P학기 결과보고서

- 블록커스 팀 -

부제 : 블록체인을 이용한 게임 재화 관리

201433813 신승빈

201635903 강병찬

201635908 곽준수

201636002 오경민

201636043 이창연

201835664 배연주

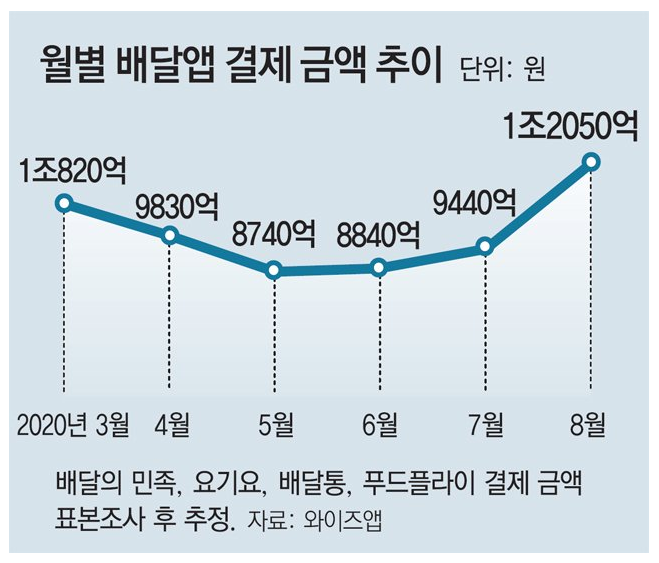
목차

1. 개요
   1. 프로젝트의 필요성
   2. 재화 버그의 원인
   3. 왜 블록체인인가?
2. 본문
   1. 구현 방법
   2. Usecase
      1. Update
      2. Buy
      3. Send
   3. 상위 설계
      1. 스마트 컨트랙트
      2. 통신 모듈
      3. 게임 모듈
   4. 하위 설계
      1. 스마트 컨트랙트
         1. Struct Player
         2. Player[] players
         3. Int[3] item\_price
         4. GetFunctions
         5. CreatePlayer
         6. Update
         7. Send
         8. Buy
      2. CommunicationModule
         1. Attribute
         2. Method
      3. Properties
         1. Attribute
         2. Method
3. 결과
   1. 기대효과

1. 개요

코로나 시대에 접어든지도 어언 1년이 되어가는 시점이다. 사람들에게는 이제 untact가 익숙해졌고, 이제는 untact를 넘어 ontact의 시대가 다가오고 있다. 뉴스에서는 코로나 바이러스의 백신에 대한 이야기가 점점 나오고있지만 백신이 출시되고 정상적인 상황으로 돌아간다 해도 그 모습이 코로나 이전의 시대와 동일할거라고 예상하는 사람은 없을 것이다.

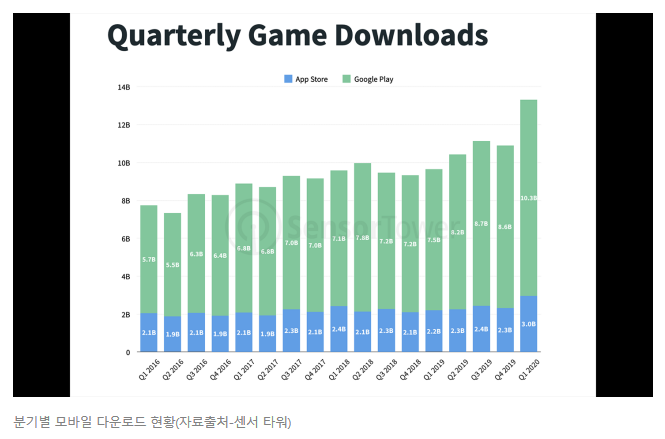
untact시대에 맞춰 많은 부분들이 변해갔다. 가장 대표적이면서 쉽게 예상할 수 있는 분야는 배달 시장일 것이다. 과거 중국집정도에서만 기대했던 배달서비스는 이제 중국집을 넘어 오히려 배달 서비스를 제공하지 않으면 사업을 운영할 수 없는 상황에 이르렀다.

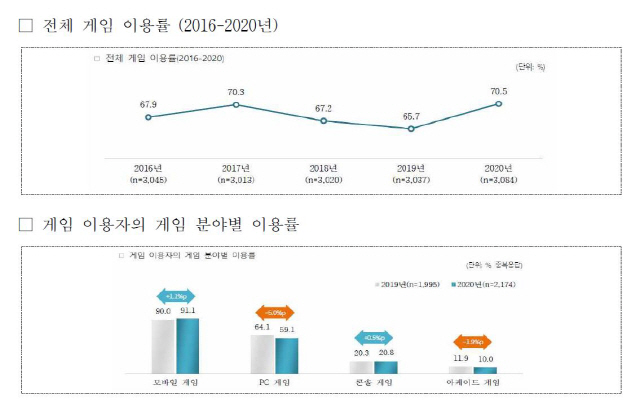


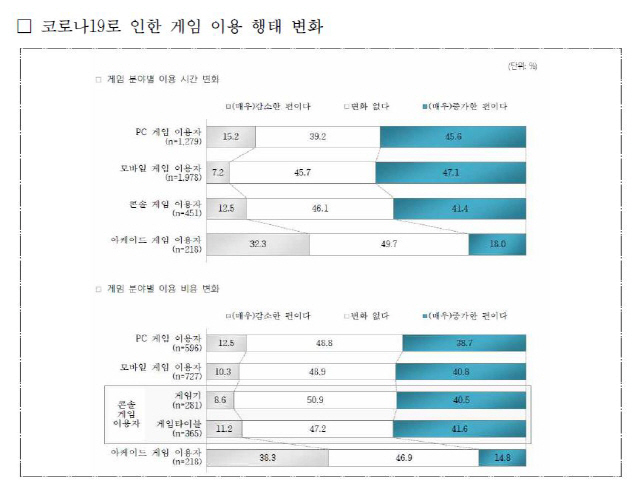
코로나 사태 이후 배달앱 결제 금액이 상승한 것을 볼 수 있다.

하지만 배달 시장 이외에도 코로나 사태로 인해 이익을 본 사업들은 꽤 많이 존재한다. 비대면 학습을 위한 온라인 강의 사업이라든지 재택 근무자를 위한 노트북, 웹캠 등 다양한 분야에서 새로운 시장이 개척되었다.

이러한 예상외의 이익을 본 다른 시장으로 게임 시장을 예로 들 수 있다. 외부와의 접촉을 최대한 자제하도록 권고하는 현 상황에서 사람들은 새로운 여가를 찾기 시작했고, 이러한 수요는 게임, 차박 등 개인이 혼자서 즐기거나 소수의 사람들만이 모여서 만남을 갖거나 혹은 다른사람과 직접 만나지 않고도 즐길 수 있는 종류의 시장으로 수요가 몰리기 시작했다.







코로나 사태 이후 게임 이용 시간이 증가한 것에 대한 통계

스마트폰이 등장한 이래 게임시장의 판도는 매우 많이 변화하였다. 과거 콘솔 또는 PC라는 단지 2지선다만이 존재했다면 이제는 스마트폰이라는 제 3의 선택지가 등장하면서 과거에비해 사람들은 더 높은 퀄리티의 게임을 더 쉽고 간편하게 접근할 수 있게 되었다.

1.1 프로젝트의 필요성

세월이 지나며 강산이 변하고 모든 것이 변화한다해도 세상에는 절대적으로 변하지 않는 것 또한 존재한다. 게임시장의 판도가 많은 변화가 있다고는 하지만 그곳에도 역시 절대로 변해서는 안되는, 또는 꼭 지켜야만하는 약속같은 것이 존재한다.

그러한 불변의 법칙이 단순히 꼭 한 개만 존재한다고는 할 수 없지만, 게임상 재화의 보전은 불변의 법칙 중 하나에 들기에 모자람이 없을 것이다.

사용자는 게임을하면 보상을 받는다. 사용자가 받은 보상의 종류는 다양하다. 경험치가 될수도 있고, 게임상의 돈, 또는 아이템이 될수도있다. 이렇게 받은 보상은 사용자가 게임상 기울인 노력의 대가이기에 사용자들은 이를 자신이 소유하고있다고 생각하게된다. 이러한 재화가 예기치못한 사건으로 손실되게 된다면 사용자들의 실망감은 이루 말할 수 없게되고 이는 게임사에 치명적으로 작용하게된다.



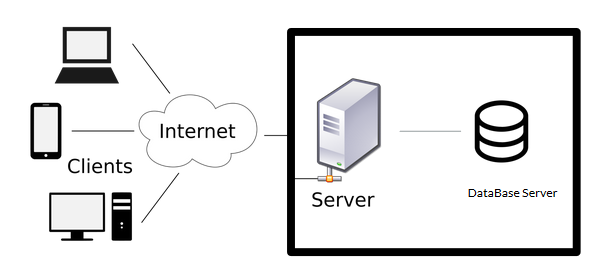
과거 점검 이후 재화 증발 사건이 발생한 후 유저들이 점검 전 자신의 재화를 미리 확인해두는 모습

콘솔게임처럼 판매와 함께 바로 이익을 회수할 수 있는 사업 구조라면 이런 버그에 대해 무시를 할 수도 있다. 자신들의 개발 역량을 뛰어넘었다던가, 더 이상의 AS에 투입할 인적 자원이 부족하다던가의 이유를 들어서 이미 판매한 게임의 AS를 포기하더라도 게임사는 이미 투자이익을 회수한 이후이기 때문에 디버깅을 포기할 수 있다. 물론 회사의 이미지는 깎일것이며 차기작에 대한 불이익은 감수해야할 것이다.

하지만 콘솔이 아닌 온라인 게임의 경우는 사정이 다르다. 유료제로 서비스를 제공했던 과거와는 달리 현재는 부분유로화가 온라인 게임 시장의 표준을 이루고있으며 이로인해 유저들은 원한다면 무료로 게임을 즐길 수 있게되었지만 게임사 입장에서는 투자비 회수 및 서비스 유지를 위한 운영비 회수에 매우 민감하게 반응할 수밖에 없고, 다시말해 재화 관련 버그는 게임사의 미래에 치명적인 문제를 일으킬 소지가 다분하다.

1.2 재화 버그의 원인

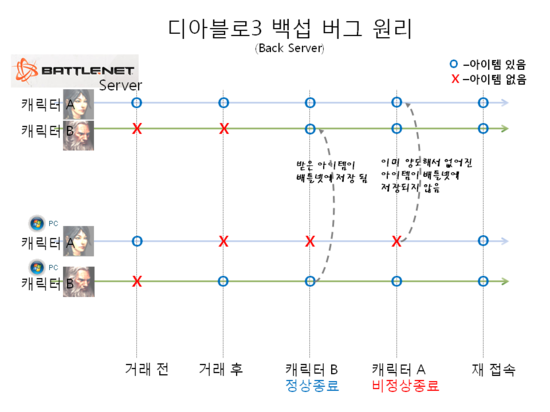
서비스 제공자에게 치명적인 원인으로 작용할 수 있는 재화 버그는 그 기본 원인은 꽤 유서깊은 역사를 가지고 있지만 최근 나온 게임에서도 꾸준히 발생하는 안타까운 현상이 존재한다.



고전적인 게임 프로젝트 구조

고전적인 게임 프로젝트는 서버-클라이언트 구조를 따르고있으며, 서버에는 항상 DB서버를 보유하고 있다. 이 DB서버에서 모든 클라이언트의 재화 정보를 저장한 후, 클라이언트가 요청하는 작업을 수행하고 업데이트하는 구조를 가지고 있다.

이런 중앙화된 서버는 이미 오랜시간 그 구조를 입증받았기 때문에 많은 게임 서비스사에서 채용하는 기본적인 구조이지만, 그로인해 그 원인이 이미 밝혀졌음에도 오랜 시간이 지난 지금도 동일한 재화 버그가 발생하는 원인이 되기도 한다.



아이템 복사 버그의 대표적 원리

요점은 트랜잭션의 타이밍문제이다. 이론상으로 유저가 하는 모든 행동을 매 단위시간마다 DB에 저장한다면 이러한 재화 버그는 발생할 수 없다. 하지만 이는 서비스하는 유저의 수가 많아질수록 DB서버에 매우 막대한 부하로 작용하게되므로 현실적으로는 불가능한 방법이다. 그렇기에 게임사들은 각자 최대한 재화 버그가 발생하지 않도록 할 수 있는 타이밍에 유저의 데이터 정보를 DB에 저장하는 방식을 채용한다. 다시말해 그 근본적인 원인은 해결되지 않은 것이다.

1.3 왜 블록체인인가?

블록체인은 모든 거래가 블록 내부에 저장되고 그 거래는 transaction을 단위로 블록이 발행된다. 블록이 발행되기 위해서는 이전 블록의 해쉬값을 포함하고 있어야하며, 모든 사용자가 블록 내용을 공유하기 때문에 조작이 불가능하다는 장점을 가지고 있다.



연속적으로 블록이 연결되면서 데이터의 무결성을 보장하는 구조

이러한 구조로 거래 내역은 무결성이 보장이된다. 이는 다시말해 데이터에 transaction이 가해지지만 않는다면 어떠한 사고가 발생하더라도 데이터가 변조가 될 일이 없다는 뜻이 되고, transactino이 발생했다면 그 값은 조작할 수 없다는 뜻이된다. 요컨대 조작이 불가능하면서 정확한 데이터를 저장할 수 있는 구조로 볼 수 있다.

2. 본문

2.1 구현 방법

기존의 블록체인의 경우 ERC 20, ERC721이라는 토큰의 전송을 transaction의 중점으로 다뤘다. 다시말해 ‘A가 B에게 토큰 a를 n개 송금’이라는 개념이 중점적이다. 하지만 스마트 컨트랙트라는 개념이 등장하고, 스마트 컨트랙트를 위한 언어인 solidity가 등장하면서 transaction의 내용은 좀 더 다양한 형태가 가능하게 되었다. 우리는 이 점을 착안하여 solidity내부에 유저 데이터를 저장한 후, 데이터 변동을 transaction으로 수행한다면 문제를 해결할 수 있지 않을까라는 생각을 갖게되었다.

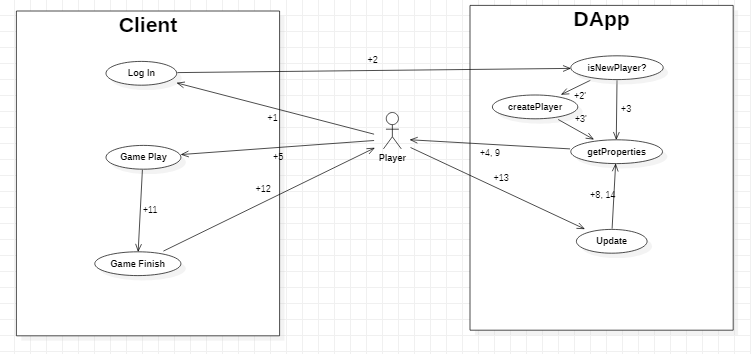


transaction의 변화

2.2 UseCase

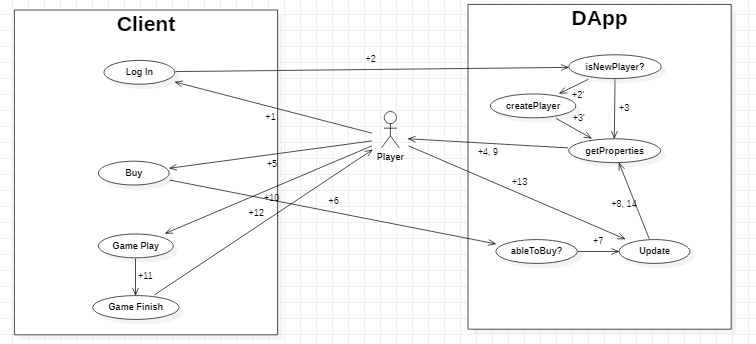
우리는 크게 3가지 usecase를 설정했다. 게임 플레이어가 게임을 끝냈을 때 얻은 보상을 얻는 update, 보상을 통해 다음 게임에서 사용할 아이템을 구매하는 buy, 자신이 가지고있는 아이템을 친구에게 선물하는 send가 그 예이다.

2.2.1 Update



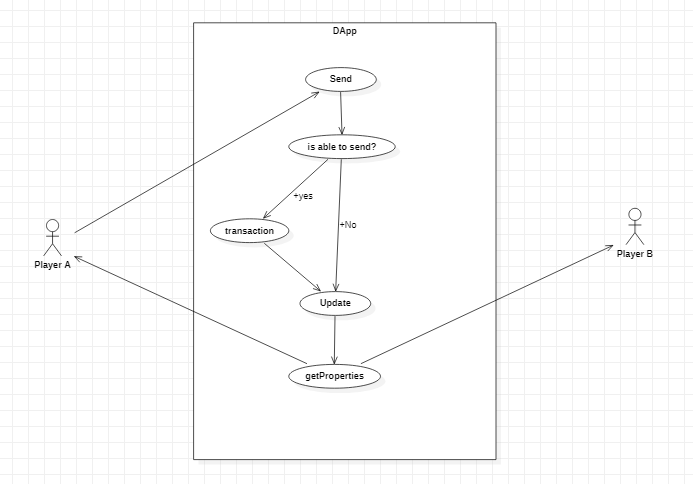
1. Player가 게임에 접속
2. 접속한 player의 유저 정보 확인
   1. 신규 유저일 경우 새 계정을 제공
   2. 기존 유저일 경우 유저가 과거 가지고있던 재화 정보 전송
3. player가 게임을 진행
4. 게임이 끝나고 유저가 얻은 보상을 update
5. 3번으로 돌아가서 반복

2.2.2 Buy



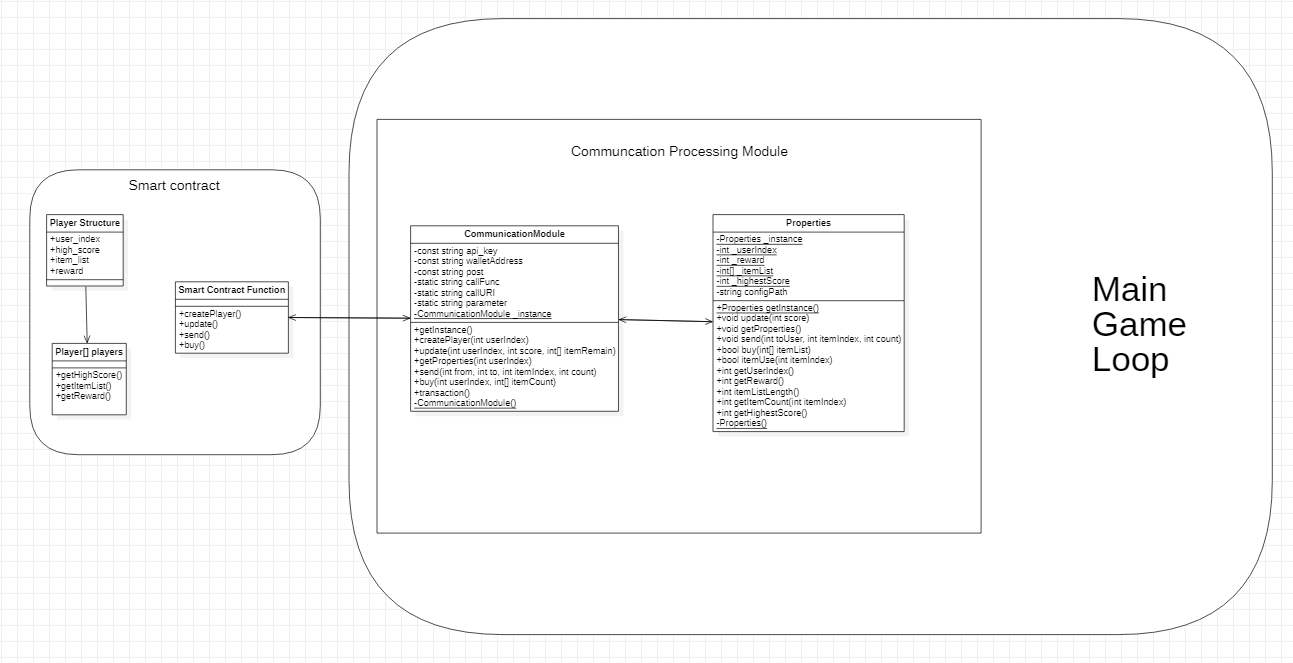
1. Player가 게임에 접속
2. 접속한 player의 유저 정보 확인
   1. 신규 유저일 경우 새 계정을 제공
   2. 기존 유저일 경우 유저가 과거 가지고있던 재화 정보 전송
3. player가 게임 실행 전 게임에서 사용할 아이템 구매
   1. 구매하고자 하는 아이템의 가격보다 보유한 재화가 충분할 경우 거래 성공
   2. 구매하고자 하는 아이템의 가격보다 보유한 재화가 부족할 경우 거래 실패
4. player 게임 진행
5. 게임이 끝나고 player가 게임간 소모한 아이템 정보 update
6. 3번으로 돌아가서 반복

2.2.3 Send



1. Player A가 Player B에게 아이템 전송
   1. Player A가 전송하고자 하는 아이템의 수량이 충분할 경우 전송 성공
   2. Player A가 전송하고자 하는 아이템의 수량이 부족할 경우 전송 실패
2. Player A와 Player B의 재화 정보 update

2.3 상위 설계



2.3.1 스마트 컨트랙트

스마트 컨트랙트 내부에는 크게 유저 재화 정보를 저장할 player struct, index값을 기반으로 player재화 정보에 접근할 get함수, 그리고 그 외에 createPlayer, update, buy, send같은 트랜잭션을 발생시킬 함수로 구성되어있다.

2.3.2 통신 모듈

통신 모듈에서는 DApp과 Rest방식으로 통신할 방법을 구현한 클래스이다. 통신 모듈에서 유저 property 데이터를 받아오면 적절한 파싱과정을 거쳐 게임 모듈에서 get함수만으로 필요한 데이터를 받아올 수 있도록 즉각적으로 사용가능하게 처리 과정을 거친다.

2.3.3 게임 모듈

게임 모듈에서는 통신 모듈에서 받아온 유저 property데이터를 기반으로 게임 데이터를 설정하고 게임을 구동시킨다.

2.4 하위 설계

2.4.1 스마트 컨트랙트

2.4.1.1 Struct Player

|  |  |  |
| --- | --- | --- |
| **Description** | 게임플레이어의 재화 관리를 위한 객체를 저장하는 부분. | |
| **type** | **식별자** | **description** |
| int | High\_score | 이전 게임에서의 최대 기록 |
| Int[3] | Item\_list | 보유중인 아이템의 개수 |
| int | reward | player의 재산 |

2.4.1.2 Player[] players

|  |  |
| --- | --- |
| **Description** | index값으로 접근할 수 있는 player데이터 |

2.4.1.3 Int[3] item\_price

|  |  |
| --- | --- |
| **Description** | 아이템 가격 |

2.4.1.4 GetFunctions

|  |  |  |
| --- | --- | --- |
| **Description** | index값으로 접근할 수 있는 player데이터 | |
| **type** | **식별자** | **description** |
| int | getHigh\_score | 이전 게임에서의 최대 기록을 가져오는 함수 |
| Int[3] | getItem\_list | 보유중인 아이템의 개수를 가져오는 함수 |
| int | getreward | player의 재산을 가져오는 함수 |

2.4.1.5 CreatePlayer

|  |  |  |
| --- | --- | --- |
| **Description** | 신규 유저일 경우 호출하는 함수 | |
| **type** | **식별자** | **description** |
| int | index | -1이 들어올 경우 신규 유저라는 의미 |

2.4.1.6 Update

|  |  |  |
| --- | --- | --- |
| **Description** | game이 끝나고 유저가 사용한 아이템, 점수들을 갱신하는 함수 | |
| **type** | **식별자** | **description** |
| int | index | 유저 구분을 위한 UID |
| int | score | 유저가 갱신한 점수 |
| Int[3] | Remain\_itemList | 사용하고 남은 아이템 개수 |

2.4.1.7 Send

|  |  |  |
| --- | --- | --- |
| **Description** | 다른 유저에게 선물하는 함수 | |
| **type** | **식별자** | **description** |
| int | from | 선물을 보내는 유저의 UID |
| int | Int to | 선물을 받는 유저의 UID |
| int | itemIndex | 선물하고자 하는 아이템의 번호 |
| int | itemCount | 선물하고자 하는 아이템의 개수 |

2.4.1.8 Buy

|  |  |  |
| --- | --- | --- |
| **Description** | 게임에 필요한 아이템을 구매하는 함수 | |
| **type** | **식별자** | **description** |
| int | index | 아이템을 구매하고자 하는 유저의 UID |
| int | itemIndex | 구매하고자 하는 아이템의 번호 |
| int | itemCount | 구매하고자 하는 아이템의 개수 |

2.4.2 Communication Module

2.4.2.1 Attribute

Rest통신에 필요한 절차를 구현한 클래스

|  |  |  |
| --- | --- | --- |
| **type** | **식별자** | **description** |
| CommunicationModule | instance | singleton패턴으로 구현 |
| string | Api\_key | 통신에 필요한 api key값을 저장 |
| string | walletAddress | 통신에 필요한 wallet address값을 저장 |

2.4.2.2 Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Return type** | | **식별자** | **description** |
| CommunicationModule | | getInstance | singleton패턴으로 구현 |
| int | | createPlayer | 생성된 UID를 반환하는 함수 |
| void | | update | 게임간 유저가 얻은 보상, 사용 아이템 정보를 갱신하는 함수 |
| **parameter** | | | |
|  | int | userIndex | 유저의 UID |
| int | score | 게임간 얻은 점수 |
| Int[] | itemRemain | 게임 후 남은 아이템 개수 |
| string | | getProperties | 유저의 재화 정보를 읽어오는 함수 |
| **parameter** | | | |
|  | int | userIndex | 유저의 UID |
| bool | | send | 선물 전달 함수 |
| **parameter** | | | |
|  | int | from | 선물을 보내는 유저의 UID |
|  | int | to | 선물을 받는 유저의 UID |
|  | int | itemIndex | 선물하고자 하는 아이템 번호 |
|  | int | itemCount | 선물하고자 하는 아이템 개수 |
| bool | | buy | 유저가 게임에 필요한 아이템을 구매하는 함수 |
| **parameter** | | | |
| int | | userIndex | 유저의 UID |
| Int[] | | itemCount | 구매하고자 하는 아이템의 개수 |

2.4.3 Properties

2.4.3.1 Attribute

Communication Module을 통해 받아온 데이터를 게임 모듈에서 사용하기 쉽게 가공하는 단계.

|  |  |  |
| --- | --- | --- |
| **type** | **식별자** | **description** |
| Properties | instance | singleton패턴으로 구현 |
| int | userIndex | 유저의 UID |
| int | reward | 유저가 가지고 있는 돈 |
| Int[] | itemList | 유저가 가지고 있는 아이템의 개수 |
| int | highestScore | 유저가 이전에 갱신한 최고 기록 |

2.4.3.2 Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Return type** | | **식별자** | **description** |
| Properties | | getInstance | singleton패턴으로 구현 |
| void | | update | 게임간 유저가 얻은 보상, 사용 아이템 정보를 갱신하는 함수 |
| **parameter** | | | |
|  | int | score | 유저가 게임에서 얻은 점수 |
| string | | getProperties | 유저의 재화 정보를 읽어오는 함수 |
| void | | send | 선물 전달 함수 |
| **parameter** | | | |
|  | int | to | 선물을 받는 유저의 UID |
|  | int | itemIndex | 선물하고자 하는 아이템 번호 |
|  | int | itemCount | 선물하고자 하는 아이템 개수 |
| bool | | buy | 유저가 게임에 필요한 아이템을 구매하는 함수 |
| **parameter** | | | |
|  | Int[] | itemCount | 구매하고자 하는 아이템의 개수 |
| int | | getUserIndex | 자신의 UID 반환 |
| Int | | getReward | 자신이 보유한 돈의 양 반환 |
| int | | getItemCount | 가지고 있는 아이템의 개수 반환 |
| **parameter** | | | |
|  | int | itemIndex | 조회하고자 하는 아이템의 번호 |
| int | | getHighestScore | 자신이 갱신한 최고 기록값 반환 |

3. 결과



게임 메인 화면



게임 플레이 화면



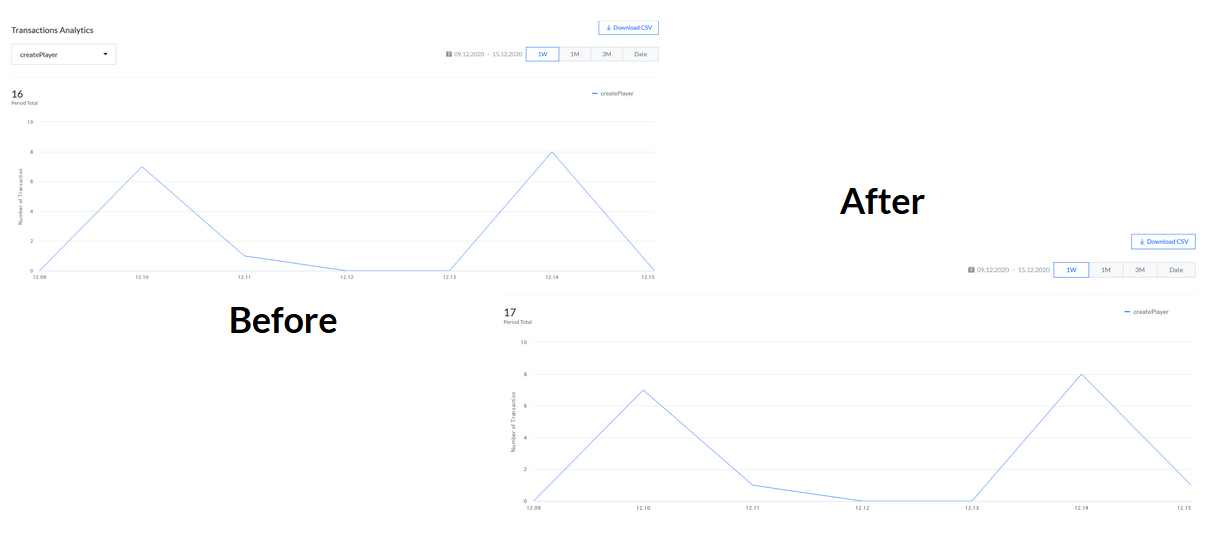
아이템 구매 화면

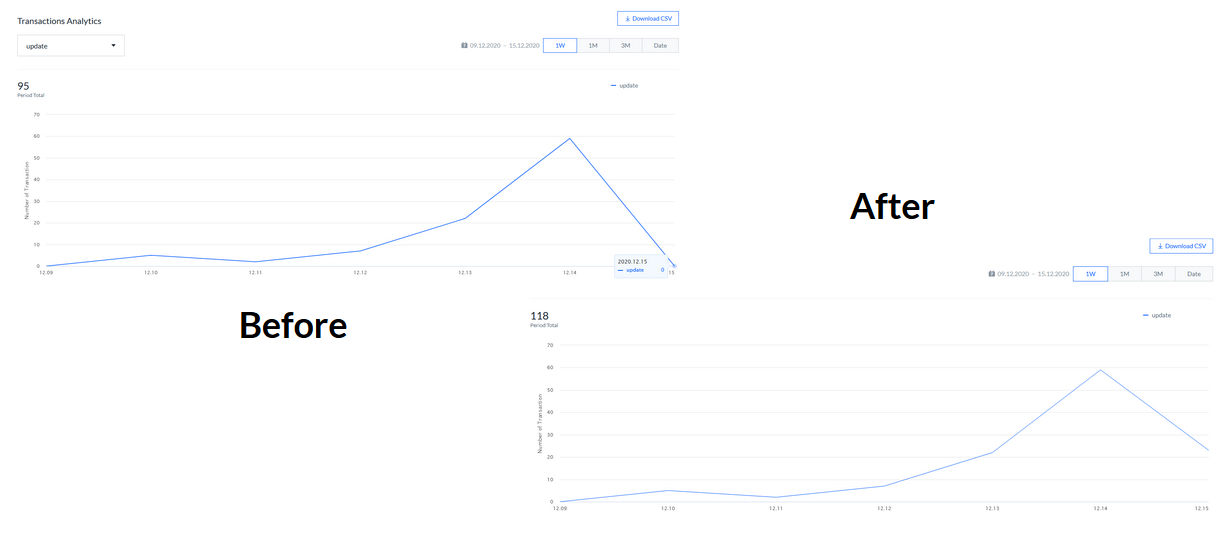


선물하기 화면

프로젝트의 구현 가능성을 위해 간단한 러닝게임을 구현해보았다. 게임 내부에는 단순한 러닝 게임 알고리즘과 그 외에 게임에 필요한 아이템 구매, 아이템 선물하기 기능을 구현했다.











게임 중 데이터의 갱신이 필요할때마다 transaction이 발생하는 것을 확인할 수 있다

게임을 진행하면서 데이터의 변동이 생기는 경우 transaction이 발생하고 발행한 transaction은 블록 내부에 안전하게 저장되는 것을 확인할 수 있다.

3.1 기대효과

최근들어 서버리스 아키텍처라는 기술이 유행하기 시작했다. 기존의 서비스 제공자들이 적절한 서비스 수요를 예측하여 서버를 구매 후, 전산실을 직접 설치한 후 서비스를 제공하던 방식은 서비스 수용의 급격한 변동에 빠르게 대응하기 힘들다는 단점이 존재했었다.



사용자 수요 예측을 너무 적게해서 발생한 입장 대기 대란

하지만 최근 유행하는 서버리스 아키텍처의 경우 아키텍처를 제공하는 회사에서 서버 자원을 유동적으로 지원해주고 자원을 사용한만큼의 요금만을 청구하는 방식으로 서버-클라이언트 방식의 서비스 제공을 고민하는 많은 회사들의 고민을 덜어주고 있다.

하지만 이러한 방식에서도 역시 기본 구조는 과거 전산실을 설치하고 서비스를 제공하던 것과 다를바가 없기 때문에 이 역시도 중앙화된 DB서버를 가지고있는 형태이다. 다시말해 이러한 방식 역시 DB관리자의 사소한 실수로인해 게임상의 재화유실 문제가 발생할 수 있다는 것을 의미한다.

지금도 블록체인, 비트코인, 이더리움을 인터넷에 검색하면 코인값이 급등했냐느니, 급락했냐느니하는 기사가 매우 많다. 하지만 우리는 이번 프로젝트를 진행하면서 블록체인의 진정한 가치란 단순하게 코인의 단가가 아닌 데이터의 수정불가능하고 무결성을 보장하는 시스템이라는 것이 가장 중요한 점이라는 깨달음을 얻게되었다.

이러한 관점에서 블록체인은 지금의 클라우드 방식의 서버리스 아키텍처와는 다른, 클라우드를 대체하는 것이 아닌 그들과 함께 서로의 단점을 보완할 수 있는 새로운 형태의 서버리스 아키텍처의 제안이 될 수 있다고 생각한다.