이준범

이메일: hardtosayilovu@gmail.com
연락처: +821040140807

안녕하세요! 7년 차 개발자 이준범입니다.
스타트업에서 블록체인 노드 운영과 암호화폐 거래소 입/출금 서비스를 개발/운영하였습니다. 수 십명에서 시작한 스타트업의 거래소에서 50만명 까지 늘어난 유저를 보유한 거래소의 암호화폐 입/출금 및 노드 운영을 담당한 적이 있었으며, 새롭게 시작하는 거래소의 암호화폐 입/출금 시스템 설계 및 개발을 책임졌던 경험이 있습니다. 다양한 규모의 암호화폐 거래소에서 개발 경험을 보유 하고 있는 것이 저의 강점이라고 생각합니다.

유저의 마음으로 꼼꼼히 살피겠습니다.
수년 간 거래소에 있으며 매번 유저 입장에서 생각 해왔습니다. 국내외 수 백개의 거래소에서 투자를 빌미로 수천건의 입/출금 거래를 해봤으며 Defi, NFT 다양한 상품도 구매해보고 거래해 오며 유저 입장에서 필요한 Needs를 몸으로 깨닫고 서비스 구축 시 반영하고자 노력하였습니다. 유저의 입장에서 유저의 마음으로 유저에게 선택을 받을 수 있는 서비스를 만들어 보겠습니다

경력

포블게이트(FOBLGATECoLtd.) **2021.08 – 현재 재직중**
Wallet 백엔드 개발팀/과장

- **암호화폐 집금(게더링) 시스템 개발** 2022.10 – 2023.01

  기술스택 : NestJS, MariaDB, Typescript
  - 수수료 절감을 위한 고객별 선택적 집금 API
  - 집금 기능 자동화를 위한 배치
  - 집금 수수료 사용 정산 API

  고객 입금 내역을 기반으로 사내 보조시스템내 실시간 잔고비율을 확인 후 필요에 의한 고객 입금내역만 콜드월렛에 집금한다.

- **상장 토큰 스마트컨트렉트 검토** 2022.01 – 2023.01

  - 신규 상장 토큰 Solidity Contract 유효성 검사 (safemath, require, modifier)
  - openzeplin 기준 interface 외의 function 에 대하여 기능 검토

  사내 규정 및 금감원 권고 사항 준수 기준 보고 작성하였습니다. 추가로 컨트렉트 내 토큰 전송 제한 기능 또는 컨트렉트 정지 및 토큰 자산의 동결 기능 에 해당하는 트렌젝션 내역에 따라 상장 여부 가능성을 검토하였습니다.

- **통합 API gateway 서비스 개발** 2022.10 – 2023.01

  기술스택 : NestJS, MariaDB, Typescript
  1. 통합 API 제공을 통한 개별 코인 처리별 연계 시스템 예외처리 간소화
  - 유사 기능을 하나로 API로 통합, 연계 시스템과 URL 기반 상호운용성 확보
  - 코인 별 처리기능을 추상화 하고, 상위 레벨에서 통합하여 파편화 되어있는 지갑시스템 연계내

역 변경사항 최소화

2. 비동기 블록체인 거래 API 요청 대응을 위한 시스템 극대화
- Queue를 통한 비동기 처리 방식의 적극 도입 및 Callback 기반의 API 구조 확립
- 병렬 처리기반으로 시스템 구조를 변경하여, 시스템 성능 극대화

통합 API Gateway 서비스의 목적은 고가용성(HA) 시스템 아키텍쳐 구성입니다. 리팩토링 계획 이전 입, 출금 시스템은 두 대의 서버로 서비스가 불가능했고 개선이 필요함에 내결함성을 목적으로 서비스를 재설계하였습니다. 통합 API Gateway의 경우 거래소 백엔드와 두 대 이상의 서버로 서비스를 구성하여 시스템 안정성을 제고하고, 성능을 향상 시키고자 하였습니다.

- **암호화폐 입, 출금 시스템 2차 리팩토링** 2022.10 - 2023.01

  기술스택 : NestJS, MariaDB, Typescript
  1. 불필요한 데이터베이스 사용 최소화
  - 서비스 모듈 초기화 시 빈번이 사용되는 데이터 캐싱

  2. Scale Out 아키텍쳐 구성(클러스터링)
  - Redis Queue, Redis 를 활용하여 순차처리 및 서비스간 메모리 데이터를 공유
  - 과부하를 대비한 병렬처리 클러스터링

  3. Wallet 관련 기능 별 데이터 아키텍쳐 수정
  - 기능별 테이블 분리(입금, 출금, 집금)
  - Redis Scan 활용 아키텍쳐 설계

  2차 리팩토링의 목적은 시스템 내결함성(FT) 위한 아케틱쳐 변경입니다. 실질적으로 고객 자산의 입금 및 출금을 처리 하는 실질적인 비즈니스 로직을 처리하는 영역이어서 시스템 기능이 계속 작동할 수 있는 구성이 필요하다고 생각하였습니다. 작업의 순서와 안정성을 위해서 Redis Queue를 사용한 순차 입,출금 처리 그리고 병렬처리 중인 서비스간의 공유 메모리로 Redis를 사용하였습니다.

  데이터베이스의 Index와 Unique 값, 질의 조건 등을 활용하여 Redis Key를 조합하여 사용하였다. 입금을 예로 WALLET:ETH:DEPOSIT:COINNAME:USER_ID:BLOCKNO:TXID 가 키 값으로 사용된다. 위와 같은 구조는 scan pattern 사용하면 질의 조건에 대한 값을 쉽게 찾을 수 있다.

- **신규 메인넷 클라이언트 API 분석 및 개발** 2022.04 - 2023.01

  기술스택 : NestJS, MariaDB, Typescript
  - 신규 상장 메인넷 코인 및 토큰 입, 출금 시스템 공통 인터페이스 API 설계 및 개발
  BSC(BNB, BEP20) 입,출금 시스템 개발
  Klaytn(Klay, KIP7) 입,출금 시스템 개발
  Solana(Sol, SPL) 입, 출금 시스템 개발
  Tron(Trx, Trc20) 입, 출금 시스템 개발
  Luna 입, 출금 시스템 개발
  Ethereum Fair 입, 출금 시스템 개발

  HD Wallet 출금 시스템을 활용한 출금지갑을 구축, 이외 통합 API Gateway 와 연계 파라미터 및 기능별 API를 통합하여 코인 별 동일한 사용성 제공하였습니다.

- **암호화폐 입,출금 시스템 리팩토링** 2022.03 - 2022.06

  기술스택 : NestJS, MariaDB, Typescript
  1. 오류코드의 분리
  - DB커넥션, 파라미터 유효성 및 누락 , 블록체인 노드 연결관련 , 비즈니스 로직 관련 오류코드 분리

  2. NestJS 프레임워크 사용
  - 기존 Node Express 프로젝트, Nestjs 프로젝트 전환
  - 외부 라이브러리 모듈화
  - CommonJS 문법 ES6 문법으로 변환 작업
  - request 인터셉터 처리를 통한 시스템 로깅
  - typeorm 사용
  - NodeJS(10->16) 버전 최신화

  프로젝트의 오류코드는 성공(0000) 또는 실패(9999) 였다. 운영지원팀에서 해당 예외상황에 자세한 내역을 알지 못하여 고객응대 시간이 평균 2일 이상 걸렸으나, 예외상황에 대한 판단 후 사내 입, 출금 보조 시스템을 활용하여 즉각적인 고객응대가 가능해졌다.

  NestJs 프레임워크를 쓰며 가장 큰 장점으로 각 암호화폐 프로젝트 아키텍쳐의 획일화 그리고 Request Lifecycle 이벤트를 활용, 컨트롤러 전 Guard 처리, Pipe를 활용한 Request 파라미터 전처리, Response 응답 획일화 등, 예외상황에 체계적 관리가 가능해졌다.

- **VerifyVASP TravelRule 트렌젝션 확인 API 개발** 2022.03 - 2022.03

  기술스택 : NestJS, MariaDB, Typescript
  - VASP 기능 제공을 위한 입금 및 출금 트렌젝션 Confirm 및 자산 확인 API

  VASP에서 사용 될 거래소 내 취급 트렌젝션에 대한 필수 유효값 제공

- **암호화폐 정산 시스템 개발 및 유지보수** 2022.01 - 2022.12

  기술스택 : NestJS, MariaDB, Typescript
  1. 암호화폐 별 일별 출금, 집금 수수료 내역 정산 배치
  - 각 코인별 출금 수수료 계
  - 고객 계정별 지갑내 자산 집금 시 사용 된 수수료 계

  2. 암호화폐 별 일별 보유 수량 기록 배치
  - 고객 데이터베이스 자산, 고객 핫월렛 자산, 회사 콜드월렛 자산, 고객 계정 내 미집금 자산 기록 자동화
  - 핫월렛 (고객 계정 내 미집금 자산 + 거래소 출금 지갑의 자산) , 콜드 월렛 자산 비율 모니터링 및 적정 비율 자동화

- **HD Wallet (BIP32, BIP39, BIP44) 활용한 출금시스템 개발** 2021.12 - 2022.01

  기술스택 : NestJS, MariaDB, Typescript
  - Mnemonic + Path 를 활용한 출금시스템
  기존 시스템은 PrivateKey를 평문 또는 일반적인 암복호화 알고리즘을 활용하여 단순 평문형태로 보관하여 사용하였다. ISMS 준비과정에서 PrivateKey 를 단순 평문 형태로 저장하여 활용하는 것이 아닌 거래소->Proxy Wallet-> 출금 로직 을 거쳐가며 각각의 서비스를 거쳐가며 파편화된 출금지갑의 구성 요소들을 만들게 하여 최종적으로 출금로직에서만 사용될 수 있도록 구

성하였다. 서명에 필요한 PrivateKey같은 경우 결정론적 난수발생(HMAC-DRBG) 활용한 니모닉 + Path 을 조합하여 PrivateKey를 만들었다.

파편화의 경우 니모닉 코드 와 Path의 분리 및 CloudHSM을 활용 암복화 하여 Wallet을 담당하는 임원외에는 파편화에 대한 단서를 알 수 없게 만들었다.

- **CloudHSM 관리 및 암/복호화 API 유지 보수** 2021.09 - 2023.01

  - SoftHSM 사용하여 PKCS#11 기반 인터페이스를 제공
  - AES_CBC_PAD(대칭키 암호화) 매커니즘을 이용한 MasterKey 관리
  - AES256 암/복호화 기능 API 관리

  HSM의 주 목적인 키의 생성을 외부에 노출 시키지 않는다는 이념보단 암/복호화를 하는 기능에 중점을 맞췄고 암/복호화에 사용되는 키를 CloudHSM 에 주입하여 사용한다. 각 지갑에 사용되는 키는 Alias를 활용하여 개별 관리 되며, 키 생성 시 필요한 초기 16바이트의 초기 벡터값을 변형하여 키를 생성했다.

- **암호화폐 노드 설치 및 운영, 입/출금 시스템 유지 보수** 2021.09 - 2023.01

  기술스택 : NestJS, MariaDB, Typescript
  1. 암호화폐 노드 별 HealthCheck API 개발
  - 사내 Admin 용도 입금 처리 중인 BlockHeight 제공
  - 암호화폐 별 블록 동기화 상태 제공

  2. RPC Config 및 Shell Script 관리
  - 코인별 RPC Config 설정 관리
  코인별 RPC 허용 설정 user, pass, ip, port 관리

  - 비트코인 계열 walletnotify 설정 관리
  config내 walletnotify 옵션을 활용 하여 쉘스크립트를 활용하여 입출금 서비스 로직으로 입금 txid를 전송

  3. 하드포크 대응 및 메인넷 클라이언트 버전 최신화
  - btc, ltc, eth, bnb, klay, eos, xrp 등 메인넷 클라이언트 설치
  - eth, bnb, klay, eos 블록 데이터 백업 (AMI)
  - ethfair, ethpow, bsc, eos, klaytn 등 메인넷 하드포크 대응
  - 메인넷 클라이언트 버전 갱신

  4. 인프라 구성 개선 (2022. 01 ~ 2022. 02)
  - 거래소 백엔드 데몬과 입출금 서비스 데몬의 분리
  체결을 제외한 거래소 기능을 담당하는 서비스 데몬(Java) 과 암호화폐 입출금을 담당하는 서비스 데몬(NodeJS)가 동일한 서버
  에 운영되어 신규 메인넷 등 신규 서비스의 증설에 따른 CPU 과부하 및 네트워크 throughput 의 부족현상 발생으로 거래소 서비스가 원활하지 않음. 거래소 기능을 담당하는 데몬과 입출금 서비스의 데몬을 물리적으로 서버를 분리하여 "거래소 〈-〉 입출금(Wallet) 〈-〉 메인넷노드" 구성형태의 프록시 Wallet 서버 구축

  5. nginx 설정관리 (2022. 01 ~ 2022. 02)
  - /etc/hosts 도메인 관리

Wallet 서버내 메인넷 서버(프라이빗존, 사설 IP 대역) 와 도메인 매핑

- worker 및 upstream 로드밸런싱 관리
active, standby 구성의 2개의 메인넷 노드를 upstream 가중치 및 fail 타임아웃 설정을 통해 RPC 요청 관리
초당 많은 rpc 요청처리를 위해 process, connection 수 관리

- **사내 입,출금 관련 운영지원 보조 기능 개발** 2021.09 - 2023.01

기술스택 : NestJS, MariaDB, Typescript
1. 암호화폐 미입금 처리
- 누락된 암호화폐 미입금 처리 내역 자동화

2. 암호화폐 오입금 처리
- 고객 실수로 인한 오입금 처리 기능 개발(memo 또는 destination tag 미입력시 admin 처리 가능)
- 고객 실수로 운영하지 않는 네트워크의 자산 전송에 대한 복구 처리 작업

3. 펜딩 트렌젝션 재처리
- 블록체인 네트워크의 과부하로 수수료 비용으로 인한 펜딩이 지속될 경우 수수료 갱신 후 재처리

4. 실시간 잔고 비율 API
- 금감원 권고사항 핫, 콜드월렛 별 적정 자산비율을 지키기 위한 모니터링 API

스포홀딩스 2020.12 - 2021.05

Wallet 백엔드 개발/책임연구원

- **인프라 구성 개선** 2019.12 - 2021.04

개선 전 상태
3티어 아키텍쳐 형태(1WEB, 1WAS, 1RDB) 로 개발되었으나 서비스 로직의 비효율성 및 슬로우 쿼리를 해결하기 위해서 Scale Up 운영

개선 작업 사항
1. 기본적인 인스턴스 및 RDB 스펙 하향 조정
- ec2 : 2xlarge =〉 large
- rds : 4xlarge =〉 xlarge

2. 로드밸런서를 활용하여 서비스 과부하 대응
- External ALB (WEB1, WEB2) - IGW 접근 허용
- Interal ALB (WAS1, WAS2) - External ALB 만 접근 허용

3. Aurora RDS Clustering
- Slow 쿼리 효율을 증대하기 위하여 읽기, 쓰기 전용 RDB 분리

4. 배포 환경 개선
- PM2 Deploy를 활용하여 배포 (WEB1, WAS1)

- AWS EFS를 사용하여 WEB1=WEB2, WAS1= WAS2 동기화

- **이더리움 입,출금 시스템 구성 및 개발** 2019.12 - 2021.04

  기술스택 : NodeJs , Php(Laravel), Mysql

  개선 전 단점 및 개선안
  1. 고객별 개인키를 Database에 평문으로 저장
  - 개인키는 필요 시 생성하는 방법으로 구성 -〉 mnemonic + derivation path(uid) 를 활용하여 고객 개인키 생성

  2. ERC20 입금 시 블록 내 트렌젝션 Input 데이터를 Decode하여 입금처리.
  - Infura(Archieve)를 활용하여 Contract.getPastEvent 사용하여 빈번한 서비스 로직 발생 제거

  추가 개발 사항
  1. 펜딩 트렌젝션으로 인한 출금지연 사태 처리
  - 출금 번호별 nonce 처리 번호, gasPrice 가격 기입에 따라 펜딩 트렌젝션 재처리로 출금 원활

  2. ERC20 토큰 직금(게더링) 자동화
  - 고객이 사내 서비스중 코인 입금 시, 입금 기록을 통하여 수수료 제출 및 직금 자동화

코인네스트 2018.02 - 2019.12

Wallet 개발 및 유지보수/사원

- **암호화폐 거래소 입출금 Wallet 개발** 2018.02 - 2019.04

  기술스택 : Java(SpringBoot) , Ubuntu, Mysql
  - 고객 입금지갑 : HD WALLET 구성
  - Extended PublicKey 를 활용하여 MasterSeed에서 고객 Wallet 생성
  - Extended PrivateKey 를 활용하여 MasterSeed에서 고객 Wallet PrivateKey 관리
  - RMI 서비스 구성을 통한 Extended PrivateKey, Wallet password 획득

  데이터베이스에 각 코인 별 Extended PublicKey를 활용하여 고객의 입금 지갑주소를 제공하였다. 집금 기능을 위하여 고객 개인키를 필요 시 RMI(원격 메소드 인터페이스)를 통해 외부 인터넷 네트워크와 단절되어있는 서버의 프로세스 메모리에 저장되어있는 키를 사용하였다.

  - rawTransaction 의 분리 (트렌젝션 오브젝트 생성 + 서명)
  출금 요청시 서명 전의 트렌젝션 오브젝트의 값을 데이터베이스에 저장하였다. 외부 인터넷 네트워크와 단절되어있는 개별 서버에는 서명을 위한 키 값들이 존재하였고 스케줄러를 통하여 트렌젝션 오브젝트에 서명 후 데이터 베이스에 갱신 하였다. 서명이 완료된 rawtransaction은 추가적인 Broadcasting 전담 스케줄러를 통하여 트렌젝션이 제출되는 구조를 만들었다.

- **트론 SR (Super Representatives) 노드 구성** 2018.06 - 2018.08

  기술스택 : AWS EC2, ELB
  - 트론 재단 요청으로 SR 참여를 위한 SR 노드 설정
  - Active, Standby ELB 구성

- 스테이킹 비율에 따른 분배 시스템 구성

- **암호화폐 노드 운영 유지보수** 2018.02 - 2020.11

  기술스택 : AWS EC2, ELB 구성
  - 대상 메인넷 : 비트코인, 이더리움 등 코인네스트 취급코인 전체
  - 메인넷 상태 모니터링 및 각 코인별 하드포크 대응

헬스케어네비 2017.01 - 2018.01

암호화폐 채굴 및 정산시스템 개발(비트코인, 이더리움)

- **암호화폐 채굴 시스템 구축** 2017.02 - 2018.01

  기술스택 : JQuery, PHP, MYSQL
  - 채굴환경 인프라 구성 (약 2,000대 컴퓨대로 구성, 채굴기 이용)
  - 채굴PC 모니터링 시스템 구성 (hashrate 측정)

- **암호화폐 채굴 정산시스템 구축** 2017.02 - 2018.01

  기술스택 : JQuery, PHP, NodeJS, MYSQL
  - 채굴실적 모니터링 및 수익분배 시스템 구축 (데이터베이스 설계 및 개발)
  - 전체 수익을 기준으로 채굴 요청 고객의 계약량에 따른 수익 분배 및 알림 서비스 구성
  : 텔레그램을 이용한 알림시스템

학력

순천향대학교 2010.03 - 2016.08

소프트웨어공학과

스킬

AWS, Git, MySQL, Node.js, Nest.js, Redis, Spring Framework, Linux, Confluence, JIRA

외국어

영어

일상회화

- **토익** 2016.01.01

  800점