

El Yousfi's & Ocone presentano



A blockchain project

Indice

Argomenti trattati

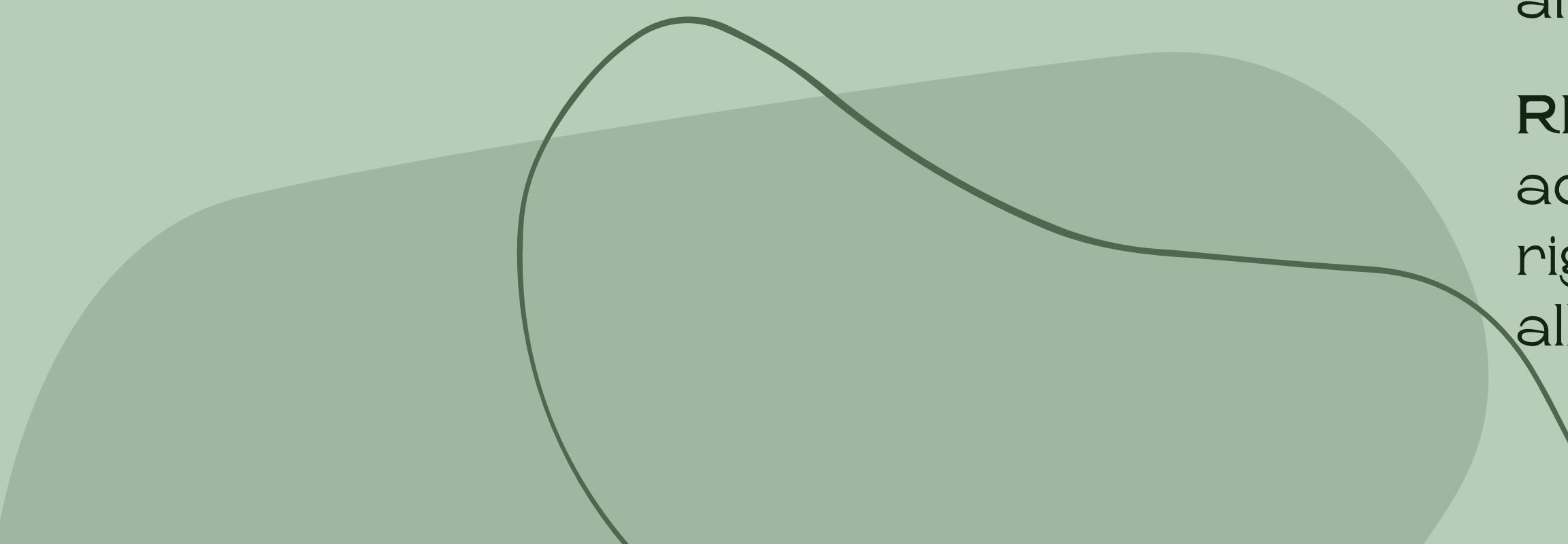
- Introduzione
- Requisiti
- Tecnologie usate
- Librerie utilizzate
- Applicazione
- ETHRoulette
- Smart Contract
- Problemi riscontrati
- Sviluppi futuri

Introduzione

Il progetto nasce dall' idea di replicare la gestione e il funzionamento dei principali casinò online utilizzando nuove tecnologie come la blockchain ed Ethereum.



Requisiti funzionali utente



RF1 - Un utente deve potersi registrare alla Piattaforma

RF2 - Un utente deve poter depositare e prelevare del saldo (Ether) dal proprio profilo.

RF3 - Un utente deve poter avviare un gioco a propria scelta, effettuare una puntata e attendere il responso di vincita.

RF4 - Un utente deve poter accedere alle sue informazioni riguardanti le giocate effettuate, alle vincite e quant' altro.

SCI: Registrazione alla DAPP

ATTORE	SISTEMA
Vuole registrarsi al sito	
	Mostra un Form con i relativi campi da compilare
L'Attore compila i form e lo sottomette	
	Verifica la correttezza dei dati sottomessi e in caso di esito positivo viene reindirizzato alla pagina di profilo
L'utente verifica che la corretta registrazione alla DAPP	

SC2: Deposito di ETH

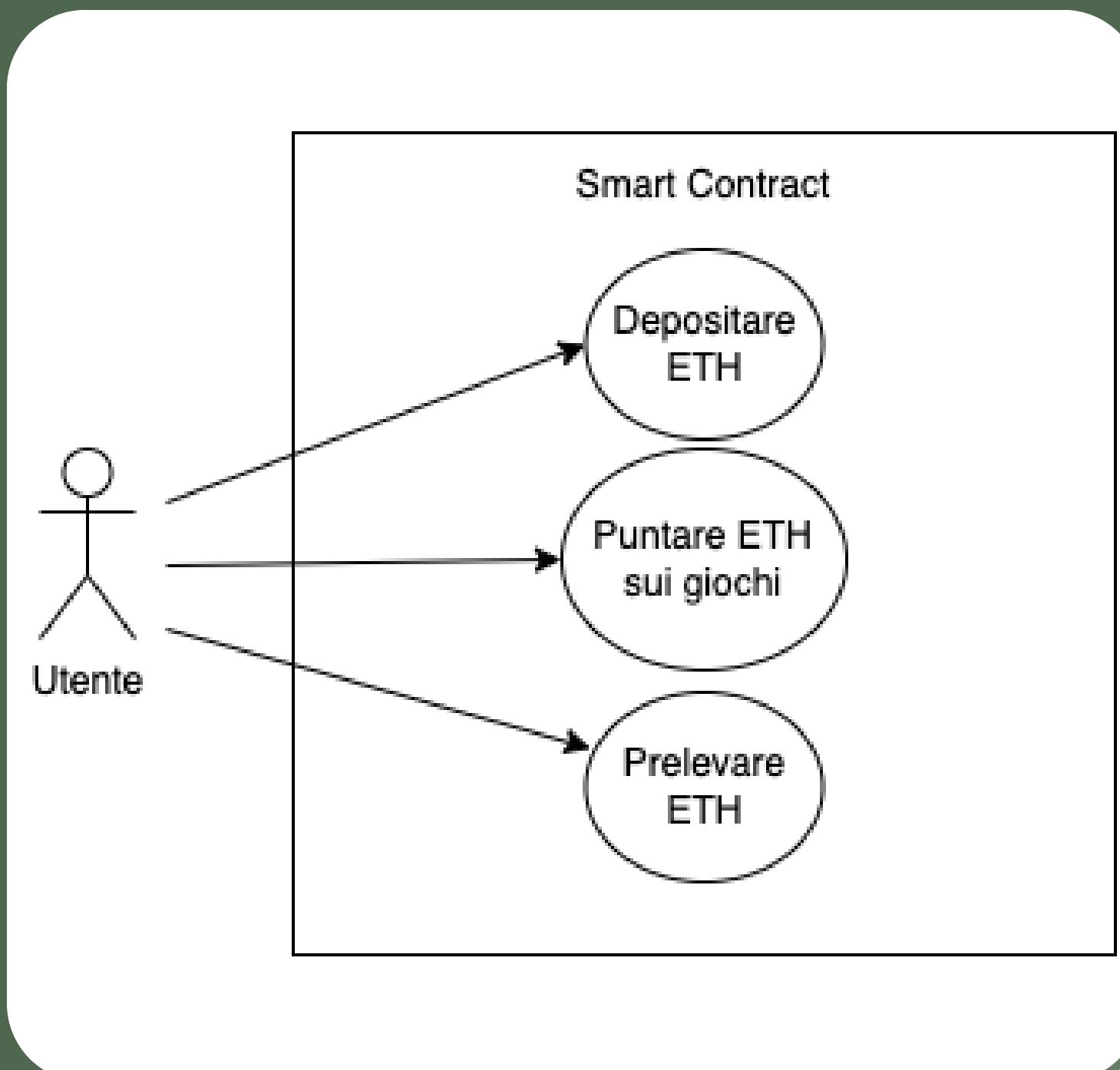
ATTORE	SISTEMA
Vuole depositare saldo nella piattaforma	
	Mostra un form per la somma che si desidera depositare
L'Attore compila i form e lo sottomette la somma	
	Verifica la correttezza dei dati sottomessi e in caso di esito positivo viene aggiornato il campo del saldo
L'utente verifica che il corretto deposito	

SC3: Puntata al gioco Roulette

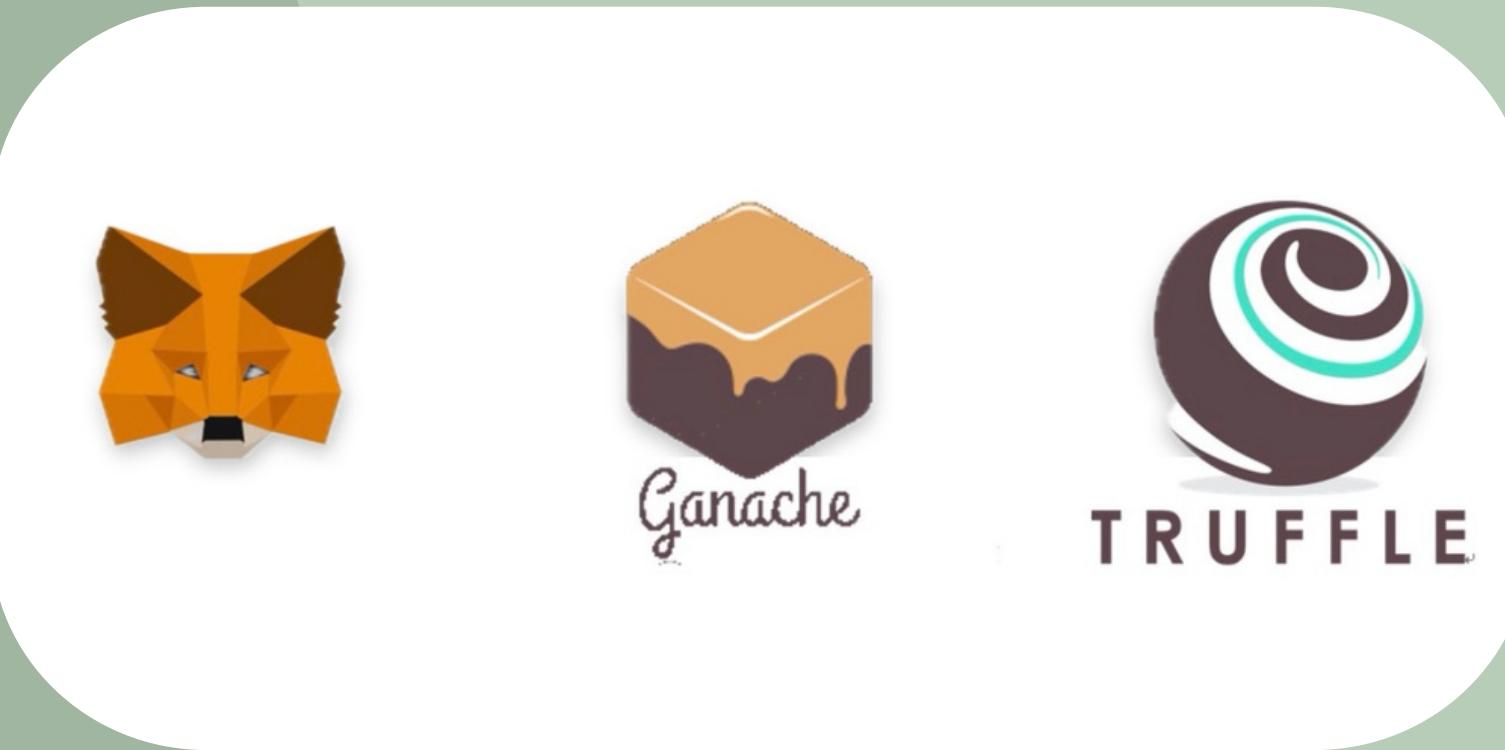
ATTORE	SISTEMA
L'utente vuole piazzare una giocata	
	Mostra le possibili combinazioni di bet
L'Attore in maniera dinamica punta le fiches e inizia la giocata con spin	
	Il sistema mostra la ruota girare e in caso di vincita mostra un pop-up con l'esito e la vincita
L'utente verifica l'esito della giocata	

Use Case Diagram

Mostriamo tramite questa illustrazione le funzionalità che gli utenti possono fare all'interno della DAPP.



Technologie utilizzate



Metamask: funge da wallet Ethereum e ci consente di interagire con applicazioni decentralizzate o dapps.

Ganache: è una blockchain personale per lo sviluppo rapido di applicazioni distribuite Ethereum.

Truffle: permette di creare, compilare, distribuire e testare smart contract



Solidity: linguaggio di programmazione ad alto livello, orientato a oggetti, per lo sviluppo di smart contract su varie blockchain, in primis Ethereum.

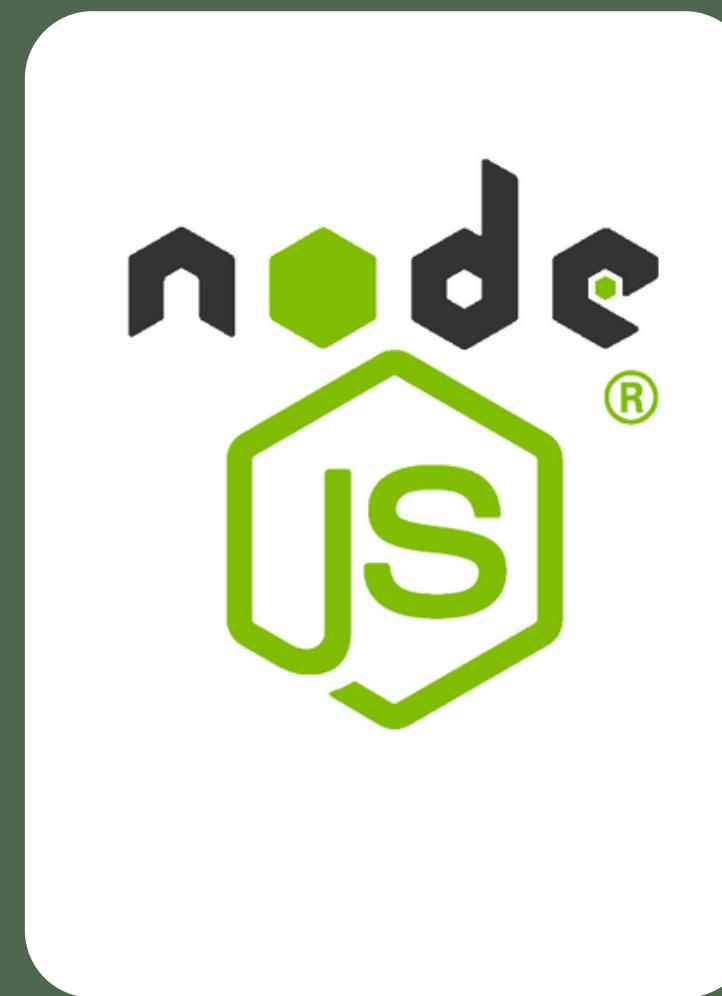
GitHub: lo sviluppo del software ed il suo versioning sono stati supportati dalla piattaforma GitHub per agevolare la collaborazione tra gli sviluppatori.

Librerie utilizzate



Web3.js

Interagisce con la blockchain di Ethereum. Può recuperare account utente, inviare transazioni, interagire con contratti intelligenti e altro ancora.



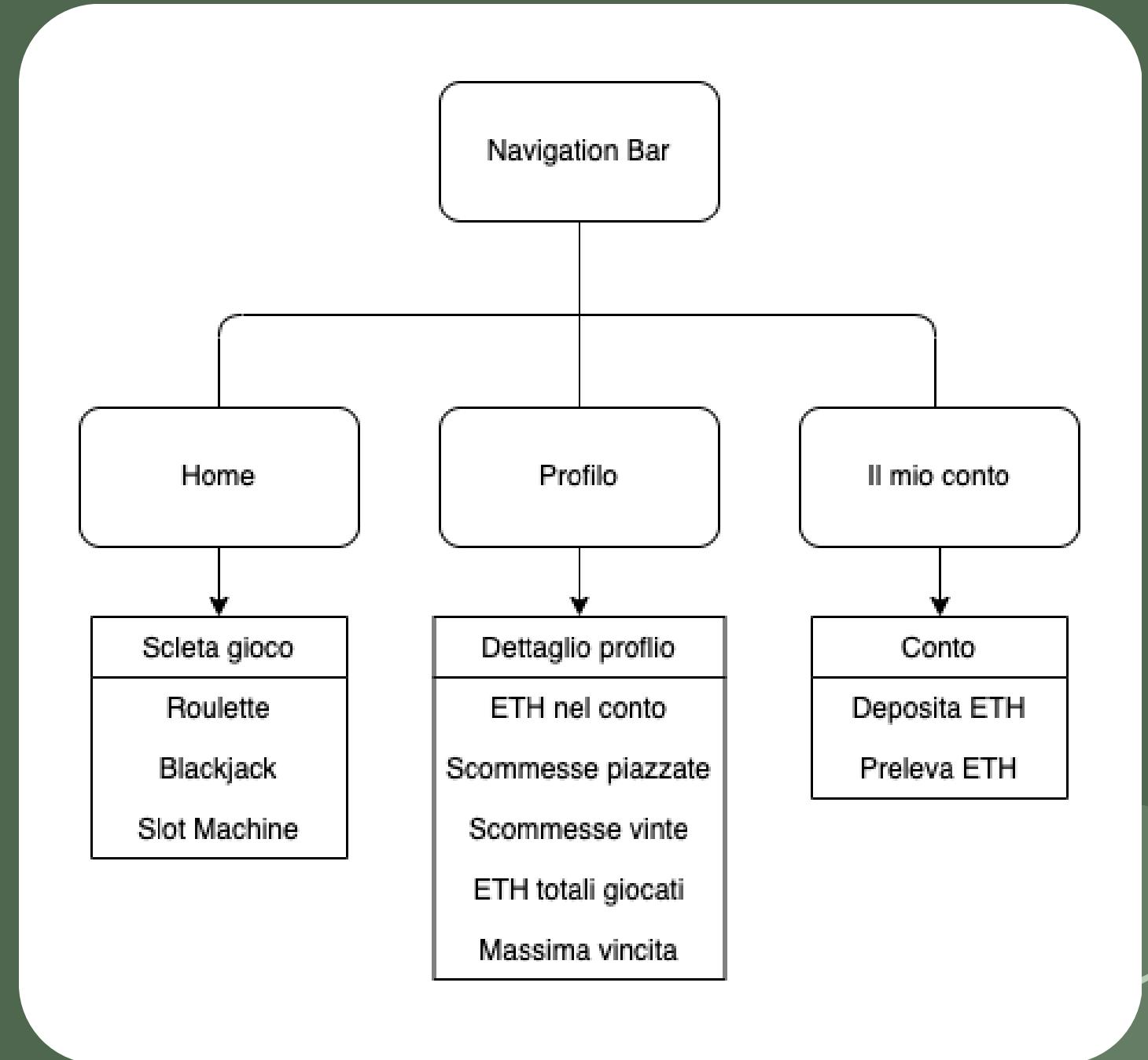
Node.js

È un runtime JavaScript guidato da eventi asincroni costruito sul motore JavaScript V8 di Chrome, e viene usato sia per lo sviluppo front-end che back-end.



Keccak256

È un membro della famiglia di funzioni hash SHA-3, è stata progettata per essere sicura contro un'ampia gamma di attacchi ed è anche efficiente, con un costo computazionale relativamente basso.



La nostra applicazione

Tramite un navigation diagram è stato mostrato come un possibile utente può muoversi all'interno dell'applicazione.

ETHRoulette

Classico gioco della roulette, dove un utente piazza una scommessa in Ether, aspetta il risultato prodotto dal gioco, e scopre se la sua scommessa è vincete o perdente. In caso di vittoria incrementa il suo saldo gioco.





Smart Contract ETHRoulette

Ci sono due principali funzioni all'interno dello smart contract ETHRoulette ovvero:

- **Bet** -> ovvero una funzione che salva all'interno della blockchain la giocata dell'utente con il corrispettivo valore in Ether piazzato.
- **SpinWheel** -> ovvero una funzione che principalmente calcola un valore 'pseudocasuale' che sarà poi il risultato che apparirà sulla roulette e successivamente controlla i bet dell'utente e applica la logica per verificare quali scommesse sono vincenti e quali perdenti e aggiorna le statistiche utente aumentando ovviamente il saldo utente in caso di vincita.



Problema generazione numero pseudocasuale

La vera casualità non è possibile nella Ethereum Virtual Machine (EVM), quindi abbiamo dovuto utilizzare lo stato attuale della blockchain per trovare un numero casuale equo, calcolando l'hash di diversi fattori e utilizzare il risultato ottenuto modulo 37 come risultato della pseudocasualità della roulette. Come fattori di calcolo abbiamo utilizzato:

- Il blockhash del blocco precedente;
- Il timestamp del blocco corrente;
- L'attuale difficoltà del blocco;
- L'ultima scommessa accettata.

Ovviamente il problema con questo approccio è che chiunque possa vedere lo stato della blockchain, può calcolare ("indovinare") il futuro numero casuale e scommettere su quel numero.

Analisi Gas Fee

	ETH	EURO
REGISTRAZIONE	0.00015459	0.50
PRELIEVO	0.0001003	0.35
DEPOSITO	0.00029474	1.03
BET	0.0003406	1.19
SPINWHEEL	0.00021351	0.75

Come funziona?

[Go to BlockBet](#)

Sviluppi futuri

L'idea della generazione del numero pseudocasuale, per come è stata gestita nel nostro progetto non è delle migliori poichè qualcuno potrebbe venire a conoscenza del prossimo numero generato, puntare e vincere sempre alla roulette facendoci andare completamente in bancarotta allora abbiamo pensato a due principali metodi per risolvere questa vulnerabilità

Soluzione: Oracle RNG Chainlink VRF

Abbiamo considerato varie possibili soluzioni ma questa è quella che riteniamo più valida per lo scopo del nostro progetto.



Chainlink VRF è un generatore di numeri casuale, che per ogni richiesta genera uno o più numeri casuali e una prova crittografica di come tali valori sono stati determinati così da non compromettere la **sicurezza** e l'**usabilità**.

Sviluppi futuri

Aggiunta di nuovi giochi



Grazie!

Prova il nostro progetto!
<https://github.com/gli-sfollati/ETHRoulette>

A. El Yousfi
M. Ocone