

CONTI MATTEO

Laboratorio Applicazioni Mobile 2018/2019

BRISCOLA

Riproduzione del classico gioco di carte italiano per due giocatori.

introduzione

Scopo

Lo scopo dell'applicazione è la riproduzione del funzionamento del gioco di carte italiano, chiamato Briscola, su dispositivi aventi sistema operativo iOS (piattaforma Apple). La versione attuale del gioco prevede la presenza di due giocatori e di due modalità di gioco.

Lo scopo era quello di creare due ambienti di gioco: il primo serviva per simulare una sorta di fase iniziale di "training" mentre il secondo serviva per testare i propri progressi confrontandosi con altri giocatori.

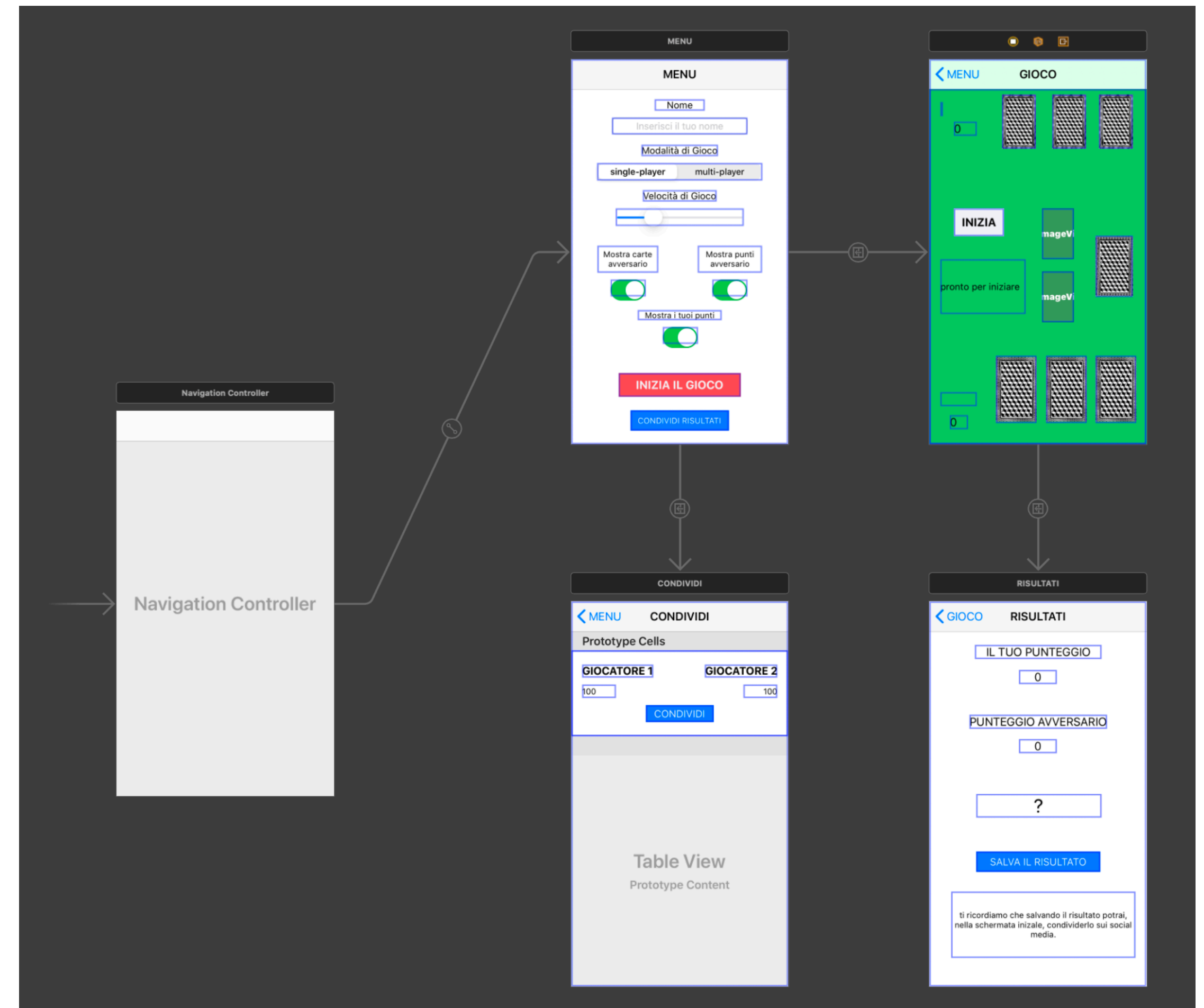
Illustreremo le scelte attuate durante la fase di progettazione, i problemi e le difficoltà incontrate, evidenzieremo alcuni dettagli che hanno inciso sulle tempistiche di consegna e mostreremo i flussi di dati presenti all'interno dell'applicazione.

MVC

Architettura Generale

Il pattern utilizzato nella realizzazione di questa applicazione è il Model-View-Controller (MVC). L'importanza di questo pattern consiste nella separazione della logica di presentazione dei dati rispetto alla logica di gestione di questi.

Per questo progetto infatti, sono stati creati diversi modelli di dati corrispondenti alle varie entità in gioco, tutti gestiti da diverse librerie di "gestione", le quali astraggono completamente il flusso di creazione/modifica dei dati rispetto alla loro presentazione.



CONTI MATTEO

Menù

qui è possibile impostare le varie opzioni di gioco e passare sia alla schermata di gioco, sia a quella “Social”.

Social

qui è possibile vedere la lista di tutte le partite salvate e, per ognuna, decidere se condividerla su Facebook.



Schermata di Gioco

qui è possibile avviare il gioco e passare alla schermata dei “Risultati”.

Condivisione Risultati

qui è possibile vedere il risultato della partita una volta terminata e decidere di salvarla.

modello Carta

Questo è il modello principale di tutto il gioco: al suo interno troviamo tutte le proprietà “naturali” di ogni carta (come punti e tipo) a cui ne seguono altre utilizzate per la fase di render e per la logica di gioco.

Questo modello inoltre contiene, al suo interno, la referenza e lo schema per l’associazione di una singola istanza alla corrispondente immagine. In questo modo è possibile avere direttamente la referenza alla risorsa.

Attributi Principali

— points

— type

— number

— name

modello Giocatore

Questo è il modello con il più alto numero di accessi nel codice sorgente: rappresenta una situazione “reale” di gioco, in cui ogni giocatore ha un numero di carte in mano e un mazzo (con le carte vinte).

Oltre alle proprietà di gioco, questo modello contiene anche alcune informazioni riguardo al ruolo che ha all'interno di questo (se è il giocatore corrente, se è emulato o se è un giocatore remoto).

Attributi Principali

— name

— cardsHand

— currentDeck

— index

— type

— deckPoints

libreria

Gestore Gioco

Il gestore del gioco è la classe che opera la “magia”: si occupa di tenere aggiornato lo stato del gioco ad ogni istante, modificando le opportune proprietà dei modelli.

Oltre a gestire lo stato del gioco, si occupa di evitare che questo finisca in uno stato “inconsistente”: ad esempio che un giocatore giochi una carta quando non è il suo turno.

Attributi Principali

— mode	— initialCards
— gameEnded	— deckCards
— aiPlayerEmulator	— cardsOnTable
— players	— trumpCard
— playerTurn	

libreria

Gestore Sessione

Questa classe viene chiamata in causa nella modalità multi-player: si occupa di attivare una sessione per il dispositivo corrente e automaticamente si mette alla ricerca di dispositivi connessi (Wifi, Bluetooth, ...) tramite una stringa di identificazione.

Durante la partita, il gestore del gioco chiede a questa classe di aggiornare lo stato del gioco (per il device in uso) a tutti i dispositivi connessi alla sessione.

Attributi Principali

— connectedPeers	— connectingPeers
— disconnectedPeers	— session
— peerID	




libreria

Gestore Database

Questa classe si occupa di astrarre completamente le entità salvate e le relazioni al loro interno. Nessuno, a parte lei, accede direttamente ai dati persistiti sul database in locale.

Questi dati vengono riutilizzati in diverse view: nella prima vengono formattati e salvati, mentre nella seconda visualizzati. Il sistema di relazioni presente nella base dati corrisponde esattamente alla rappresentazione dello stato finale del gioco.

Entità Principali

-  **Players** contiene solo i campi incide, nome e tipo.
-  **Results** contiene la serializzazione delle carte vinte e la somma dei loro punti.
-  **Matches** contiene la coppia di relazioni sopra definite, sia per il giocatore locale che per quello remoto/emulato.

Libreria Emulatore

È stata realizzata, infatti, come una macchina a stati che in base alla classificazione real-time delle carte in tavola e quelle in mano (dell'emulatore) elabora uno stato finale, che si traduce nella scelta di una carta da giocare (chiaramente tra quelle che ha in mano).

Il primo stato è quello nel quale l'emulatore si trova a dover giocare per primo, mentre il secondo è quando si trova a giocare per secondo (e abbiamo quindi già una carta in tavola).



POSSIBILI SCELTE DELL'EMULATORE

Briscola

1. gioca il liscio più alto che ho.
2. gioco il carico più basso che ho sotto i 10 punti.
3. gioco la briscola più bassa che ho (solo se non vale dei punti).
4. gioco la briscola più bassa che ho (solo se la carta in tavola vale dei punti e la mia supera quella in tavola).
5. gioco il carico più basso che ho.
6. gioco la briscola più bassa che ho.

Carico

1. gioco il carico più alto che ho di questo tipo ma solo se supera la carta in tavola.
2. gioco la briscola più bassa che ho.
3. gioco il liscio più alto che ho
4. gioco il carico più basso che ho.

Liscio

1. gioco il carico più alto che ho di questo tipo (solo se è un re, un tre o un asso).
2. gioco il liscio più alto che ho non di questo tipo.
3. gioco il liscio più basso che ho di questo tipo, a patto di non superare la carta in tavola.
4. gioco il carico più alto che ho di questo tipo.
5. gioco il liscio più basso che ho.
6. gioca il carico più basso che ho (fante o cavallo o re)
7. gioca la briscola più bassa che ho senza punti.
8. gioco il carico più basso che ho.
9. gioco la briscola più bassa che ho.

funzionalità

Grafica e Dati

Avendo illustrato nel capitolo precedente le 4 view principali all'interno dell'applicazione, è facile dedurre che queste corrispondono esattamente alle 4 funzionalità principali:

Configurazione del Gioco

Gioco (in modalità single/multi-player)

Salvataggio dei risultati

Condivisione dei risultati

funzionalità

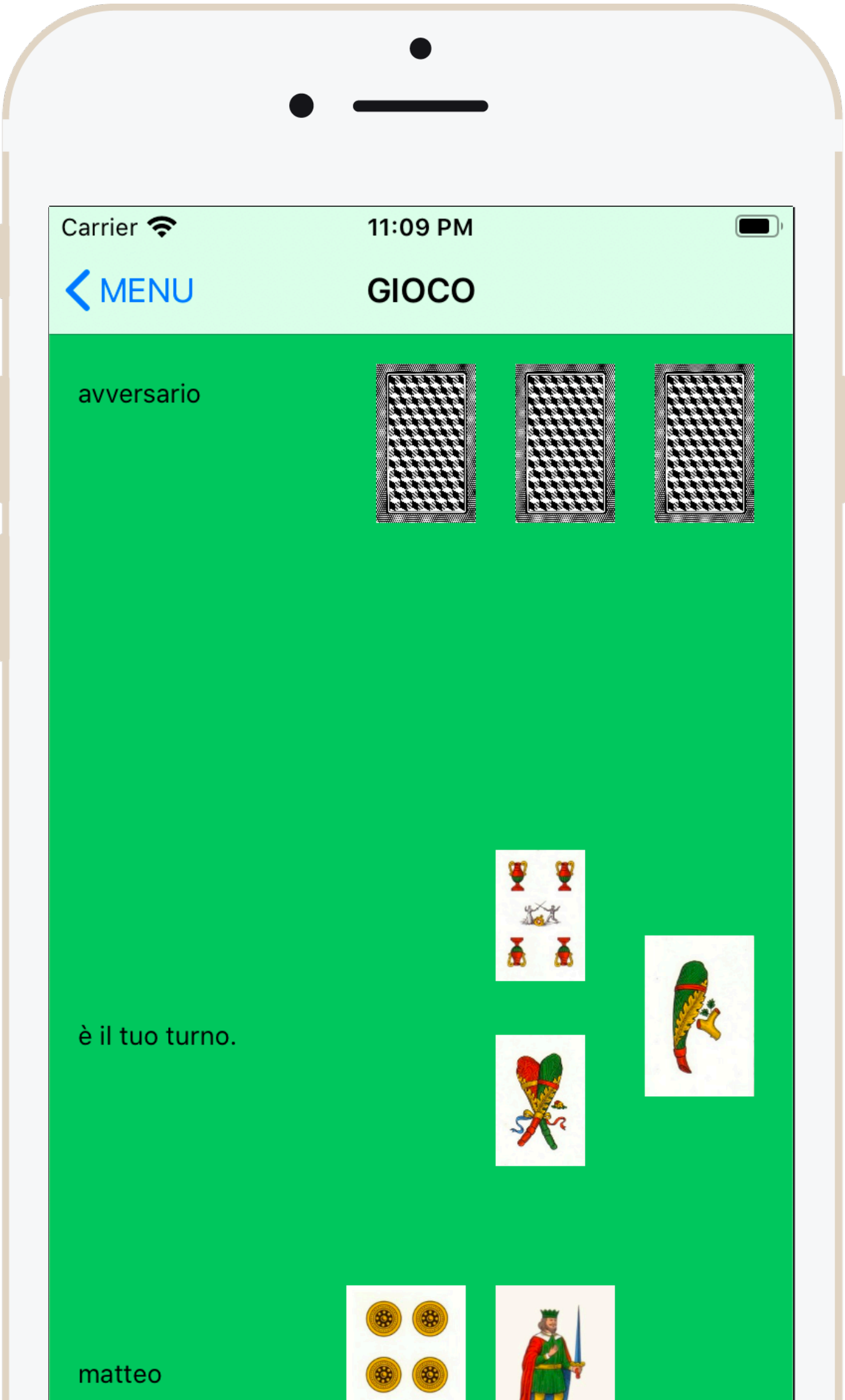
Interfaccia Grafica

Notiamo come le nostre carte sono scoperte (al contrario di quelle dell'avversario) e per poter giocare una carta è necessario toccarla. Una volta che anche il nostro avversario (emulatore/giocatore remoto) avrà giocato una carta, il vincente della mano si aggiudicherà le carte, ed esse verranno aggiunte al suo mazzo (non visibile), e automaticamente verrà distribuita una nuova carta (ad ogni giocatore) e la mano sarà pronta per iniziare (partirà il giocatore che ha vinto quella precedente)



Scalabilità

Questa disposizione grafica consente facilmente l'aggiunta di ulteriori 'postazioni' di gioco (per ulteriori giocatori).



Componenti Visibili

- Nome
- Punti (in real-time)
- Carte in Mano
- Carta in Tavola
- Messaggio di aiuto

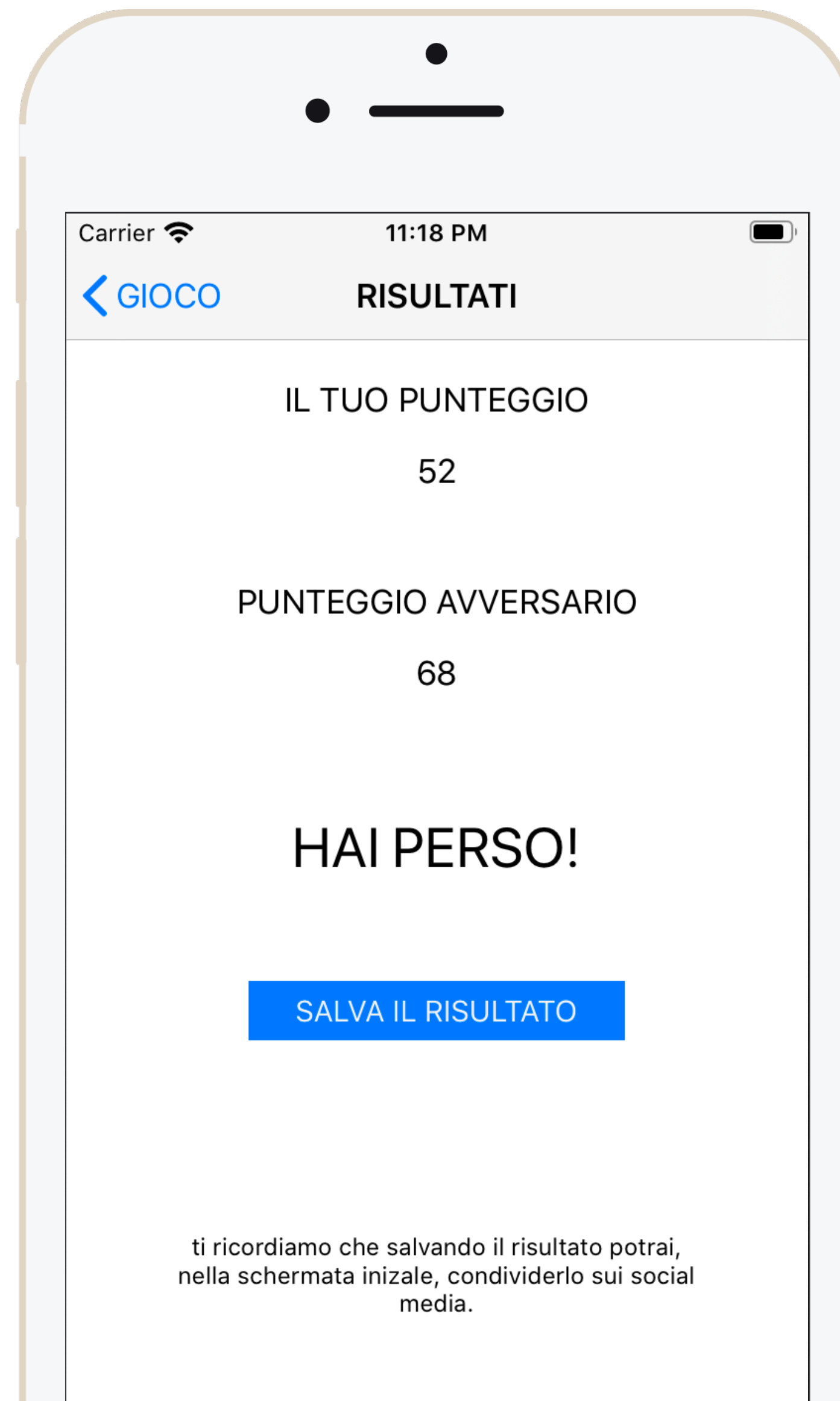
funzionalità Flusso Dati

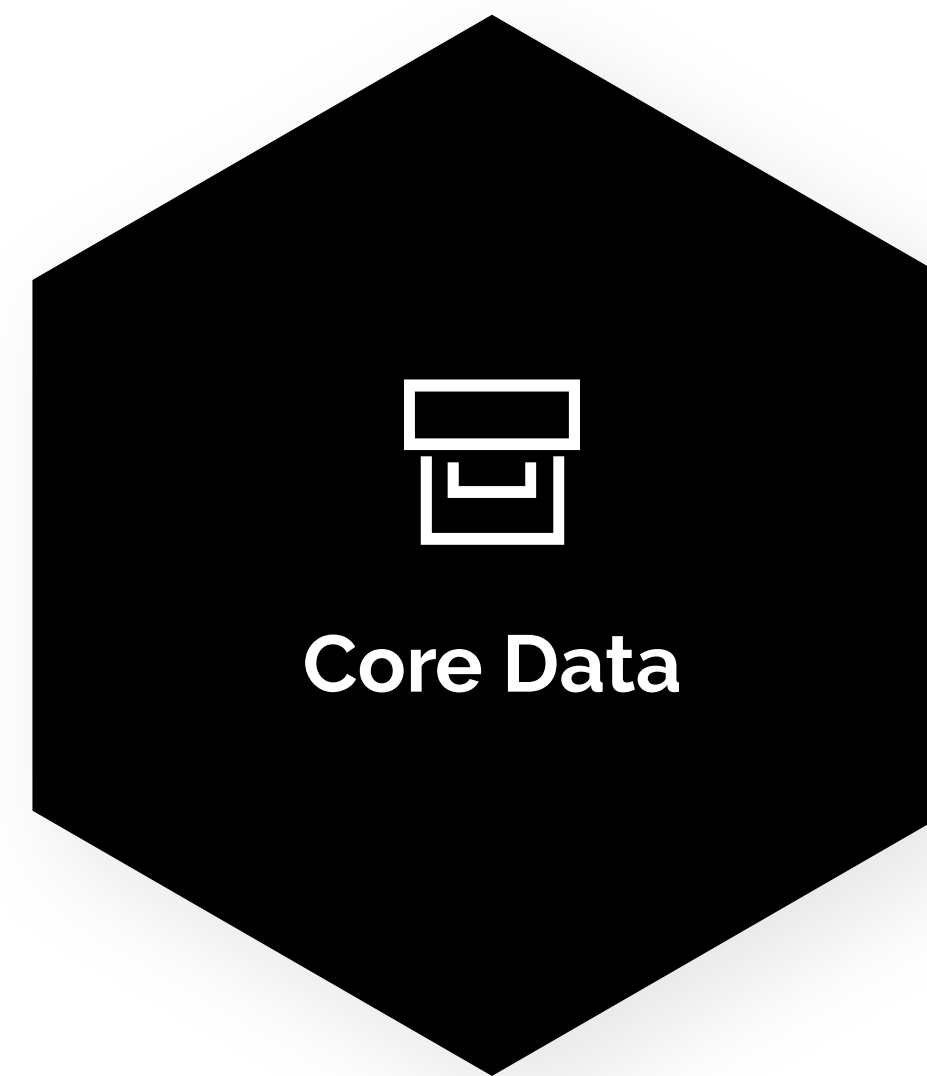
Il flusso dei dati persistiti è piuttosto semplice:

1. Appena avviata l'applicazione (nell'apposita schermata) è possibile visualizzare l'elenco delle partite salvate sul database interno.
2. Al termine di ogni partita è possibile salvare il risultato in modo facile e veloce.

Nessun dato viene salvato automaticamente e non è possibile salvare nessuna informazione sulla partita in corso finché questa non è terminata.

Inoltre non esiste nessun meccanismo di salvataggio dati in remoto: ovvero nel caso un utente cancelli l'applicazione, tutti i dati al suo interno verranno persi.

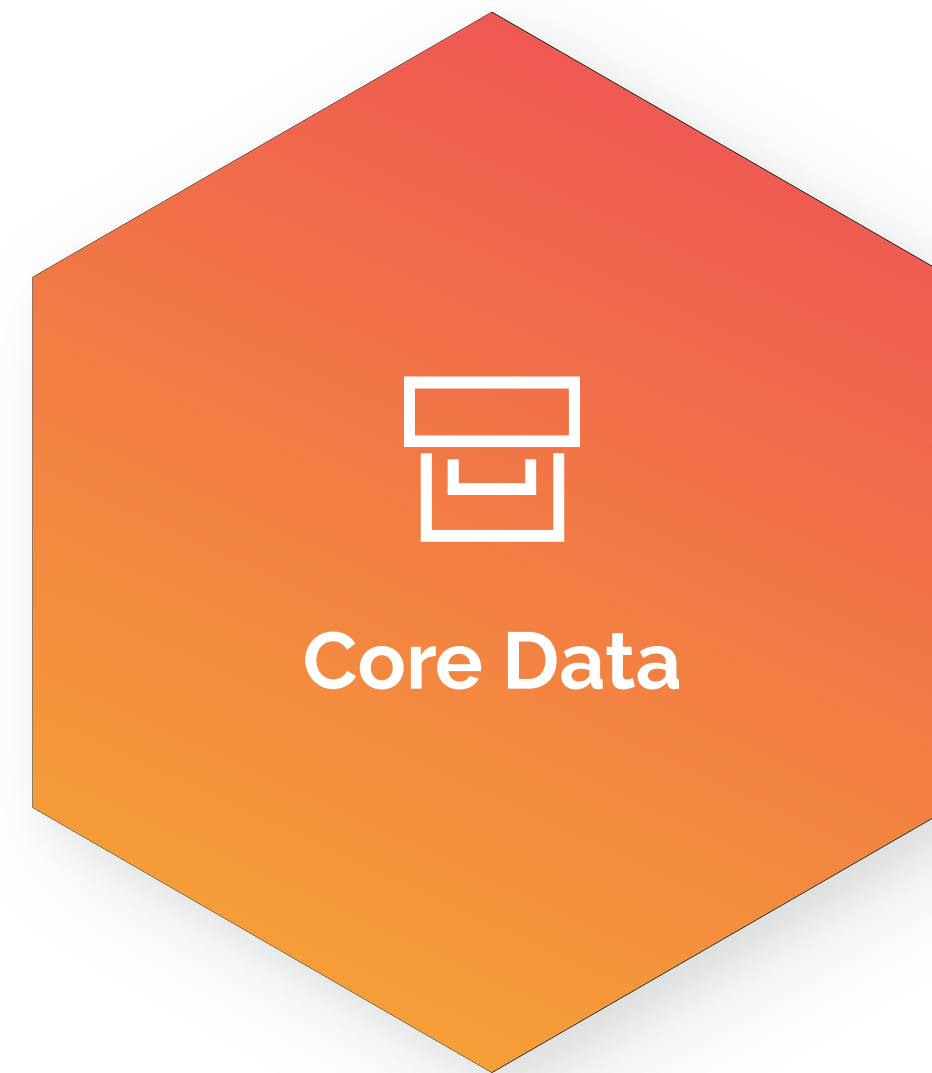




tecnologie utilizzate

MultiPeer Connectivity

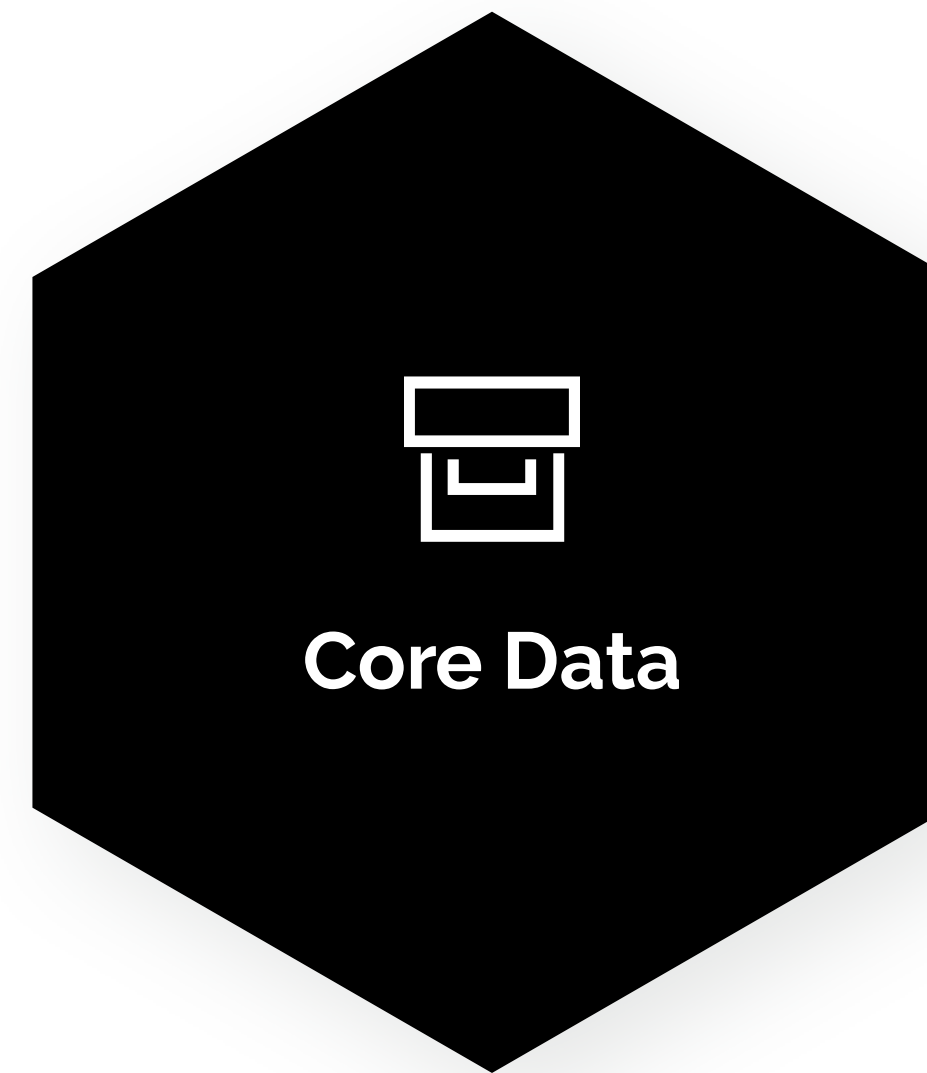
questa libreria supporta la scoperta di servizi forniti da dispositivi vicini e supporta la comunicazione con tali servizi attraverso dati basati su messaggi. In iOS, il framework utilizza reti Wi-Fi infrastrutturali, Wi-Fi peer-to-peer e reti personali Bluetooth per il trasporto sottostante.



tecnologie utilizzate

Core Data

questa libreria consente di salvare i dati permanenti di un'applicazione per l'utilizzo offline e di memorizzare nella cache i dati temporanei (database in locale).



tecnologie utilizzate

Facebook SDK

questa libreria consente di utilizzare tutte le principali SDK di Facebook per applicazioni iOS. In particolare in questo progetto sono state utilizzare le API per la condivisione di link e testo.

progettazione

Schermata Gioco

Il controller viene creato (con all'interno l'istanza della configurazione di gioco scelta, passata dalla view precedente) e crea un'istanza di tutte le librerie che utilizza.

L'handler crea tutti i modelli, inizializza il gioco e salva il loro stato al suo interno.

Il controller non interagisce mai direttamente con i modelli, ma sfrutta sempre gli handlers per applicare modifiche allo stato del sistema

Il controller prepara tutti gli assets, inizializza la sessione condivisa (se la modalità è multi-player) e chiama la funzione render().

Il controller riceve un outlet dalla view e viene chiamato di conseguenza il metodo corrispondente all'interno dell'handler. Una volta terminato questo, il controller richiamerà render() per aggiornare la grafica.

progettazione

Flusso Dati Esterno



1. Connessione

Un giocatore inizia il gioco avviando una sessione e rimane in attesa del numero di partecipanti richiesto (in questo caso due).

2. Inizializzazione

Il giocatore che ha cliccato sul pulsante d'inizio condividerà con l'altro lo stato iniziale del mazzo. Fatto ciò entrambi i giocatori creeranno due istanze di gioco identiche.



3. Gioca e Aspetta

Ogni giocatore a turno giocherà una carta e rimarrà in attesa che l'altro faccia lo stesso. Una volta terminato il gioco la sessione viene disconnessa.



sviluppo

Aspetti Rilevanti

Inizialmente, nello sviluppo, mi sono concentrato nel cercare di costruire una base solida e facilmente estendibile. Tutte le principali strutture dati sono rappresentate con la struttura dati Array. La caratteristica di questa struttura è il fatto che può contenere un numero variabile di elementi e questo chiaramente si sposava a pieno con l'idea, un giorno, di proseguire lo sviluppo aggiungendo anche le modalità a 3, 4 e 5 giocatori.

Come alternativa, avrei potuto usare la struttura dati 'Tuple', fissando il numero di elementi di ciascuna tupla (ovvero il numero di carte in mano), ma, si potrebbe utilizzare lo stesso gestore gioco a prescindere del numero di carte in mano a ciascun giocatore.

Il gestore del gioco opera a prescindere del numero di giocatori e del numero di carte in mano ad ognuno di essi.

conclusione

Difficoltà Incontrate

Come gestire tutto il flusso di dati?

Dove salvare l'istanza dei modelli di dati utilizzati? Negli handlers o nei controllers?

Come gestire la sincronizzazione della configurazione del gioco tra due device all'interno di una stessa sessione?

Che livello di astrazione creare tra i modelli e il loro utilizzatore ?





Nuovi Livelli di Difficoltà

Ovvero rendere configurabile la capacità dell'emulatore di giocare una carta.



Grafica e Animazioni

Aggiungere le animazioni, i constraint per tablet e landscape mode.



Accoppiamento Dispositivi

Dare la possibilità di scegliere il dispositivo con cui giocare (all'interno della sessione).



Grafica in AR

Disegnare le carte in AR (tramite ARKit)



Modalità 3-4-5 giocatori

Aggiungere anche le modalità con 3, 4 e 5 giocatori.



Sincronizzazione Database

Sincronizzare il database in locale con uno remoto in modo da non perdere i progressi

conclusione

Commenti Finali

Sicuramente si potrebbero apportare diversi miglioramenti all'emulatore, ma essendo già presente una classificazione real-time delle carte in gioco, questi non dovrebbero richiedere molto sforzo.

Riassumendo, posso affermare che l'aspetto principale su cui mi sono concentrato, durante tutto lo sviluppo, è stato quello della progettazione e della gestione del flusso di dati. Ho cercato di creare una code-base facilmente estendibile e molto modularizzata.



CONTI MATTEO

Laboratorio Applicazioni Mobile 2018/2019

BRISCOLA

Riproduzione del classico gioco di carte italiano per due giocatori.