

블록체인을 활용한 포트폴리오 관리 플랫폼



201624586 조병우

201712159 조현우

201724490 서지원

지도교수 권 동 현

목 차

1. 연구 소개	1
1.1. 배경 및 필요성	1
1.2. 기술 및 시장 현황	2
1.3. 개선 가능성	3
1.4. 연구목표	5
2. 연구 배경	6
2.1. 연구 개요	6
2.2. 블록체인	7
2.3. 서버	14
2.4. 프론트엔드	22
2.5. 전체 시나리오	26
2.6. 시스템 구성도	28
3. 연구 결과 분석 및 평가	29
3.1. 설계 변경 내역 및 한계점	29
3.2. AI-블록체인 아이디어 경진대회	30
3.3. 사용자 설문조사 결과 분석	31
4. 결론 및 향후 연구 방향	36
5. 개발 일정 및 역할 분담	37
5.1. 개발 일정	37
6. 참고 문헌	38

1. 연구 소개

1.1. 배경 및 필요성

4차 산업혁명 시대로의 발전은 다양한 산업 분야에 수많은 변화를 가져왔다. 이런 변화를 반영하듯 각 산업에서 필요로 하는 인력에도 변화가 일어나고 있다. 각 기업은 변화 하는 기술에 대처하기 위해 새로운 직원을 채용하는 여러 방안을 모색하고 있다. 하지만 자격증, 성적, 학력, 수상 내역 등의 자기 정보들을 위조하여 부정 입학 및 위장 취업하는 등의 문제는 꾸준히 발생하고 있다.

입사 지원 및 직원 채용 과정을 살펴보면 먼저 특정 기업이 직원 채용 공고를 게시하고 해당 기업에 입사하고자 하는 지원자들은 직접 작성한 포트폴리오를 해당 기업에 제출한다. 이러한 일련의 절차를 통해 해당 기업은 새로운 직원을 모집하고 입사 지원자의 조건이 채용 조건에 적합하다고 판단이 되면 최종적으로 해당 직원을 채용하게 된다. 기존의 채용 플랫폼은 사용자 본인이 직접 자격증, 수상 내역, 학력 등의 자기 정보를 입력하고 포트폴리오를 작성하는 형태로 구성되어 있다. 이때, 기업은 포트폴리오에 입력된 자기 정보에 대해 추가적인 검증을 시행하지 않기 때문에 입사 지원자가 악의적인 목적으로 해당 정보들을 위조로 작성하여 기업에 제출하게 되면 기업에서 해당 데이터들을 검증하는 과정이 불가피하다. 추가로 대부분 기업은 입사 지원자들의 모든 개인 정보 및 관련 데이터를 자사 데이터베이스에 저장해두고 관리하는 데 이는 기업의 나쁜 의도에 의해 개인정보 유출 및 자기 정보의 오남용으로 이어질 수 있다.

채용절차에서 입사지원자의 개인정보는 어떻게 관리해야 할까

최근 채용절차에서 입사지원자의 개인정보가 외부로 유출되는 사례가 연이어 발생했다. 일정 규모 이상의 기업에는 인사팀 등 채용절차를 전담하는 부서가 별도로 마련되어 있는 경우가 일반적이므로, 채용절차에서 수집되는 서류 및 증빙자료를 관리하는 절차와 프로세스가 잘 정립된 편이다.

반면 스타트업의 경우, 특정 직무의 인력이 필요한 시점에 채용절차를 진행하여 현업부서가 전형에서 선발 등 모든 채용과정을 직접 진행하는 경우가 흔하므로, 채용절차에서 입사지원자의 개인정보를 어떻게 보호하고 관리하는지에 대한 내부 지침이 마련되어 있지 않은 경우가 많다.

그림 1 - 채용 절차에서의 입사 지원자 개인정보 유출 (출처 - NEPLA)

본 과제는 개인 정보의 유출과 오남용 및 데이터의 위변조와 같은 문제를 해결하기 위해 자기 정보 통제 및 데이터 무결성 측면에서 장점을 가지는 블록체인을 활용한 개인 포트폴리오 관리 플랫폼을 제안한다. 제안하는 시스템은 블록체인에 사용자 자기 정보를 저장하고 블록체인 네트워크는 프라이빗 네트워크의 구현체인 하이퍼레저 패브릭을 기반으로 동작한다. 개인 포트폴리오 항목의 해시값을 블록체인에 저장함으로써 공유 데이터의 무결성 검사가 가능하게 한다. 현재 세종 텔레콤, 몰리다 등 여러 학사 정보 및 포트폴리오 관리 플랫폼들이 존재하지만, 기능 및 성능 측면에서 보완할 점을 다수 발견했다. 이를 개선하고자 새로운 개인 포트폴리오 관리 플랫폼을 제안한다.

1.2. 기술 및 시장 현황

1.2.1. 세종텔레콤 스마트 학사정보 관리(SER)

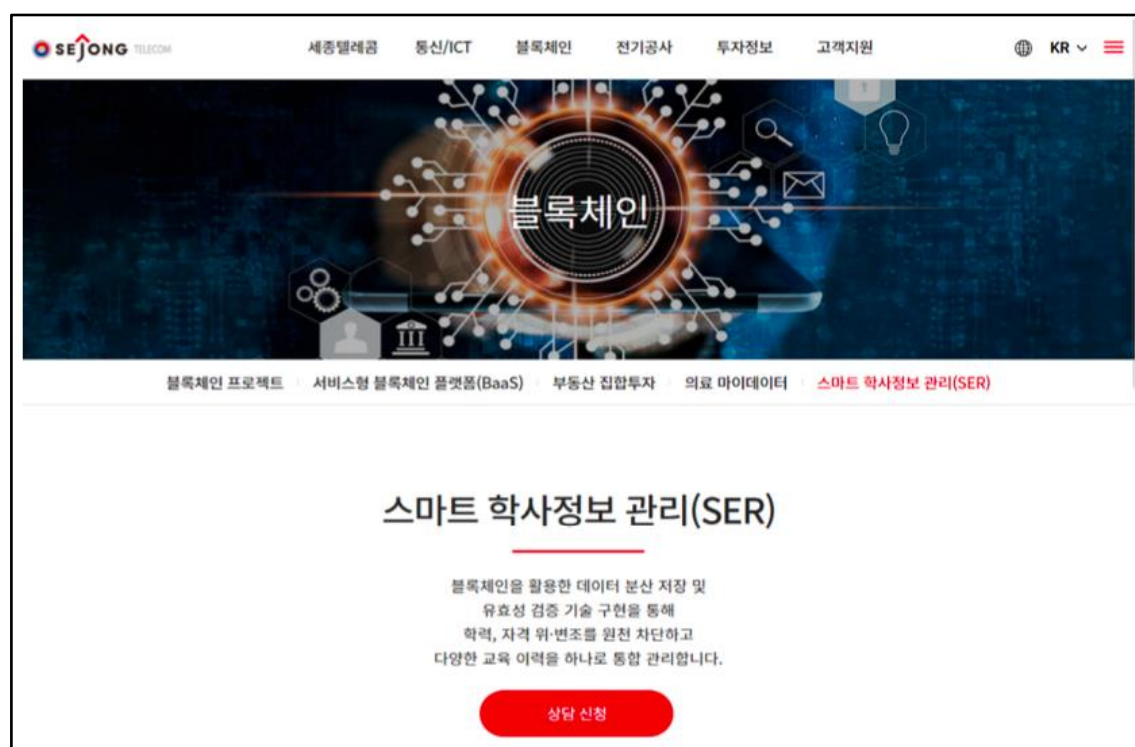


그림 2 - 세종텔레콤 스마트 학사정보 관리(SER)

세종 텔레콤 스마트 학사정보 관리 플랫폼은 블록체인을 활용한 데이터 분산 저장 및 유효성 검증 기술 구현을 통해 학력, 자격 위·변조를 원천 차단하고 다양한 교육 이력을 하나로 통합 관리한다.

1.2.2. 몰리다 – 직무학습이력 관리 플랫폼

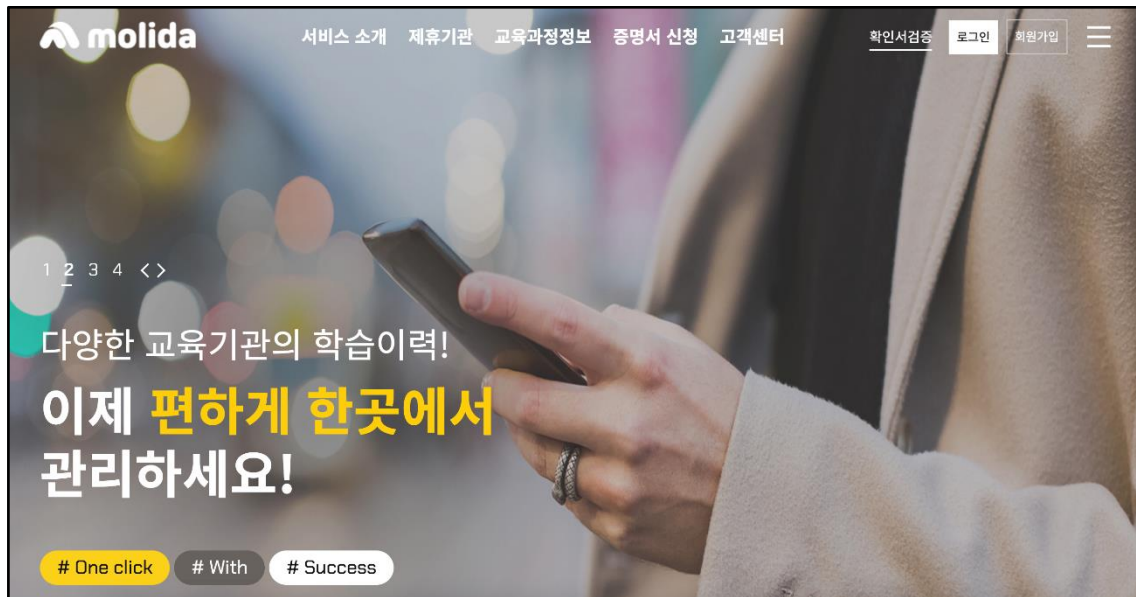


그림 3 - 몰리다 직무학습이력 관리 플랫폼

몰리다 직무학습 이력 관리 플랫폼은 블록체인을 기반으로 증명서를 발급 및 관리한다. 중소 교육기관의 증명서 발급 업무를 돕기 위해 산업통상자원부와 협력하여 개발한 플랫폼이다.

1.3. 개선 가능성

기존에 존재하는 다른 플랫폼의 경우 자기 정보 및 포트폴리오 관리는 다음과 같은 절차로 진행된다.

1. 지원자가 기업에 포트폴리오를 제출할 때 블록체인에 등록되어 있는 자신의 ID를 함께 제출한다.
2. 기업은 지원자의 포트폴리오를 수령하고 수령 여부를 통지한다.
3. 기업은 블록체인 관리자에게 지원자의 포트폴리오 내부의 학력과 경력의 위조 여부에 대한 검증을 요청한다
4. 블록체인 관리자는 지원자에게 블록체인 내의 경력 확인 동의 여부를 확인한다.
5. 지원자는 블록체인 관리자에게 본인의 데이터를 조회할 수 있도록 권한을 부여하는 메시지를 발송한다.

6. 블록체인 관리자는 체인코드를 실행하여 블록체인의 내부 데이터를 확인하고 이때 사용자의 고유 ID를 사용하여 사용자 데이터를 검색한다.

7. 블록체인 관리자는 블록체인에 등록한 데이터를 회사에 전달하여 위변조 여부를 확인할 수 있도록 한다.

표 1 - 기존 플랫폼의 자기 정보 및 포트폴리오 관리 절차

기존 플랫폼의 자기 정보 및 포트폴리오 관리 절차에 따르면 사용자가 입사하고자 하는 기업에 본인이 작성한 포트폴리오를 제출한 이후 해당 기업은 블록체인 관리자에게 포트폴리오 내부의 학력과 자격증 및 수상 내역 등의 자기 정보 위조 여부에 대해 검증을 추가로 요청해야 한다. 하지만 본 과제에서 제안하는 시스템은 이를 개선하여 이미 검증이 되어 신뢰성이 보장되는 데이터로 구성된 정해진 양식의 포트폴리오 URL을 기업에 제출한다면 해당 기업 입장에서는 포트폴리오 내부의 데이터에 대한 추가적인 검증을 진행할 필요가 없게 된다. 위의 표에 나와 있는 자기 정보 및 포트폴리오 관리 절차에 따른 일련의 과정들이 모두 단축되는 것이다.

본 과제에서 개발하고자 하는 플랫폼의 핵심 기능으로는 사용자의 자기 정보 데이터 및 포트폴리오를 프라이빗 블록체인의 구현체인 하이퍼레저 패브릭을 활용하여 무결성을 보장하고 이를 저장하는 것이다. 개인 포트폴리오를 생성하기 위해 블록체인 네트워크를 조회하여 데이터를 불러오고 정해진 양식에 맞춰 이를 시각화한다. 블록체인 네트워크로부터 조회된 자기 정보 데이터들을 일종의 블록 형태로 서버 및 프론트엔드에 제공한다. 이는 수정이 불가능한 읽기 전용의 블록 데이터 형태이고 사용자는 해당 데이터 블록을 재배포하여 포트폴리오의 형태를 수정할 수 있다. 이를 통해 사용자에게 포트폴리오의 시각적 다양성을 제공한다. 또한 사용자는 기업에서 요구하는 최소한의 자기 정보만을 포트폴리오에 선택적으로 포함해 자기 정보의 외부로 유출을 최소화한다. 이를 통해 최종적으로 완성된 포트폴리오는 URL 링크를 통해 기업에 제공된다.

더불어, 기존의 플랫폼들과는 달리 경력 및 증명을 발급해주는 기관뿐만 아니라 입사 지원자와 해당 기업에도 편의를 제공하기 위해 자기 경력에 대한 코멘트 또는 자기소개서 편집 및 작성 기능을 제공한다. 이로써 제안하는 시스템은 기존의 플랫폼들을 완전히 대체할 수 있다.

1.4. 연구목표

본 졸업 과제는 프라이빗 블록체인 기반의 개인 포트폴리오 관리 플랫폼 개발을 목표로 한다. 개인 정보의 오용 및 남용을 비롯한 각종 개인 정보 관련 보안 측면의 문제점들을 해결하기 위해 블록체인을 활용하여 개인의 자기 정보를 관리한다는 명목이다. 블록체인의 공유 원장에는 개인의 자격증과 학력, 수상 내역 등의 개인 포트폴리오에 기재할 수 있는 전반적인 내용을 기록하며 이를 내부 합의 알고리즘을 사용하여 검증하고 블록으로 생성하는 형태로 동작한다.

제안하는 시스템은 개인 포트폴리오에 포함될 자기 정보들을 블록체인에 저장하고 포트폴리오 생성 시 체인코드를 사용한 정보 조회를 통해 데이터를 가져온다. 정해진 양식에 맞춰 데이터를 시각화하고 포트폴리오를 생성하여 이를 URL 링크를 통해 사용자에게 제공하는 형태이다. 개인의 자기 정보를 사전에 검증되어 신뢰성이 보장되는 상태로 기업에 전달할 수 있는 방식이다.

또한 입사 지원자는 기존의 방식대로 진행되던 원본 데이터들로 구성된 직접 작성한 포트폴리오의 제출이 아닌 제안하는 플랫폼에서 생성된 짧은 수명의 URL 링크를 제출함으로써 기업은 읽기 권한만 가지게 되며 해당 포트폴리오에 대한 편집 및 저장은 불가능하게 한다. 입사 지원자들의 자기 정보 데이터는 블록체인을 통해 무결성이 보장되며 정해진 목적이 아닌 다른 이유로 기업이 해당 개인 정보 조회 및 사용을 요청할 경우 항상 사용자의 동의를 얻어야 한다. 기업에 제공한 URL 링크는 정해진 시간이 지나면 자동으로 소멸하게 하고 유효 시간은 사용자가 설정할 수 있게 한다. 이를 통해 사용자의 개인 정보와 자격증 및 수상 내역의 자기 정보는 개인정보의 유출 및 오남용으로부터 안전하게 관리될 수 있다.

2. 연구 배경

2.1. 연구 개요

1. 프라이빗 블록체인의 구현체인 하이퍼레저 패브릭을 활용해 자기 정보 데이터와 개인 포트폴리오의 무결성 및 신뢰성을 보장한다.
2. 사용자가 입사하고자 하는 기업에 이력서 및 포트폴리오를 제출했을 때 해당 기업이 이를 검증하는 절차가 존재한다. 이때, 지원자가 사전에 검증된 자기 정보들로 구성된 포트폴리오 URL 링크를 제출함으로써 기존에 진행해왔던 검증 절차를 간소화하여 시간을 단축한다.
3. 해당 기업은 입사 지원자들의 모든 개인 정보 및 관련 데이터를 수집하고 이를 자사 데이터베이스에 저장해두고 관리한다. 이는 기업의 나쁜 의도 혹은 보안상의 문제에 의해 정보의 오남용으로 이어질 수 있다. 이를 방지하기 위해 지원자들은 기존의 방식대로 진행되던 원본 데이터의 제출이 아닌 해당 플랫폼에서 생성된 짧은 수명의 URL 링크를 제출한다.
4. URL 링크는 정해진 시간이 지나면 자동으로 소멸하도록 하고 해당 페이지에서 복사, 스크린 샷, 드래그 등의 기능은 제한된다. 이를 통해 사용자의 개인 정보와 자격증 및 수상 내역, 학력 등의 자기 정보는 안전하게 관리될 수 있다.
5. 사용자가 포트폴리오에 따라 학력, 자격증, 수상 내역 등의 자기 정보에 대한 노출 여부를 선택할 수 있게 하여 각 기업에 서로 다른 포트폴리오를 보여주고 이에 따라 기업이 요구하는 최소한의 정보만을 노출할 수 있게 한다.
6. 지원자는 마크다운 기능을 활용할 수 있고 이를 통해 개인의 포트폴리오를 비롯하여 자기소개서도 생성 및 수정할 수 있다.

2.2. 블록체인

2.2.1. 배경지식

A. Hyperledger Fabric

Hyperledger 프로젝트는 IBM에서 시작하게 되어 2015년부터 리눅스재단이 이끌고 있는 오픈소스 블록체인 프로젝트이다. HyperLedger 프로젝트에는 Fabric, Iroha, Indy, Composer 등의 여러 프로젝트가 포함되어 있는데, 그중 가장 활발하게 운영 중인 Fabric은 어플리케이션이나 솔루션을 개발할 수 있는 블록체인 프레임워크이다. 특징으로는 멤버십 서비스나 합의 등 핵심 기술 요소들이 plug-and-play 방식으로 제공되어 수정이나 변경하고 싶은 코드를 다른 코드의 영향 없이 쉽게 교체할 수 있다는 점이 있다. 또한, 분산 원장과 스마트컨트랙트를 체인코드로 구현하여 제공하며, 멤버십 서비스를 통해 개인정보 보호 및 접근을 제어하는 허가형 프라이빗 블록체인이다.

프라이빗 블록체인은 네트워크에 참여한 참여자들이 동일한 목적을 갖고 참여하므로 퍼블릭 블록체인과 달리 네트워크에 기여를 이끌어 내기 위한 합의 알고리즘과 보상이 불필요하다. 따라서, 새로운 블록 생성에 소요되는 시간과 비용이 퍼블릭 블록체인에 비해 빠르기 때문에 고속의 transaction을 요구하는 도메인에서 사용하게 된다. 퍼블릭 블록체인과 프라이빗 블록체인의 비교는 아래와 같다.

구분	퍼블릭 블록체인	프라이빗 블록체인
읽기 권한	누구나 읽기 가능	허가된 기관만 읽기 가능
트랜잭션 검증 및 승인	누구나 트랜잭션 검증 및 승인 가능	허가된 기관과 감독 기관만 트랜잭션 검증 및 승인 가능
트랜잭션 생성	누구나 트랜잭션 생성 가능	법적 책임을 지는 기관만 생성 가능
합의 알고리즘	부분 분기를 허용하는 작업 증명이나 지분 증명 알고리즘	부분 분기를 허용하지 않는 CFT계열의 합의 알고리즘
속도	7~20 TPS	1000 TPS 이상
권한 관리	누구나 같은 권한을 갖고 있음	채널과 Authorization 을 통한 읽기/쓰기 권한 관리 가능
구현체	비트코인, 이더리움 등	Hyperledger Fabric, R3 Corda

표 2 - 퍼블릭 블록체인과 프라이빗 블록체인의 비교표

B. 분산된 원장

블록체인 네트워크의 핵심은 네트워크 내에서 기록된 모든 트랜잭션(Transaction)을 저장한 원장(ledger)을 분산시키는 것이다. 블록체인 원장이 탈중앙화되었다고 말하는 것은 네트워크의 여러 참여자가 복제된 원장을 가지고 있기 때문이다. 이 네트워크 내의 참여자들은 해당 네트워크를 유지하기 위해 상호 간에 합의된 상태이다. 이런 탈중앙화와 합의는 실제 세상의 상품과 서비스를 거래하는 비즈니스에서 강력한 특성으로 작용한다.

탈중앙화와 합의에 이어 블록체인의 또 다른 중요한 특성은 암호화 기술을 사용하여 원장에 기록된 트랜잭션은 수정이 불가능하고 Append-Only 속성을 만족한다는 것이다. 또 다른 이름으로 불변성이라고 하며 이에 따라 정보의 기원(Root 또는 Parent)을 찾기 쉽다. 이에 따라 블록체인은 증명하는 시스템에 사용된다.

C. 체인 코드와 스마트컨트랙트

정보의 일관성, 트랜잭션, 쿼리 명령문 등 원장의 고유 기능들을 보장하기 위해 원장들 사이에 사용되는 계약서를 스마트컨트랙트라고 한다. 스마트컨트랙트는 단순히 정보를 캡슐화하여 네트워크 위에서 단순화시킬 뿐 아니라 네트워크 참여자들에게 일종의 트랜잭션을 자동으로 실행할 수 있게 하기 위해 사용한다. 체인코드는 하이퍼레저 패브릭에서 스마트컨트랙트를 발급할 수 있는 일종의 패키지를 의미한다. 체인 코드는 외부의 어플리케이션이 원장과 상호작용을 원할 때 사용되어 외부 어플리케이션에 의해 동작된다. 하이퍼레저 패브릭은 Go, Node.js, Java SDK 등을 통해 체인코드를 작성할 수 있도록 지원한다.

D. 합의

네트워크 사이에서 원장이 동기화 되어 있도록 유지하는 과정을 합의라고 한다. 이러한 과정을 통해 트랜잭션이 네트워크 내의 특정 참가자들로부터 증명될 때만 원장이 갱신되도록 보장하고 원장이 갱신될 때 동일한 오더의 트랜잭션을 업데이트하도록 보장한다. 따라서 블록체인 네트워크의 구성을 한마디로 정리하면 스마트컨트랙트로 업데이트되고 합의로 동기화를 보장하는 서로 공유되고 복제된 트랜잭션 시스템이라 할 수 있다.

E. 컨소시움과 채널

컨소시움의 사전적 정의는 “함께 협력하기로 동의한 사람 또는 집단”으로 하이퍼레저 패브릭에서 조직(Organization)들이 하나의 컨소시움을 구성하면 그 조직들은 트랜잭션 내역을 공유할 수 있다. 하나의 하이퍼레저 패브릭 네트워크에서 여러 컨소시움을 구성할 수 있으며 언제든지 설정을 통해 컨소시움에 조직을 추가할 수 있고 삭제할 수 있다.

하이퍼레저 패브릭에서 채널은 둘 이상의 특정 블록체인 네트워크의 참여자들 사이에 통신을 지원하기 위한 프라이빗(Private)한 서브넷(Subnet)이다. 다시 말해 채널은 컨소시움 내 커뮤니케이션 메커니즘이다. 채널은 채널의 구성원인 조직과 구성원당 대표 피어(Anchor Peer), 공유된 원장, 체인코드와 오더러(Orderer)로 정의된다. 네트워크의 각 트랜잭션은 채널 위에서 실행되며 거래자들은 해당 채널에서 거래할 수 있도록 인증 또는 승인을 받아야 한다. 채널에 참여하는 각 피어는 서비스에 인증하는 MSP(Managed Service Provider)에서 제공하는 고유한 ID를 갖게 된다.

하이퍼레저 패브릭에서 원장은 각 채널에 종속된다. 하나의 채널당 하나의 원장을 갖게 되고 이 장부를 피어들이 물리적으로 호스팅하여 한 채널에 종속된 피어들은 동일한 내용의 장부 복사본을 갖게 된다. 장부의 갱신은 여러 피어들의 합의를 통해 이루어지기 때문에 일관성을 유지하게 된다. 또한, 해당 채널에 속하지 않은 구성원은 원장을 확인하지 못하기 때문에 채널의 분리는 효율적인 데이터 관리와 공유를 가능하게 만든다. 이는 퍼블릭 블록체인과 구분되는 가장 큰 차별점이다.

F. 조직

네트워크를 구성하는 구성원의 역할을 하는 “조직”은 블록체인 네트워크 공급자에 의해 블록체인 네트워크에 가입된다. 조직은 MSP(Membership Service Provider)를 네트워크에 추가하여 네트워크에 가입할 수 있다. MSP는 해당 조직에서 발급한 서명이 유효한 ID로부터 생성되었는지 어떻게 네트워크의 다른 조직이 확인할 수 있는지 방법에 대해 정의하게 된다. MSP 내부 ID의 특정 액세스 권한은 조직이 네트워크에 가입할 때 동의하는 정책에 의해 관리된다. 조직은 다국적 기업만큼 클 수도 있고, 일반 개인만큼 작을 수도 있다. 조직의 모음을 컨소시움이라하고, 조직에서 발생하는 transaction의 endpoint를 peer라 한다. Peer는 원장을 유지, 관리하고 원장에 대한 읽기/쓰기 작업을 수행하기 위해 체인코드 컨테이너를 실행하는 네트워크의 엔티티이다. Peer는 조직이 소유하고, 유지, 관리한다.

2.2.2. 수행 내역

A. 블록체인 네트워크 구조

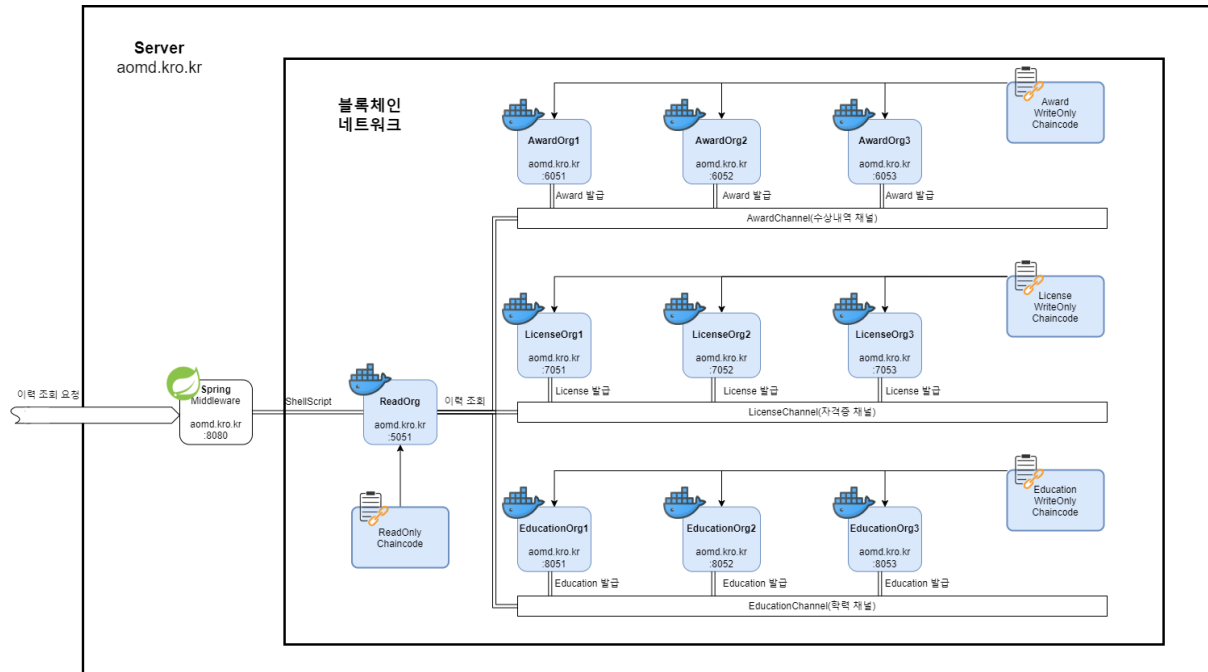


그림 4 - 블록체인 네트워크 구조

최종적으로 구성된 블록체인 네트워크 구조는 위와 같다. 해당 플랫폼에서 수상내역(Award), 자격증(License), 학력(Education)의 보관을 지원하기 때문에, 각 asset을 저장하고 읽기 위한 채널들을 생성해주었다. 각 채널에는 각 asset을 발급하는 임의의 조직들을 할당하여 주었고, 채널에 종속된 조직들에게 각각 동일한 체인코드를 발급하여 주었다.

B. 조직 구성

이름	포트
ReadOrg	5051
EducationOrg1	6051
EducationOrg2	6052
EducationOrg3	6053
AwardOrg1	7051
AwardOrg2	7052
AwardOrg3	7053
LicenseOrg1	8051
LicenseOrg2	8052
LicenseOrg3	8053
OrdererOrg	9050

표 3 - 조직들의 포트 매핑표

각 조직은 실제 기업이나 단체에서 운영하는 서버로, 본 과제를 수행하는데 있어서 실제 서버를 수 대 대여하는 것은 어려움이 따르므로 Docker를 사용하여 서로 다른 port를 통해 접근하도록 설정하였다. 여기서 조직은 각각 하나의 peer를 갖게 되며, 각 peer는 채널에 종속되는 원장의 복사본을 유지, 관리하고 읽기/쓰기 작업을 직접 수행하는 역할을 한다. 초기 모델은 각 채널당 1개의 조직을 할당했었는데, 새로운 조직을 추가하는 작업에 대한 테스트와 네트워크모델의 고도화를 위해 각 채널당 3개의 조직으로 증가시켰다.

네트워크의 간소화를 위해 채널당 CA(Certificate Authority)를 1개씩 두어 ca-educationOrg1, ca-awardOrg1, ca-licenseOrg1, ca-readOrg, ca-ordererOrg의 총 5개의 CA를 구성했다. 실제 docker 컨테이너로 구현되는 것이 아닌 추상적인 개념인 MSP(Member ship Service Provider)는 각 조직당 1개씩 할당하였다.

각 채널에 소속된 조직뿐만 아니라 모든 채널에 소속된 ReadOrg 조직을 별도로 네트워크에 추가하였다. ReadOrg는 읽기전용 체인코드를 따로 발급하기 위해 고안된 조직이다. 해당 조직은 네트워크에 정의된 모든 채널에 참여하여 각 원장의 내용을 읽을 수 있다. 읽은 원장의 내용을 이력 조회를 요청한 백엔드 서버에 반환하여 서버에서 비즈니스로직에 활용하거나 다시 사용자에게 반환할 수 있도록 한다.

C. 체인코드

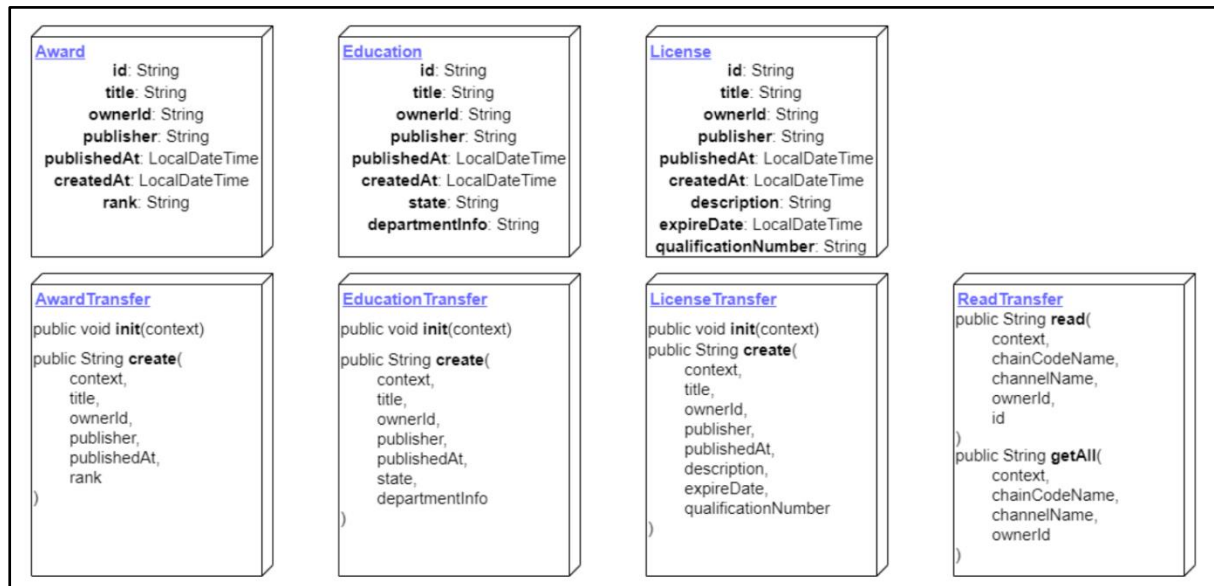


그림 5 - 블록체인 네트워크에 정의된 Asset과 체인코드(Transfer)

채널에 종속된 조직들은 각 채널에 이력을 발급할 수 있는 체인코드를 발급하여 주었다. 이 체인코드는 쓰기 작업을 하는 코드만 포함되어 있고, 원장의 내용은 읽을 수 없게 하였다. 이로써 한 발급기관이 다른 발급기관이 발급한 개인의 이력을 조회하는 문제가 발생할 수 있는 여지를 가장 아래 단계인 체인코드에서부터 방지하였다.

읽기전용인 ReadOrg에는 읽기전용 체인코드를 따로 발급하였다. 각 채널에 정의된 체인코드에 읽기 작업을 위한 코드를 추가할 수 있지만, 체인코드 단계에서 쓰기 작업을 방지하였다. 예기치 못한 공격으로 인해 코드 일부가 탈취되거나 하는 위협을 차단하기 위해서이다.

해당 체인코드는 모두 Java로 작성되었으며, 각 체인코드 당 하나의 Java 프로젝트가 된다. 대다수의 Hyperledger Fabric 샘플 코드를 보면 Go 언어로 작성되어 있지만, 해당 프로젝트에서 사용된 프레임워크인 Spring과 언어를 맞추어 하나의 프로그래밍 언어로 구현할 수 있도록 하기 위해서 Java 언어로 작성하였다.

D. 최종 도커 컨테이너

```
huthi@huthi-server:~/workspace/graduate/chainNetwork$ docker ps --format "table {{.Names}}\t{{.Ports}}\t{{.Status}}"
```

NAMES	PORTS	STATUS
dev-peer0.licenseOrg2.aamd.com-readCC_1.0-bd3ef310267e61ed7a48be30f824f90ff5a4d87f352b4c80fafe72cccae9862		Up 26 minutes
dev-peer0.licenseOrg3.aamd.com-readCC_1.0-bd3ef310267e61ed7a48be30f824f90ff5a4d87f352b4c80fafe72cccae9862		Up 26 minutes
dev-peer0.licenseOrg1.aamd.com-readCC_1.0-bd3ef310267e61ed7a48be30f824f90ff5a4d87f352b4c80fafe72cccae9862		Up 26 minutes
dev-peer0.awardOrg2.aamd.com-readCC_1.0-bd3ef310267e61ed7a48be30f824f90ff5a4d87f352b4c80fafe72cccae9862		Up 26 minutes
dev-peer0.awardOrg3.aamd.com-readCC_1.0-bd3ef310267e61ed7a48be30f824f90ff5a4d87f352b4c80fafe72cccae9862		Up 26 minutes
dev-peer0.awardOrg1.aamd.com-readCC_1.0-bd3ef310267e61ed7a48be30f824f90ff5a4d87f352b4c80fafe72cccae9862		Up 26 minutes
dev-peer0.educationOrg3.aamd.com-readCC_1.0-bd3ef310267e61ed7a48be30f824f90ff5a4d87f352b4c80fafe72cccae9862		Up 26 minutes
dev-peer0.readOrg.aamd.com-readCC_1.0-bd3ef310267e61ed7a48be30f824f90ff5a4d87f352b4c80fafe72cccae9862		Up 26 minutes
dev-peer0.educationOrg2.aamd.com-readCC_1.0-bd3ef310267e61ed7a48be30f824f90ff5a4d87f352b4c80fafe72cccae9862		Up 26 minutes
dev-peer0.educationOrg1.aamd.com-readCC_1.0-bd3ef310267e61ed7a48be30f824f90ff5a4d87f352b4c80fafe72cccae9862		Up 26 minutes
dev-peer0.licenseOrg1.aamd.com-licenseCC_1.0-566700232ecf5ebbbb958ff0fe1f393c926d164b9965fe85d20b5222a47246		Up 27 minutes
dev-peer0.licenseOrg2.aamd.com-licenseCC_1.0-566700232ecf5ebbbb958ff0fe1f393c926d164b9965fe85d20b5222a47246		Up 27 minutes
dev-peer0.readOrg.aamd.com-licenseCC_1.0-566700232ecf5ebbbb958ff0fe1f393c926d164b9965fe85d20b5222a47246		Up 27 minutes
dev-peer0.licenseOrg3.aamd.com-licenseCC_1.0-566700232ecf5ebbbb958ff0fe1f393c926d164b9965fe85d20b5222a47246		Up 27 minutes
dev-peer0.awardOrg3.aamd.com-awardCC_1.0-ac7177513f797745e83b4e64b1ffb468aba9a98146f0993b798369f4bc10f0c		Up 28 minutes
dev-peer0.awardOrg1.aamd.com-awardCC_1.0-ac7177513f797745e83b4e64b1ffb468aba9a98146f0993b798369f4bc10f0c		Up 28 minutes
dev-peer0.readOrg.aamd.com-awardCC_1.0-ac7177513f797745e83b4e64b1ffb468aba9a98146f0993b798369f4bc10f0c		Up 28 minutes
dev-peer0.awardOrg2.aamd.com-awardCC_1.0-ac7177513f797745e83b4e64b1ffb468aba9a98146f0993b798369f4bc10f0c		Up 28 minutes
dev-peer0.readOrg.aamd.com-educationCC_1.0-6aa55d155b04ee5478eadf350f16f2eb1e48bd880684afab74ecaa124532f032		Up 28 minutes
dev-peer0.educationOrg3.aamd.com-educationCC_1.0-6aa55d155b04ee5478eadf350f16f2eb1e48bd880684afab74ecaa124532f032		Up 28 minutes
dev-peer0.educationOrg1.aamd.com-educationCC_1.0-6aa55d155b04ee5478eadf350f16f2eb1e48bd880684afab74ecaa124532f032		Up 28 minutes
dev-peer0.educationOrg2.aamd.com-educationCC_1.0-6aa55d155b04ee5478eadf350f16f2eb1e48bd880684afab74ecaa124532f032		Up 28 minutes
peer0.licenseOrg1.aamd.com	0.0.0.0:8053->8053/tcp, :::8053->8053/tcp, 7051/tcp, 0.0.0.0:18053->18053/tcp, :::18053->18053/tcp	Up 31 minutes
peer0.licenseOrg1.aamd.com	0.0.0.0:7052->7052/tcp, :::7052->7052/tcp, 7051/tcp, 0.0.0.0:17052->17052/tcp, :::17052->17052/tcp	Up 31 minutes
peer0.licenseOrg1.aamd.com	0.0.0.0:8051->8051/tcp, :::8051->8051/tcp, 7051/tcp, 0.0.0.0:18051->18051/tcp, :::18051->18051/tcp	Up 30 minutes
cli		Up 32 minutes
peer0.awardOrg3.aamd.com	0.0.0.0:7053->7053/tcp, :::7053->7053/tcp, 7051/tcp, 0.0.0.0:17053->17053/tcp, :::17053->17053/tcp	Up 31 minutes
peer0.educationOrg2.aamd.com	0.0.0.0:6052->6052/tcp, :::6052->6052/tcp, 0.0.0.0:16052->16052/tcp, :::16052->16052/tcp, 7051/tcp	Up 31 minutes
peer0.awardOrg1.aamd.com	0.0.0.0:7051->7051/tcp, :::7051->7051/tcp, 0.0.0.0:17051->17051/tcp, :::17051->17051/tcp	Up 31 minutes
peer0.educationOrg3.aamd.com	0.0.0.0:6053->6053/tcp, :::6053->6053/tcp, 0.0.0.0:16053->16053/tcp, :::16053->16053/tcp, 7051/tcp	Up 30 minutes
peer0.licenseOrg2.aamd.com	0.0.0.0:8052->8052/tcp, :::8052->8052/tcp, 7051/tcp, 0.0.0.0:18052->18052/tcp, :::18052->18052/tcp	Up 31 minutes
orderer.aamd.com	0.0.0.0:9050->9050/tcp, :::9050->9050/tcp, 7050/tcp, 0.0.0.0:18050->18050/tcp, :::18050->18050/tcp	Up 31 minutes
peer0.educationOrg1.aamd.com	0.0.0.0:6051->6051/tcp, :::6051->6051/tcp, 0.0.0.0:16051->16051/tcp, :::16051->16051/tcp, 7051/tcp	Up 30 minutes
peer0.readOrg.aamd.com	0.0.0.0:5051->5051/tcp, :::5051->5051/tcp, 0.0.0.0:15051->15051/tcp, :::15051->15051/tcp, 7051/tcp	Up 31 minutes
ca_readOrg	0.0.0.0:5055->5055/tcp, :::5055->5055/tcp, 0.0.0.0:15055->15055/tcp, :::15055->15055/tcp, 7054/tcp	Up 32 minutes
ca_licenseOrg1	0.0.0.0:8055->8055/tcp, :::8055->8055/tcp, 7054/tcp, 0.0.0.0:18055->18055/tcp, :::18055->18055/tcp	Up 33 minutes
ca_orderOrg	0.0.0.0:9055->9055/tcp, :::9055->9055/tcp, 7054/tcp, 0.0.0.0:18055->18055/tcp, :::18055->18055/tcp	Up 33 minutes
ca_awardOrg1	0.0.0.0:7055->7055/tcp, :::7055->7055/tcp, 7054/tcp, 0.0.0.0:17055->17055/tcp, :::17055->17055/tcp	Up 32 minutes
ca_educationOrg1	0.0.0.0:6055->6055/tcp, :::6055->6055/tcp, 0.0.0.0:16055->16055/tcp, :::16055->16055/tcp, 7054/tcp	Up 32 minutes

그림 6 - 블록체인 네트워크 구성을 위한 도커 컨테이너가 모두 구성된 모습

총 39개의 Docker 컨테이너가 구성되었으며, 5개의 CA 컨테이너, 10개의 조직 컨테이너, write 전용 체인코드의 승인과 발급을 위한 12개의 컨테이너, read 전용 체인코드의 승인과 발급을 위한 12개의 컨테이너로 구성되어 있다.

2.3. 서버

2.3.1. 웹 애플리케이션 서버

A. SpringBoot

Spring Boot를 사용해 애플리케이션 서버를 구성하였다. 국내 사용자가 가장 많기 때문에 많은 레퍼런스가 존재한다. 따라서 빠른 트러블 슈팅이 가능한 점을 이유로 사용하게 되었다.

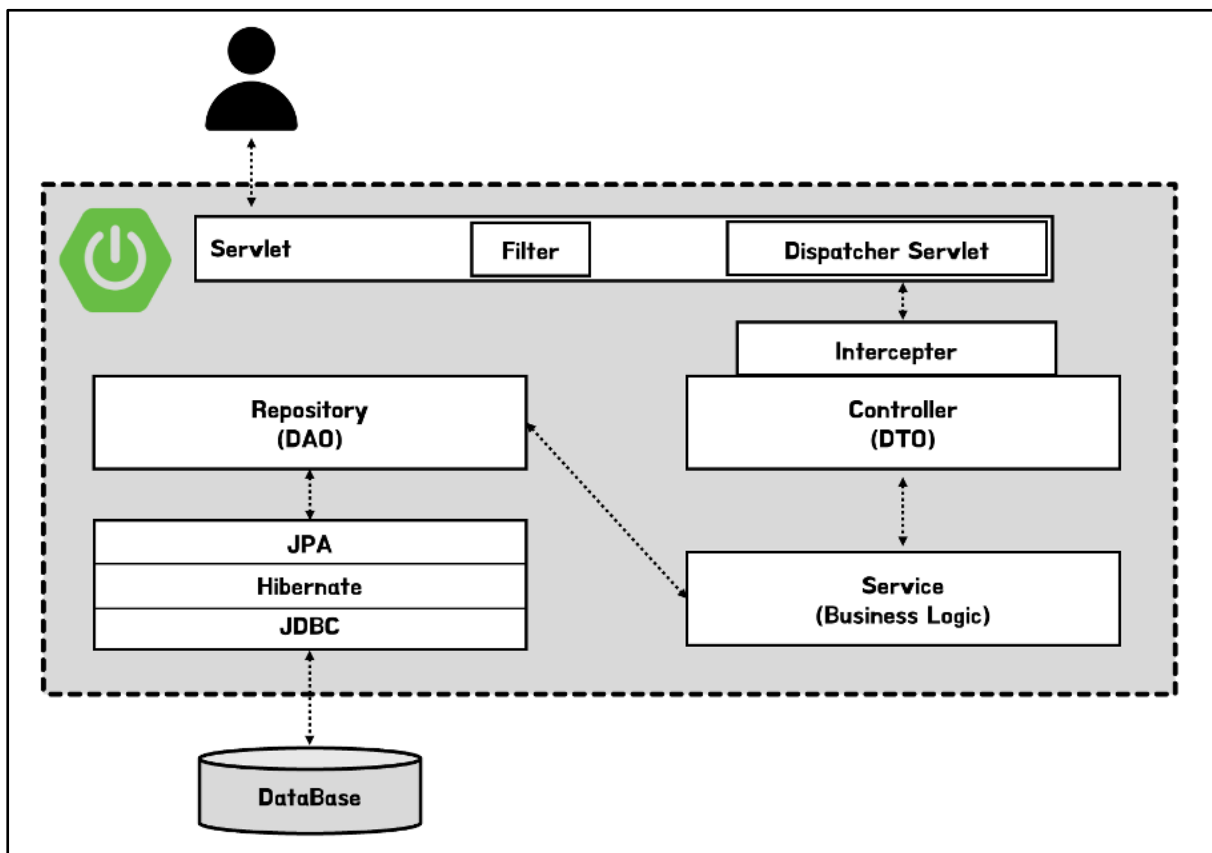


그림 7 - AOMD 서비스 아키텍처

애플리케이션은 크게 3 Layer로 나누어서 개발을 진행했다.

- Presentation Layer : URL에 해당하는 클라이언트의 요청을 받아들이는 계층이다. Controller라고 부르며 HttpRequest 객체를 전달받는다.
- Business Layer : 애플리케이션의 세부 요구사항을 구현하는 계층이다. Service라고 부르며 Controller와 Repository 사이의 연결 역할을 수행한다.
- Persistence Layer : 데이터베이스와의 직접적인 접근을 통해 데이터의 영속성을 관리하는 계층이다. Repository라고 부르며 필요한 데이터를 가져오는 질의를 요청한다.

B. SpringSecurity

Spring Security는 스프링 기반 애플리케이션의 보안(인증과 권한, 인가 등)을 담당하는 스프링 프레임워크이다. Spring Security는 인증과 권한에 대한 부분을 Filter의 흐름에 따라 처리한다. 보안과 관련해서 체계적으로 많은 옵션을 제공해주기 때문에 개발자가 일일이 보안 관련 로직을 작성하지 않아도 된다는 장점이 있다.

C. JWT

JWT 토큰을 사용한 인증을 기반으로 로그인 및 회원가입을 구현했다. JWT는 Json Web Token의 약자로 모바일이나 웹의 사용자 인증을 위해 사용하는 암호화된 토큰을 의미한다. JWT 정보를 request에 담아 사용자의 정보 열람, 수정 등 개인적인 작업들을 수행할 수 있다. JWT는 프론트엔드와 백엔드의 완전한 분리를 도와준다는 점과 유저의 로그인 상태를 브라우저에서 간단하게 저장할 수 있어 서버의 부담이 적다는 장점이 있기 때문에 사용했다.

2.3.2. 인프라

애플리케이션의 개발과 배포를 분리하고 동적인 서버 스케일링을 위해 쿠버네티스에서 동작할 수 있도록 인프라를 구축하였다. 아래 표는 인프라 구축에 사용된 기술들의 버전 정보를 나타낸 것이다.

Docker	20.10.18
K8s	1.23.6+k3s
Nginx	1.21.1
Azure	Ubuntu 20.04 LTS

표 4 - 도커 버전 정보

A. 도커(Docker)

Docker는 컨테이너(Container)를 관리하는 런타임이다. 다양한 프로그램, 실행환경을 컨테이너로 추상화하고 동일한 인터페이스를 제공하여 프로그램의 배포 및 관리를 단순하게 해준다. 백엔드 애플리케이션, 데이터베이스, 메시지 큐 등 어떤 프로그램도 컨테이너로 추상화할 수 있고 조립 PC, AWS, Azure, Google cloud 등 어디에서든 실행할 수 있다.

컨테이너(Container)는 개별 Software의 실행에 필요한 실행환경을 독립적으로 운용할 수 있도록 기반 환경 또는 다른 실행환경과의 간섭을 막고 실행의 독립성을 확보해주는 운영체제 수준의 격리 기술이다. 본 프로젝트에서는 컨테이너 런타임으로 Docker를 사용하였다.

B. 쿠버네티스(Kubernetes)

쿠버네티스는 구글이 개발한 오픈소스 컨테이너 오케스트레이션 도구이다. 각기 다른 배포 환경으로 컨테이너화된 애플리케이션을 관리하는 데 도움을 준다. 특히 애플리케이션이 언제나 사용할 수 있도록 높은 가용성을 보장한다. 퍼포먼스가 중요한 애플리케이션일수록 빠르게 컨테이너를 확장시켜서 응답속도를 보장할 수 있는 높은 확장성을 가진다. 또한 서버가 장애를 겪어서 모든 데이터를 잃을 때 데이터 백업 메커니즘이 활성화되어 가장 최근의 데이터를 불러올 수 있다. 그리고 컨테이너는 회복된 상태에서 아무 일 없었다는 듯이 다시 동작하는 높은 회복성을 가진다.

K3s를 활용하였는데 이는 가벼운 Kubernetes로 쉽게 설치하고 적은 메모리/binary 파일을 사용하여 Edge/IoT 환경 혹은 CI/Dev 환경에서 k8s를 쉽게 사용할 수 있도록 도와주는 도구이다. 쿠버네티스의 핵심 엔진(api-server, kubelet 등)은 동일하기 때문에 K8s를 대체해서 사용이 가능하다.

C. NginX

NginX는 경량화된 소프트웨어 웹 서버이다. Nginx는 Single-thread로 동작하며 비동기 non-blocking I/O 이벤트 기반으로 요청을 처리한다. 따라서 적은 자원으로 효율적인 트래픽 처리가 가능하다. 특징으로 2가지를 꼽을 수 있는데 첫 번째는 HTTP Server로서 정적 파일을 Serve 해준다는 점이다. 클라이언트(유저)로부터 요청받았을 때 WAS를 거치지 않고 요청에 맞는 정적 파일을 응답해주는 HTTP server로써 활용할 수 있다. HTML, CSS

같은 정적인 리소스에 대한 요청을 Nginx가 처리해준다. React의 build 된 파일들도 정적인 리소스라고 볼 수 있고 따라서 Nginx가 index.html 같은 메인 페이지를 렌더링해 줄 수 있다.

두 번째로 Reverse Proxy Server로서 Client와 Server를 중개해준다. Reverse Proxy Server로서 Client의 Request와 Server의 Response를 중개하는 서버로 동작하게 할 수 있다. 이 과정에서 nginx는 로드밸런서의 역할을 수행할 수 있다. 동적으로 계산되거나 전달되어야 하는 사항들은 WAS에게 맡긴다.

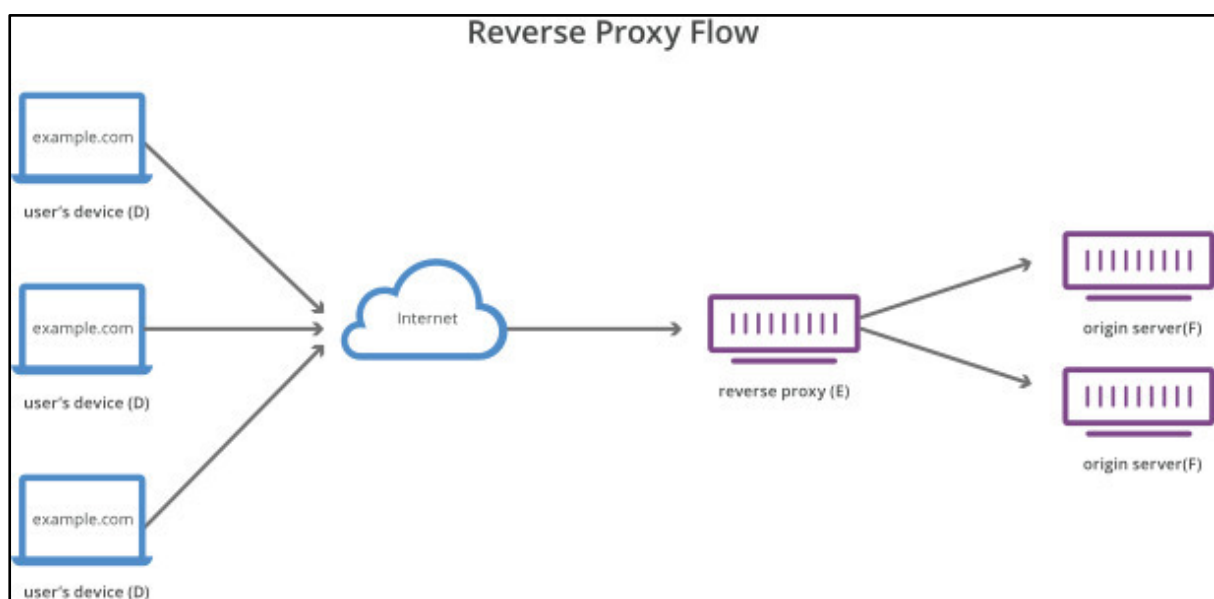


그림 8 - 리버스 프록시 흐름도

클라이언트와 서버 사이에(서버 앞에) 위치하여 보안, 로드 밸런싱 역할을 한다. 그래서 클라이언트가 특정 리소스를 요청하면 프록시 서버가 WAS에 요청 후 응답받은 리소스를 클라이언트에게 전달해주는 개념이다. 예를 들자면 이 방식은 특정 ip주소에서 제공하는 api서버를 호출하기 위해서 인터넷에 있는 클라이언트가 리버스 프록시 서버에 api를 요청하여 응답을 받는 방식이다. 리버스 프록시 서버는 실제 서버가 어디서 동작하는지 감추는 역할을 한다. 클라이언트는 리버스 프록시를 통해서 리소스를 요청하기 때문에 서버의 IP주소를 알 수 없다. 그리고 리버스 프록시 서버는 실제 서버들에 대한 주소를 매핑하고 있어야 한다.

D. 클라우드 서비스

비용 문제로 인해 무료 학생 크레딧을 받을 수 있는 Azure 클라우드 서비스를 이용하였다. Azure를 사용하였는데 이는 전 세계의 Microsoft 데이터 센터에서 응용 프로그램을 빌드, 배포, 관리할 수 있는 유연한 Public 클라우드 플랫폼이다. IaaS(Infrastructure-as-a-Service)라 불리는 서버, 네트워크, 스토리지 등의 인프라 자원을 가상으로 제공해주는 서비스를 이용하였다. 아래의 그림들은 클라우드 서버의 구성을 도식화 한 것이다.




 aomd-master	가상 머신	Azure for Students	aomd	Korea Central
 aomd-worker1	가상 머신	Azure for Students	aomd	Korea Central
 aomd-worker2	가상 머신	Azure for Students	aomd	Korea Central

그림 9 - 서버 구성

노드 이름	사양	운영체제(OS)
Master	Standard_B2s (2core 4GB)	Linux 20.04 LTS
Worker 1	Standard_B1ms (1core 2GB)	Linux 20.04 LTS
Worker 2	Standard_B1ms (1core 2GB)	Linux 20.04 LTS

표 5 - 서버 구성

또한 추가적인 비용 절감 및 효율적인 리소스 사용을 위해서 Azure에서 제공하는 AKS(Azure Kubernetes Service)를 사용하지 않고 K3s를 사용해 직접 쿠버네티스 클러스터를 구성하였다. 아래의 4가지 순서는 클러스터를 구성하는 과정을 나열한 것이다.

1. 가상 네트워크(vnet) 구성 : 10.0.0.0/16의 private IP 공간을 가지는 가상 네트워크를 구성한다.
2. 네트워크 보안 설정 : 사용할 포트(80, 443, 22 ... etc)에 대한 인바운드 규칙을 설정한다.
3. 가상 머신 생성 : Master, Worker 1, Worker 2 노드를 이전에 생성한 가상 네트워크 위

에 생성한다.

4. 각 노드에 K3s를 설치하고 Master와 Worker 관계를 구성한다.

E. Github Action

빌드 서버가 따로 존재하지 않는 점과 배포 서버에서 빌드 시 리소스 소모가 크다는 점을 이유로 Github Actions를 사용하게 되었다. Github Action은 repository가 공개되어 있으면 일부 제한된 기능을 무제한 사용할 수 있다. 빌드 과정을 실시간으로 확인할 수 있고, 빌드 로그 목록을 쉽게 볼 수 있다. 그리고 빌드과정의 오류도 파이프라인 구축 지식 없이 쉽게 확인할 수 있다.

F. ArgoCD

배포를 위해서 GitOps 방법론을 사용하였고 이를 올바르게 사용하기 위해 4가지의 원칙을 고수하였다. 먼저 선언형 배포 작업 정의서 (YAML) 생성하였다. 배포 방식을 명령형으로 정의된 것이 아니라 선언형으로 정의했다. 사용자가 바라는 상태(desired state)를 선언적으로 Git에 정의하면 그것이 쿠버네티스에 반영되도록 한다.

두 번째로 Git을 이용해 배포 버전을 관리했다. Git에 모든 배포에 관련된 정보가 정의되어 있어야 하며, 각 버전이 Git 저장소에 기록이 되어 있어야 한다. 이를 통해 개발자가 쉽게 예전 버전으로 롤백하거나 새로운 버전으로 업그레이드를 할 수 있도록 했다.

세 번째로 변경 사항 반영을 자동화했다. 사용자가 Git 저장소에 선언형 정 의서를 선언하면 실제 배포는 자동으로 이루어져야 한다. 따라서 이것을 책임지는 GitOps 구현체는 ArgoCD와 같은 도구를 사용하였고, 자동화를 통해 인적 오류를 줄이고 지속적 배포를 가능하게 했다.

마지막으로 이상 탐지 및 자가 치유를 적용하였다. GitOps 구현체는 YAML 파일을 클러스터에 반영하는 것뿐만 아니라 배포된 리소스가 이상이 없는지 확인하고 유지하는 역할도 담당할 수 있도록 했다. 결과적으로 이러한 원칙(GitOps)을 가지는 배포 소프트웨어 중 하나인 ArgoCD를 사용하여 지속적 배포(CD : Continuous Deploy)를 구현하였다.

쿠버네티스에 배포된 애플리케이션이 Git 저장소와 Sync된 상태임을 보여준다. 하나의 애플리케이션을 배포하기 위해서 생성된 다양한 쿠버네티스 컴포넌트 간의 연결 관계를 보여준다. 배포 현황은 아래의 그림과 같다.

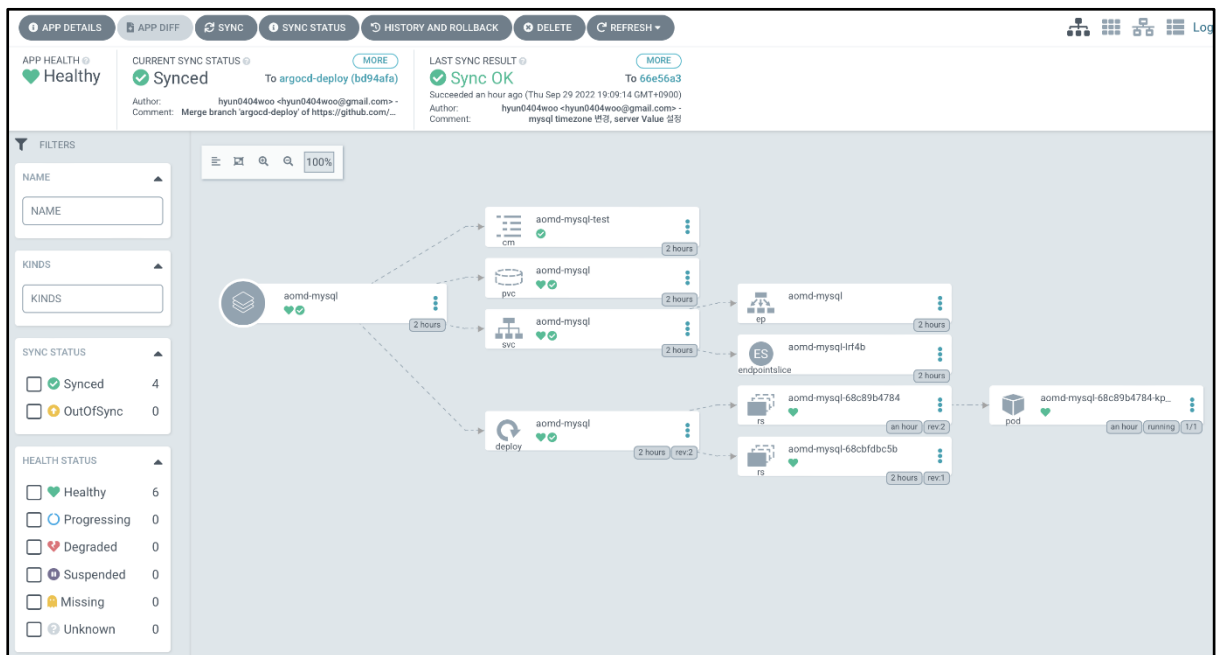


그림 10 - MySQL 배포 현황

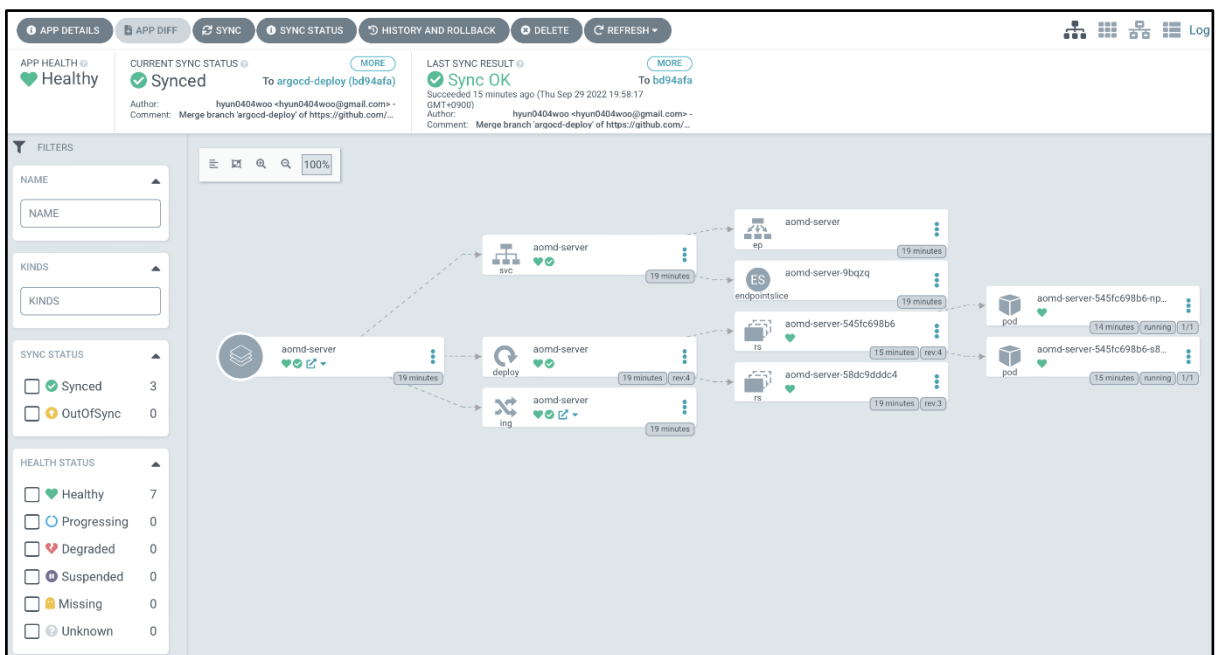


그림 11 - 웹 어플리케이션 서버 배포 현황

G. Helm

Helm은 쿠버네티스 패키지 매니저이다. apt, yum, pip 툴과 비슷하게 플랫폼의 패키지를 관리한다. 개발한 서버 애플리케이션을 가동하기 위해 필요한 모든 요소를 선언해 놓을 수 있다.

2.3.3. 시스템 아키텍처

A. Request Flow

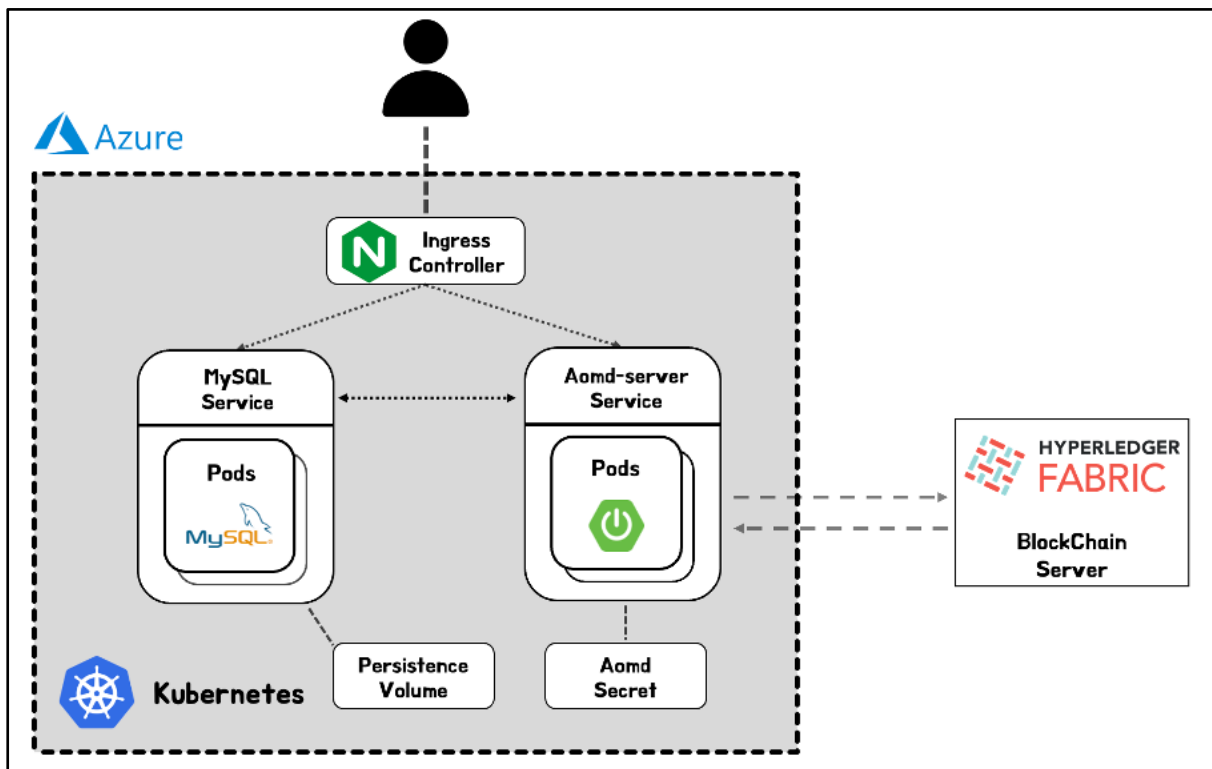


그림 12 - 인프라 요청 흐름도 (Request Flow)

B. CI/CD 파이프라인

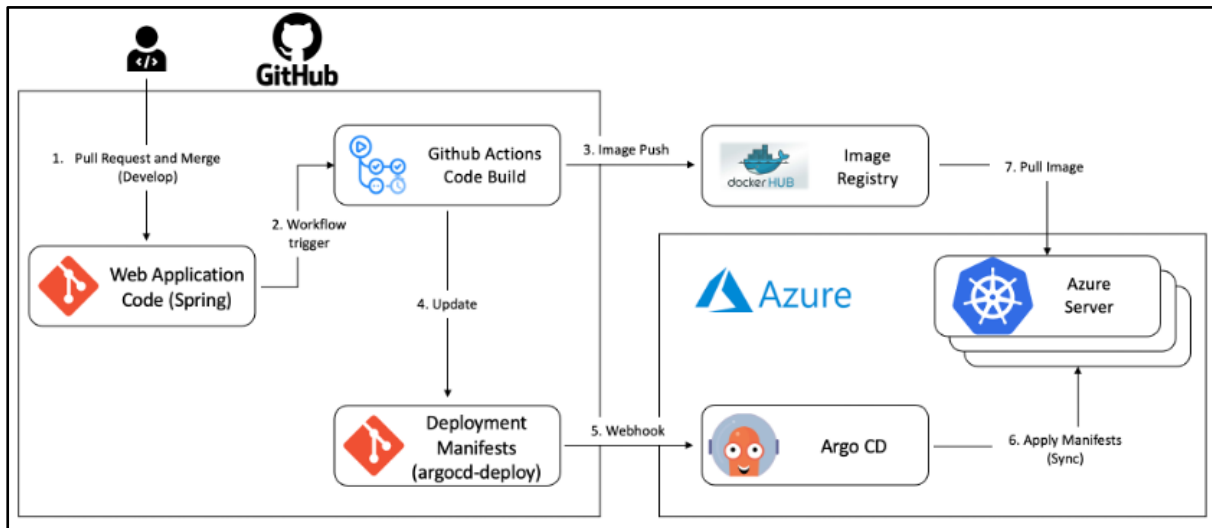


그림 13 - 지속적 통합 및 배포 흐름도 (CI/CD 파이프라인)

2.4. 프론트엔드

2.4.1. React

제안하는 시스템의 프론트엔드 사이드 전반적인 부분은 리액트 프레임워크를 기반으로 설계되었다. 프로젝트 규모가 커지면서 다양한 유저 인터랙션이 전달된다면 그만큼 DOM 요소들 또한 변화가 이루어져야 한다. 본 플랫폼은 서버에 정의된 API로부터 학력과 자격증을 비롯한 자기 정보 데이터들을 조회하는 통신이 빈번히 발생하고 포트폴리오 생성 및 수정 등의 다양한 유저 인터랙션이 요구되기 때문에 DOM 요소들이 자주 변화하게 되고 그에 따라 브라우저가 많은 연산을 수행하게 된다. 이러한 과정이 반복된다면 전체적인 프로세스의 비효율성을 야기한다. 프론트엔드 프레임워크는 DOM 관리와 상태 변화 관리를 최소화하여 개발자가 오직 기능과 사용자 인터페이스 개발에 집중할 수 있도록 도와준다는 이점이 있다.

또한 리액트 프레임워크는 컴포넌트를 단위로 한 코드 작성을 통해 생산성과 유지 보수를 용이하게 한다. 하나의 요소가 변화함에 따라 다른 요소들에게까지 영향을 미치는 복잡한 로직을 업데이트하는 까다로운 작업의 경우 컴포넌트의 재사용 기능으로서 이를 보완할 수 있다. 각각의 컴포넌트는 자바스크립트의 확장 구문인 JSX 문법을 통해 손쉽게 구성할 수 있다. 또한 사용자의 인터랙션에 의해 상태 변화가 일어나면 브라우저 작동

원리에 의해 렌더링 과정을 반복하게 되는데 이때 비효율성을 최소화하기 위해 Virtual DOM을 사용한다. 유저 인터랙션에 의해 변화가 발생하여 10개의 노드를 수정해야 할 때 실제 DOM에 바로 적용하는 것이 아니라 Virtual DOM에 먼저 변경 사항을 적용해보고 일련의 연산이 끝나고 나면 최종적인 변화를 실제 DOM에 전달하는 방식이다. Virtual DOM은 이러한 과정에서 어떤 부분이 변경되었는지 종합적으로 파악하여 필요한 부분만을 실제 DOM 트리에 변경 및 적용할 수 있게 하여 렌더링 과정에서의 효율적인 렌더링 속도에 이점이 있다.

2.4.2. Redux

리덕스는 애플리케이션의 상태 정보를 효율적으로 관리하기 위한 오픈소스 자바스크립트 라이브러리이다. 복잡한 상태 관리가 이루어지는 단일 페이지 애플리케이션인 SPA(Single Page Application)에서 특히 유용하게 사용된다. 블록체인과 서버의 통신을 통해 발급된 사용자 자기 정보 데이터들을 비롯한 개인 정보들은 포트폴리오를 생성 및 수정하고 본인 인증 과정을 위해 프론트엔드 사이드에서 넘겨받게 된다. 이때 해당 정보들은 프론트엔드 애플리케이션 내부에서 상태(State)의 형태로 관리된다. 단일 페이지 애플리케이션의 경우 이를 구성하고 있는 컴포넌트 간의 데이터 교류가 복잡하여 이를 효율적으로 관리할 방법이 필수적으로 요구되는데 리덕스는 이러한 복잡한 상태 관리를 효율적으로 할 수 있게 해주는 이점이 있다.

2.4.3. REST API

REST API는 두 컴퓨터 시스템이 인터넷을 통해 정보를 안전하게 교환하기 위해 사용하는 인터페이스이다. 본 시스템은 블록체인과 서버 및 클라이언트 간의 통신을 REST API를 통해 구현하였다. 이는 안전하고 신뢰할 수 있으며 특히 클라이언트와 서버의 상호작용을 최적화한다. REST를 활용한 웹 서비스는 완전하게 클라이언트와 서버를 분리하여 지원하는데 서버 사이드의 기술 변경이 클라이언트 사이드에 영향을 주지 않아 각 부분이 독립적으로 발전할 수 있도록 다양한 서버 구성 요소를 단순화하고 분리한다. 또한 REST API는 사용되는 기술과 독립적이어서 API 설계에 영향을 주지 않으며 다양한 프로그래밍 언어로 클라이언트 및 서버 애플리케이션을 설계할 수 있다.

2.4.4. 페이지 라우팅 구조

제안하는 플랫폼은 전체 페이지를 하나의 페이지에 담아 동적으로 화면을 바꿔가며 표현하는 단일 페이지 애플리케이션이다. 아래의 그림은 플랫폼의 주요 페이지 간의 라우팅 구조를 도식화 한 것이다.

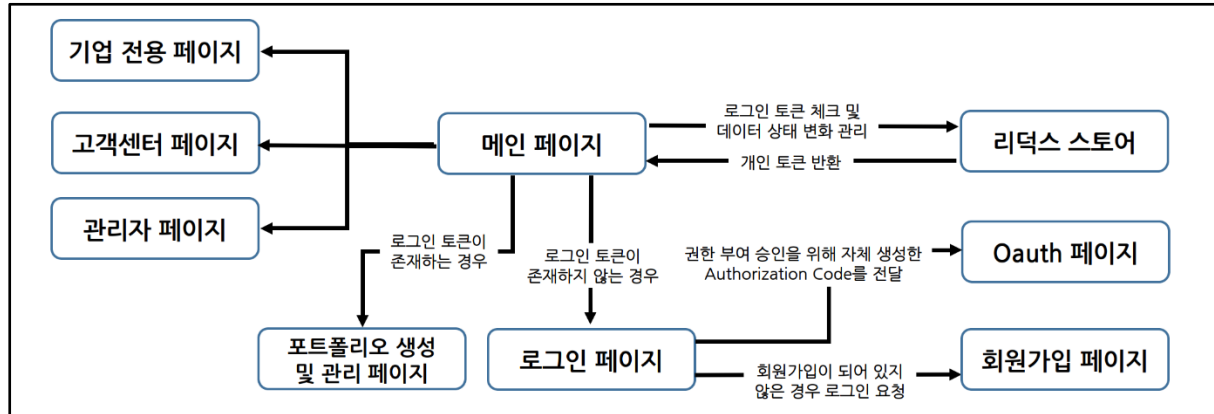


그림 14 - 페이지 라우팅 구조

2.4.5. 드래그 앤 드랍 컨텍스트 구조

제안하는 플랫폼은 최상의 사용자 경험을 제공하기 위하여 드래그 앤 드랍(DND, Drag and Drop) 기능을 적용하였다. 아래의 그림은 드래그 앤 드랍 컨텍스트의 구조를 도식화 한 것이다.

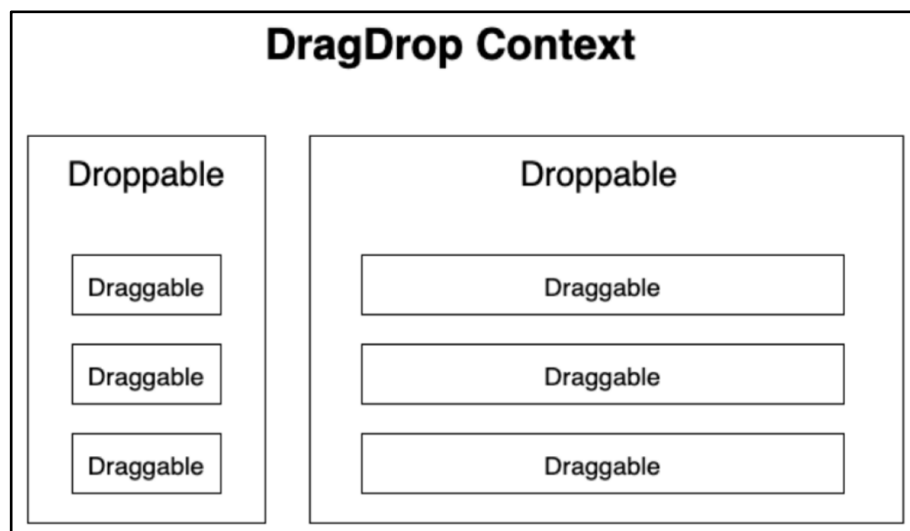


그림 15 - 드래그 앤 드랍 컨텍스트 구조 (DND Context)

2.4.6. 리덕스 통신 구조

제안하는 플랫폼은 리덕스 라이브러리를 통해 애플리케이션의 데이터 상태 변화를 관리한다. 아래의 그림은 리덕스 통신의 프로세스 구조를 도식화 한 것이다.

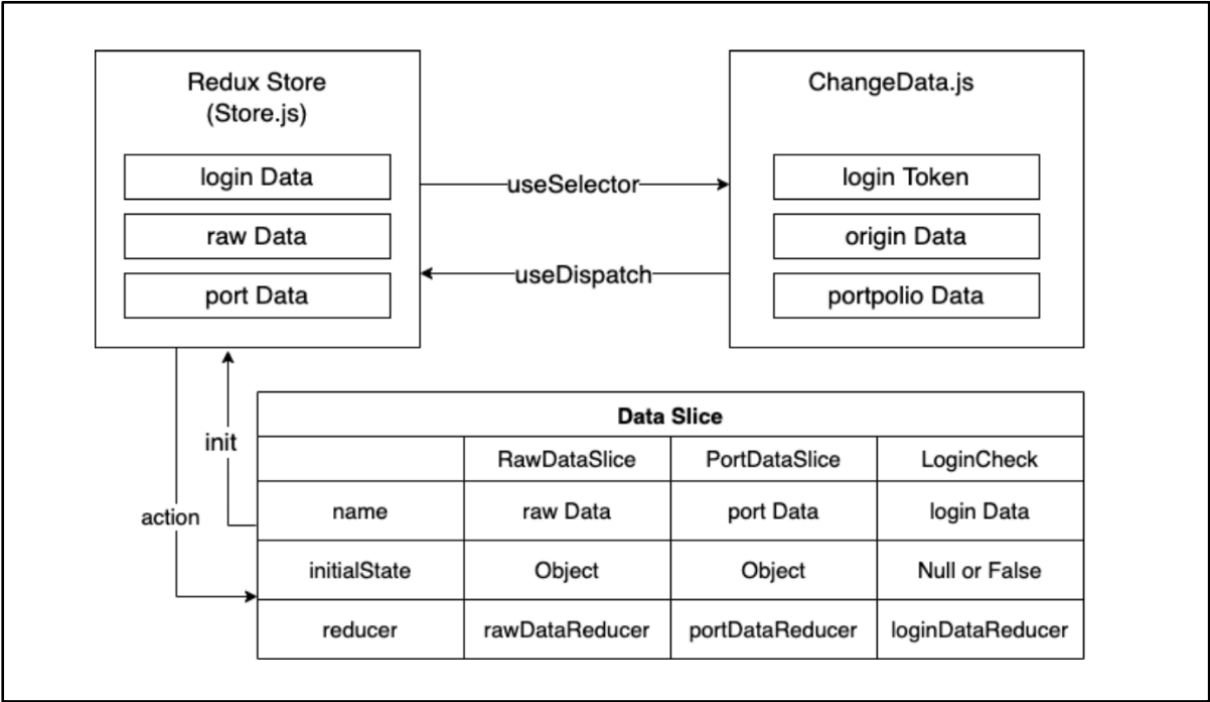


그림 16 - 리덕스 통신 구조

2.5. 전체 시나리오

2.5.1. 프로세스 다이어그램

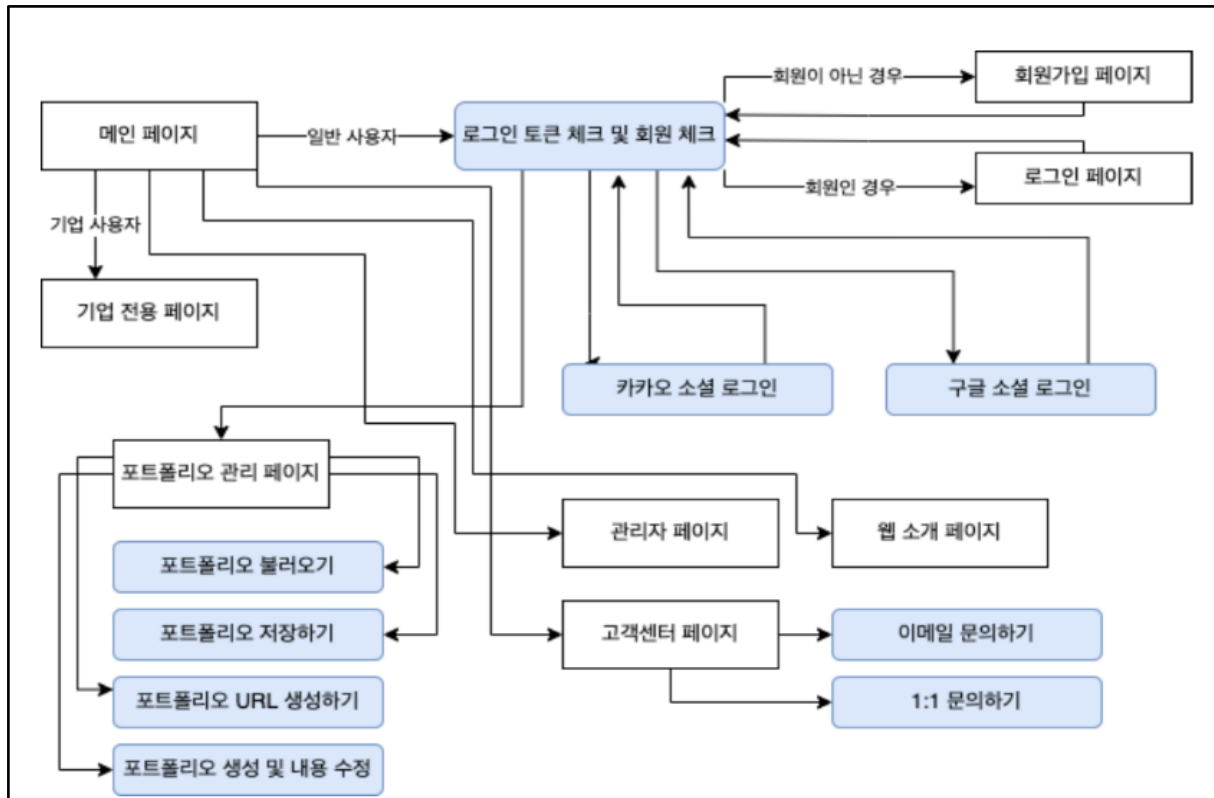


그림 17 - 프로세스 다이어그램

2.5.2. 플로우 차트

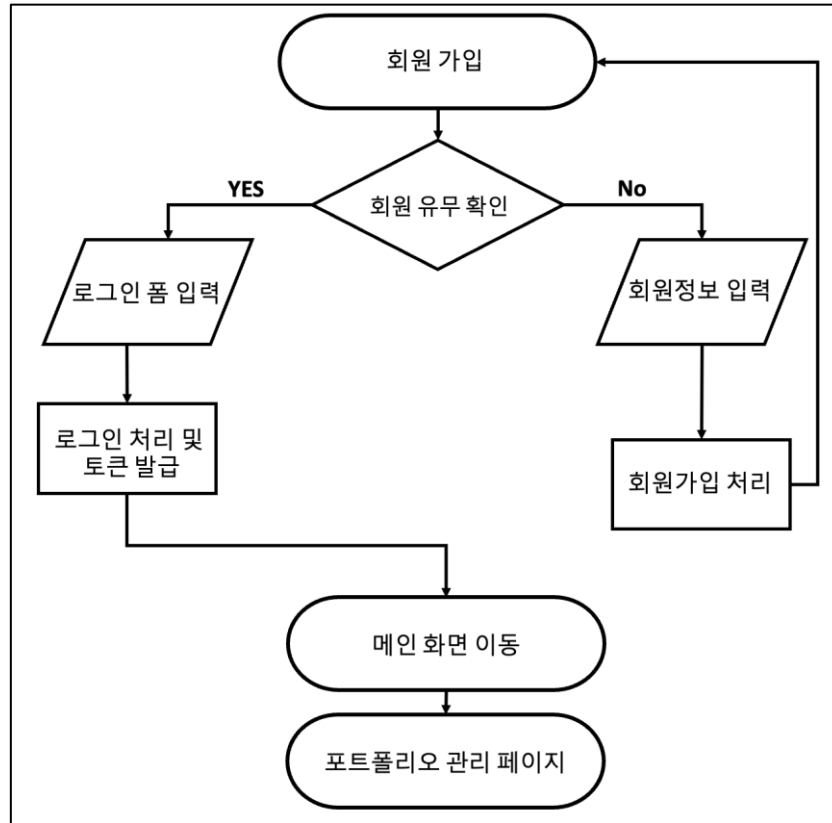


그림 18 - 회원가입 및 로그인 플로우차트

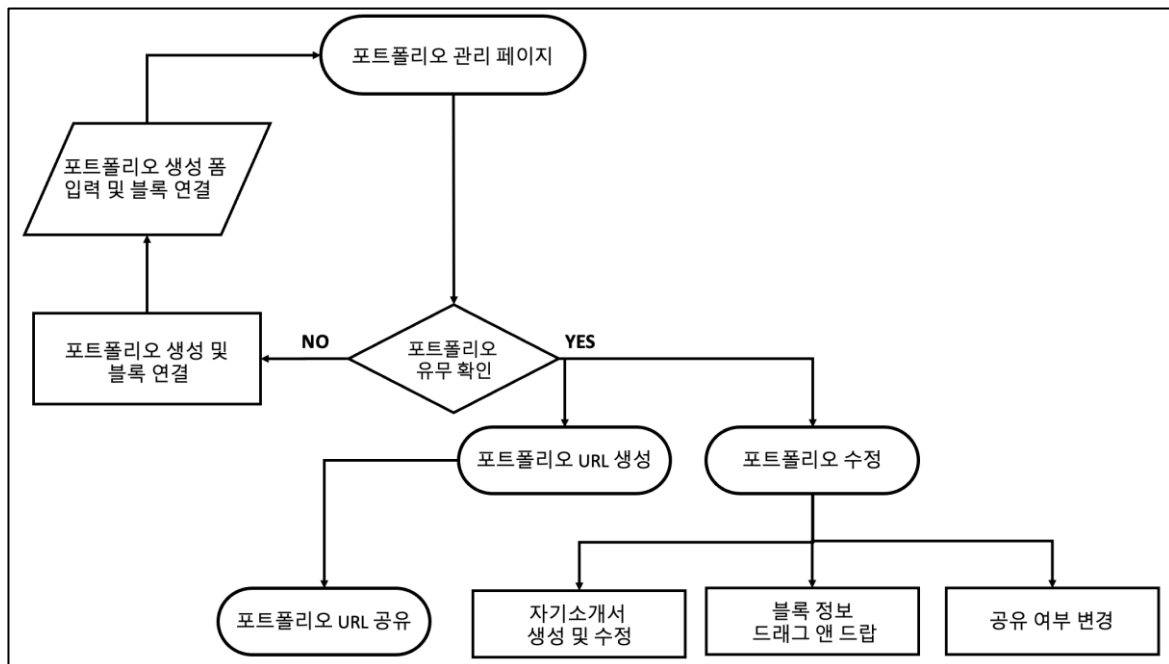


그림 19 - 포트폴리오 관리 플로우차트

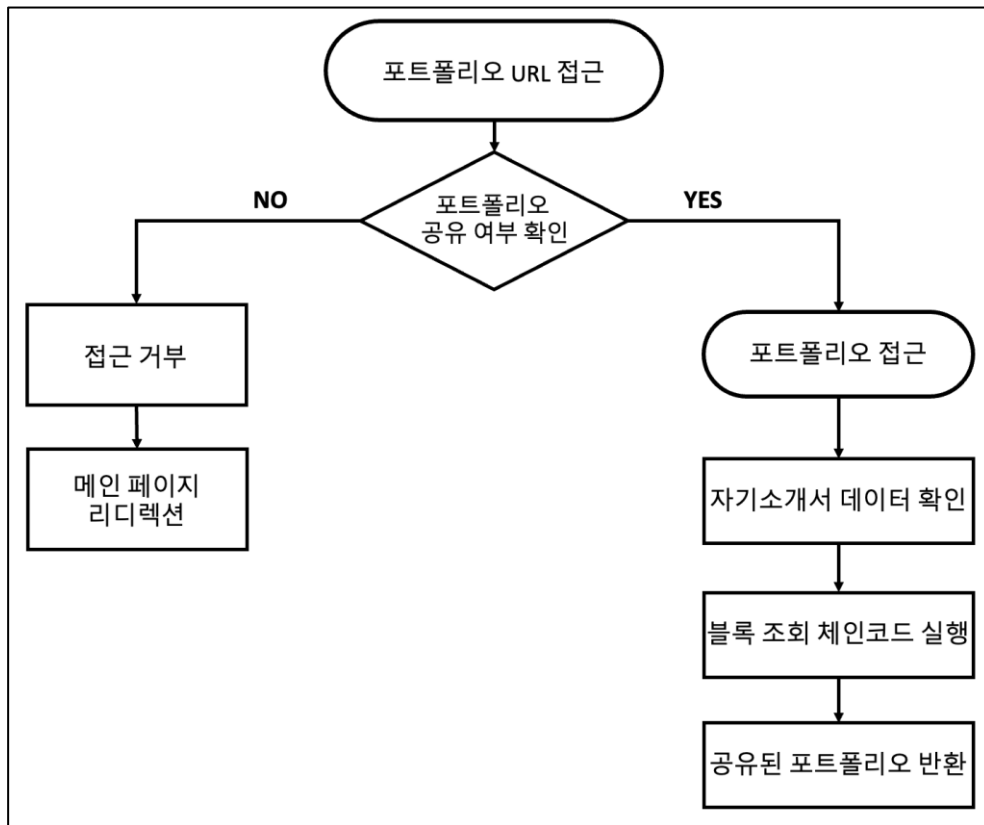


그림 20 - 포트폴리오 조회 플로우차트

2.6. 시스템 구성도

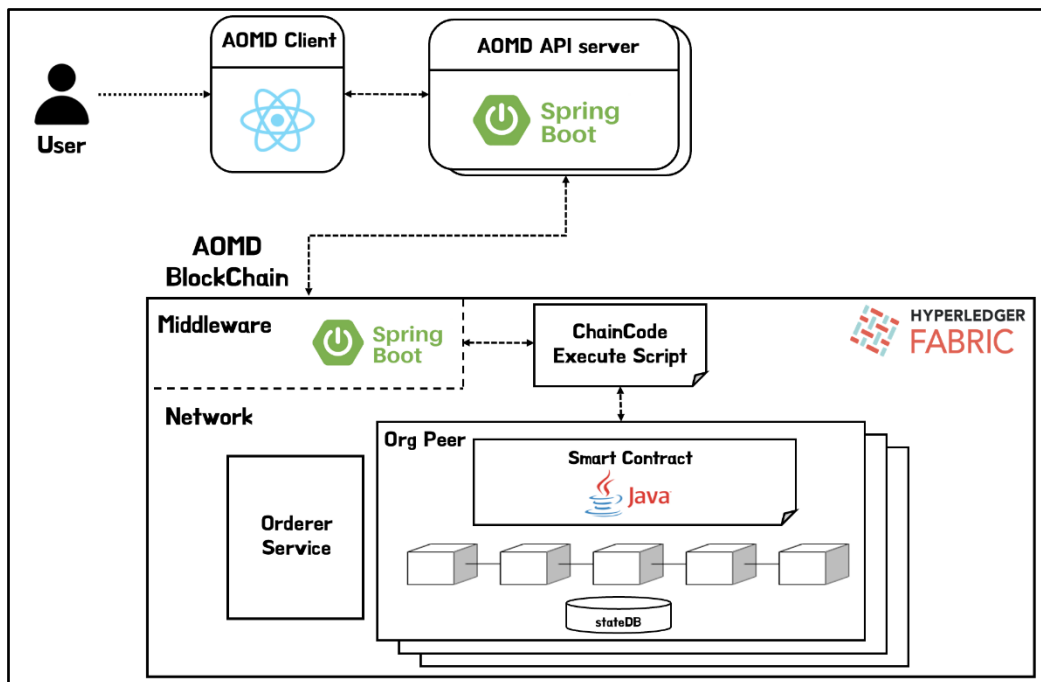


그림 21 - 시스템 구성도

3. 연구 결과 분석 및 평가

3.1. 설계 변경 내역 및 한계점

3.1.1. JVM의 메모리 사용량으로 인한 네트워크 고도화의 한계

Hyperledger Fabric에서 Go, Javascript, Typescript 그리고 Java로 체인코드를 제공한다. 그중 Server에서 사용된 Spring과 언어의 통일과 개발자의 언어 스펙을 고려하여 Java로 체인코드를 작성하였다. 문제는, Java로 작성된 체인코드가 하나의 컨테이너 위에서 모두 실행되는 것이 아니라, {조직 * 체인코드 수} 만큼 컨테이너가 생성된다는 점이다. 각 컨테이너에 JVM이 올라간 채로 실행되기 때문에 메모리 사용량이 다른 언어에 비해 상당히 높다. 하나의 체인코드 컨테이너당 약 480MB의 메모리공간을 사용하기 때문에 더 많은 수의 네트워크 엔티티를 구성하는 것에 한계가 있었다.

3.1.2. Application SDK 연결 실패

Hyperledger Fabric에서는 블록체인 네트워크와 통신을 하기 위한 Application SDK를 지원한다. Go, Javascript, Java 언어를 지원하며, 스크립트 파일이 아닌 해당 SDK를 통해 유저를 생성하고, 등록하는 작업을 가능하게 한다. 그뿐만 아니라 SDK를 통해 외부에서 체인코드를 실행하여 스마트컨트랙트를 발행할 수 있고, 비동기적으로 트랜잭션을 listening 하는 등 여러 가지 기능을 제공한다. 이러한 기능을 통해 Docker Container를 통해 내부적으로 구현되었던 조직들을 외부 서버(비즈니스 모델에서 이력 발급기관)에 설치하여 블록체인 네트워크를 쉽게 구성할 수 있도록 지원한다.

마찬가지로 언어의 통일을 위해 Java Application SDK를 설치하여 외부에서 블록체인 네트워크에 연결하고, 스마트컨트랙트를 생성하는 코드를 작성하고 테스트해보았다. 하지만 블록체인 네트워크에 연결하는 과정에서 HTTP 프로토콜이 아닌 gPRC 프로토콜을 통해 연결하게 되는데, 연결에 실패하였다. 포트 확인부터 처음부터 블록체인 네트워크를 다시 구성하는 등 여러 시도를 해보았지만 모두 실패했다.

임시방편으로 체인코드를 통해 스마트컨트랙트를 생성하는 쉘스크립트를 작성하고, 해당 쉘 스크립트를 외부에서 실행할 수 있도록 중계하는 Middleware를 Spring으로 구현하였다. 해당 방법으로 Application SDK를 완전히 대체할 수 있지만, 네트워크와 로직의 복잡도가 증가하고 SDK의 기능을 쉘 스크립트로 구현하게 됨으로써 개발기간이 연장되었다.

3.2. AI-블록체인 아이디어 경진대회

3.2.1. 개요

제안하는 플랫폼을 비즈니스적인 측면에서 사업화를 위한 방향으로 발전시켜 '블록체인을 활용한 포트폴리오 관리 플랫폼'에서 '블록체인을 활용한 포트폴리오 거래 플랫폼'이라는 새로운 아이디어를 고안해냈고 이를 기획 배경, 차별점 및 개선 가능성, 구현 가능성, 사업화 추진 계획 등으로 일목요연하게 정리하여 AI-블록체인 아이디어 경진대회에 제출했다. 해당 대회에 관해 학과 내부에서 많은 참여를 독려하며 홍보하기도 했고 현재 진행하고 있는 졸업과제의 제안하는 플랫폼과 중복되는 부분이 상당히 많다고 판단되었기에 나름의 유의미한 결과를 도출해 낼 수 있을 거로 생각했다.

3.2.2. 아이디어 기획 및 설계 멘토링 진행

포트폴리오 관리라는 핵심 키워드에서 비즈니스 로직을 접목할 수 있는 방향으로 '거래'라는 키워드를 추천해주셨고 플랫폼 사용 대상을 명확히 하여 아이디어를 보다 구체화할 수 있었다. 멘토님은 제안하고자 하는 시스템을 비즈니스 모델을 통하여 사업성 및 수익성 등의 완전히 새로운 관점에서 바라볼 수 있게 해주셨고 기존에 존재하는 플랫폼들과는 다른 확실한 차별점의 유무를 특히 강조하셨다. 해당 부분은 원래 하고자 했던 주제인 블록체인을 활용한 포트폴리오 관리 플랫폼을 설계 및 기획하는 단계에서도 긍정적인 영향을 미쳤다.

3.2.3. 회고

해당 대회를 통해 제안하는 플랫폼을 졸업과제의 측면이나 현재 상황에 대한 개선점 연구가 아닌 비즈니스적인 측면으로 혹은 사업화 아이디어로써 바라보고 이를 조금 더 발전시켜 새로운 아이디어를 고안해냈다. 제안하는 아이디어의 출발점이 졸업과제 및 연구이니만큼 프로젝트를 기획 및 설계하는 단계에서 자연스럽게 현재 상태에 대한 문제점 해결이나 구현 가능성에 대한 고민에 많은 시간을 투자하였다. 이러한 고민이 아이디어 경진대회의 취지에 안 맞지 않겠느냐고 우려도 있었지만, 기획이 어느 정도 구체화하고 이를 실제로 구현하는 단계에 접어들면서 프로젝트 초기에 했었던 고민은 오히려 팀원들을 독려하고 생산성을 향상했다. 그 결과 우수상이라는 좋은 결과를 만들어낼 수 있었다.

3.3. 사용자 설문조사 결과 분석

사전에 만들어진 제안하는 플랫폼의 프로토타입을 활용하여 사용자 설문조사를 진행했다. 설문 대상은 취업을 준비하는 학부생들로 구성되어 있으며 연령대는 20대 중반에서 후반까지 매우 다양했다. 포트폴리오 관리 플랫폼의 필요성과 요구사항 등을 비롯하여 해당 플랫폼의 프로토타입을 직접 사용해보고 난 후 느낀 점 등을 작성하도록 했다. 아래는 설문조사 결과에 대한 분석 내용이다.

3.3.1. 포트폴리오 관리 플랫폼에 대한 배경 지식

설문 조사 첫 번째 질문으로 기존에 서비스되는 포트폴리오 관리 플랫폼에 대한 배경 지식을 조사했다. 그 결과 취업을 준비하는 4학년 학부생들의 대부분이 현재 운영되고 있는 포트폴리오 관리 플랫폼에 대해 모르고 있었다. 이는 서론에 언급했던 문제점 중 하나인 해당 플랫폼들이 교육 기관과 기업들만을 대상으로 서비스하고 있기 때문에 일반 사용자가 접하기 어렵기 때문이라고 해석할 수 있다. 두 번째 질문으로는 포트폴리오를 작성해본 경험 또는 취업을 준비해본 경험이 있는지를 조사했다. 아래는 각각의 결과를 나타낸 그래프이다.

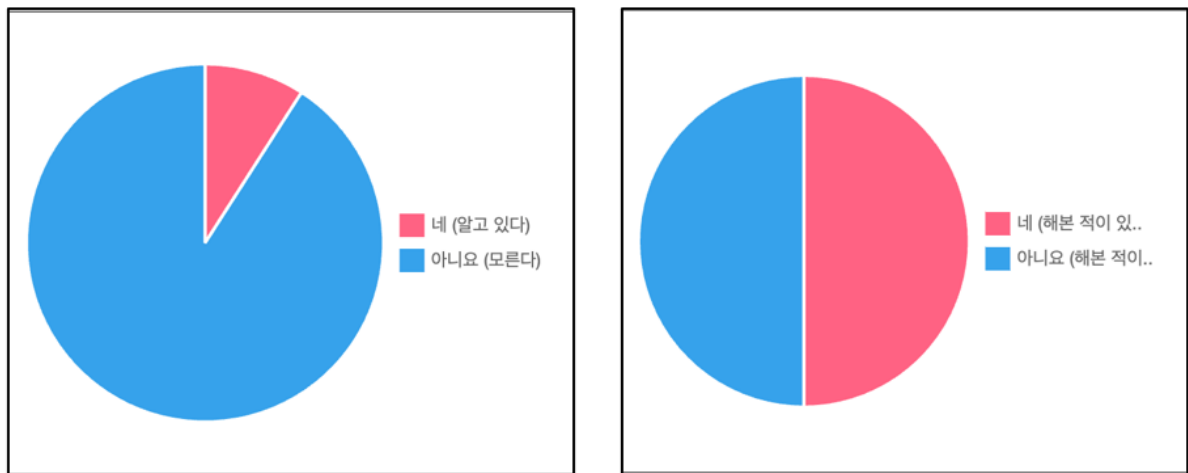


그림 22 - “세종텔레콤 스마트 학사 정보 관리(SER)와
물리다 직무학습이력 관리 플랫폼에 대해 알고 계십니까?” 에 대한 답변 (좌),
“포트폴리오를 작성한 경한 경험 또는 취업을 준비해본 경험이 있나요?” 에 대한 답변 (우)

설문 대상이 현재 4학년 학부생임을 고려했을 때 대부분 답변자가 포트폴리오를 작성하거나 취업을 준비 해봤을 거라고 예상했지만 예상과는 다르게 설문 결과는 각각 절반의 확률을 보여줬다.

위의 질문에서 포트폴리오를 작성한 경험 또는 취업을 준비해본 경험이 있다는 답변자를 대상으로 추가 질문을 했다. 질문 내용은 “경험이 있다면 불편했던 점이 있나요? 혹시 있다면 특히 어떤 점이 불편했나요?”였다. 이에 대한 답변은 다양했고 답변자들의 경험에서 비롯된 사실이므로 상세했다. 그중 몇 가지를 선별하여 아래의 표에 나열하였다.

1. 포트폴리오에 어떤 내용이 들어가야 하는지 잘 모르겠다.
2. 어떤 템플릿이 좋은 템플릿인지를 잘 모르겠다
3. 인터넷에 무단으로 배포될까 걱정된다.
4. 다양한 회사에 포트폴리오를 제출해야 하는 경우, 제출할 때마다 포트폴리오를 새롭게 작성해야하는 점이 불편했다.
5. 내 경력 정보들을 한번에 관리할 수 있는 방법이 없다는 점이 불편했다.
6. 포트폴리오를 작성하고 다시 찾아보기가 힘들었다.
7. 워크넷 사람인 등 취업 사이트마다 입력해야 해서 번거로웠다. 그리고 개인정보들이 빠져나갈까 노심초사하며 잠을 한 숨도 못 잤던 적도 있다.

표 5 - “경험이 있다면 불편했던 점이 있나요?
혹시 있다면 특히 어떤 점이 불편했나요?”에 대한 답변

세 번째는 “플랫폼을 통해 개인의 포트폴리오를 직접 편리하게 관리하고, 이를 블록체인을 통해 안전하게 보관할 수 있다면 해당 플랫폼을 사용할 의향이 있으십니까?”라는 질문으로 제안하는 플랫폼의 주 사용층이 될 취업준비생 및 학부생들에게 포트폴리오 관리 플랫폼에 대한 필요성을 조사했다. 아래의 그래프는 해당 질문에 대한 내용을 막대그래프로 시각화한 것이다.

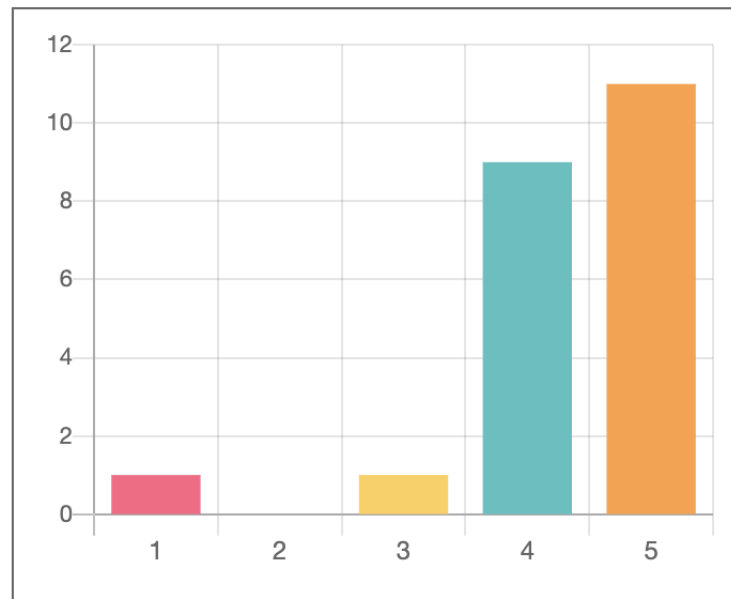


그림 23 - "플랫폼을 통해 개인의 포트폴리오를 직접 편리하게 관리하고, 이를 블록체인을 통해 안전하게 보관할 수 있다면 해당 플랫폼을 사용할 의향이 있으십니까?" 에 대한 답변

답변 5인 "네(쓴다)"라는 답변이 가장 많이 나왔다. 이를 통해 플랫폼의 주 사용층이 될 취업준비생 및 학부생들의 수요를 확인하였다.

3.3.2. AOMD 플랫폼 만족도 평가 및 피드백

A. 만족도 평가

현재 AOMD 플랫폼은 AWS를 통해 배포 되어있는 상태이다. 다섯 번째 질문으로 해당 플랫폼에 대한 전반적인 만족도를 조사했다. 결과는 예상보다 훨씬 긍정적이었다. 아래는 해당 질문의 설문 결과를 시각화한 막대그래프다.

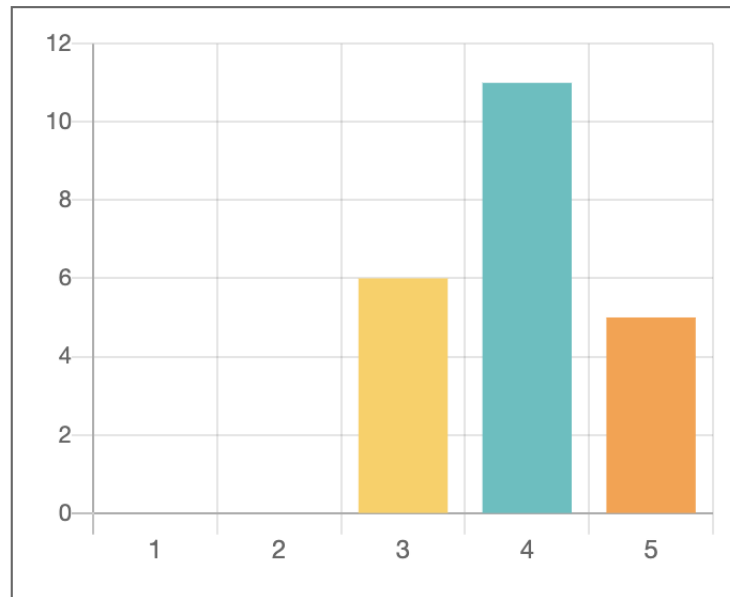


그림 24 - AOMD 플랫폼 만족도 평가 결과

B. 특히 좋았던 점

AOMD 플랫폼에 대한 만족도 평가 후 다음으로 특히 좋았던(만족스러웠던) 점을 서술 형으로 응답 받았다. 다양한 응답들이 있었지만, 그 중 몇 가지를 선별하여 아래의 표에 나열하였다.

1. 이력서를 넣으면서 해당 직무에 맞게 포트폴리오 혹은 자기소개서를 수정해야하는 일이 빈번한데 개인적인 스펙이나 인적사항 등을 드래그 앤드랍으로 빠르게 작성하는 부분이 좋았다.

2. 데이터 쉽게 추가하고 삭제할 수 있어서 편했다. 그리고 각 기업마다 요구하는 자기소개서가 다른데 포트폴리오 여러 개를 한 사이트에서 관리할 수 있는게 편리한 것 같다.
3. 초보자도 사용하기 쉽게 만들었다. 편리하다.
4. 디자인과 부드러운 UI, 드래그 앤 드랍 기능으로 간편하게 포트폴리오를 관리할 수 있는 점이 좋다.
5. 자신의 경력이나 자격, 학력을 한눈에 볼 수 있어서 좋다
6. UI/UX 가 특히 사용하기에 매우 편리하게 구성되어 있었다. 한 눈에 보기 편했고, 페이지가 깔끔하다.

표 6 - 특히 만족스러운(마음에 들었던) 부분이 있다면 어떤 부분입니까? 에 대한 답변

C. 특히 불편했던 점

추가로 특히 불편했던 점을 서술형으로 응답 받았다. 응답들은 향후 연구 방향에 최대한으로 적용할 예정이다. 다양한 응답들이 있었지만, 그 중 몇 가지를 선별하여 아래의 표에 나열하였다.

1. 처음 포트폴리오를 작성하는 사람들을 위한 기본 템플릿이 있으면 좋을 것 같다.
2. Github 같은 것들과 연동하여 간편하게 프로젝트 정보들을 볼 수 있으면 좋을 것 같다.
3. 눈에 크게 보이는 단점은 없지만 굳이 뽑자면 페이지가 깔끔한 만큼 심플하다.
4. 공인 인증 교육기관(교육부, 국가 기관 인턴 활동,etc...)에서 발급된 내용들은 바로 왼쪽에 노출이 되지만 비공인 교육기관이라든지 사설 기관에서 발급된 자격증이나 교육 수수료 관련 내용은 하나하나 협력을 맺어서 하기엔 약간의 어려움이 보인다.

5. 사용자의 개인 정보 보안이라는 부분을 가장 신경을 쓴 것 같지만 다른 외부 어플리케이션을 통한 캡처 등의 문제에는 아직 미흡한 모습을 보인다.
6. 로그인이 필요한 경우에 바로 로그인 화면으로 옮겨 졌으면 좋겠다.
7. 네이버로 소셜 로그인이 안되는 부분이 불편했다
8. 공고가 열린 기업의 자기소개서 문항이 자동으로 완성되면 좋을 것 같다.

표 7 - 특히 불편했던 부분이 있다면 어떤 부분입니까? 에 대한 답변

4. 결론 및 향후 연구 방향

현재 세종 텔레콤, 물리다 등 여러 학사 정보 및 포트폴리오 관리 플랫폼들이 존재한다. 하지만 기능 및 성능에 보완할 점이 다수 존재한다는 것을 발견했고 또한 기업 및 기관이 아닌 일반 사용자가 사용하기엔 무리가 있다는 점에 초점을 맞춰 이를 개선하고자 새로운 개인 포트폴리오 관리 플랫폼을 제안했다.

블록체인을 통해 자가 정보 및 포트폴리오의 무결성과 신뢰성을 보장하고자 하였으며 이를 통해 개인정보 유출 및 자기 정보의 오남용을 방지하고자 하였다. 또한 기존에 존재하는 포트폴리오 내부 데이터에 대한 복잡한 검증 과정을 개선하여 이미 검증이 된 신뢰성이 보장되는 데이터로 구성된 정해진 양식의 포트폴리오 URL을 기업에 제출하는 형태로 일련의 과정들을 모두 간소화할 수 있다.

설문 조사 결과를 바탕으로 기존에 있는 기능들의 효율성 향상을 첫 번째 목표로 두어 유지 보수를 해 나갈 예정이고 이와 더불어 설문 조사 질문 중 “특히 불편했던 부분”에 대한 답변을 위주로 향후 연구할 예정이다.

5. 개발 일정 및 역할 분담

5.1. 개발 일정

5.1.1. 초기 개발 일정

5월					6월					7월					8월					9월				
1주	2주	3주	4주	5주	1주	2주	3주	4주	5주	1주	2주	3주	4주	5주	1주	2주	3주	4주	5주	1주	2주	3주	4주	5주
블록체인 스터디																								
					개발 프레임워크 스터디																			
								블록체인 네트워크 구축																
								체인코드 개발																
													서버 개발											
													웹 UI 설계 및 개발											
																	테스트 및 수정							
																				최종 점검 및 발표 준비				

그림 25 - 초기 개발 일정

5.1.2. 변경된 개발 일정

5월					6월					7월					8월					9월				
1주	2주	3주	4주	5주	1주	2주	3주	4주	5주	1주	2주	3주	4주	5주	1주	2주	3주	4주	5주	1주	2주	3주	4주	5주
블록체인 스터디																								
					개발 프레임워크 스터디																			
								블록체인 네트워크 구축																
								체인코드 개발																
													서버 개발											
													웹 UI 설계 및 개발											
																	테스트 및 수정							
																				사용자 테스트 및 인사이트 도출				
																				점검 및 최종 보고서 작성				

그림 26 - 변경된 개발 일정

5.2. 역할 분담

이름	역할
조병우	- Hyperledger Fabric 네트워크 구축 - 체인코드 개발
조현우	- SpringBoot 서버 개발 - REST API 설계
서지원	- React를 통한 플랫폼 개발 - UI/UX 설계

표 8 - 구성원별 역할

6. 참고 문헌

- [1] Kakao REST API Documentation, Available:
<https://developers.kakao.com/docs/latest/ko/kakaologin/rest-api>
- [2] Google REST API Documentation, Available:
<https://cloud.google.com/identity-platform/docs/use-rest-api?hl=ko>
- [3] Kubernetes Official Documentation, Available :
<https://kubernetes.io/ko/docs/home/>
- [4] ArgoCD Official Documentation, Available :
<https://argo-cd.readthedocs.io/en/stable/>
- [5] Spring Official Documentation, Spring Boot , Available :
<https://spring.io/projects/spring-boot>
- [6] Spring Official Documentation, Spring Security, Available :
<https://spring.io/projects/spring-security>
- [7] React Documentaion, Available:
<https://ko.reactjs.org/>

[8] Redux Documentation, Available:

<https://ko.redux.js.org/introduction/getting-started/>

[9] Linux Foundation, Hyperledger Project:

<https://www.hyperledger.org/>

[10] Hyperledger Fabric, release-2.2 Documentation:

<https://hyperledger-fabric.readthedocs.io/en/release-2.2/>

[11] Hyperledger Fabric, Java SDK Documentation:

<https://hyperledger.github.io/fabric-gateway-java/>

[12] Hyperledger Fabric, Java Chaincode Documentation:

<https://hyperledger.github.io/fabric-chaincode-java/>

[13] Aws Documentation, Query Chaincode Data in the State Database:

<https://docs.aws.amazon.com/managed-blockchain/latest/hyperledger-fabric-dev/hyperledger-couchdb.html>

[14] HashWiki, 하이퍼레저 패브릭:

http://wiki.hash.kr/index.php/%ED%95%98%EC%9D%B4%ED%8D%BC%EB%A0%88%EC%A0%80_%ED%8C%A8%EB%B8%8C%EB%A6%AD

[15] 하이퍼레저 블록체인 개발, 패브릭과 컴포저로 탈중앙화 dApp 만들기. 한빛 미디어, 2018

[16] B. Kim, K. Kwon, K. Kim, **"An Implementation of medial record storage system that guaranties integrity using private blockchain technology"**, ETRI, PP.106-107, 2020.

[17] 김호원. **"오픈소스 블록체인 플랫폼 동향"**, 월간 SW 중심사회 9월호 포커스3, 2021.

[18] Y. Kang, J. Kim, Y. Cho, **"Proposal of Military Career Certificate Issuing System**

Based on Hyperledger Fabric Platform" KCS, VOL.60, NO.02, PP.0048-0050, 2019.

[19] S. Bae, Y. Shin, "**Design of Personal Career Records Management and Distribution using Block Chain**", KIIECT, VOL.13, NO.3, PP.235-242, 2020.

[20] M. Kwon, J. Jang, J. Lee, H. Yu, "**Performance Optimization of the Endorsement Phase Using Channels in Hyperledger Fabric**", PKIPSC, VOL26, NO.2, PP.110-112, 2019.

[21] W. Hwang, H. Kim, "**A Study on Implementation of BlockChain Voting System using Hyperledger Fabric**", KIIECT, VOL.13, NO.4, PP.298-305, 2020.

[22] H. Kim, "**The Method for Securing Reliability of Students' Records Using Blockchain**", 2020.

[23] J. KIM, N, PARK, "**A Proposal of the Chain Code Agreement Mechanism for the Block Data of the School Records based on Hyperledger Fabric**", JKIIIT, VOL.19, NO.3, PP.121-128, 2021.