GambleBoard

Ronin Chellakudam, Lauri Tahvanainen, Md Rezuanul Haque and Martin Frick

Department of Informatics IfI University of Zurich UZH
Binzmühlestrasse 14, CH---8050 Zürich, Switzerland
{ronin.chellakudam, lauri.tahvanainen, mdrezuanul.haque, martin.frick}@uzh.ch

Abstract. We present GambleBoard, a decentralized betting platform for back and lay bets. GambleBoard provides its users with the ability to create and place bets on virtually any types of events without any trusted third parties. This is accomplished by using different blockchain technologies and services; Ethereum smart contracts, IPFS (Interplanetary File System) for decentralized storage, The Graph protocol for indexing blockchain data and Kleros for dispute resolution. The final implementation accomplishes its objectives as a working proof-of-concept in a local environment.

1 Introduction

Our goal was to create a platform (DApp) where players can bet against each other without a centralized provider demanding a commission in between. The type of bets we offer are "back"- and "lay"-bets (livetipsportal, 2021). The players have the possibility to back or to lay a bet which allows them to either bet for or against a certain outcome and to offer a bet to other players or to accept a bet another player has offered.

The bet can be almost on any type of event. A champions league football match, a beer league ice hockey match, a political event or even a funny bet with a friend about being able to throw a ping pong ball in to a glass while blind. The way we can achieve having bets of so many types is through having the default situation be that the players agree on the outcome of the event. After a bet has been laid, the players stake is frozen until the bet is over or removed. A user who lays a bet also must decide on some options. These include: the outcomes, the odds, the stake on the outcome and a description of the bet. By deciding the odds of the outcomes and betting on one of these outcomes the layer also fixes the possible stake on the other outcome.

adfa, p. 1, 2011.

2 Solution/DApp Design

A bet pool is created by an interaction with a smart contract from now on referenced as “the GambleBoard contract”. In a simple 1v1 pool the creator configures the pool and adds stake and waits for someone to agree on the odds and provide their stake. The stakes would then be locked in the pool for a certain time or until both parties vote on the result. When the outcome is known, both parties interact with the the GambleBoard contract to broadcast their opinion on the result. If these opinions match the rewards are delegated to the winner accordingly. If there is a dispute on the result, the player can send the dispute to Kleros court for arbitration. The jury will make a decision and the GambleBoard contract will react to the decision (Kleros, 2021).

In our project, we were guided by some of the biggest centralized betting platforms and their type of bets they offer. Or goal was to create a similar platform but decentralized and with any kinds of bets. As this was a learning project, we hoped to learn how to create our own DApp and to get familiar with Solidity as well as with the tools needed like Ganache, MetaMask and many more.

For our DApp, it was crucial to implement a functioning smart contract, as it forms the basis of the project and allows the players to interact with each other. It was also highly relevant to integrate the Kleros mechanism to have a decision if there is a disagreement about the outcome of the bet. Furthermore, we wanted to have a user interface that appeals to the users and is easy to use. To determine success, we ran tests on both the backend and frontend to make sure the DApp was working properly (Kleros, 2021).

2.1 Use cases

A user can either create a new bet on the platform or accept a bet created by another user (e.g. place a bet). Thus, the user can view the list of all bets created and filter for his own bets or for specific bets he wants to back. All bets are placed in Ether. When the deadline is reached, the user can vote on the outcome of the bet. If there is agreement on the outcome, the winner of the bet can claim their winnings. In the event of a disagreement, the user has the option of triggering the Kleros mechanism. In the rare case that there is no result, e.g. if a match is cancelled, the user can claim a refund.

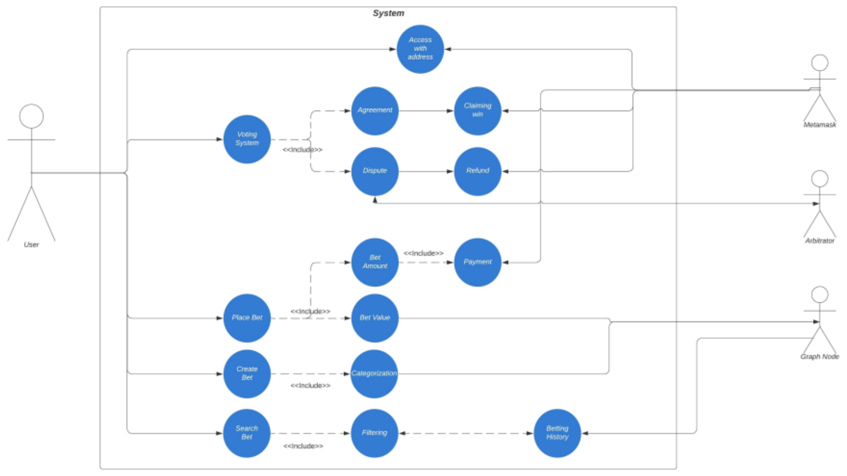


Figure 1: Use Case Diagram

2.2 Implementation choices

Our core functionality was implemented as an Ethereum smart contract. The contract was written in Solidity (Soliditylang.org, 2021). Our use case needs to query bet related structures from the contract that are not explicitly present. This is the case for example in an event, that groups bets with similar attributes under itself. To efficiently query the smart contract data, we integrated the GambleBoard contract with The Graph Protocol (The Graph, 2021), that makes it possible to easily map smart contract events to a decentralized database and query this database. Apollo client, a javascript GraphQL state management library, was used in the front-end to structure the queries to the Graph Node (Apollo Graph Inc, 2021).

Since bets are resolved by the players voting on the outcome of the event, there needs to be a mechanism for resolving possible disputes about the outcome. Our smart contract implements the ERC 792 Arbitration (Lesaege, 2017) and ERC 1497 Evidence (Vitello, Lesaege, & Piqueras, 2018) standards the GambleBoard contract being an arbitrable contract. This way we can use the services of any ERC 792 compatible Arbitrator contract to resolve the bets that have a disagreement about the outcome. Out implementation uses Kleros (Kleros, 2021). IPFS, a decentralized file storage system was used for saving user submitted evidence about disputed bets (Protocol Labs, 2021).

The front-end application was implemented using React.js for quick front-end development (Facebook, 2021), web3.js for interacting with the smart contracts (Web3 Foundation, 2016), MetaMask for integrating with user accounts (MetaMask, 2021), archon for producing and evaluating evidence hashes (Kleros, 2018).

The testing environment consisted of a local Ganache (Truffle Suite, 2021) blockchain, a local IPFS node, a local Graph Protocol node and the front-end running as a local web server. Everything was developed in the local environment for speed of development, and because the requirements did not demand for the application to be deployed to a main net blockchain.

2.3 Software Architecture

Figure 2 shows all the components of our implementation. A user interacts with the front end, creating smart contract transactions and querying data about the smart contract. If there is an agreement about the bet between the players, the transactions only involve the users, the GambleBoard contract and IPFS.

IPFS is used to store evidence and metaevidence about a bet. Metaevidence is saved when the bet is created, evidence only when a player wants to submit evidence to a disputed bet. Metaevidence is evidence that contains information about the bet which the arbitrator needs to rule on the dispute. Evidence in our system is a text input given by the user, that contains freeform arguments about why the dispute should be ruled to the advantage of the player.

If there is a disagreement one player can raise a dispute and send the bet to Arbitration. Kleros is a service that implements an Arbitrator interface. It is another smart contract running on the Ethereum blockchain and thus when raising a dispute, the GambleBoard contract will call the arbitrator contract. The arbitrator contract application will search for the GambleBoard contract's events for an IPFS link to metaevidence and evidence about the dispute. In case of Kleros the evidence is presented to random jurors, who will decide on the outcome of the dispute. A resolution of a dispute is done by a function call from the Arbitrator contract to the GambleBoard contract. This will change the bets state in the GambleBoard contract to agreement. Now the player(s) can withdraw their funds depending on the outcome (Kleros, 2018).

The querying of smart contract data is done completely from the Graph Node, which works by listening to events raised in the GambleBoard contract and then executing a predefined function as a reaction to a specific event. With these mapping functions we fill the Graph Nodes database according to a predefined schema. Code sample 1 shows a part of the schema and an event entity. The front-end queries the node with GraphQL queries to fill the front-end with data.

For example, the event entity groups bets with the same description, country, category and date. This way we can find the bets under this event with a simple query show in code sample 2. If this querying would be implemented on the smart contract, we would have to go through all the selected events of the contract to find the information. Another option would also be to set up a centralized custom database.

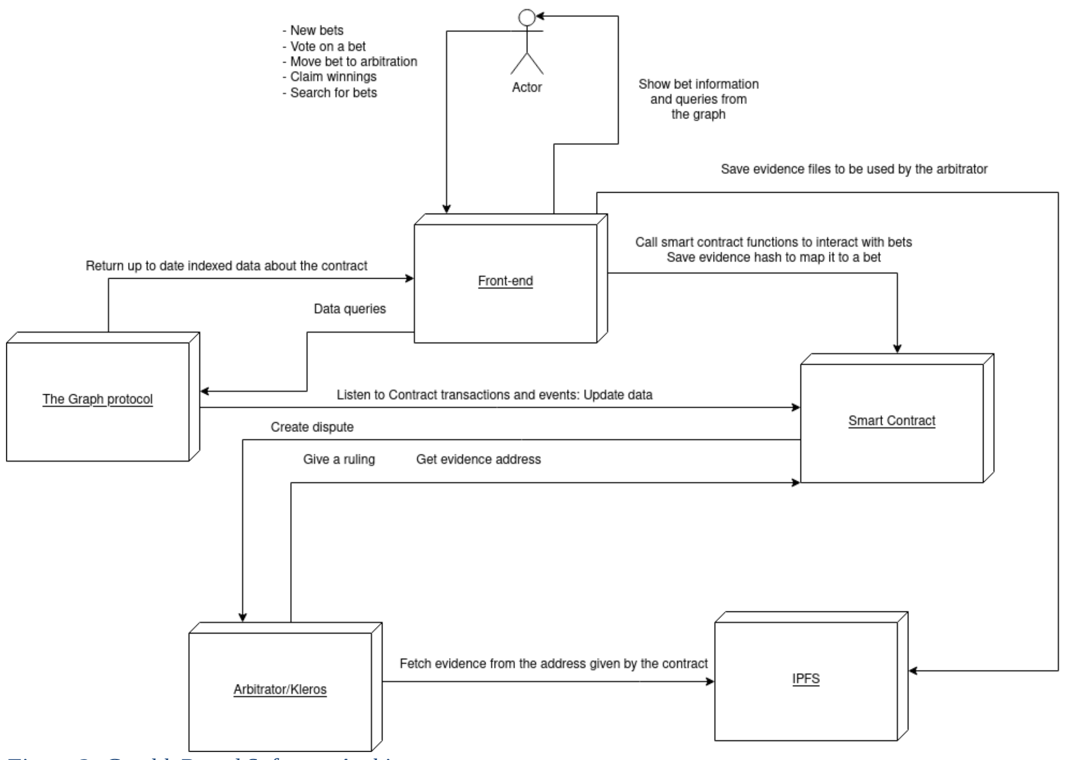

- New bets
- Vote on a bet
- Move bet to arbitration
- Claim winnings
- Search for bets
Actor
Show bet information and queries from the graph
Save evidence files to be used by the arbitrator
Call smart contract functions to interact with bets
Save evidence hash to map it to a bet
Return up to date indexed data about the contract
Front-end
Data queries
The Graph protocol
Listen to Contract transactions and events: Update data
Smart Contract
Create dispute
Give a ruling
Get evidence address
Arbitrator/Kleros
Fetch evidence from the address given by the contract
IPFS

Figure 2: GambleBoard Software Architecture

2.4 Program Code

Back-end

The functions below represent the interface of the GambleBoard contract and how it interacts with the arbitrator. The functions map almost one to one to users interactions on the front-end.

A bet is saved as struct as shown below. Each bet has an unique ID:

```
Struct Bet {
	uint256 stakingDeadline;
	uint256 votingDeadline;
	uint256 backerStake;
	uint256 creatorStake;
	RulingOption outcome;
	State state;
	bytes1 voteEvidenceBools;
	address payable creator;
	address payable backer;
```

```
        string description;
        string creatorBetDescription;
    }
```

Players can interact with the GambleBoard contract using the following interfaces:

CreateBet(description, creatorBetDescription, league, country, categroy, staking-Deadline, timeToVote, creatorOdd, metaEvidence)
-A player creates a bet, giving all the necessary information about the bet. The bet is then mapped to an id when created with all the necessary information.
-Calculates the amount a backer has to stake from the creator's stake and odd.

PlaceBet(betID)
-A player backs a bet, giving the betID as parameter.

VoteOnOutcome(betID, outcome)
-A player interacts with the contract to give their suggestion on the outcome of the bet
-If all have given their suggestion the following happens:
a: All players agree and the winnings can be claimed by the winner
b: If Players do not agree, bet can be sent to Kleros

Refund(betID)
-If no players voted in time or if the votes were on NO_OUTCOME, the stakes can be refunded by triggering this function.

ClaimWinnings(betID)
-If only one player voted within the time to vote, the winner of the bet will be chosen based on the one voting.
-Player who won the bet can claim winning but can also be called by loser.

CreateDispute(betID, fee)
-A player creates a dispute about a bet.
-Calls the arbitrator contract to create the dispute.
-DisputeID from the arbitrator is mapped to BetID in the GambleBoard contract.

ProvideEvidence(betID, evidence)
-A player submits evidence on the dispute of a bet. This is done by emitting an evidence event as by the ERC 1497 standard.

ExecuteRuling(disputeID, ruling) external
-Is called by the arbitrator to give a ruling on the bet.

GraphQL Schema and query

Code sample 1

```
type Event @entity {
  id: ID! # set to EventDescription-Country-Category-
  description: String!
  startTime: BigInt!
  league: String!
  country: Int!
  category: Int!
  bets: [Bet!]! @derivedFrom(field: "event")
}
```

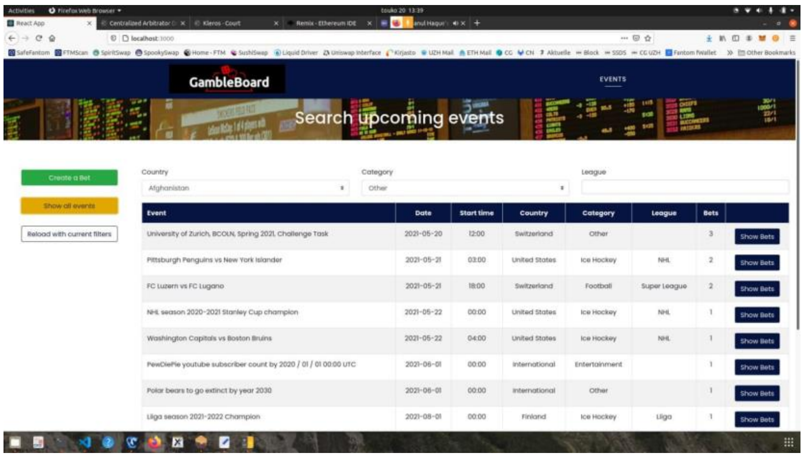
Code sample 2

```
query getEvents($country: Int, $category: Int, $league: String) {

    events(where: {country: $country, category: $category, league_contains: $league}, orderBy: startTime) {
      id
      bets {
        id
      }
    }
}
```

Front-End / Graphical User Interface

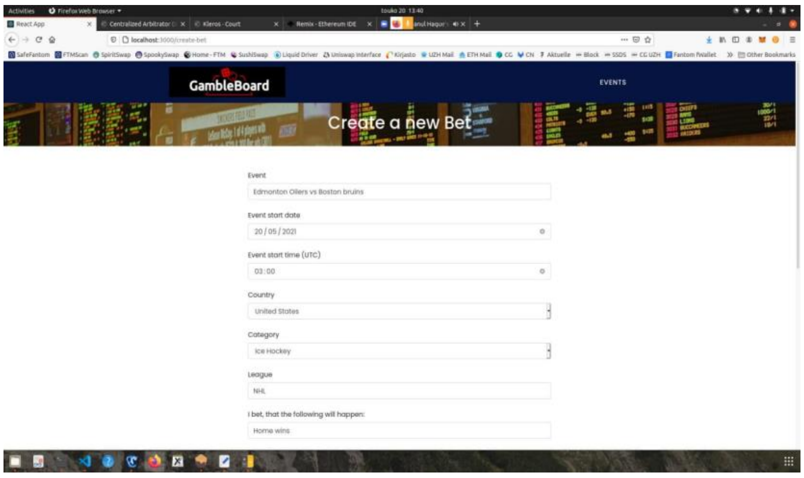
A user can check and search events based on country, category and league.



Country: Afghanistan
Category: Other
League:

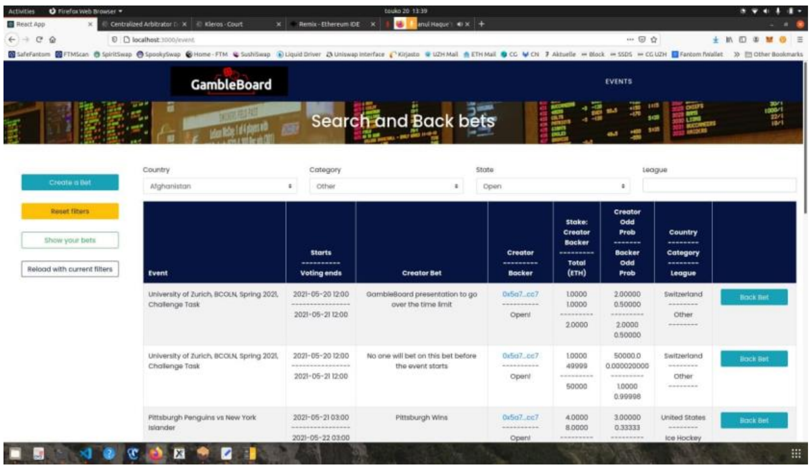
Event	Date	Start time	Country	Category	League	Bets	
University of Zurich, BCOLN, Spring 2021, Challenge Task	2021-05-20	12:00	Switzerland	Other		3	Show Bets
Pittsburgh Penguins vs New York Islander	2021-05-21	03:00	United States	Ice Hockey	NHL	2	Show Bets
FC Luzern vs FC Lugano	2021-05-21	18:00	Switzerland	Football	Super League	2	Show Bets
NHL season 2020-2021 Stanley Cup champion	2021-05-22	00:00	United States	Ice Hockey	NHL	1	Show Bets
Washington Capitals vs Boston Bruins	2021-05-22	04:00	United States	Ice Hockey	NHL	1	Show Bets
PewDiePie youtube subscriber count by 2020 / 01 / 01 00:00 UTC	2021-06-01	00:00	International	Entertainment		1	Show Bets
Polar bears to go extinct by year 2030	2021-06-01	00:00	International	Other		1	Show Bets
Liiga season 2021-2022 Champion	2021-08-01	00:00	Finland	Ice Hockey	Liiga	1	Show Bets

A user can create a new bet:



Event	Edmonton Oilers vs Boston bruins
Event start date	20 / 05 / 2021
Event start time (UTC)	03 : 00
Country	United States
Category	Ice Hockey
League	NHL
I bet, that the following will happen:	Home wins

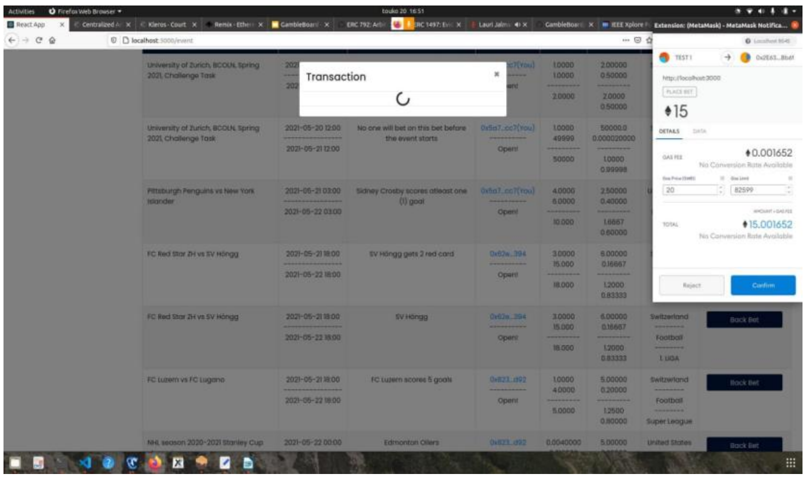
As shown below, other users can back the bet.



Country	Afghanistan
Category	Other
State	Open
League	

Event	Starts ---------- Voting ends	Creator Bet	Creator ---------- Backer	Stake: Creator Backer ---------- Total (ETH)	Creator Odd Prob -------- Backer Odd Prob	Country -------- Category -------- League	
University of Zurich, BCOLN, Spring 2021, Challenge Task	2021-05-20 12:00 ---------------- 2021-05-21 12:00	GambleBoard presentation to go over the time limit	0x5a7...cc7 ---------- Open!	1.0000 1.0000 --------- 2.0000	2.00000 0.50000 --------- 2.0000 0.50000	Switzerland --------- Other ---------	Back Bet
University of Zurich, BCOLN, Spring 2021, Challenge Task	2021-05-20 12:00 ---------------- 2021-05-21 12:00	No one will bet on this bet before the event starts	0x5a7...cc7 ---------- Open!	1.0000 49999 --------- 50000	50000.0 0.000020000 --------- 1.0000 0.99998	Switzerland --------- Other ---------	Back Bet
Pittsburgh Penguins vs New York Islander	2021-05-21 03:00 ---------------- 2021-05-22 03:00	Pittsburgh Wins	0x5a7...cc7 ---------- Open!	4.0000 8.0000 ---------	3.00000 0.33333 ---------	United States --------- Ice Hockey	Back Bet

When placing a bet, MetaMask pops up and asks to confirm the transaction.


Transaction

Extension: (MetaMask) - MetaMask Notification

TEST 1 → 0x2E63...Bb6f

http://localhost:3000

PLACE BET

♦15

DETAILS | DATA

GAS FEE ♦0.001652
No Conversion Rate Available

Gas Price (GWEI): 20 | Gas Limit: 82599

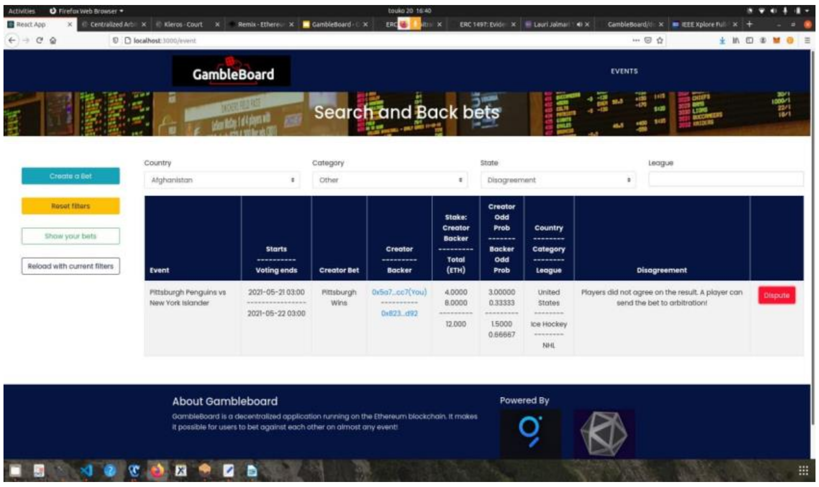
AMOUNT + GAS FEE

TOTAL ♦15.001652
No Conversion Rate Available

Reject | Confirm

University of Zurich, BCOLN, Spring 2021, Challenge Task	202 ------- 202		c7(You) ------- ent	1.0000 1.0000 ------- 2.0000	2.00000 0.50000 ------- 2.0000 0.50000		
University of Zurich, BCOLN, Spring 2021, Challenge Task	2021-05-20 12:00 ------- 2021-05-21 12:00	No one will bet on this bet before the event starts	0x5a7...cc7(You) ------- Open!	1.0000 49999 ------- 50000	50000.0 0.000020000 ------- 1.0000 0.99998		
Pittsburgh Penguins vs New York Islander	2021-05-21 03:00 ------- 2021-05-22 03:00	Sidney Crosby scores atleast one (1) goal	0x5a7...cc7(You) ------- Open!	4.0000 6.0000 ------- 10.000	2.50000 0.40000 ------- 1.6667 0.60000		
FC Red Star ZH vs SV Höngg	2021-05-21 18:00 ------- 2021-05-22 18:00	SV Höngg gets 2 red card	0x62e...394 ------- Open!	3.0000 15.000 ------- 18.000	6.00000 0.16667 ------- 1.2000 0.83333		
FC Red Star ZH vs SV Höngg	2021-05-21 18:00 ------- 2021-05-22 18:00	SV Höngg	0x62e...394 ------- Open!	3.0000 15.000 ------- 18.000	6.00000 0.16667 ------- 1.2000 0.83333	Switzerland ------- Football ------- 1. LIGA	Back Bet
FC Luzern vs FC Lugano	2021-05-21 18:00 ------- 2021-05-22 18:00	FC Luzern scores 5 goals	0x823...d92 ------- Open!	1.0000 4.0000 ------- 5.0000	5.00000 0.20000 ------- 1.2500 0.80000	Switzerland ------- Football ------- Super League	Back Bet
NHL season 2020-2021 Stanley Cup	2021-05-22 00:00	Edmonton Oilers	0x823...d92	0.0040000	5.00000	United States	Back Bet

In case of disagreement, the user can create a dispute. The Kleros court will then take on the dispute.



Country: Afghanistan | Category: Other | State: Disagreement | League:

Event	Starts ------- Voting ends	Creator Bet	Creator ------- Backer	Stake: Creator Backer ------- Total (ETH)	Creator Odd Prob ------- Backer Odd Prob	Country ------- Category ------- League	Disagreement	
Pittsburgh Penguins vs New York Islander	2021-05-21 03:00 ------- 2021-05-22 03:00	Pittsburgh Wins	0x5a7...cc7(You) ------- 0x823...d92	4.0000 8.0000 ------- 12.000	3.00000 0.33333 ------- 1.5000 0.66667	United States ------- Ice Hockey ------- NHL	Players did not agree on the result. A player can send the bet to arbitration!	Dispute

About Gambleboard

GambleBoard is a decentralized application running on the Ethereum blockchain. It makes it possible for users to bet against each other on almost any event!

Powered By

3 Conclusions/Critique

The final product of the GambleBoard DApp provides a good variety of basic functionalities described in the original design of the project in a 1 vs 1 Back-lay system. In addition, we were able to implement one of the optional requirements, which we designed to have a more complete platform, which is the ability for users to resolve bet disputes using Kleros. Due to the lack of time, we were not able to implement some of the optional requirements that we set up to have a final product such as the introduction of private bets accessible through invitation and pari mutual bets.

During the project's realization, we had many meetings to update the whole team about the work progress and issues. The documentation created in some of these sessions helped us greatly to stay on track. Our project was created and tested only in a local environment on a private blockchain, as a result, we can consider it as a proof-of-concept implementation. Furthermore, we would have liked to test it also in a Testnet to learn the actual performance of our project and optimize it to the best of our efforts.

4 Future Work

This project was a great learning opportunity for us, which we used to learn Solidity and all the other necessary tools to run a DApp. We also used the opportunity to familiarise ourselves with Kleros and the Graph and how we can integrate them into our project.

For the future, we might consider continuing our project and adding more features. There are several ways we could improve our DApp. For example, we could allow the user to choose an oracle instead of Kleros to handle disagreements. So, when laying/backing a bet, the user could add an oracle which will decide on the outcome. Also, we could let the users choose between private and public bets.

Another functionality that could be added is a rating system. The idea behind this is that honest users get a better rating, while dishonest users get a worse rating. With this mechanism we could reduce the intention of users to cheat.

Finally, we could offer pari mutual betting in addition to back and lay betting. In pari-mutual betting, all bets of a certain type are placed together in a pool, while the odds are calculated by dividing the pool among all winning bets. This would allow users to participate not only in p2p betting but also in many-to-many betting.

5 Figures

6 References

Literaturverzeichnis

Apollo Graph Inc. (20. May 2021). *Introduction to Apollo Client*. Von Apollo Graph-Website: https://www.apollographql.com/docs/react/ abgerufen

Facebook. (20. May 2021). *React*. Von React-Website: https://reactjs.org/ abgerufen

Kleros. (20. May 2018). *archon - Ethereum Arbitration Standard API*. Von Kleros-Website: https://archon.readthedocs.io/en/latest/index.html abgerufen

Kleros. (20. May 2021). *The justice Protocol*. Von Kleros-Website: https://kleros.io abgerufen

Lesaege, C. (6. December 2017). *ERC 792: Arbitration Standard* . Von Github-Website: https://github.com/ethereum/EIPs/issues/792 abgerufen

livetipsportal. (19. March 2021). *livetipsportal*. Von livetipsportal-Website: https://www.livetipsportal.com/en/betting-strategies/back-lay/ abgerufen

MetaMask. (2021. May 2021). *A crypto wallet & gateway to blockchain apps*. Von MetaMask-Website: https://metamask.io/ abgerufen

Protocol Labs. (20. May 2021). *IPFS powers the Distributed Web*. Von IPFS Website: https://ipfs.io/ abgerufen

Soliditylang.org. (20. May 2021). *Solidity Documentation*. Von Solidity-Website: https://docs.soliditylang.org/en/v0.8.0/ abgerufen

The Graph. (20. May 2021). *APIS for a vibrant decentralized future*. Von Graph-Website: https://thegraph.com abgerufen

Truffle Suite. (20. May 2021). *Ganache*. Von Truffle Suite-Website: https://www.trufflesuite.com/ganache abgerufen

Vitello, S., Lesaege, C., & Piqueras, E. (16. October 2018). *ERC 1497: Evidence Standard*. Von Github-Website: https://github.com/ethereum/EIPs/issues/1497 abgerufen

Web3 Foundation. (April. 30 2016). *web3.js - Ethereum JavaScript API*. Von web3-Website: https://web3js.readthedocs.io/en/v1.3.4/index.html abgerufen