Bingo

Laboratorio di applicazioni mobili

A.c.: 2021/2022

Carboni Carlotta

Num. Matricola: 893185

Carlotta.carboni2@studio.unibo.it

Alsina Gabriel Riccardo

Num.Matricola: 890823

Gabriel.alsina@studio.unibo.it

[serve come dritta poi è da cavare]

1. Manuale utente

* Descrizione 3 bottoncini
* Partita
* Spiegare funzionamento crediti + Market
* Vincita

1. Codice

* Tecnologie usate
  + Angular+Ionic+capacitor
  + Firebase
* scelte progettuali
  + algoritmi
    - generatore schede
    - bossolo
    - partita
  + db no socket

- manuale utente:

La prima volta che si apre l’app, dopo averla scaricata, si aprirà su una pagina che permette di registrarsi o di loggarsi. Per registrarsi verranno richiesti dei dati, se non vengono inseriti tutti, o se l’username scelto è già in uso comparirà un alert per segnalare il problema. Una volta registrati o loggati, si verrà reindirizzati alla home page. [foto home]

In alto a sinistra si trovano i crediti, necessari per poter giocare. Nel search-box è possibile inserire il codice di una partita: questo box è l’unico modo per poter accedere alle partite private, ma si può anche utilizzare per cercare partite pubbliche. Le partite pubbliche sono comunque tutte visibili nella tabella sottostante.

Per entrare in una partita basta cliccare sul bottone entra, mentre per crearla a basterà cliccare su crea partita, in entrambi i casi è necessario avere almeno un credito, ovvero il cotso di una scheda. Se non si hanno abbastanza crediti si verrà avvisati tramite un alert.

Dopo aver cliccato su crea partita si dovrà scegliere il tipo di partita: pubblica o privata, la differenza tra le due tipologie è che quella pubblica sarà visibile a tutti, mentre la privata solo a chi è a conoscenza del codice della partita.

Eseguita la scelta entriamo nell’istanza della partita. [foto]

In alto si trovano i dati della partita: codice, proprietario, partecipanti e montepremi (che è in crediti), questi ultimi 2 vengono aggiornati con l’entrata di nuovi giocatori e con l’acquisto di nuove cartelle.

I bottoni per gestire la partita, che saranno diversi nel caso non si sia il proprietario, [foto non proprietario] e un bottone per la visualizzazione della chat. Nel pre-partita il proprietario e i giocatori potranno ancora uscire dalla partita, venendo rimborsati dei crediti spesi, tramite i rispettivi bottoni di uscita o tramite i tabs. Se a uscire dalla partita è il proprietario la partita verrà cancellata e tutti i giocatori verranno reindirizzati alla home, avvisati da un alert che spiega il motivo dell’uscita, mentre se a uscire è un giocatore verrà semplicemente ridotto il numero dei giocatori. Se i giocatori escono a partita iniziata non verranno rimborsati, mentre se a partita iniziata è il proprietario ad uscire i giocatori verranno rimborsati dei crediti.

Quando si entra/si crea una partita viene generata e comprata una scheda, ma tramite al bottone compra una scheda se ne potranno comprare altre fino ad un massimo di 3 per giocatore. Il bottone compra scheda è visibile finché non inizia la partita o finché non si comprano 3 schede, anche se non si hanno crediti per comprare altre schede il bottone non viene disabilitato, ma in assenza di sufficienti crediti, cliccandolo, comparirà un alert che permetterà di acquisire 10 crediti senza abbandonare la partita ( è comunque possibile rifiutare l’acquisto dei 10 crediti).

Sopra alle schede è presente un toggle: “Evidenzia caselle chiamate”, attivandolo quando un numero presente nella cartella verrà estratto questo cambierà colore. ATTENZIONE: il fatto che la cella si illumini non significa sia stato segnato, il cambio colore serve solo per facilitare la verifica dei numeri estratti, bisognerà sempre poi cliccare sulla cella!

Per poter iniziare una partita il proprietario deve cliccare su start partita, per poterla iniziare devono esserci almeno 2 giocatori, se si clicca senza che il numero minimo sia rispettato il proprietario verrà avvisato da un alert che ricorderà il numero minimo di partecipanti necessari.

Una volta che viene fatta iniziare una partita (questa non sarà più trovabile nell’elenco delle partite) comparirà il tabellone e inizierà l’estrazione dei numeri, mentre scompariranno la chat, i dati della partita e la possibilità di comprare una nuova scheda.

I bottoni per il bingo e la cinquina sono di default disabilitati e si abilitano solo quando sono presenti le condizioni per la vincita: nel caso della cinquina bisogna avere **segnato** tutta una riga, mentre per il bingo bisogna aver **segnato** tutta la cartella. ATTENZIONE: bisogna SEGNARE, non basta che i numeri vengano estratti, finché i numeri non sono segnati (le celle non sono segnabili finché il loro numero non viene estratto) non si abiliteranno i bottoni. Se qualcuno fa cinquina verrà segnalato con un fumetto con scritto “cinquina” (basterà cliccarci sopra per farlo sparire) e ovviamente non si potrà più fare cinquina. Mentre se qualcuno fa bingo la partita finisce, il tabellone scompare e comparirà un riepilogo della partita: con riportato chi ha vinto, cosa, quanto e con quanti numeri estratti è stato fatto bingo.

Non spariranno le schede per permettere agli utenti di continuare a guardarle e dire “daiiii, me ne mancava solo uno”.

Finita la partita sarà possibile uscire o cliccare su continua per giocare un’altra partita con le stesse persone. Se il proprietario clicca su esci non sarà possibile continuare la partita e, sia che si sia già cliccato su continua o no, si verrà reindirizzati alla home page insieme a un alert che informerà del motivo della chiusura della partita. Una volta che il proprietario fa finire la partita questa verrà cancellata dal db.

Altra funzionalità della nostra app è il Market. Il market è raggiungibile tramite il tabs Market in basso al centro, oppure cliccando l’immagine dei soldi affianco i crediti nella home page.

[foto]

In questa pagina è possibile comprare nuovi timbri o crediti, i timbri si comprano con i crediti mentre i crediti si comprano con i soldi.

L’ultimo tabs è quello che indirizza al profilo, in questa pagina è possibile visualizzare i propri dati e le statistiche di gioco: partite fatte, quante volte abbiamo fatto bingo, quante volte cinquina e quante volte il super bingo, è sempre bingo, ma per farlo non devono essere stati estratti più di X numeri essendo così difficile nel caso lo si riesca a fare il montepremi diventa di 1000 crediti indipendentemente dal numero delle schede in gioco.

Sempre qui è possibile cambiare il proprio timbro, il timbro che attualmente è in uso è quello evidenziato di giallo.

Come ultima cosa in questa pagina è possibile effettuare il logout.

- codice:

- tecnologie usate

Per la realizzazione di questo progetto abbiamo deciso di creare un’app ibrida sviluppata con Angular e Ionic.

La motivazione principale che ci ha fatto sviluppare un'app ibrida con questo framework è il fatto che sia Cross-platform: Ionic consente agli sviluppatori di creare app eseguibili su più piattaforme, tra cui iOS, Android e Web, con un'unica base di codice.

Un altro punto a favore per Angular è il fatto che utilizzi il pattern architetturale Model-view-controller (MVC), Angular implementa questo pattern tramite la sua struttura a componenti, dove ogni componente funge da modello, vista e controller allo stesso tempo, gestendo i propri dati e la propria logica. Ogni componente viene sviluppato come un elemento separato e autonomo. Questi componenti possono essere riutilizzati in diverse parti dell'applicazione, rendendo il codice più organizzato, modulare e facile da gestire.

Come database abbiamo utilizzato Firebase, perché facile da implementare e utilizzare, non è necessario configurare server o gestire la scalabilità, e in più è un Real-time database, ovvero un database NoSQL in tempo reale, che significa che i dati vengono automaticamente sincronizzati su tutti gli elementi in ascolto. Questo rende facile costruire app che supportano la collaborazione in tempo reale tra gli utenti.

- scelte progettuali:

- algoritmi:

Mostrare tutto il codice in un documento è impossibile, quindi abbiamo deciso di spiegare le parti che abbiamo ritenuto più interessanti:

* Generatore-scheda.ts

Per la creazione delle schede abbiamo creato un servizio: generatore-scheda.ts.

La generazione di una cartella la possiamo dividere in 3 punti:

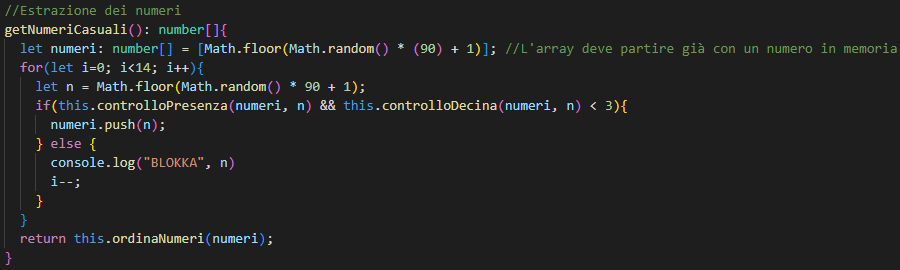
* Prendere i numeri;
* Sistemare le caselle bianche
* Ritornare i valori

Prendere i numeri:

I numeri della cartella vengono estratti in maniera casuale ma ci sono delle regole:

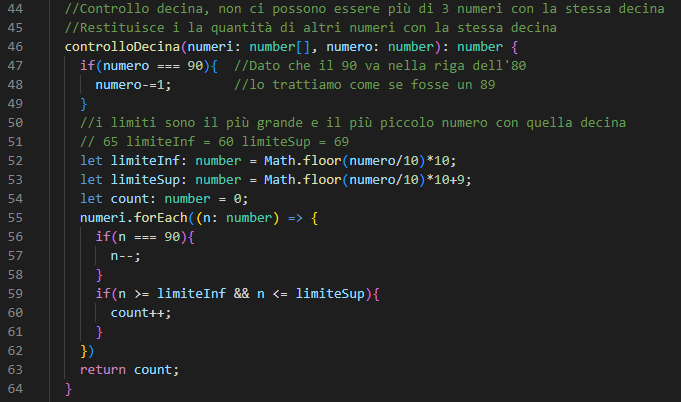
* Non ci possono essere due numeri uguali (banale)
* Non ci possono essere più di 3 numeri con la stessa decina, inoltre, il 90 fa parte della decina dell’80.

Quindi, per ogni numero estratto, facciamo controlli su queste due regole.



Per la prima regola è facile, basta controllare che il numero generato non sia già presente nell’array numeri tramite il metodo controlloPresenza().

La seconda è un attimo più complicata:



Innanzitutto, dato che il 90 lo dobbiamo trattare come se facesse parte della decina del 8, lo decrementiamo di uno . Calcoliamo quelli che abbiamo denominato limiteInferiore e limiteSuperiore, i limiti sono il più piccolo e il più grande numero appartenenti alla decina presa in considerazione (es. 65: limiteInf = 60 e limiteSup = 69), vengono calcolati facendo l’arrotondamento di numero/10 e lo moltiplichiamo per 10, per esempio:

65 / 10 = 6.5 troncato diventa 6 per 10 fa 60, il limite superiore è lo stesso ma più 9.

Successivamente scorriamo tutti i numeri che abbiamo già salvato e controllo quanti numeri fanno parte dell’intervallo [limiteInf, limiteSup], se un numero ne fa parte incremento una variabile count, alla fine del metodo ritorno count. Se count è maggiore di 3 non posso aggiungere il numero calcolato all’array di numeri.

Finito il metodo richiamo il metodo per avere tutti i numeri in ordine.

Una volta ottenuti tutti i numeri parte la seconda parte dell’algoritmo, ovvero sistemare le caselle bianche, per farlo dividiamo questo problema in due sottoproblemi:

* Inserire le caselle bianche
* Ordinare le caselle nella cartella.

Per inserire le caselle bianche utilizziamo il metodo aggiungiCaselleVuote()

Immagine che contiene testo

Descrizione generata automaticamente

Le variabili fondamentali di questo metodo sono tre:

* *indexNumeri*: che sarebbe la decina che stiamo controllando;
* *count*: il numero di caselle vuote che dobbiamo aggiungere;
* *numeriConZero*: che è l’array dove salviamo tutti i 27 numeri.

Scorriamo tutti i numeri, facciamo un controllo sulla decina per verificare se sia la stessa di *indexNumeri*, se è la stessa aggiungo il numero a *numeriConZero* e decremento *count*, se invece è diversa dobbiamo aggiungere delle caselle vuote che rappresentiamo con zero. Il numero di zeri da aggiungere per questa decina è il valore di count più 3 \* (la decina che stiamo controllando - indexNumeri - 1), questo perché si potrebbe saltare una decina, per capire meglio facciamo un esempio:

l’ultimo numero che abbiamo aggiunto è il 22, quindi *indexNumeri* sarà 2, lo aggiungiamo e decrementiamo count, il prossimo numero che aggiungiamo sarà 30, fa parte della 3° decina, quindi dobbiamo aggiungere degli zeri, aggiungiamo il valore di *count* + ( 3 \* ( 3 - 2 - 1)) quindi solo il valore count, questa formula serve se un prossimo numero salta la decina, facciamo l’esempio ora che il prossimo numero da aggiungere sia 56, fa parte della 5° decina, e aggiungiamo *count* + ( 3 \* (5 - 3 - 1)), questa formula aggiunge gli zeri per la 4° che non c’è.

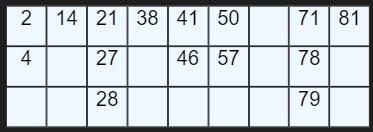
I controlli appena spiegati vengono eseguiti su tutti gli elementi dell’array numeri tranne sul numero 90 (nel caso ci sia), perché questo andrà sicuramente infondo ed è inutile quindi farci dei controlli (ricordiamo che l’array numeri è anche già stato ordinato).

Dopo aver aggiunto gli zeri aggiungiamo il numero che stiamo controllando e aggiorniamo *indexNumeri* e *count.*

Alla fine, aggiungiamo degli zeri in fondo, se ce ne mancano, fino ad arrivare a 27 numeri.

Passeremo da avere un array [2, 4, 14, 21, 27, 28, 38, 41, 46, 50, 57, 71, 78, 79, 81] ad avere un array [2, 4, 0, 14, 0, 0, 21, 27, 28, 38, 0, 0, 41, 46, 0, 50, 57, 0, 0, 0, 0, 71, 78, 79, 81, 0, 0].

Ora dovremmo sistemarli, perché fatto così la scheda ci rimane così.



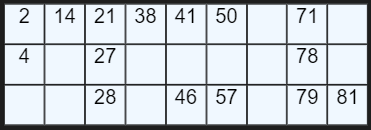
Ogni riga deve avere 5 caselle con numeri (e di conseguenza 4 caselle bianche). Per sistemare la caselle bianche usiamo il metodo sistemaBianche().

[services/generatore-cartella.service.ts righe: 155-209]

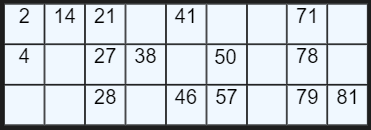
Prima di tutto sistemiamo i numeri dell’array in una matrice identica alla cartella sopra e contiamo il numero di caselle vuote nell’ultima riga.

Partiamo a sistemare la cartella sempre dall’ultima riga, mentre la scelta della colonna da cui partire viene fatta casualmente, per rendere la creazione della cartella il più casuale possibile. Se l'ultima riga nell'indice appena estratto ha zero proviamo a scambiarlo con il numero della riga sopra (matrice[1]), se anche quello è zero proviamo con la prima, se anche quella è vuota aggiungiamo un giro al for ( ), perché questo è andato a vuoto e quindi la nostra riga ha ancora lo stesso numero di caselle vuote.

Alla fine di questo pezzo, avremo sistemato l’ultima riga e avremo la cartella così:



Facciamo la stessa cosa ma per la seconda riga scambiandola solo con la prima, alla fine di tutto avremo una cartella completa.



Ultimo passaggio di questo servizio sarà estrarre i dati, per ritornare i dati al component che ha richiamato il servizio: ritorniamo un array con i numeri e gli 0 nell'ordine precedentemente fatto, solo i numeri e le tre cinquine estratte dalla matrice.