare se il prodotto in esame rispetta i principi fondamentali dell'usabilità.

In questa fase sono coinvolte due categorie di individui:

- Esperti di usabilità: Gli autori dell'analisi di usabilità stessa
- Esperti di dominio: Utenti considerati esperti del contesto da studiare (nel caso GamePlate esperti di videogiochi e/o ristoranti)

L'applicazione viene presentata agli utenti esperti singolarmente e si chiede di analizzarla in modo critico, individuando eventuali problemi.

Al termine delle interviste, gli esperti di usabilità si riuniscono e rielaborano i risultati raccolti, cercando di classificare i problemi individuati.

6.1.1 Ambiente di valutazione e composizione del campione

Nel caso di GamePlate, allo scopo di simulare in tutto e per tutto la prima esperienza di un utente che approccia il servizio, si è scelto di effettuare i test con l'app appena installata sul dispositivo.

Si è ritenuto, infatti, che anche le fasi iniziali dell'esperienza di utilizzo (registrazione o login) necessitassero di essere messe alla prova.

Questi momenti iniziali sono decisivi: la prima impressione dell'utente è determinante per ottenere poi la permanenza dello stesso.

In questa fase si è scelto di far partecipare due esperti di usabilità e tre esperti di dominio, descritti nella tabella 3 e nella figura 47.

A questi ultimi inizialmente è stato concesso di esplorare liberamente l'applicazione nella 'Release 1' per qualche minuto (analisi olistica) e successivamente sono stati sottoposti tre task da svolgere (Appendice A.1).

ID	Genere	Professione	Età	Familiarità con app simili
EU1	Maschio	Studente informatica	Tra i 16 e i 24 anni	Alta
EU2	Maschio	Studente informatica	Tra i 16 e i 24 anni	Alta
ED1	Maschio	Impiegato	Tra i 16 e i 24 anni	Media
ED2	Maschio	Studente superiori	Tra i 16 e i 24 anni	Bassa
ED3	Maschio	Studente biologia	Tra i 16 e i 24 anni	Alta

Tabella 3
Partecipanti alla valutazione euristica.
(EU = esperti di usabilità, ED = esperti di dominio)

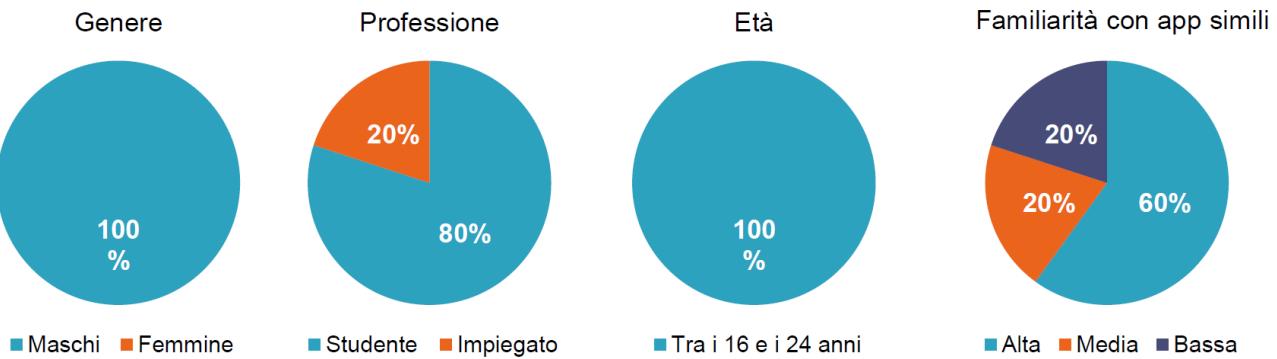


Figura 47
Composizione del campione di utenti

Sebbene a prima vista il campione possa sembrare poco variegato per quanto riguarda genere ed età, in questa fase si è ritenuto più importante il livello di familiarità con app simili.

6.1.2 Euristiche

I problemi individuati durante le interviste sono stati classificati seguendo le 10 linee guida redatte nel 1990 dall'esperto di usabilità Jakob Nielsen^[24], qui riportate nella loro traduzione in italiano:

- **E1: Far vedere lo stato del sistema**

L'utente dovrebbe avere sempre la percezione di cosa sta facendo. Perché questo si verifichi bisogna fornirgli un adeguato feedback necessario per chiarire la corrispondenza tra le azioni che può compiere e gli effetti che ne conseguono.

- **E2: Corrispondenza tra sistema e mondo reale**

Il linguaggio utilizzato deve essere semplice e di immediata comprensione. A questo scopo è meglio evitare tecnicismi e termini stranieri e strutturare i contenuti in maniera logica, chiara e funzionale.

- **E3: Controllo utente e libertà**

L'utente deve essere messo in grado di rimediare ai suoi errori attraverso "uscite d'emergenza" evidenti e funzioni che evitano percorsi complicati e snervanti.

- **E4: Assicurare consistenza**

È opportuno usare le convenzioni generalmente acquisite e mantenerle inalterate per evitare di confondere gli utenti.

- **E5: Riconoscimento piuttosto che memoria dell'utente**

Per ridurre lo sforzo mnemonico dell'utente, l'interfaccia dovrebbe rendere visibili gli strumenti e le opzioni disponibili e fornire le istruzioni per un suo corretto utilizzo.

- **E6: Assicurare flessibilità ed efficienza d'uso**

È opportuno prevedere diversi livelli di utilizzo del sistema, supportando una navigazione di tipo gerarchico per i navigatori meno esperti ma inserendo anche opportune scorciatoie per gli utenti abituali.

- **E7: Visualizzare tutte e sole le informazioni necessarie**

È opportuno evitare di inserire informazioni irrilevanti in quanto ogni dato superfluo toglie visibilità a ciò che è invece effettivamente importante, distraendo l'utente ed appesantendo la pagina.

- **E8: Prevenire gli errori**

Oltre a prevedere eventuali messaggi d'errore, è opportuno cercare di limitare il più possibile gli sbagli dell'utente.

- **E9: Permettere all'utente di correggere gli errori e non solo di rilevarli**

I messaggi di errore dovrebbero essere espressi in un linguaggio chiaro e non attraverso codici, indicando precisamente in cosa consiste il problema e come fare a risolverlo.

- **E10: Help e documentazione**

Anche se il sistema dovrebbe essere usabile senza documentazione, potrebbe essere necessario inserire strumenti di aiuto e istruzioni. In questo caso ogni informazione dovrebbe essere facilmente rintracciabile, focalizzata sul compito dell'utente e strutturata in più passaggi.

6.1.3 Prioritizzazione

Una volta completata la valutazione euristica, è stato sottoposto al campione un questionario volto a determinare la gravità dei problemi individuati. Per ciascuno di essi si è chiesto di esprimere il grado di importanza in una scala definita secondo 5 livelli:

- 0: Non sono d'accordo che questo sia un problema
- 1: È solo un problema “estetico”, non serve risolverlo a meno che non avanzi tempo
- 2: È un problema secondario, la sua risoluzione non è urgente
- 3: È un problema rilevante, va risolto con alta priorità
- 4: È un problema gravissimo, va assolutamente risolto prima che l'app sia rilasciata

Sulla base di tale valutazione è stato eseguito un test binomiale per confrontare il numero di volte in cui ciascun problema si è ritrovato in prima fascia (tra i 5 più gravi) e il numero di volte in cui si è trovato in seconda fascia (non tra i 5 più gravi).

Dal risultato di tale test si sono stabilite tre fasce di priorità:

- A: fascia alta con priorità significativa
- B: fascia intermedia di non significatività (non si hanno dati sufficienti per stabilire il livello di priorità)
- C: fascia bassa con priorità significativa

6.1.4 Risultati

A seguito della valutazione euristica sono stati individuati 22 problemi, di cui 1 con priorità A, 17 con priorità B e 4 con priorità C (lista completa in appendice A.2)

Di seguito la lista di quelli che, sulla base dei risultati ottenuti, sono stati ritenuti i più importanti.

P13: Messaggio di validazione della mail poco leggibile

Quando ci si registra tramite e-mail, per completare l'operazione è necessario confermare la propria identità tramite un link inviato al suddetto indirizzo. Questa informazione viene data all'utente tramite un messaggio, mostrato nella figura 48, che compare nella parte bassa dello schermo per pochi istanti. Se quest'ultimo non viene notato, la registrazione non può essere completata.

Il problema non rispetta la prima e la settima euristica di Nielsen ed è stato notato da 4 utenti su 5.

Con una severità media pari a 3 e una deviazione standard pari a 0, gli è stata assegnato il massimo livello di priorità (A).

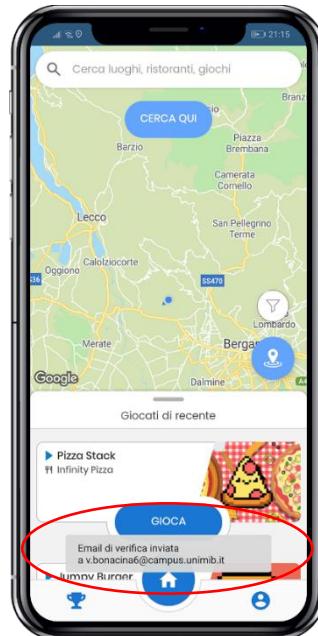


Figura 48

Screenshot da GamePlate. Il messaggio è piccolo e poco visibile, come indicato dal problema P13

P3: Filtri di ricerca poco individuabili

Le ricerche possono essere filtrate utilizzando un apposito pulsante, che però l'utente fatica a trovare, poiché posizionato in una zona dello schermo poco coerente.

Il problema non rispetta la quinta euristica di Nielsen ed è stato notato da 4 utenti su 5.

Con una severità media pari a 3 e una deviazione standard pari a 0.89, gli è stato assegnato il livello di priorità B.

P4: Comportamento inatteso del tasto ‘Localizzami’

Il tasto ‘Localizzami’ effettua una ricerca dei ristoranti vicini, utilizzando i criteri di filtraggio (se specificati). Questo comportamento differisce da quello che l'utente si aspetterebbe, ovvero visualizzare la propria posizione sulla mappa.

Il problema non rispetta la quinta euristica di Nielsen ed è stato notato da 3 utenti su 5.

Con una severità media pari a 2.8 e una deviazione standard pari a 1.09, gli è stato assegnato il livello di priorità B.

P14: Simbolo del tasto filtro poco riconoscibile

Il simbolo usato per il tasto dei filtri è stato considerato poco riconoscibile. L'utente non comprende la funzione del pulsante fino a quando non lo preme.

Il problema non rispetta la seconda e la quinta euristica di Nielsen ed è stato notato da 1 utente su 5.

Con una severità media pari a 1.4 e una deviazione standard pari a 1.79, gli è stato assegnato il livello di priorità B.

P21: Tasto ‘Localizzami’ troppo isolato e fuori posizione

Il tasto ‘Localizzami’, illustrato nella figura 49, è stato considerato poco visibile e difficile da individuare.

Il problema non rispetta la quinta euristica di Nielsen ed è stato notato da 1 utente su 5.

Con una severità media pari a 2.4 e una deviazione standard pari a 1.04, gli è stato assegnato il livello di priorità B.

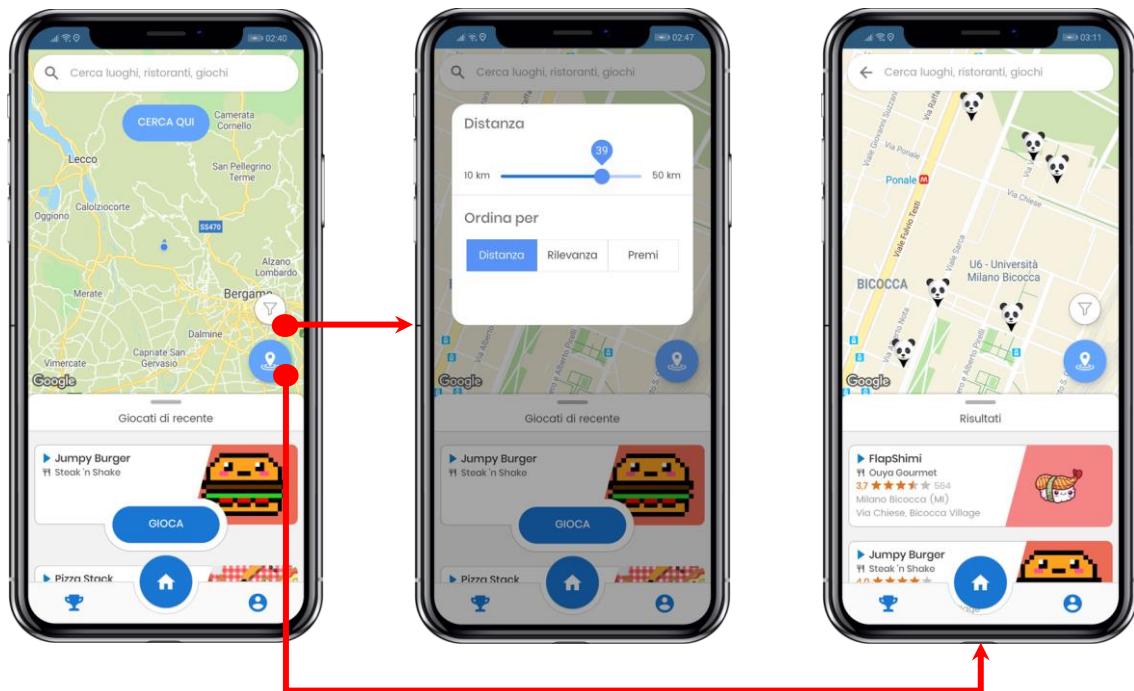


Figura 49
Screenshot da GamePlate, rappresentazione del sistema di ricerca con filtri. L’interazione con l’utente è poco chiara, come dimostrato dai problemi P3, P4, P14 e P21

6.2 Release 2

A seguito della valutazione euristica, basandosi sui problemi riscontrati, è stata messa a punto una nuova versione dell’applicazione che risolvesse le principali criticità.

Pur trattandosi solo di un prototipo dalle funzionalità più limitate, è stata sufficiente per procedere con i test successivi e valutare la qualità delle modifiche effettuate.

Di seguito i principali cambiamenti, messi in relazione con i problemi risolti rispetto alla prima versione dell’app.

Messaggio di invio dell'e-mail di validazione

Il messaggio scarno e di breve durata che avvisava l'utente di dover validare il proprio account è stato sostituito da una Dialog a tutto schermo, visibile nella figura 50, chiudibile soltanto premendo sul tasto ‘Ok’.

Questa modifica è volta a risolvere il problema P13 della valutazione euristica, ovvero l'unico con livello di priorità massimo.

Processo di ricerca dei ristoranti vicini

Dalla valutazione euristica sono emersi diversi problemi (P3, P4, P14, P21) legati alla meccanica di ricerca, in particolare per quanto riguarda la ricerca con filtri e quella dei ristoranti vicini.

Tutto il processo è stato dunque ripensato, trasformando le due fasi parallele d'impostazione dei filtri e pressione del pulsante di ricerca in fasi sequenziali, come mostrato nella figura 51, così che l'utente non possa confondersi.

Inoltre, il tasto ‘Localizzami’ scompare quando viene selezionata la barra di ricerca, così da non confondere le due modalità.



Figura 50
Screenshot da GamePlate. Nuovo messaggio di avviso per la verifica dell'account

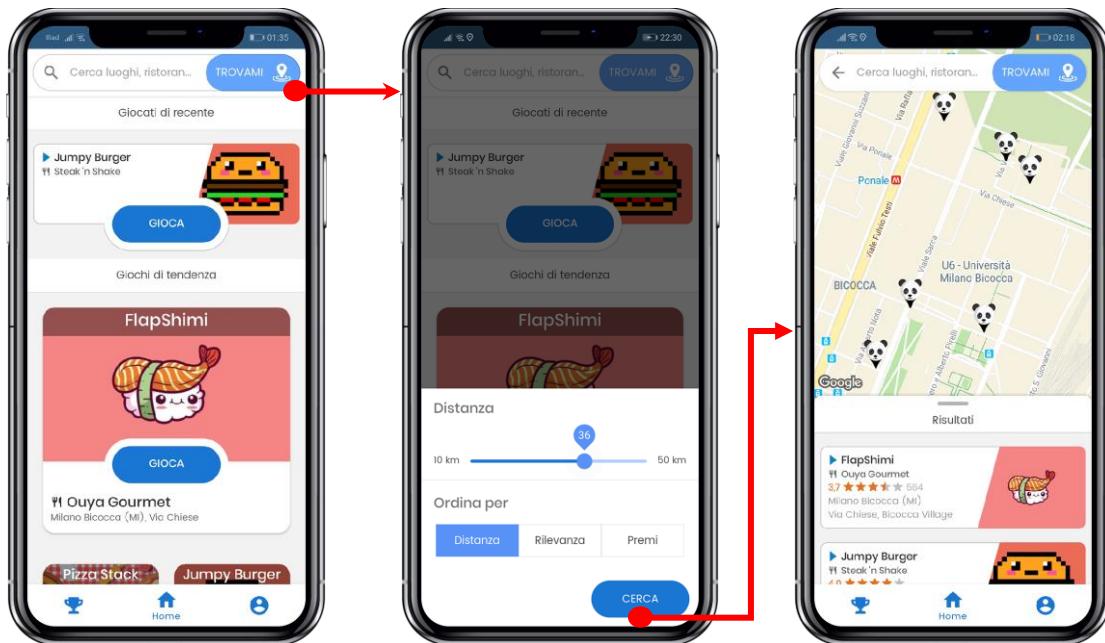


Figura 51
Screenshot da GamePlate. Meccanica di ricerca rinnovata

Aggiunta di una Homepage

Fondamentalmente, nella prima versione dell'app, non esisteva una vera e propria Homepage. Questo rendeva gli utenti spaesati di fronte alla prima schermata dell'app, come evidenziato dai problemi P17 e P10.

Nella Release 2 è stata introdotta una schermata, illustrata nella figura 52, che permette di vedere, oltre alla lista dei giochi recenti, anche quella dei giochi di tendenza.

Questa nuova funzionalità permette agli utenti di avere un migliore impatto al primo approccio con la piattaforma, permettendo loro di entrare subito in partita.

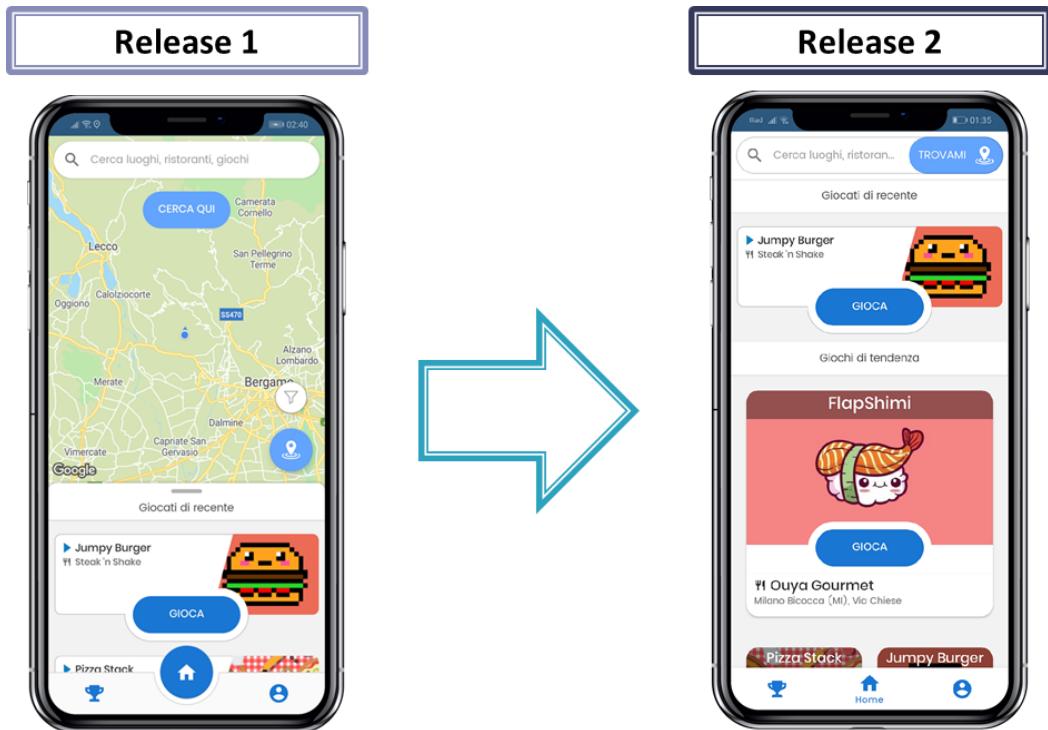


Figura 52
Screenshot da GamePlate. Introduzione di una homepage nella Release 2

Sostituzione del pulsante Home

Diversi utenti erano confusi dall'estetica del Floating Action Button utilizzato nella barra di navigazione, come evidenziato dai problemi P2 e P22. Di conseguenza, l'elemento è stato rimosso, a favore di un'interfaccia più classica e facile da comprendere, mostrata nella figura 53.

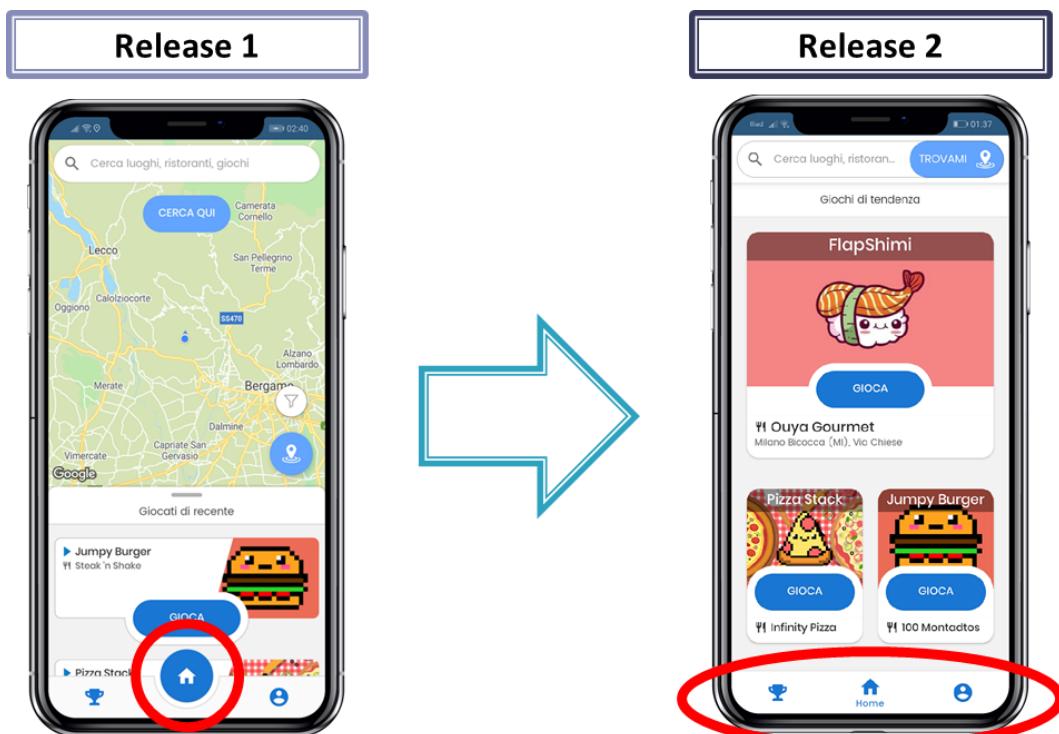


Figura 53
Screenshot da GamePlate. Sostituzione del pulsante Home

Migliorata la visibilità dei tasti nella schermata ‘Dettaglio Gioco’

Nella scheda ‘Dettaglio Gioco’ i pulsanti in alto (‘Indietro’ e ‘Classifiche’) rischiano di essere poco visibili se l’immagine dietro è troppo chiara. Lo stesso vale per il titolo del gioco, posizionato poco più in basso. Anche questo problema era stato individuato nella fase di valutazione euristica (P15).

Per migliorare la visibilità si è scelto di aggiungere uno strato nero semitrasparente dietro gli oggetti interessati, in modo da creare il contrasto necessario alla visione degli stessi, come si mostra nella figura 54.

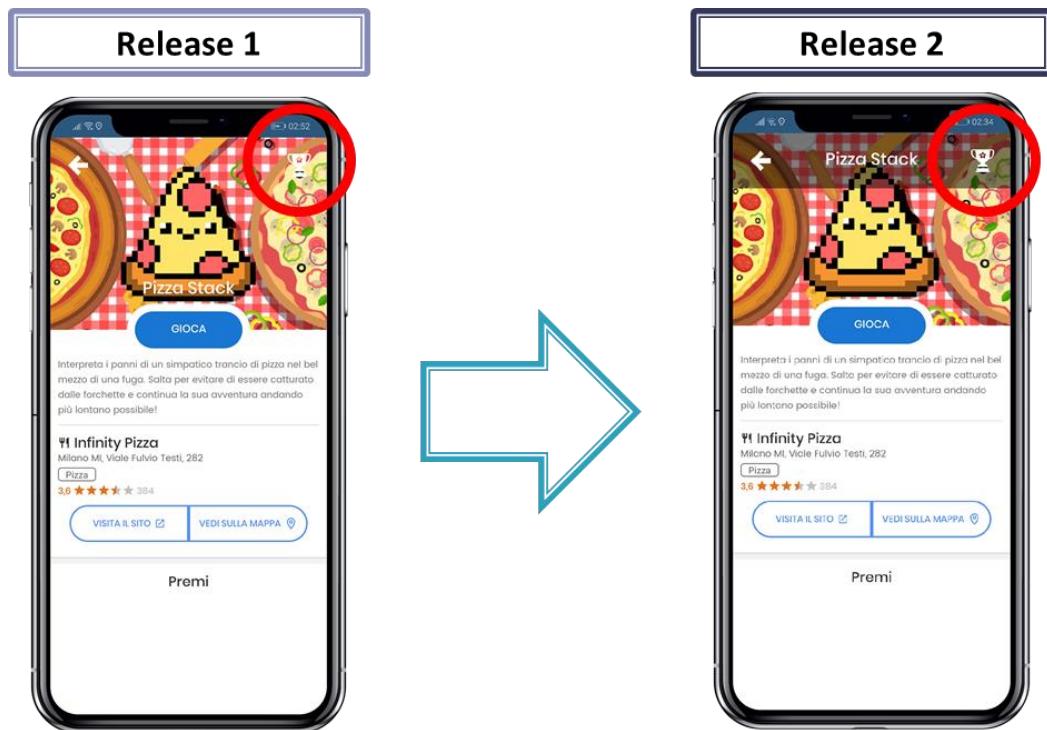


Figura 54
Screenshot da GamePlate. Migliorata la visibilità dei tasti nella Release 2

Click del pulsante ‘Maggiori informazioni’ di un premio

Il tasto ‘Maggiori informazioni’, situato nell’angolo in alto a destra di un premio, è stato giudicato troppo piccolo dagli utenti (P20).

A tal riguardo si è deciso di rimuovere completamente il tasto ed estendere la funzione di click a tutta la card, come mostrato nella figura 55.

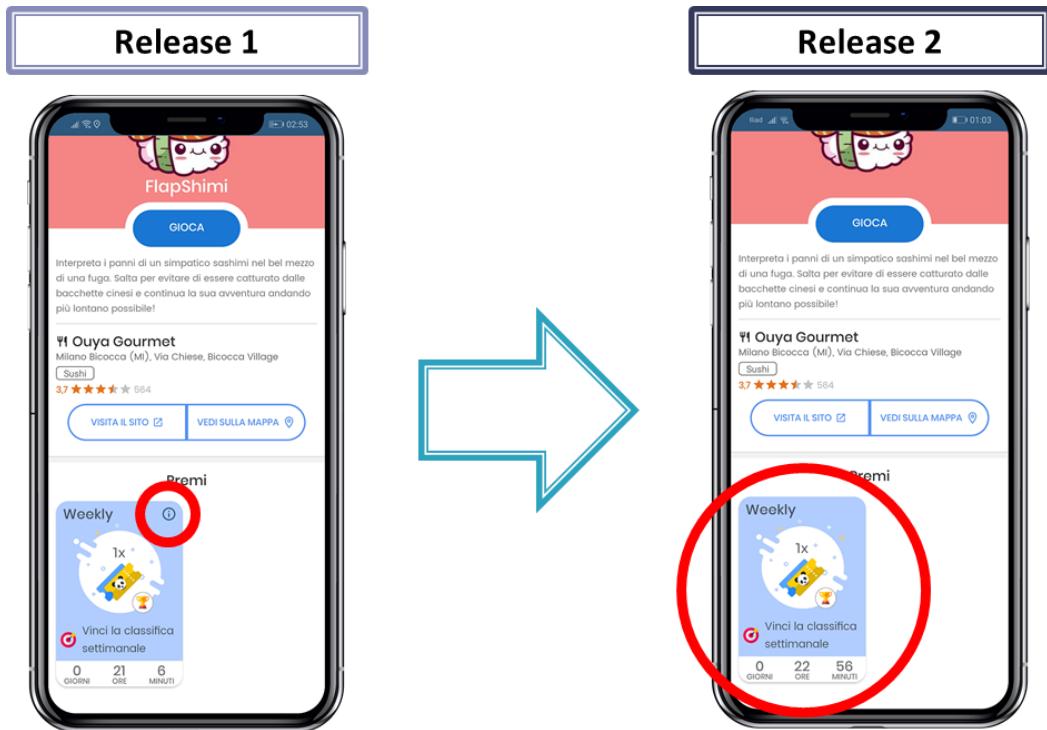


Figura 55
Screenshot da GamePlate. Modifica della funzione 'Maggiori informazioni' di un premio

6.3 Test Utente

I test utente consistono in una serie di interviste, svolte allo scopo di confrontare quantitativamente l’efficacia e l’efficienza della prima e della seconda versione dell’app.

A tal fine sono state coinvolte 12 persone, alle quali è stato chiesto di svolgere 3 semplici task (compiti), scelti in modo tale da ricoprire tutte e 3 le principali sezioni dell’app (premi, home e profilo).

In seguito ai test effettuati sulla ‘Release 1’, le stesse persone sono state sottoposte ad un ulteriore sessione di test, identica alla prima, sulla ‘Release 2’.

Al fine di spiegare i compiti da svolgere limitando il più possibile l’interazione col responsabile del test, ad ogni utente è stata mostrata una presentazione (consultabile nell’appendice A.1).

Di seguito i tre task sottoposti agli intervistati (per conoscerne lo svolgimento ottimale consultare l’appendice A.1):

- Effettua una ricerca filtrata dei ristoranti “intorno a te” ed evidenziane uno
- Vedi i dettagli di un premio vinto
- Effettua il logout dal tuo account

6.3.1 Composizione del campione

Per svolgere i test si è cercato di formare un campione, descritto nella figura 56, sufficientemente diversificato per quanto riguarda genere, professione, età e livello di familiarità con applicazioni simili.

I tre esperti di dominio interpellati nella valutazione euristica sono stati inclusi nel gruppo.

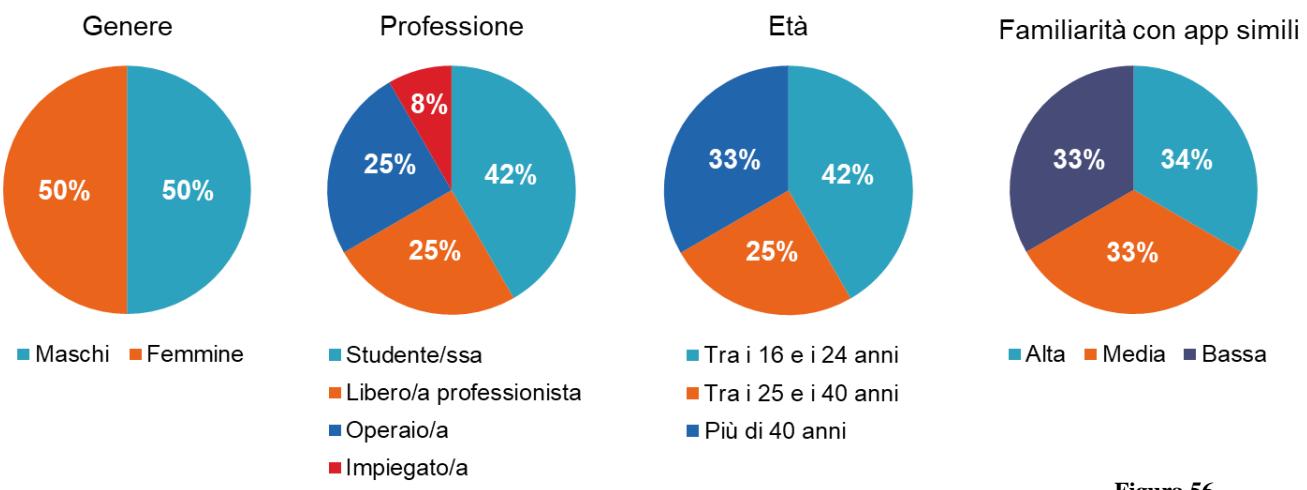


Figura 56
Composizione del campione di utenti

6.3.2 Test statistici

Per studiare l'efficacia sono stati osservati gli utenti nello svolgimento dei task, controllando se questi portassero a termine i compiti assegnati nel modo migliore. È stata tenuta traccia del numero di insuccessi e di successi, differenziandoli tra autonomi (senza aiuti esterni) e assistiti (con aiuti esterni).

Per studiare l'efficienza, invece, si è tenuto conto di tutti i tempi d'esecuzione dei task di ogni utente.

Ogni dato ricavato è stato infine confrontato a livello statistico tra i risultati delle sessioni della Release 1 e quelli della Release 2.

Il livello di confidenza adottato in tutti i test statistici è del 95%.

Si ritengono quindi statisticamente significative le differenze associate a un p-value inferiore al 5%.

L'ipotesi nulla H_0 assunta per ciascun test statistico effettuato è la seguente: “*Non ci sono differenze statisticamente significative tra l'applicazione nella Release 1 e l'applicazione nella Release 2*”.

6.3.3 Efficacia

	TASK 1		TASK 2		TASK 3	
ID	RELEASE 1	RELEASE 2	RELEASE 1	RELEASE 2	RELEASE 1	RELEASE 2
UT1	Successi					
UT2	Insuccessi					
UT3	Insuccessi					
UT4	Successi		Successi assistiti	Successi assistiti		
UT5	Successi					
UT6	Successi		Successi assistiti			
UT7	Successi		Successi assistiti			
UT8	Successi					
UT9						
UT10						
UT11						
UT12						

█ Successi
 █ Successi assistiti
 █ Insuccessi

Tabella 4
Risultati ottenuti nei task

Per confrontare le due versioni dell'app, è stato eseguito un test esatto di Fisher sui risultati di ogni task, mostrati nella tabella 4. In questo modo è stato possibile stabilire se esistessero o meno delle differenze statisticamente significative.

L'aiuto fornito agli intervistati è stato reputato decisivo per il corretto svolgimento del task, pertanto si è scelto di conteggiare i successi assistiti come errori.

Di seguito i risultati ottenuti dai test statistici condotti sull'efficacia di ogni task:

Task 1:

Il task 1 chiedeva di effettuare una ricerca dei ristoranti nei dintorni e di selezionarne uno.

Nella Release 1 si sono avuti 5 successi e 7 insuccessi/successi assistiti.

Nella Release 2, invece, si sono registrati solo successi.

In base al test statistico effettuato, si è trovata una differenza statisticamente significativa, con un p-value pari a 0.0045.

Questo risultato suggerisce che gli utenti compiono un numero significativamente maggiore di errori nell'esecuzione del task 1 usando la Release 1 rispetto a quando utilizzano la Release 2.

Task 2:

Il task 2 chiedeva di andare a vedere i dettagli di un premio appena vinto.

Nella Release 1 si sono avuti 8 successi e 4 insuccessi/successi assistiti.

Nella Release 2, invece si sono registrati 11 successi e 1 insuccesso.

In base al test statistico effettuato, si è trovata una differenza statisticamente non significativa, con un p-value pari a 0.32.

Questo risultato suggerisce che gli utenti non compiono un numero significativamente maggiore di errori nell'esecuzione del task 2 usando la Release 1 rispetto a quando utilizzano la Release 2.

Task 3:

Il task 3 chiedeva di effettuare il logout dal proprio account.

In entrambe le release si sono registrati 12 successi e nessun insuccesso/successo assistito.

In base al test statistico effettuato, non si è trovata alcuna differenza tra le versioni dell'app e il p-value è risultato pari a 1.

Questo risultato suggerisce che gli utenti non compiono un numero diverso di errori nell'esecuzione del task 3 usando le due release dell'applicazione.

6.3.4 Efficienza

Per valutare le differenze tra le due release in termini di efficienza, è stato eseguito un t-test per campioni dipendenti sui tempi di esecuzione di tutti i task.

Di seguito gli esiti di tale attività statistica:

Task 1 (risultati in figura 57):

In base al test statistico effettuato, si è trovata una differenza statisticamente significativa tra i tempi di esecuzione per la Release 1 ($M=92 \pm 37s$) e la Release 2 ($M=26 \pm 7s$).

Il valore di t è risultato pari a 4.41, mentre il p-value è risultato uguale a 0.001.

Questo risultato suggerisce che gli utenti impiegano un tempo significativamente minore per eseguire il task 1 usando la Release 2 rispetto a quando usano la Release 1.

Task 2 (risultati in figura 58):

In base al test statistico effettuato, si è trovata una differenza statisticamente significativa tra i tempi di esecuzione per la Release 1 ($M=38 \pm 23s$) e la Release 2 ($M=23 \pm 21s$).

Il valore di t è risultato pari a 3.1, mentre il p-value è risultato uguale a 0.01.

Questo risultato suggerisce che gli utenti impiegano un tempo significativamente minore per eseguire il task 2 usando la Release 2 rispetto a quando usano la Release 1.

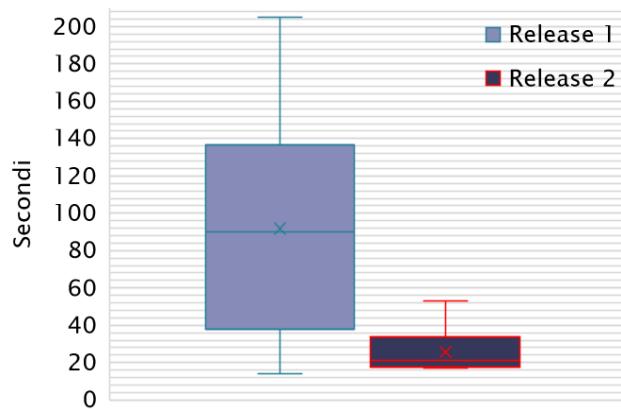


Figura 57
Risultato del test per il task 1

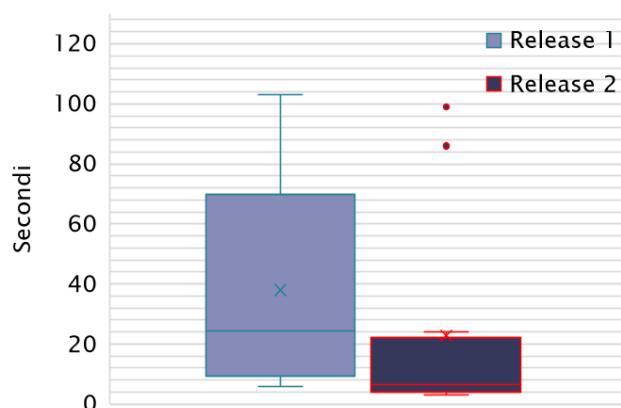


Figura 58
Risultato del test per il task 2

Task 3 (risultati in figura 59):

In base al test statistico effettuato, si è trovata una differenza statisticamente significativa tra i tempi di esecuzione per la Release 1 ($M=16 \pm 13s$) e la Release 2 ($M=8 \pm 7s$).

Il valore di t è risultato pari a 2.7, mentre il p -value è risultato uguale a 0.02.

Questo risultato suggerisce che gli utenti impiegano un tempo significativamente minore per eseguire il task 2 usando la Release 2 rispetto a quando usano la Release 1.

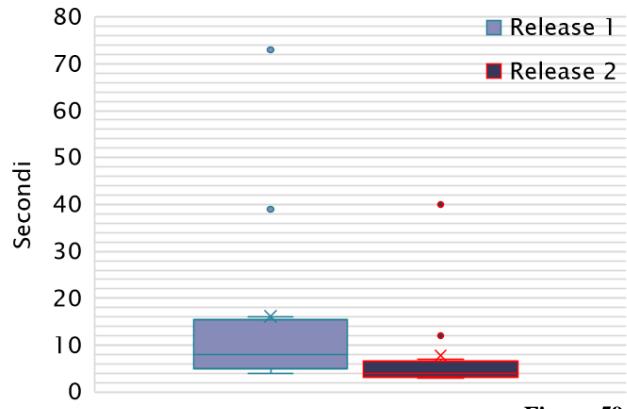


Figura 59
Risultato del test per il task 3

6.4 Questionario

Allo scopo di misurare il livello di gratificazione degli utenti, si è scelto di somministrare un questionario a 24 soggetti (12 dei quali già coinvolti nei test utente) in merito alla ‘Release 2’ dell’applicazione.

6.4.1 Test UEQ

Il questionario selezionato è lo User Experience Questionnaire^[25] (UEQ), realizzato nel 2006 da A.Hinderks, M.Schrepp e J.Thomashevsky.

Si tratta di una serie di 26 domande, elencate nella figura 60, cui è possibile rispondere usando una scala da 1 a 7.

Si è scelto di usare lo UEQ per la sua semplicità e per i tool di analisi forniti (comprensivi di benchmark).

	1	2	3	4	5	6	7		
fastidioso	<input type="radio"/>	piacevole	1						
incomprensibile	<input type="radio"/>	comprendibile	2						
creativo	<input type="radio"/>	privo di fantasia	3						
facile da apprendere	<input type="radio"/>	difficile da apprendere	4						
di grande valore	<input type="radio"/>	di poco valore	5						
noioso	<input type="radio"/>	appassionante	6						
non interessante	<input type="radio"/>	interessante	7						
imprevedibile	<input type="radio"/>	prevedibile	8						
veloce	<input type="radio"/>	lento	9						
originale	<input type="radio"/>	convenzionale	10						
ostrettivo	<input type="radio"/>	di supporto	11						
buono	<input type="radio"/>	scarso	12						
complicato	<input type="radio"/>	facile	13						
repellente	<input type="radio"/>	attraente	14						
usuale	<input type="radio"/>	moderno	15						
sgradevole	<input type="radio"/>	gradevole	16						
sicuro	<input type="radio"/>	insicuro	17						
attivante	<input type="radio"/>	soporifero	18						
conforme alle aspettative	<input type="radio"/>	non conforme alle aspettative	19						
inefficiente	<input type="radio"/>	efficiente	20						
chiaro	<input type="radio"/>	confuso	21						
non pragmatico	<input type="radio"/>	pragmatico	22						
ordinato	<input type="radio"/>	sovraffatto	23						
invitante	<input type="radio"/>	non invitante	24						
congeniale	<input type="radio"/>	ostile	25						
conservativo	<input type="radio"/>	innovativo	26						

Figura 60
Lista di domande del questionario UEQ
(fonte: <https://www.ueq-online.org/>)

I quesiti sono suddivisi in 6 sottogruppi:

- Attrattività:
 - D1: Fastidioso – Piacevole
 - D12: Buono - Scarso
 - D14: Repellente - Attraente
 - D16: Sgradevole - Gradevole
 - D24: Invitante - Non invitante
 - D25: Congeniale - Ostile
- Apprendibilità:
 - D2: Incomprensibile - Comprensibile
 - D4: Facile da apprendere - Difficile da apprendere
 - D13: Complicato - Facile
 - D21: Chiaro - Confuso
- Efficienza:
 - D9: Veloce - Lento
 - D20: Inefficiente - Efficiente
 - D22: Non pragmatico - Pragmatico
 - D23: Ordinato - Sovraccarico
- Controllabilità:
 - D8: Imprevedibile - Prevedibile
 - D11: Ostruttivo - Di supporto
 - D17: Sicuro - Insicuro
 - D19: Conforme alle aspettative – Non conforme alle aspettative
- Stimolazione:
 - D5: Di grande valore - Di poco valore
 - D6: Noioso - Appassionante
 - D7: Non interessante - Interessante
 - D18: Attivante - Soporifero
- Originalità:
 - D3: Creativo - Privo di fantasia
 - D10: Originale - Convenzionale
 - D15: Usuale - Moderno
 - D26: Conservativo – Innovativo

I risultati integrali dei questionari sono riportati nell'appendice A.3

6.4.2 Confronto statistico tra sistemi

Sulla base dei risultati del test UEQ, sono stati fatti dei confronti statistici tra la Release 2 di GamePlate e un modello ideale creato attraverso dati fintizi.

Quest'ultimo è stato impostato in modo tale che la sua mediana valga un valore sopra la metà della scala del questionario, nel caso in cui le risposte siano mappate da sinistra a destra, oppure un valore sotto la metà di tale scala nel caso in cui le risposte abbiano mapping opposto.

Sono stati condotti 5 test di MANN-WHITNEY su alcune domande ritenute particolarmente importanti:

- Domanda 1: Fastidioso (1) – Piacevole (7)
- Domanda 3: Creativo (1) – Privo di fantasia (7)
- Domanda 6: Noioso (1) – Appassionante (7)
- Domanda 9: Veloce (1) – Lento (7)
- Domanda 21: Chiaro (1) – Confuso (7)

Il livello di confidenza adottato in tutti i test statistici è del 95%.

Si ritengono quindi statisticamente significative le differenze associate a un p-value inferiore al 5%.

L'ipotesi nulla H_0 assunta per ciascun test statistico effettuato è la seguente: “*Non ci sono differenze statisticamente significative tra l'applicazione nella Release 2 e il modello ideale*”.

Di seguito i risultati ottenuti da questa analisi:

Piacevolezza (risultati in figura 61):

In base al test effettuato, si è trovata una differenza statisticamente significativa tra le mediane della piacevolezza della Release 2 (Mediana = 6) e del modello ideale (Mediana = 5).

Il valore z-score è risultato -3.12 mentre il p-value 0.001.

Questo risultato suggerisce che gli utenti considerano la ‘Release 2’ significativamente più piacevole rispetto al modello ideale.

Creatività (risultati in figura 62):

In base al test effettuato, si è trovata una differenza statisticamente significativa tra le mediane della creatività della Release 2 (Mediana = 2) e del modello ideale (Mediana = 3).

Il valore z-score è risultato -3.9 mentre il p-value 0.0001.

Questo risultato suggerisce che gli utenti considerano la ‘Release 2’ significativamente più creativa rispetto al modello ideale.

Domanda 1: Fastidioso(1) – Piacevole(7)

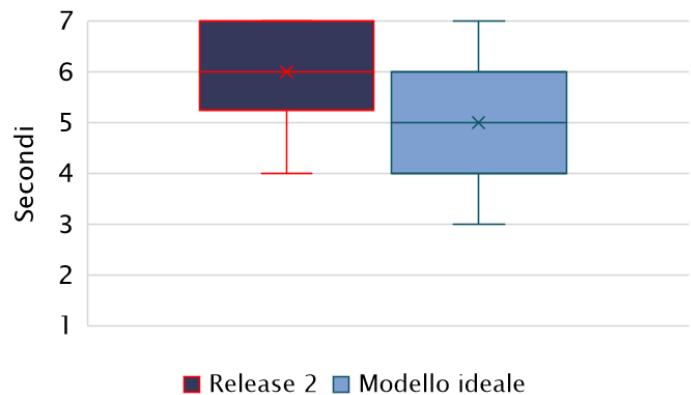


Figura 61
Risultato del confronto sulla piacevolezza

Domanda 3: Creativo(1) – Privo di fantasia(7)

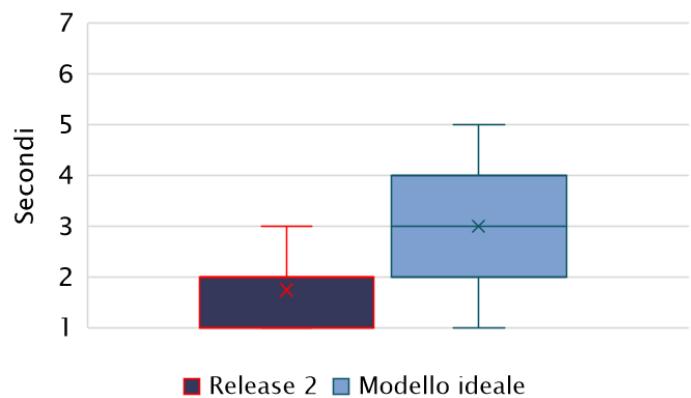


Figura 62
Risultato del confronto sulla creatività

Coinvolgimento (risultati in figura 63):

In base al test effettuato, si è trovata una differenza statisticamente non significativa tra le mediane del coinvolgimento della Release 2 (Mediana = 6) e del modello ideale (Mediana = 5).

Il valore z-score è risultato 1.66 mentre il p-value 0.09.

Questo risultato suggerisce che gli utenti considerano la Release 2 e il modello ideale ugualmente coinvolgenti.

Domanda 6: Noioso(1) – Appassionante(7)

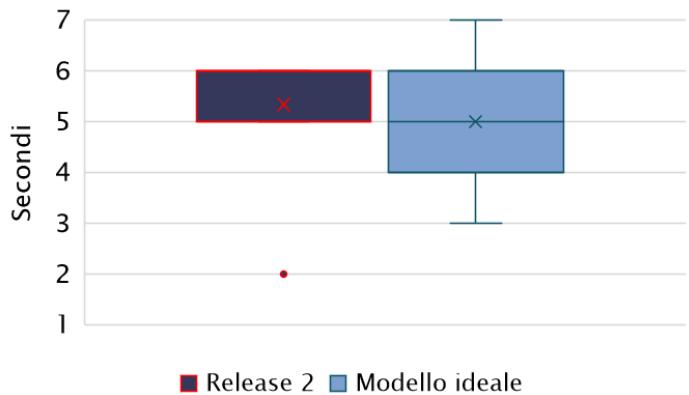


Figura 63

Risultato del confronto sul coinvolgimento

Velocità (risultati in figura 64):

In base al test effettuato, si è trovata una differenza statisticamente significativa tra le mediane della velocità della Release 2 (Mediana = 2) e del modello ideale (Mediana = 3).

Il valore z-score è risultato -2.34 mentre il p-value 0.01.

Questo risultato suggerisce che gli utenti considerano la Release 2 significativamente più veloce rispetto al modello ideale.

Domanda 9: Veloce(1) – Lento(7)

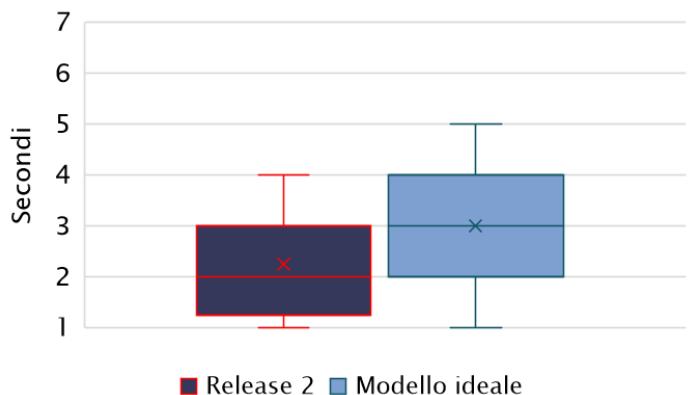


Figura 64

Risultato del confronto sulla velocità

Chiarezza (risultati in figura 65):

In base al test effettuato, si è trovata una differenza statisticamente non significativa tra le mediane della chiarezza della Release 2 (Mediana = 2) e del modello ideale (Mediana = 3).

Il valore z-score è risultato -1.46 mentre il p-value 0.14.

Questo risultato suggerisce che gli utenti considerano la Release 2 e il modello ideale ugualmente chiari.

Domanda 21: Chiaro(1) – Confuso(7)

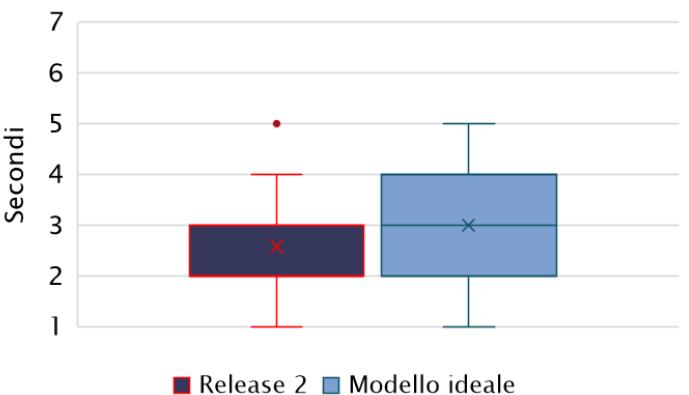


Figura 65

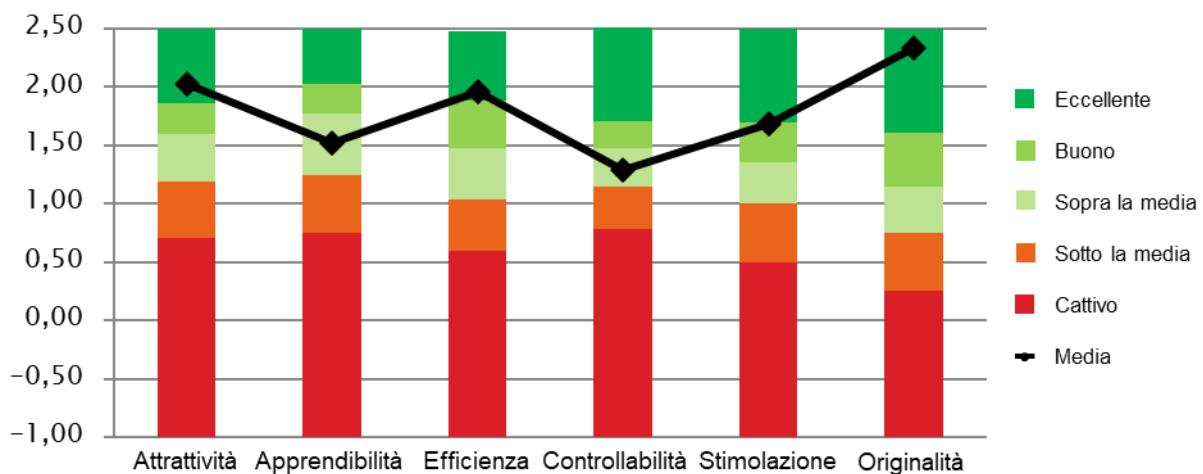
Risultato del confronto sulla chiarezza

6.4.3 Benchmark

UEQ, oltre al questionario, fornisce uno strumento di benchmark che confronta i risultati con un dataset di 401 studi, riguardanti prodotti informatici, per un totale di 18483 utenti intervistati.

Per ogni domanda, le risposte del questionario (da 1 a 7) vengono trasformate in una scala da -3 a +3, dunque vengono eseguiti dei confronti con i risultati salvati nel dataset.

Il risultato del benchmark, illustrato nella tabella 5, ha mostrato che GamePlate si trova sopra la media in tutte le 6 categorie prese in esame dal test.



Sottogruppo	Media	Comparazione con Benchmark	Interpretazione
Attrattività	2,03	Eccellente	Nel 10% dei migliori risultati
Apprendibilità	1,52	Sopra la media	25% dei risultati migliori, 50% peggiori
Efficienza	1,96	Eccellente	Nel 10% dei migliori risultati
Controllabilità	1,29	Sopra la media	25% dei risultati migliori, 50% peggiori
Stimolazione	1,69	Buona	10% dei risultati migliori, 75% peggiori
Originalità	2,33	Eccellente	Nel 10% dei migliori risultati

Tabella 5
Risultati del benchmark in forma grafica (sopra) e tabellare (sotto)

6.4.4 Net Promoter Score

Il Net Promoter Score (NPS) è un indicatore che può essere usato per valutare la fedeltà e l'apprezzamento di un utente nei confronti di una certa azienda, servizio, applicazione.

Si calcola sottponendo un'unica domanda all'utente: “*Con quale probabilità consiglieresti questo prodotto/servizio a un amico o a un collega?*”.

Le risposte possibili sono in una scala da 0 a 10 e vengono interpretate nella maniera seguente:

- DETRATTORI: Risposte tra 0 e 6 → Utenti insoddisfatti
- PASSIVI: Risposte tra 7 e 8 → Utenti soddisfatti ma indifferenti
- PROMOTORI: Risposte tra 9 e 10 → Utenti pienamente soddisfatti

L’NPS si calcola con la seguente formula: %PROMOTORI - %DETRATTORI.

Nel caso di GamePlate, il Net Promoter Score, calcolato sulle risposte degli stessi 24 utenti coinvolti nei questionari, è risultato pari a 42 (come si mostra nella figura 66).

Un risultato così positivo potrebbe essere viziato tuttavia dall'effetto novità dell'app, la quale ad oggi non ha ancora nessun competitor con cui paragonarla.

Inoltre, nello specifico caso di questa analisi, l’NPS è fortemente soggetto a courtesy bias.

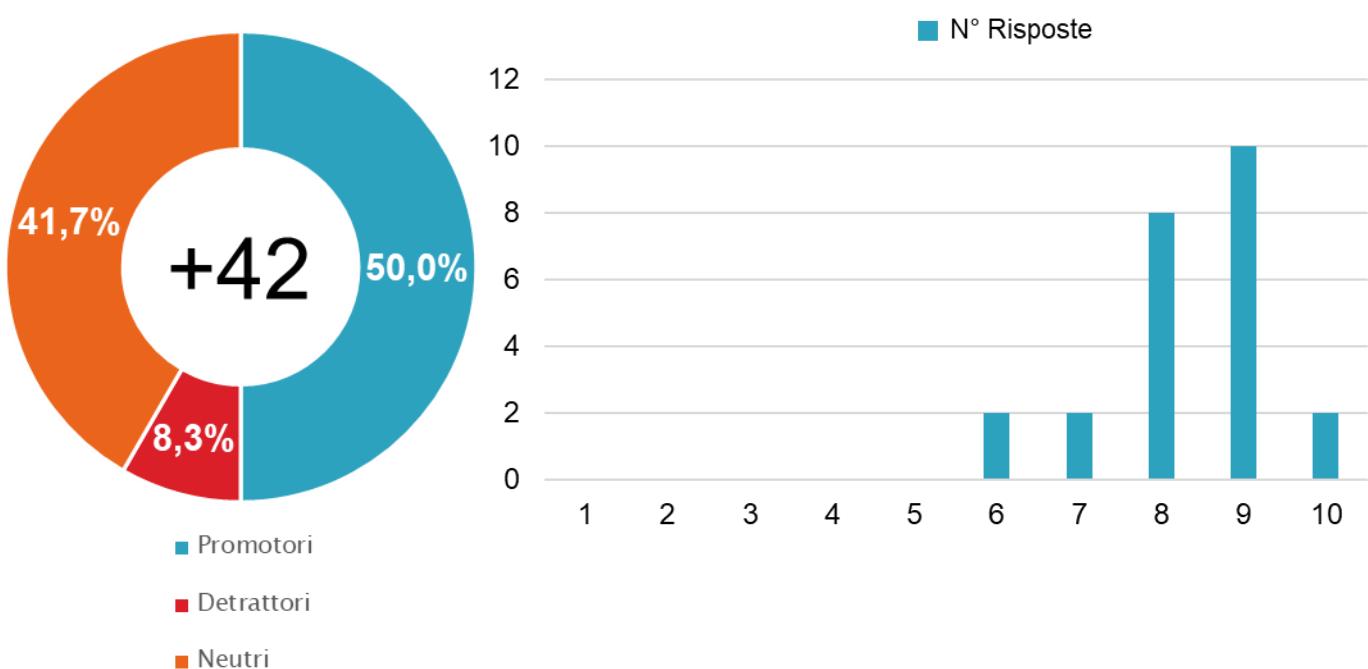


Figura 66
Risultati del Net Promoter Score per GamePlate

6.5 Considerazioni finali

Considerando i risultati dell'analisi eseguita, si ritiene che la Release 2 abbia portato dei significativi miglioramenti all'usabilità generale dell'app.

In particolare, si è registrato un netto aumento dell'efficacia del primo task proposto nei test utente e un aumento dell'efficienza in generale.

Tutto ciò è sintomo di una **rinnovata e migliorata interazione con l'utente**, che infatti ha mediamente valutato positivamente l'applicazione sia in occasione del questionario UEQ che nel caso del Net Promoter Score.

Conclusioni

Il progetto GamePlate ha avuto origine dall'idea di promuovere locali e ristoranti in modo nuovo e non invasivo, dando anche la possibilità ai consumatori di vincere premi e sconti.

Il percorso che ha portato alla realizzazione di una piattaforma del genere ha richiesto un lavoro particolarmente ampio e variegato.

Si è spaziati dalla progettazione di database allo sviluppo Android nativo, dalla realizzazione dell'interfaccia grafica all'analisi della sua usabilità, passando per la creazione di giochi HTML5 e del plugin in Java-JavaScript per farli comunicare con la piattaforma.

Durante i mesi impiegati per la realizzazione del progetto sono stati riscontrati diversi feedback positivi dagli utenti ai quali è stata presentata l'applicazione, in particolare nel corso dell'analisi di usabilità. Ci si è convinti pertanto che l'idea sia valida e degna di essere portata avanti anche successivamente, con una serie di rifiniture e nuove funzionalità.

In cima alla lista dei progetti futuri si trova senza dubbio l'implementazione completa della Release 2, presentata nel capitolo sull'analisi di usabilità, che per ora si trova allo stato prototipale.

Sarà inoltre necessaria la realizzazione di un considerevole numero di giochi, in modo da creare una rete di ristoranti abbastanza fitta da convincere gli utenti ad utilizzare il servizio.

Oltre a questi piani, i quali verranno messi in atto nel breve periodo, si è ideata una serie di nuove funzionalità che andranno implementate nel medio/lungo periodo.

Innanzitutto, si è pensato alla realizzazione di una Dashboard dove trovare informazioni sui nuovi giochi pubblicati e sulle promozioni dei ristoranti nei dintorni. Si implementerebbe anche un sistema di following così che l'utente sia sempre aggiornato sulle novità dei locali che più gli interessano.

Un'altra funzionalità, che si prevede di integrare, consiste nella possibilità di effettuare prenotazioni direttamente dall'app.

Si sta inoltre valutando la possibilità di inserire nell'applicazione un sistema di monete, ottenibili all'interno dei giochi, che consentano di accedere ad altri premi.

Queste sono solo alcune delle idee a cui si è pensato negli ultimi mesi, e sicuramente ne verranno prese in considerazione altre in futuro.

Quel che è certo è che GamePlate è ancora aperto a una moltitudine di possibilità che la porteranno a migliorarsi e arricchirsi sempre di più.

Appendice A

A.1 Task

Descrizione dettagliata dei task

Release 1			
Descrizione breve	Precondizioni	Post condizioni	Procedura
Effettua una ricerca filtrata dei ristoranti 'intorno a te' e evidenziane uno. (TASK 1)	<ul style="list-style-type: none"> L'utente deve essersi registrato e loggato col suo account. L'utente deve vedere la schermata di home nell'applicazione. 	<ul style="list-style-type: none"> L'utente sta visualizzando il dettaglio di un ristorante. 	<ul style="list-style-type: none"> Cliccare l'icona del filtro e impostare i filtri che preferisce. Cliccare l'icona della ricerca 'intorno a me'. Alzare la 'BottomSheet' (elemento scrollabile situato sul basso dello schermo). Cliccare un elemento della lista visualizzata.

Release 2

Descrizione breve	Precondizioni	Post condizioni	Procedura
Effettua una ricerca filtrata dei ristoranti 'intorno a te' e evidenziane uno. (TASK 1)	<ul style="list-style-type: none"> L'utente deve essersi registrato e loggato col suo account. L'utente deve vedere la schermata di home nell'applicazione. 	<ul style="list-style-type: none"> L'utente sta visualizzando il dettaglio di un ristorante. 	<ul style="list-style-type: none"> Cliccare il tasto in alto a destra 'TROVAMI'. Impostare i filtri desiderati tra quelli che si rendono visibili. Alzare la 'BottomSheet' (elemento scrollabile situato sul basso dello schermo). Cliccare un elemento della lista visualizzata.

Release 1 / Release 2

Descrizione breve	Precondizioni	Post condizioni	Procedura
Visualizza i dettagli di un premio vinto. (TASK 2)	<ul style="list-style-type: none"> L'utente deve cliccare il tasto 'GIOCA' nella pagina di dettaglio di un ristorante e quindi vedere la schermata di un gioco. L'utente deve aver vinto un premio e quindi aver ricevuto la notifica che lo avvisa della vincita. 	<ul style="list-style-type: none"> L'utente sta visualizzando il dettaglio del premio che ha appena vinto. 	<ul style="list-style-type: none"> Cliccare la freccia in alto a sinistra per uscire dal gioco, oppure cliccare la notifica ricevuta. Uscire dalla pagina di dettaglio, aperta precedentemente per poter accedere alla schermata di gioco. Cliccare l'icona del trofeo in basso a sinistra per aprire la sezione dei premi vinti Cliccare l'elemento che rappresenta il premio appena vinto

Release 1 / Release 2

Descrizione breve	Precondizioni	Post condizioni	Procedura
Effettua il logout del tuo account. (TASK 3)	<ul style="list-style-type: none"> L'utente deve essersi registrato e loggato col suo account. L'utente deve vedere la schermata di home nell'applicazione. 	<ul style="list-style-type: none"> L'utente sta visualizzando la home dell'applicazione ma senza essere loggato. 	<ul style="list-style-type: none"> Cliccare l'icona del profilo in basso a destra per accedere alla sezione del profilo. Cliccare l'icona dell'ingranaggio per poter accedere alle impostazioni. Cliccare sul tasto 'ESCI' per effettuare il logout

Presentazione dei task usata durante le interviste



VALUTAZIONE DI USABILITÀ DELL'APPLICAZIONE
GamePlate

Ti ringraziamo per il tuo aiuto!

SE NON CI CONOSCESSI

GamePlate nasce con l'intento di fare pubblicità a locali e ristoranti in modo innovativo e originale attraverso i videogiochi. Ad ogni ristorante è affiancato un particolare videogioco basato su punteggi e gli utenti hanno la possibilità di vincere buoni e sconti, spendibili in quel locale, attraverso il superamento di determinati obiettivi nel gioco correlato.

Ogni settimana viene eseguito l'azzeramento di una classifica per ogni gioco e il vincitore riceverà in premio un buono gratuito per mangiare nel ristorante associato.

Questa strategia permette sia di incentivare il cliente a giocare (attraverso l'erogazione dei premi), sia di dare l'opportunità a quest'ultimo di poter visitare e testare il luogo sponsorizzato, permettendo che la pubblicità effettuata diventi attiva a tutti gli effetti.

CIAO!

Siamo molto contenti per averci concesso il tuo tempo!
Questo test ha lo scopo di valutare l'usabilità dell'applicazione in esame, per noi è molto importante raccogliere informazioni e opinioni sulla piattaforma dalla gente al fine di migliorarla sempre di più nel tempo.
In fondo la nostra app è costruita per essere adoperata proprio da voi!
Ti chiediamo solamente di svolgere 3 semplici compiti su due versioni differenti dell'applicazione.

Prima di iniziare, ricorda:

NON SEI TU AD ESSERE VALUTATO, MA L'APPLICAZIONE. QUINDI NON SENTIRTI SOTTO ESAME.



GamePlate
Release 1

TASK 3

▶ [Effettua il logout del tuo account](#)

- Entra nella sezione dedicata al tuo account
- Accedi alle impostazioni ed effettua il logout





GamePlate
Release 2

Torna alla [Slide 5](#) per completare i task

Altrimenti [Continua](#)

A.2 Risultati della valutazione euristica

Problemi individuati

ID	Problema	Descrizione problema	Euristiche violate	Valutatori	Popolarità	Severità media	Dev. Std. Severità	Priorità
P13	Messaggio validazione mail poco leggibile	Il messaggio che invita l'utente a confermare la propria mail appare in basso per pochi istanti. L'utente che perde questa informazione non può completare la registrazione al 100%	E1, E7	ED1,ED2,ED3,EU2	4	3	0	A
P3	Filtri di ricerca poco individuabili	I filtri inseriti nella home non sono facilmente trovati dagli utenti	E5	EU1, ED1,ED2,ED3	4	3	0,89	B
P4	Comportamento inatteso del tasto 'Localizzami'	Il tasto 'Localizzami' effettua una ricerca dei ristoranti vicini, utilizzando i filtri (se specificati). Spesso quindi non mostra la posizione dell'utente (evento che ci si attenderebbe)	E2, E5	EU1, ED1,ED2	3	2,8	1,09	B
P9	Il tasto 'Indietro' del telefono a volte chiude l'app	Quando ci si trova nelle schermate 'Premi' o 'Profilo', se si preme il tasto 'Indietro', l'applicazione si chiude invece che tornare alla Home	E6, E8	EU1, ED2	2	2,8	0,83	B
P19	Il click della notifica di vittoria di un premio non fa quello che ci si aspetta	Quando si clicca sulla notifica che indica la vittoria di un premio, l'app si limita ad aprirsi. Ci si sarebbe aspettati di vedere i dettagli del premio appena vinto.	E2	ED3,EU2	2	2,8	0,43	B
P10	Mancanza di un tutorial	Il tutorial termina una volta arrivati alla fase di registrazione, una volta ottenuto l'accesso all'app l'utente non ha un aiuto su come muoversi tra le schermate. Oltretutto il tutorial non è nemmeno consultabile in un secondo momento	E10	ED1,ED2	2	2,6	0,52	B
P7	Mancanza di feedback in caso di mancanza di rete	In mancanza di rete l'applicazione non funziona correttamente, e ciò non viene segnalato. Se si effettua una nuova ricerca viene solo riportato un messaggio con scritto "Errore". Se si apre l'app si rimane costantemente nella splashscreen senza ricevere feedback.	E1, E9	EU1	1	2,4	1,65	B

ID	Problema	Descrizione problema	Euristiche violate	Valutatori	Popolarità	Severità media	Dev. Std. Severità	Priorità
P17	Schede 'Giochi recenti' e 'Risultati ricerca' confondibili tra loro	I giochi recenti e i risultati di una ricerca sono inseriti nella stessa scheda in basso, a seconda dello stato dell'app (normalmente giochi recenti, quando invece si effettua una ricerca mostra i risultati). Le differenze tra le due situazioni sono minime.	E5, E7	ED2	1	2,4	0,76	B
P21	Tasto 'Cerca intorno a me' troppo isolato e fuori posizione	Il tasto risulta poco visibile e quindi difficile da individuare	E5	EU2	1	2,4	1,04	B
P5	Impossibilità di annullare un'operazione quando essa è iniziata	Se si inizia a fare una ricerca di qualsiasi tipo, i pulsanti per tornare indietro non sono attivi finché l'operazione non è terminata. La stessa problematica avviene quando si apre il dettaglio relativo a un ristorante.	E3, E6, E9	EU1	1	2,2	0,71	B
P15	Tasto classifiche poco visibile	Nella schermata di dettaglio su un gioco, ci sono dei casi in cui il pulsante delle classifiche (una coppa bianca) non è ben visibile a causa della immagine di background	E7	ED1	1	2,2	0,52	B
P2	Il tasto home non sempre è premibile	Il tasto home ha come scopo quello di tornare alla schermata centrale se ci si trova nella sezione 'Profilo' o 'Premi'. Se si è già nella sezione Home il pulsante non fa niente alla pressione e non cambia la sua estetica (non è chiaro che è già stato selezionato)	E1,E5,E7	EU1,ED2,ED 3	3	2	0,71	B
P20	Mancanza di consistenza grafica tra premi nel 'Dettaglio Gioco' e nella schermata 'Premi'	Nel primo caso il premio non è cliccabile, per vederne i dettagli serve premere sul pulsante 'Info' in alto a destra. Nel secondo caso il pulsante 'Info' è assente e l'intero premio è cliccabile	E4	EU1	1	1,8	0,71	B
P14	Simbolo del tasto filtro poco riconoscibile	Il simbolo utilizzato per il tasto 'Filtro' non è comprensibile. La funzione del pulsante non è chiara all'utente	E2, E5	ED1	1	1,4	1,79	B
P6	Tasto 'Esci' difficile da individuare	Per effettuare il Logout è necessario spostarsi nella schermata 'Profilo', premere il pulsante 'Impostazioni' e infine premere il pulsante 'Esci'	E5	EU1, ED1	2	1,2	1,51	B

ID	Problema	Descrizione problema	Euristiche violate	Valutatori	Popolarità	Severità media	Dev. Std. Severità	Priorità
P1	Poca chiarezza su come chiudere le dialog pop-up	Alla voce 'Dettagli Offerta' (presente sia consultando un premio, sia consultando i dettagli di un gioco) appare una finestra popup priva di pulsanti per chiuderla (la si può chiudere premendo all'esterno della stessa). La stessa situazione si presenta consultando le classifiche della schermata di dettaglio sui giochi e aprendo i filtri nella schermata di Home	E5	EU1	1	1,2	1,23	B
P12	Terminologia sui premi poco chiara	Il premio 'Weekly' non chiarisce in modo immediato di cosa si tratti. L'utente non capisce quale sia la ricompensa in palio	E2	ED1	1	1,2	1,13	B
P18	Scadenze premi poco visibili nella schermata di 'Dettaglio Gioco'	Il countdown di un premio, che appare sotto il premio stesso, è poco visibile poichè si confonde con lo sfondo dietro.	E7	ED3	1	1,2	0,48	B
P8	Inconsistenza di terminologia nei dettagli premio	Quando si consulta un premio nella schermata dedicata il pulsante 'Dettagli offerta', se premuto, apre una dialog che ha come titolo 'Dettagli premio'	E4	EU1	1	1	0	C
P11	L'opzione 'Registrami più tardi' non viene riconosciuta come un pulsante	Nella schermata di accesso, l'opzione 'Registrami più tardi' appare solo sotto forma di testo cliccabile. L'utente non sa che si tratta di una opzione cliccabile	E2, E5	ED1	1	1	0,84	C
P22	Tasto 'Home' si sovrappone ad altri elementi grafici	Quando il tasto home si sovrappone a elementi dello stesso colore, risulta scarsamente leggibile	E7	EU2	1	1	0,84	C
P16	Tasto 'Login' e 'Registrati' confondibili	La posizione dei due tasti è invertita rispetto a quello che solitamente si vede (sopra 'Login' e sotto 'Registrati').	E5	ED2	1	0,6	0,83	C

Matrice valutatori/problemi

P/V	P1	P5	P7	P8	P20	P11	P12	P14	P15	P16	P17	P18	P21	P22	P6	P9	P10	P19	P4	P2	P3	P13
EU2																						
ED3																						
ED2																						
ED1																						
EU1																						

Risultati del questionario per la prioritizzazione

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22
2	2	1	3	3	0	1	1	3	0	0	0	3	0	1	0	0	3	3	1	0	0
0	2	4	3	1	0	0	1	2	3	1	0	3	0	3	0	4	1	3	2	4	2
3	3	4	3	3	3	1	4	3	2	2	3	4	3	2	3	1	3	2	4	1	
1	1	2	1	2	3	4	1	3	4	2	3	3	3	2	1	2	1	2	3	2	2
0	2	4	4	2	0	4	1	2	3	0	1	3	0	2	0	3	0	3	1	2	0

A.3 Risultati dei questionari

Risposte al questionario UEQ

Genere	Età	Professione	Familiarità app simili	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20	T21	T22	T23	T24	T25	T28	NPS
Femmina	tra i 16 e i 24 anni	Operai/o/a	Bassa	6	6	1	2	2	2	5	4	3	2	3	2	5	5	6	5	3	2	3	5	2	4	2	3	4	5	8
Femmina	tra i 25 e i 40 anni	Studente/ssa	Alta	6	6	1	1	3	6	6	5	2	2	4	1	6	7	6	6	2	1	2	7	3	6	1	2	1	6	9
Femmina	maggiore di 40 anni	Liberoprofessionista	Bassa	5	3	3	4	4	6	6	3	2	2	4	3	3	4	6	6	4	2	3	6	5	4	2	3	3	6	7
Maschio	maggiore di 40 anni	Operai/o/a	Bassa	6	4	2	4	2	6	6	4	4	3	6	2	4	4	6	6	4	4	2	6	3	6	3	2	2	6	8
Maschio	maggiore di 40 anni	Liberoprofessionista	Media	7	5	2	3	4	6	6	4	2	2	6	2	6	6	7	6	5	2	2	6	4	6	2	2	2	6	6
Maschio	tra i 16 e i 24 anni	Studente/ssa	Alta	5	5	3	2	4	5	7	4	1	1	6	1	5	6	7	6	2	4	1	7	3	4	2	1	2	7	10
Femmina	maggiore di 40 anni	Liberoprofessionista	Bassa	7	6	1	2	2	6	6	4	4	2	6	1	6	6	7	7	1	2	2	7	2	2	1	7	9	9	
Maschio	tra i 25 e i 40 anni	Operai/o/a	Media	6	5	1	2	2	6	6	5	3	1	6	2	5	6	7	6	3	1	2	6	2	6	1	1	3	7	9
Maschio	tra i 16 e i 24 anni	Studente/ssa	Alta	7	7	1	1	2	5	7	6	1	1	7	1	7	7	7	7	2	1	1	7	1	1	1	7	9	9	
Femmina	tra i 16 e i 24 anni	Studente/ssa	Media	7	6	2	4	6	7	7	2	3	6	2	6	5	7	5	1	2	6	7	1	2	2	7	8	8		
Femmina	tra i 16 e i 24 anni	Impiegato/a	Media	4	5	2	2	3	5	6	2	1	2	6	2	7	6	7	6	2	1	1	6	2	6	1	1	2	6	8
Maschio	tra i 16 e i 24 anni	Studente/ssa	Alta	6	6	2	1	2	5	6	6	2	3	5	1	6	6	7	6	2	1	1	6	2	7	1	2	2	7	9
Maschio	tra i 16 e i 24 anni	Studente/ssa	Media	5	5	2	2	3	5	6	2	2	5	2	7	6	7	6	2	1	6	6	2	6	1	2	1	6	8	
Femmina	tra i 16 e i 24 anni	Studente/ssa	Alta	6	6	3	2	2	5	6	6	2	3	6	1	6	6	7	6	2	1	1	6	2	7	1	2	2	7	8
Maschio	tra i 16 e i 24 anni	Impiegato/a	Alta	4	5	2	1	4	5	7	4	1	1	6	1	5	6	7	6	1	4	1	7	3	4	2	1	2	7	9
Maschio	tra i 25 e i 40 anni	Operai/o/a	Media	7	6	1	2	2	6	6	4	4	2	6	1	6	6	7	7	2	2	6	2	7	2	1	1	7	10	
Maschio	tra i 16 e i 24 anni	Impiegato/a	Media	7	5	1	2	2	6	6	5	3	1	6	2	5	6	7	6	3	2	2	7	2	6	2	2	3	7	9
Femmina	tra i 25 e i 40 anni	Liberoprofessionista	Media	6	5	2	3	4	5	6	4	2	2	6	2	6	7	7	6	5	1	2	6	4	6	1	2	2	5	6
Maschio	maggiore di 40 anni	Liberoprofessionista	Alta	7	6	1	1	2	6	7	6	1	1	7	1	7	6	7	7	2	1	1	7	1	1	1	7	9	9	
Maschio	tra i 16 e i 24 anni	Studente/ssa	Media	7	7	2	2	4	6	7	7	1	3	6	2	4	5	7	5	1	2	1	7	2	2	1	1	7	9	
Femmina	maggiore di 40 anni	Liberoprofessionista	Bassa	6	3	3	4	2	6	5	4	4	3	6	2	6	4	6	6	4	2	6	3	6	3	2	2	6	8	
Maschio	tra i 25 e i 40 anni	Operai/o/a	Bassa	5	6	1	2	2	6	3	3	2	3	2	5	5	6	5	3	2	3	5	3	4	2	3	4	6	7	
Maschio	tra i 16 e i 24 anni	Studente/ssa	Alta	6	6	1	1	4	6	5	2	2	4	1	6	7	6	6	2	1	2	7	3	6	1	2	1	6	9	
Femmina	tra i 25 e i 40 anni	Studente/ssa	Bassa	6	4	2	4	3	6	6	4	2	2	4	3	3	4	6	6	4	2	3	6	5	4	2	3	3	6	8

Sitografia

- [¹] <https://www.superdataresearch.com/2019-year-in-review>
- [²] <https://it.wikipedia.org/wiki/Advergame>
- [³] <https://developer.android.com/studio/intro/>
- [⁴] https://en.wikipedia.org/wiki/Android_Studio
- [⁵] <http://www.construct.net>
- [⁶] <https://cordova.apache.org/docs>
- [⁷] <https://en.wikipedia.org/wiki/Firebase>
- [⁸] <https://firebase.google.com/>
- [⁹] <https://cloud.google.com/maps-platform>
- [¹⁰] <https://opencagedata.com/api>
- [¹¹] <https://firebase.google.com/docs/reference/android/com/google/firebase/auth/FirebaseAuth#public-taskauthresult-createuserwithemailandpassword-string-email,-string-password>
- [¹²] <https://firebase.google.com/docs/reference/android/com/google/firebase/auth/FirebaseAuth#public-taskauthresult-signinwithemailandpassword-string-email,-string-password>
- [¹³] <https://www.pubnub.com/learn/glossary/what-is-geohashing/>
- [¹⁴] <https://github.com/MichaelSolati/geofirestore-js/tree/v3.4.3>
- [¹⁵] <https://github.com/googlemaps/android-maps-utils>
- [¹⁶] <https://github.com/google/volley>
- [¹⁷] https://it.wikipedia.org/wiki/Cyclic_redundancy_check
- [¹⁸] <https://it.emcelettronica.com/cyclic-redundancy-check-crc-concetti-basilari>
- [¹⁹] <https://www.html.it/pag/16477/la-crittografia-a-chiave-pubblica-e-rsa/>
- [²⁰] [https://it.wikipedia.org/wiki/RSA_\(crittografia\)](https://it.wikipedia.org/wiki/RSA_(crittografia))
- [²¹] <https://cron-job.org/en/>
- [²²] <https://github.com/zxing/zxing>
- [²³] https://it.wikipedia.org/wiki/Organizzazione_internazionale_per_la_normazione
- [²⁴] <https://www.nngroup.com/articles/ten-usability-heuristics/>
- [²⁵] <https://www.ueq-online.org/>