

Learn, Find, Figure Out

STARTERGATE / 최호승

me@startergate.dev

백엔드 엔지니어
3학년 고등학생 @ 광주소프트웨어마이스터고등학교

Backend Engineer

[GitHub](#) [StackShare](#)



백엔드 엔지니어입니다. 눈에 보이지 않는 걸 만듭니다.

서버에서 돌아가는 프로그램을 만듭니다. 직접 해야할 때는 모바일 앱이나 웹 앱도 직접 만듭니다.

컴퓨터에는 항상 관심이 많았습니다. 초등학교 때에는 컴퓨터 방과후를 다녔고,

중학교 때에 생활코딩으로 코딩을 시작했습니다.

코딩에 관심이 많아져서 소프트웨어 마이스터고등학교로 진학했습니다.

항상 변화하며 새로운 것을 시도합니다. 새로운 것은 항상 더 나은 것이라 생각하고 있습니다.

Experiences

광주소프트웨어마이스터고등학교

SW개발과 2018.3 ~ 2021.2 (예정)

웹 개발 동아리 빈실 부장

게임 개발 동아리 Tiny Beluga 부장



백엔드 개발자 (현장실습) 2020.1 ~ 2020.2

계정 인증 미들웨어 및 일부 기능 리팩토링

Skills



Activities

글로벌 비즈쿨 CO-TDM 창업경진대회

2018.6, 최우수상

GSM 팀프로젝트발표회

2020.9, 장려상

GSM 전공 동아리 해커톤 대회

2020.9, 장려상

GSM 창의알고리즘 콘테스트

2018.4, 장려상

Python

Django와 챗봇 개발을 위해 사용

알고리즘 문제 해결을 위해 사용

파이썬 백엔드 개발 현장실습 경험, 배포된 코드 작성 경험

Node.js

Node.js 8부터 시작, TypeScript와 함께 사용 중

Apollo + Koa로 GraphQL 개발 경험

다양한 종류와 규모의 프로젝트 개발

빈실

학교 내부에서 사용되는 서비스를 개발하는 동아리

TypeScript, Node.js, Koa.js 사용

테스팅, 문서화, 스프린트 개발 경험

Project Highlights

보여주고 싶은 게 많지만, 중요한 것만 적었어요



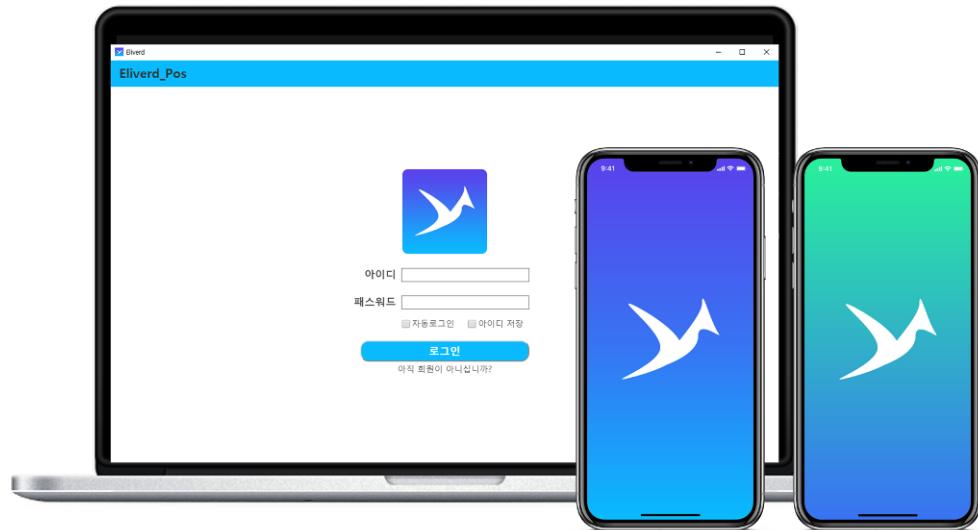
Eliverd

Eliverd

Details

Deliver Your Life, Eliverd.

소상공인을 위한 온라인 상점, 배달 서비스



개발 기간

2020.04.14. ~ 2020.10.18. 개발 완료

상태

팀원 및 역할



최호승

팀장, 기획, 서버 개발, 데이터베이스 구축



박준영

디자인, 모바일 앱 개발



박종효

데스크탑 앱 개발

기술 스택

Python, JavaScript, Dart, MariaDB, Docker, AWS (EC2, S3)

주요 라이브러리

Django, Django REST Framework, Flutter, Electron, React

기타

수상: 교내 팀프로젝트발표회 (2020.09, 장려상)



번호	기능 코드	Depth1	Depth2	Depth3	구현 대상				작업 표소	관련자	기능 정의	
					PC	Store	Mobile	Consumer	디자인	피피라운	개발	
1	eliverdapp-signin-01	메인	로그인 요청	-	○	○	○	X	○	○	○	X
2	eliverdapp-signin-02	메인	회원 가입 페이지 Navigation	-	○	○	○	X	○	○	○	X
3	eliverdapp-store-signup-01	메인	회원가입 요청	-	○	○	○	X	○	○	○	X
4	eliverdapp-signout-01	헤더	로그아웃	-	○	○	○	X	○	○	○	X
5	eliverd-store-mgmt-01	헤더	제고 및 터치(제고 세부 설정)	-	○	○	○	X	○	○	○	X
6	eliverd-store-mgmt-02	헤더	제고 등록	-	○	○	○	X	○	○	○	X
7	eliverd-store-mgmt-03	메인	제고 등록	나코드 등록	○	○	○	X	○	○	○	X
8	eliverd-store-mgmt-04	메인	제고 수령	-	○	○	○	X	○	○	○	X
9	eliverd-store-mgmt-05	메인	제고 접수	-	○	○	○	X	○	○	○	X
10	eliverd-store-mgmt-06	메인	제고 삭제	-	○	○	○	X	○	○	○	X
11	eliverd-store-mgmt-07	메인	사업자 등록	-	○	○	○	X	○	○	○	X
12	eliverd-store-mgmt-08	메인	사업자 등록	사업자 결제	○	○	○	X	○	○	○	X
13	eliverd-store-mgmt-09	메인	사업자 등록	사업장 위치 지정	○	○	○	X	○	○	○	X
14	eliverpos-mgmt-01	헤더	리远处방법 선택여부	리远处방법 선택여부	○	○	X	X	○	○	○	○
15	eliverpos-mgmt-02	메인	PoS 환경 설정	POS 환경 설정	○	○	X	X	○	○	○	○
16	eliverpos-mgmt-03	메인	제고 수령	제고 수령	○	○	X	X	○	○	○	○
17	eliverpos-mgmt-04	메인	사업장 부록 등록	사업장 부록 등록	○	○	X	X	○	○	○	○
18	eliverpos-mgmt-05	메인	제고 정보 조회	제고 정보 조회	○	○	X	X	○	○	○	○

기능 정의

필요도와 시급성을 정리할 수 있었음
프로젝트 구조 이해에 도움
서버와 클라이언트의 개발 효율화

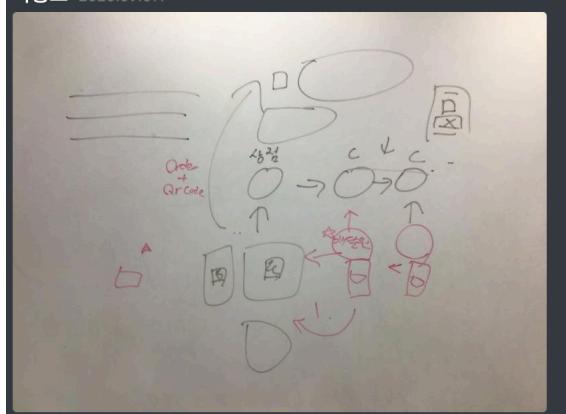
기획

여러 전략과 분석을 진행하며 이 프로젝트에서 무엇을 중점으로 보고 개발해야 하는지, 무엇을 강조해야 하는지 정리할 수 있었다.

프로젝트 기획서	
팀명	운영네 마트
팀원 및 역할 (분야별 책임자)	사업부: 최효정 소비자부: 엄 백운영 POB: 박종우 상장영: 강, 윤경 개발
프로젝트 주제	온라인마트, 소상공인 중심의 상장 및 배달 시스템
제작 배경 및 필요성	세계화되는 이마트 같은 대형마트와 위한 디자인 서비스도 중요하다. 구글 포드 푸드, 베리풀, 맥도날드, 유플러스 등 다양한 경쟁사가 있지만 대형마트는 엔터테인먼트, 프로모션과 협업을 이루고 있으므로 영세상인은, 이들처럼 상장 같은 소상공인들에게는 불편한 점이 많기 쉽지 않아 있다.
사업 및 유사 제품 현황	최근에 Google Now에서 서비스형 브랜드가 버려와 서비스형 온라인 판매점이 가중치가 되고 있는 것을 감지해 소상공인과 협업을 할 수 있는 힘이라는 걸 느꼈고 그들을 위한 세종은 물론, 베리풀, 맥도날드, 페스티벌 등과 협업을 느꼈다.
제작 내용	구글 포드 푸드나, 베리풀의 만족 같은 배달 서비스가 시장에 출시, 이미 같은 서비스(마트)를 운영하는 업체는 예상 수익률을 요구하는데도 시장 자체는 커서 거래하는 편이다. 소비자부에서는 돈을 내야하는데도 편리함을 추구하는 성향이 되어가고 있음.
기대 효과 및 활용방안	- 소비자: 중앙, 면밀, 편의점, 등의 가격을 비교하고, 기준의 배달 서비스에서는 지원 하지 않던 서비스도 지원하는 소비자들이 유통망을 확장해 소비를 확장할 수 있다. - 소상공인: 편의점: 품질에 대한 고민과 함께 베리풀이나 맥도날드 모델의 변화를 제작해 배달을 상장에서도 적용해 품질을 확장할 수 있다. - 대기업: 편의점, 배달을 상장에서도 적용해 품질을 확장하는 가능. 수익성이 좋은 낮은 매장을 특징 하고, 배달을 대상화하면서 비용 절감 가능. 새로운 지역으로 진출하는 경우 신규 매장 이 영업 점수를 얻을 가능.

(1) 계약 배경 - 외부 환경 분석 (PEST / STEEP)	
정책적 배경	중소벤처기업부의 스마트 대한민국 정책 (중소기업의 ICT화 지원)
경제적 배경	비당 일계의 상장 언택트 비즈니스의 활성화
사회적 배경	코로나19로 인해 놀라워지기 시작한 온라인 결제 소비자들은 배달 서비스 이용 증가
트렌드 배경	배민의 수주율 인상이나, 광고 정책 변경으로 인해 축발된 새로운 배달 업체 대량 도입
기술적 배경	(전국민의 스마트폰 보급율이 높아지면서) 모바일 앱의 사용 비중 증가
제도적 배경	정부 차원에서의 새로운 배달업 요구

(2) 계약 배경 - 내부 환경 분석 (3C)	
개인적 능력	소상공인을 통한 서비스 운영 변화를 유연하게 수용할 수 있는 구조
경쟁제품/기술 /특허 분석	타마트: 동네 마트와 겨루는 것이 아닌 자체 물류 센터 운영 아트포모: 페스티벌 운영 로마켓
교역 분석	고객분류 <ol style="list-style-type: none"> 대기업 판매자 영세 상인 (온라인 배송을 이용하는 모든) 소비자
	목표고객 <ol style="list-style-type: none"> 1순위: 영세 상인 2순위: 소비자



6. 배달순서
- 6-1 상점용 앱, 데스크탑에서 주문 생성
 - 6-2 주문 생성에서 받은 문자열을 QR코드로 변환
 - 6-3 소비자용 앱에서 배달원이 해당 QR코드를 스캔
 - 6-4 배달원의 화면에 QR코드와 지도, 상세 정보가 뜸
 - 6-5 배달원이 소비자에게 도착
 - 6-6 소비자가 소비자용 앱으로 배달원의 QR코드를 스캔
 - 6-7 계정정보 일치 시 배달 완료.

문서화

drf-yasg를 사용해 문서화 진행
클라이언트와 서버 개발자 간의 의사소통을 간소화
기능 개발 진행 상황 공유
백엔드 실제 개발 상황 정리

원활한 언택트 의사소통

기능, 디자인 추가/변경마다 간단한 회의
좋은 디자인과 기능 개발할 수 있었음

GET /purchase/{oid}/approve/

Response samples

200

Content type
application/json

```
{
    "oid": 0,
    "tid": "string",
    "status": "pending",
    "is_delivery": true,
    "customer": 0,
    "partials": [
        0
    ]
}
```

Copy Expand all Collapse all



server 1

Latest version

► Pull image from the command line:

Learn more

```
$ docker pull docker.pkg.github.com/junyeong-market/eliverd-api/server:1
```

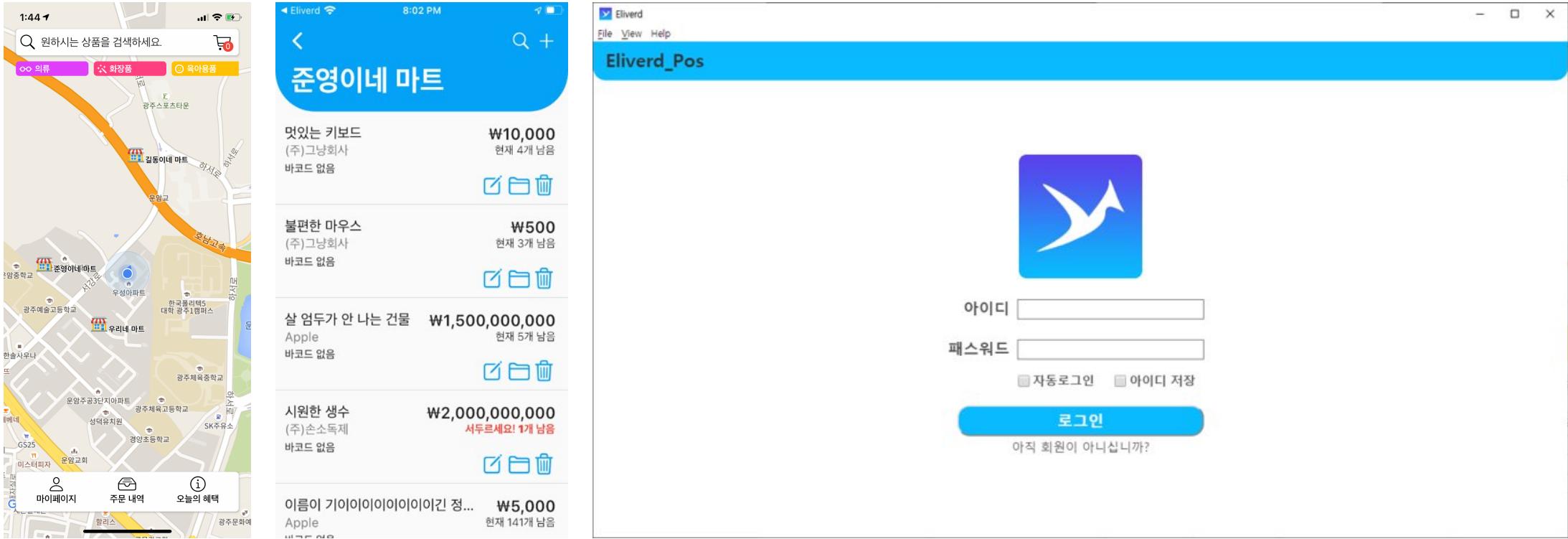
◀ Use as base image in DockerFile:

```
FROM docker.pkg.github.com/junyeong-market/eliverd-api/server:1
```



컨테이너화

기능 추가가 잦은 프로젝트의 특성 상 많은 배포가 필요
바이너리 의존성이 있는 프로젝트다 보니 빌드용 shell script 제작
docker-compose를 활용하여 빌드 및 배포 진행
AWS ECS와 GitHub Action을 활용해 자동으로 배포되도록 했으면 더 좋았을 것 같음



다양한 클라이언트

프로젝트의 특성 상 상점과 소비자 애플리케이션이 모두 필요
앱은 소비자용과, 영세 상인이나 노점을 대상으로 상점용을 개발
POS와 유사한 역할을 해줄 데스크탑 프로그램 개발
이 클라이언트들을 하나의 서버에서 처리할 수 있도록 개발함

DOTORI

DOTORI

GSM 기숙사 관리 시스템

The screenshot shows the DOTORI application's home screen. At the top, it displays a message: "Beta 서비스 중입니다. 정식 서비스: 2020년 1학기 예정". Below this, there are several cards:

- 기숙사 공지사항**: Includes a "로그아웃" button and three "Test" items from 2020년 00월 00일.
- 대나무 숲 게시글**: Includes a "디보기" button and five "Test" items from 2020년 00월 00일.
- 기상음악**: Includes a "더보기" button and three "Test" items from 2020년 00월 00일.
- 사실 아무것도 없다!**: Includes a "더보기" button and three "Test" items from 2020년 00월 00일.

On the left sidebar, there are navigation links: 홈, 공지사항, 노트북 대여, 기상음악, 상벌점, 설정, 학교 홈페이지, 학교 슬랙, and 버그 제보. At the bottom, there is a copyright notice: © 2019 GSM. All Rights Reserved. 개인정보취급방침 | 이용약관 | 라이선스 | Github.

Details

개발 기간

2020.02.14. ~ 2020.08.26.

상태

미완

팀원 및 역할



최호승
부장, 서버 개발



동아리원들
디자인, 서버 및 프론트엔드 개발



기술 스택

Node.js, TypeScript, MariaDB, Docker, AWS (EC2, S3)

주요 라이브러리

Koa.js, Sequelize.js, React

합의를 통한 진보

개발의 중요한 분기점이 되는 결정을 회의를 통해 결정
팀원 모두의 합의를 거치기에 결정된 사항에 대한 참여도를 높일 수 있었음
코딩 컨벤션, 테스팅, 문서화 등을 회의를 거쳐 성공적으로 도입

✓ master	default
develop	
feature/account-app	
feature/board	
feature/documentation	
feature/laptop	
feature/middlewares	
feature/music-apply	
feature/nginx-proxy	

git-flow

협업의 편의성을 위해 git-flow 브랜치 전략 사용
개발 초반에는 release branch에 대한 필요성을 느끼지 못함
이후 hotfix를 반영하는 방법을 찾다가 필요성을 느낄 수 있었음

회의-채팅-20200130

회의-채팅-20200214

회의-채팅-20200306

회의-채팅-20200313-백엔...

회의-채팅-20200322-도...

회의-채팅-20200408

회의-채팅-20200426-도...

회의-채팅-20200509



테스팅

실제로 출시할 코드였기에 테스팅을 개발 초기 단계부터 진행
테스트를 완성해야 PR Merge 될 수 있도록 함
생각치 못했던 에러도 여러 개 찾을 수 있었음

The screenshot shows the Bean API documentation. The left sidebar has a navigation tree with categories like Authentication, General, Account, Score Archive, User, Board, Laptop, and Music. The 'Account' category is currently selected. The main content area displays the 'Account' section of the API. It includes a search bar, a 'HEADER PARAMETERS' table with an 'Authorization' row, and a 'Responses' table containing two rows: one for a successful response (200) and one for an unauthorized response (401). Below this is a detailed description of the 'Account' endpoint.

This screenshot shows the 'session' creation section of the Bean API documentation. It features a 'GET /account/v1/session' request sample and a 'POST /account/v1/session' request sample. The 'POST' sample includes a 'Payload' table with fields 'email' and 'pw'. Below the requests are 'Response samples' for both GET and POST methods, showing JSON examples of the responses.

File	%Stmts	%Branch	%Funcs	%Lines	Uncovered Line #s
All files	100	100	100	100	
board.v1.controller.ts	100	100	100	100	
index.ts	100	100	100	100	
laptop.controller.ts	100	100	100	100	
music.controller.ts	100	100	100	100	
score.controller.ts	100	100	100	100	


```
===== Coverage summary =====
Statements : 100% ( 147/147 )
Branches  : 100% ( 6/6 )
Functions  : 100% ( 23/23 )
Lines     : 100% ( 146/146 )
=====
```



```
Test Suites: 11 passed, 11 total
Tests:       64 passed, 64 total
Snapshots:   0 total
Time:        26.855s
Ran all test suites.
```

문서화

프론트와의 편한 협업을 위해 문서화를 진행
Redoc을 사용함

Node.js CI

160 results			
	Event ▾	Status ▾	Branch ▾
Actor ▾			
✓ dep: Joi 업데이트	feature/update-joi	4 months ago	...
Node.js CI #245: Pull request #77 synchronize by startergate		1m 43s	
✓ [FIX] ctx.assert 잘못 적음	hotfix/assertion-error	6 months ago	...
Node.js CI #242: Pull request #75 opened by startergate		1m 41s	
✓ [FIX] 로그인 API 변경	hotfix/name-error	6 months ago	...
Node.js CI #241: Pull request #74 opened by startergate		1m 21s	
✓ [FIX] 해시 변수 필요할 때마다 만들기	hotfix/email-must-be-uni...	6 months ago	...

Continuous Integration

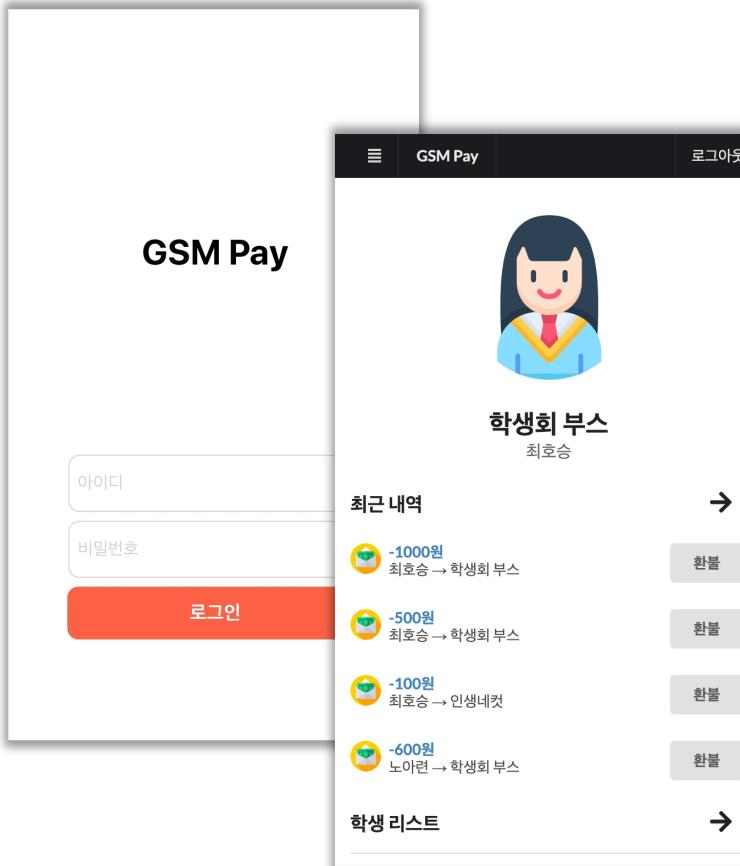
PR을 진행할 때마다 코딩 컨벤션을 자동으로 검사하는 GitHub Action 사용
다양한 환경에서 빌드를 수행하여 테스팅 과정에서 발견하지 못한 버그 검사
CI 통과를 의무화해서 코드 퀄리티 유지에 도움이 되었음
테스트까지 자동으로 돌리게 되었으면 코드 리뷰 하는데 더 편했을 것



GSM Pay

Details

온라인 메모 웹 앱



개발 기간

상태

2020.06.17. ~ 2020.09.23. 개발 완료

팀원 및 역할



최호승
부장, 서버 개발



동아리원들
디자인, 서버 및 프론트엔드 개발



기술 스택

Node.js, TypeScript, JavaScript, MySQL, Docker, AWS EC2

주요 라이브러리

Koa.js, Sequelize.js, Apollo, React

기타

수상: 교내 전공동아리 해커톤 대회 (2020.09, 장려상)

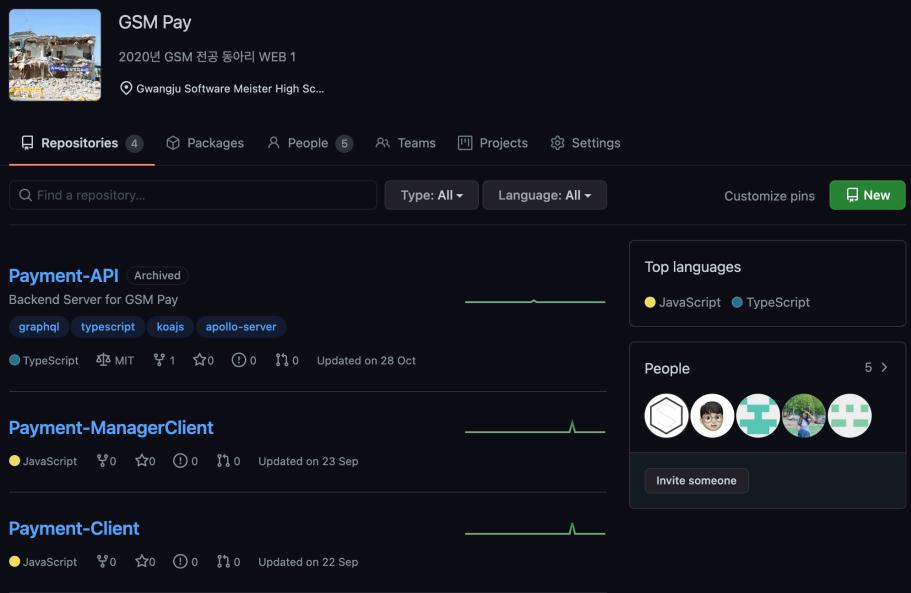
```
const query = gql`  
  type Query {  
    me: User  
    users: [User]  
    booth(bid: Int!): Booth  
    booths: [Booth]  
    myBooths: [Booth]  
    transactionsInBooth(bid: Int): [Transaction]  
    transactionsInUser: [Transaction]  
    transactions: [Transaction]  
  }  
`;
```

GitHub을 통한 협업

git-flow 적용하여 개발

GraphQL

Apollo를 통한 GraphQL 사용

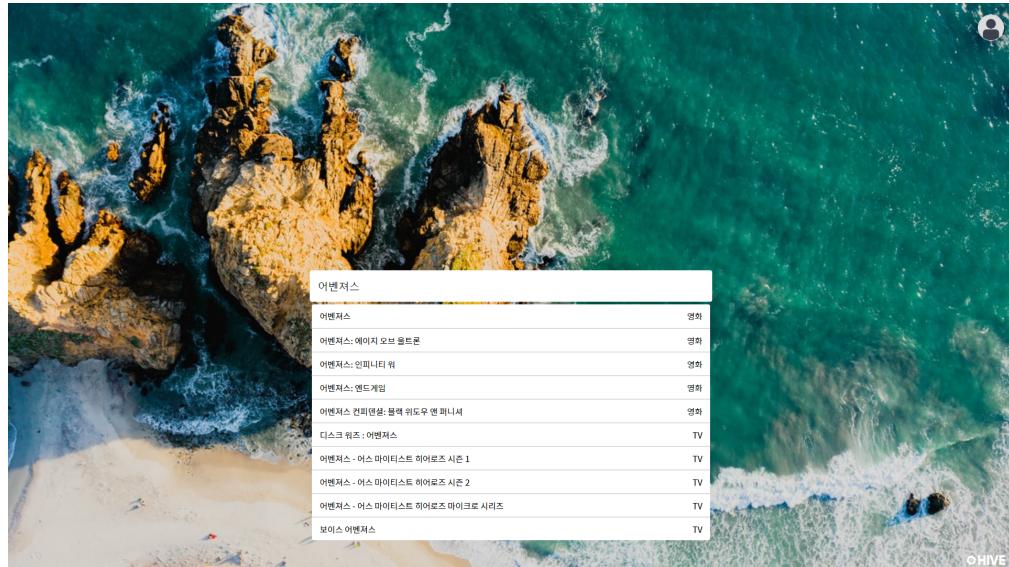




HIVE

Details

영화 평점 크롤링 서비스



개발 기간

2019.06.25. ~ 2019.07.10. 개발 완료

팀원 및 역할



최호승

팀장, 기획, 서버 개발, 데이터베이스 구축



정민우

디자인, 프론트엔드 개발

기술 스택

Node.js, JavaScript, MongoDB, AWS EC2

주요 라이브러리

Express.js, cheerio





문제에 맞는 기술 활용

여러 소스에서 데이터를 크롤링해야 했음
JS Promise를 활용하여 각 크롤러의 실행 순서와 시점을 제어함

사용자 경험 중시

다양한 소스에서 데이터 수집
이를 수집하는 시간이 유저에게 드러나지 않도록 구성

```
db.findMovie(watchaID[j], (err, res) => {
  if (err) throw err;
  if (res) {
    this.update(watchaID[j]);
    callback();
    return;
  }
  callback2();
  const instanceWatcha = createInstance("https://watcha.com/ko-KR/");
  const instanceImdb = createInstance("https://www.imdb.com/");
  //const instanceRotten = createInstance("https://www.rottentomatoes.com/");
  const instanceNaver = createInstance("https://movie.naver.com/movie");
  instanceWatcha.get('/contents/${watchaID[j]}', {
    timeout: 1000
  }).then(response => {
    const $ = cheerio.load(response.data.split('\"').join(''));
    const $movieinfo = $('.css-13h49w0-PaneInner');
    const nameKO = $movieinfo.find($('.css-13a04pq-Title')).text();
    const releaseYear = $movieinfo.find($('.css-w4pu2t-Detail')).text().split('·')[0];
    db.insertMovie({
      wid: watchaID[j],
      title: nameKO,
      released: releaseYear,
      lastUpdate: new Date(0)
    });
    return new Promise((resolve, reject) => {
      resolve(nameKO, releaseYear);
    });
  });
}
```



저에 대해 더 알고 싶나요? 자세한 정보는

<https://startergate.dev>