Progetto sviluppato da : Alessandro Petrini 110198

COS'È PBCOMPANION

PBcompanion è un'applicazione web creata per permettere agli utenti del gioco immaginario "PlanetB" di restare connessi anche al di fuori della piattaforma di gioco. Sulla web app in questione è infatti possibile restare aggiornati su tutte le novità del gioco e i suoi aggiornamenti, consultare la classifica dei giocatori piú forti, interagire e gestire la propria lista amici e la propria squadra.

Per quanto invece riguarda la gestione del proprio profilo utente, oltre a chiaramente la possibilità di registrarsi e accedere con il proprio profilo, è possibile aggiungere una propria immagine del profilo con cui farsi riconoscere anche dagli altri utenti. È inoltre possibile effettuare un reset della password, in caso la si voglia cambiare o venga dimenticata.

TECNOLOGIE UTILIZZATE

Per la creazione di questa web app sono state utilizzate principalmente le funzionalità offerte dai framework angular e ionic permettendo quindi all'applicazione di essere responsive, e di poter quindi girare in maniera soddisfacente su dispositivi con caratteristiche hardware e software diverse. I principali linguaggi utilizzati sono quindi HTML, Typescript e SCSS. Per quanto invece riguarda il lato backend sono stati sfruttati i servizi offerti dall'infrastruttura firebase, quindi authentication, storage e database.

SVILUPPO

MENU

Uno dei primi componenti creati è stato sicuramente il menu di navigazione laterale, che rappresenta il componente di navigazione principale per dispositivi mobile. In quanto tale, il menu di navigazione funge da punto di accesso per le varie pagine su cui è possibile navigare all'interno dell'applicazione, ed è per questo molto importante il suo routing-module.

Nel routing-module del menu, oltre ad essere elencate le varie rotte verso cui è possibile navigare, sono presenti anche importanti istruzioni che si occupano di far rispettare le regole di accesso ad alcune pagine che hanno bisogno che l'utente sia autenticato (come la pagina relativa al proprio account o quella della community), per cui respingeranno, attraverso le auth-guard, gli utenti che non hanno effettuato l'accesso che tentano di accedere ad aree riservate agli utenti autenticati del sito.

Piú precisamente sono state utilizzate le guard canActivate (che si occupa di controllare una certa condizione prima che l'utente venga indirizzato sulla pagina in questione) e le pipe redirectUnauthorizedTo() (che si occupa di reindirizzare gli utenti non autenticati ad una diversa rotta) e redirectLoggedInTo (che si occupa di reindirizzare gli utenti autenticati ad una diversa rotta).

Infine, anche la parte html del menu, così come anche molti altri componenti della web app, è stata definita in modo tale da non dover essere modificata in caso si voglia in futuro aggiungere altre pagine, questo perché la lista di pagine navigabili viene generata in base ai contenuti di una struttura dati presente nella pagina typscript.

Un'altra caratteristica molto importante del menu è che presenta anche al suo interno il tag che identifica l'header (app-header) permettendo quindi di essere visualizzato in tutte le pagine ed il router-outlet per permettere la navigazione applicando le regole implementate.

HEADER

Anche l'header, così come il menu laterale, costituisce uno dei componenti piú importanti dell'applicazione poiché costituisce parte integrante di ogni pagina e viene infatti condiviso tra tutte le pagine. Oltre a fornire

da punto di accesso per le altre pagine come il menu laterale, rende molto piú rapida ed intuitiva la navigabilià all'interno dell'applicazione. Per essere presente in ogni pagina, l'header è stato inserito all'interno della pagina html del menu utilizzando il suo apposito tag app-header.

Una particolarità dell'header è il fatto che al suo interno presenta in realtà due distinti header, in modo tale da permettere sempre una corretta visualizzazione a prescindere dal dispositivo in uso. Infatti, uno dei due, molto piú minimale nei contenuti, si rivolge agli utenti che accedono alla web app da dispositivi di piccole dimensioni, considerabili quindi mobile, mentre l'altro, piú carico in termini di contenuti e di stile, si rivolge a dispositivi con risoluzioni piú alte come ad esempio i computer.

Un'implementazione di questo tipo è stata principalmente possibile grazie al tag @media nel SCSS con cui si va ad effettuare una differenziazione dello stile a seconda della larghezza dello schermo dell'utente (max-min-width).

LOGIN-REGISTER

Per quanto riguarda la pagina che gestisce l'accesso e la creazione di nuovi profili, la parte html è stata implementata sfruttando un ion-segment offerto dal framework ionic, che permette di alternare la visualizzazione dei contenuti della pagina a seconda della scelta dell'utente, che può quindi scegliere se visualizzare il form relativo al login o quello relativo alla registrazione.

Entrambi i form sono stati realizzati sfruttando il FormGroup e il FormControl del framework angular, utili per tenere traccia dei contenuti e della validità degli input dell'utente, ma anche per accedervi da lato Typescript. Sui vari input vengono infatti applicati vari controlli di validità, tra cui il rispetto della lunghezza, l'uguaglianza delle password nella registrazione ed altri.

Per quanto riguarda le logiche di accesso e registrazione presenti in questo modulo, abbiamo una semplice integrazione di alert e popup di caricamento (questi ultimi offerti dal LoadingService, servizio appositamente creato) per fornire all'utente un responso visivo di ciò che sta accadendo e di eventuali problemi durante l'esecuzione del comando. Ovviamente la vera logica di registrazione ed accesso viene completamente delegata ad un servizio appositamente creato, chiamato AuthService, mentre altre funzioni accessorie sempre relative ad altri dati utente vengono affidate allo UserService.

HOME

La home può essere considerata la piú semplice tra le pagine principali della web app, infatti si limita a presentare all'utente i dati ottenuti dal server firebase, elencando sotto forma di ion-cards i contenuti da visualizzare, che consistono principalmente di immagini e contenuti testuali.

Per quanto riguarda il lato typescript, abbiamo semplicemente una chiamata al servizio che si occupa di ottenere i dati dal server, ovvero il DataService, restituendo un observable da cui si estraggono i dati.

ACCOUNT

Questa pagina ha l'obiettivo di mostrare all'utente i dati relativi al proprio profilo e fornire alcune possibilità di gestione del proprio account, come il cambio password o il caricamento di una nuova immagine del profilo.

Per quanto riguarda la logica di questa pagina, le vere e proprie funzioni che realizzano le attività di cambio immagine del profilo, logout e reimpostazione password, vengono richiamate dal servizio che si occupa principalmente di gestire i profili utente, ovvero il UserService. Il tutto anche qui accompagnato da popup di caricamento del LoadingService per una migliore esperienza visiva per l'utente.

Per quanto riguarda la scelta dell'immagina da cambiare è stato utilizzato il plugin API capacitor/camera, permettendo quindi la selezione di una foto nella propria galleria o l'utilizzo della fotocamera per ottenere l'immagine da caricare sul server come immagine del profilo.

ABOUT

Questa pagina non presenta alcuna interazione né da parte dell'utente, né da parte del server poiché viene utilizzata semplicemente per mostrare informazioni statiche riguardo al gioco e al suo sviluppo che non sono destinate a modifiche o altre funzioni.

RESET PASSWORD

Il principale scopo di questa pagina è quello di presentare la funzionalità del reset password all'utente e presenta semplicemente un form costituito da un solo input per l'email, dove eventualmente verranno inviate le istruzioni per il cambio password. Anche qui è stato utilizzato un semplice FormGroup, i cui dati vengono passati all'AuthService, che si occupa di gestire i dati e l'autenticazione dell'utente tramite firebase.

VERIFY EMAIL

Lo scopo principale di questa pagina è quello di notificare l'utente del fatto che è stato inviato il link di conferma di registrazione del proprio account alla propria email.

LEADERBOARDS

Questa è la pagina che si occupa di mostrare la classifica dei giocatori iscritti nella piattaforma e le loro statistiche di gioco.

Nella parte html, così come ad esempio la pagina home, viene utilizzato all'interno di una ion-row (righe della tabella) un controllo *ngFor, che scorre la lista di utenti registrati nella piattaforma ed mostrando i dati di ognuno di essi all'interno di ogni ion-col, ovvero le colonne che compongono la tabella.

L'ordine con cui gli utenti vengono mostrati viene definito da un campo chiamato orderType, al quale viene attribuito un numero che corrisponde alla caratteristica con cui si vuole ordinare la classifica (1-ordine alfabetico, 2-ordine di kill, 3-ordine di morti e così via), mentre cliccando per una seconda volta sulla caratteristica per cui si vuole ordinare verrà invertita la classifica (crescente/decrescente e viceversa).

I dati relativi ai giocatori vengono ottenuti dal server firebase attraverso il servizio UserService.

COMMUNITY

Questa pagina permette ad un utente autenticato di gestire e visualizzare la propria lista amici.

Anche qui per quanto riguarda la parte html, la lista amici viene visualizzata attraverso un controllo *ngFor, mentre grazie ad un controllo *ngIf sulla lunghezza della lista amici, se non si ha alcun amico nella lista amici verrà mostrata all'utente una nota al riguardo.

Oltre a poter visualizzare i propri amici e le relative immagini del profilo, è possibile aggiungere o rimuovere degli utenti dalla propria lista. In particolare, per aggiungere un amico basterà inserire il suo nome e cliccare il relativo pulsante e, se corretto, lo ritroveremo nella lista amici, mentre per cancellarlo bastera cliccare il relativo pulsante affiancato al suo nome, facendolo scomparire dalla lista amici.

Queste funzioni vengono realizzate richiamando le funzioni presenti nello UserService.

SERVIZI

AUTH SERVICE

Questo servizio si occupa di gestire la logica di accesso e registrazione alla web app, sfruttando i servizi messi a disposizione da Firebase Authentication, in particolare utilizzando le funzioni createUserWithEmailAndPassword per la creazione utente, signInWithEmailAndPassword per l'accesso, signOut per effettuare il logOut dal proprio account, sendPasswordResetEmail per reimpostare la password e sendEmailVerification per inviare una mail di conferma agli account appena creati. Questi servizi permettono inoltre di salvare all'interno dell'infrastruttura firebase i dati relativi agli utenti senza tener traccia però delle password, che sono hashate e mantenute dal provider.

DATA SERVICE

È il servizio che si occupa di fornire i dati ottenuti da firebase storage e firebase database grazie ai servizi messi a disposizione da Firestore come collectionData, che fornendogli un riferimento del firestore e della directory da cui ottenere i dati, restituisce un observable contentente tutti i dati presenti nella cartella HomeContents del database e getDownloadURL per ottenere l'url delle immagini caricate nel firebase storage e poterle mostrare nella home. I dati verranno poi utilizzati quindi per popolare i contenuti della home, altrimenti vuota, attraverso la sottoscrizione all'observable.

USER SERVICE

È il servizio in generale più utilizzato poiché fornisce i servizi necessari agli utenti della piattaforma.

Uno dei servizi più importanti è quello che si occupa di creare ed inizializzare i dati del nuovo utente nel database in fase di registrazione, utilizzando la funzione setDoc del servizio Firestore per popolare il profilo utente nel database con i dati passati.

Molto importante è anche il servizio di upload di immagini del profilo. Questo si occupa innanzitutto di caricare l'immagine selezionata dall'utente sul firestore storage e successivamente di ottenere un url relativo all'immagine che verrà salvato sul profilo utente tramite il servizio updateDoc (simile al setDoc, ma utilizzabile anche su profili gia inizializzati poiché aggiorna solamente il campo desiderato), per poter essere mostrata come immagine del profilo.

Per quanto riguarda la pagina community, abbiamo il servizio che restituisce la lista amici del profilo attualmente autenticato con una semplice query che va a restituire i profili degli utenti che hanno nella propria lista amici l'id dell'utente attualmente autenticato.

Infine, abbiamo i servizi utilizzati per aggiungere e rimuovere utenti dalla propria lista amici, anche questi realizzati principalmente andando a modificare le liste amici degli utenti in questioni con un comando updateDoc. In particolare, verrà utilizzato all'interno di updateDoc, il comando arrayRemove per rimuovere dall'array, che costituisce la lista amici, l'id dell'amico da rimuovere, ripetuto per entrambi i profili. Simile comportamento si ha anche per l'aggiunta di un nuovo amico, ma in questo caso all'interno di updateDoc si andrà ad utilizzare un comando arrayUnion a cui si passa l'id dell'amico da aggiungere in modo tale da aggiungere alla lista un nuovo id.

Molto utile è anche il servizio che, a partire da un nickname, restituisce un oggetto di tipo utente contenente tutti i dati relativi all'utente in questione, effettuando una semplice query sul database e restituendo il risultato come oggetto utente.