



롤 전적사이트
OP.GG Clone

LOLify

장성훈 박진우

LOLify 구조

백엔드

크롤러

OP.GG , FOW에서 데이터를 가져와요

MySQL 데이터베이스

챔피언 데이터, 소환사 데이터를 저장해요

API 서버

데이터 베이스에 필요한 정보를 업데이트하고
원하는 형태로 조회할 수 있게 해요

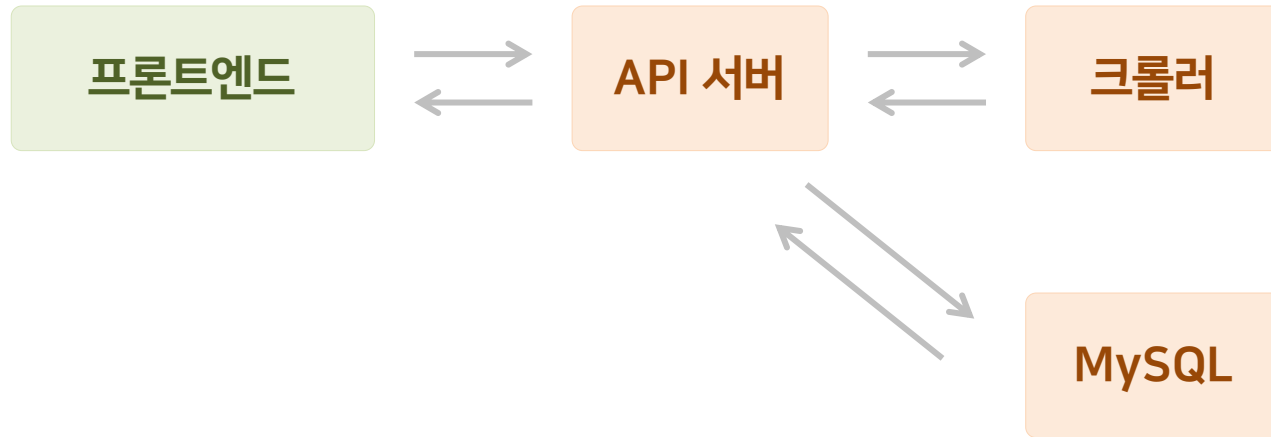
프론트엔드

리액트

다양한 UI를 렌더링해요.

백엔드에서 받은 데이터를
state로 설정해서
state가 변경됐을 때
적절하게 UI를
변경해요.

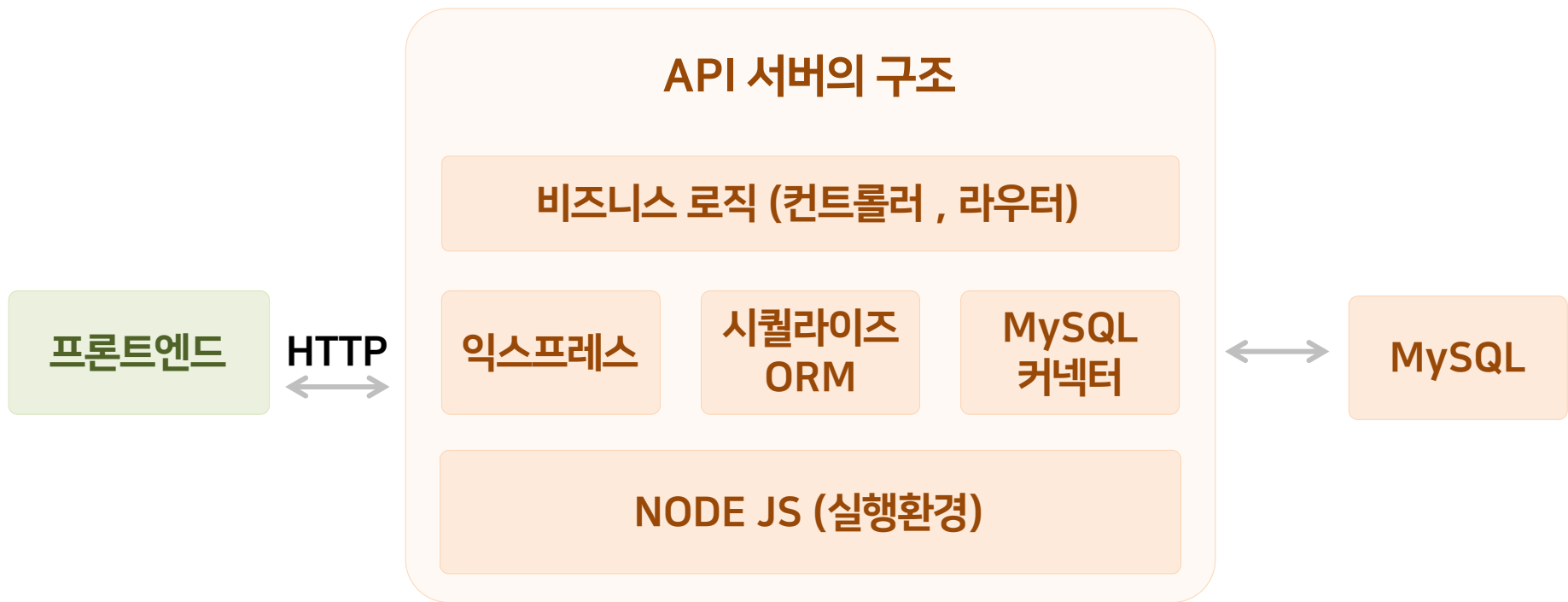
LOLify 구조



API 서버

API 서버는 크롤러가 수집해온 데이터를 MySQL에 보관했다가 클라이언트가 요청할 때마다 건네주는 역할을 해요.

API 서버는 용량이 큰 이미지파일, JSON 데이터 파일 등을 클라이언트가 요청할 때마다 건네주는 역할도 해요.



설명

Node js는 기본 실행환경 역할을 해요.

익스프레스는 해당 비즈니스 로직을 라우팅해요.

시퀀라이즈 ORM은 자바스크립트 객체와 데이터베이스 MySQL 테이블을 매칭해주는 라이브러리에요.

MySQL 커넥터는 말 그대로 ORM 과 MySQL을 연결(연동)해주는 라이브러리에요.

컨트롤러는 데이터를 조회(삽입, 업데이트, 삭제)하는 역할을 해요.

예를 들면, 챔피언 '아리'의 데이터를 조회(삽입, 업데이트, 삭제)하는 역할을 해요.

라우터는 특정 경로에 HTTP 요청이 왔을 때 해당 경로에 맞는 컨트롤러 함수로 요청을 전달하는 길안내 역할을 해요.

예를 들면, '아리' 라는 요청이 왔을 때, '챔피언/아리' 로 요청을 전달하는 거예요.

그리고 컨트롤러에서 요청을 처리하는 거죠.

크롤러

크롤러는 자동화된 방법으로

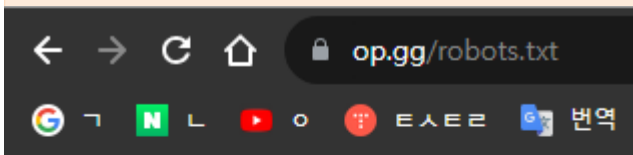
웹 페이지 전체 혹은 일부를 추출하여 정보를 얻는 프로그램이에요.

- 1 Axios 라이브러리를 통해서 HTML을 로드해요.
Cheerio 라이브러리를 통해 로드한 HTML 을 파싱해요. (완료)
- 2 Axios 라이브러리를 통해서 HTML을 로드해요.
Cheerio 라이브러리를 통해 로드한 HTML 을 파싱해서 DOM을 생성해요. (완료)
- 3 빈 객체를 선언하고, 객체 안에
CSS 셀렉터 문법을 사용해서 검색한 DOM요소를 할당해요. (완료)
- 4 DOM 요소가 들어있는 객체를
Sequelize(ORM)을 이용해서 MySQL에 저장해요. (미완료)

[문제!] 크롤링 불가능

OP.GG는 크롤링이 정책적으로는 가능했지만
기술적으로는 고난도이거나 불가능했어요.

OP.GG의 크롤링 봇 허용 여부 확인



User-agent: *
Allow: /

모든 봇은 모든 페이지에
접근가능하다는 뜻

Sitemap: <https://www.op.gg/sitemap.xml>

메인 HTML 내부에 콘텐츠가 없고,
데이터를 동적으로 불러오는 사이트

```
4      (function(t, l) {  
5          try {  
6              let w = window,  
7                  d = document,  
8                  s = d.createElement("script"),  
9                  f = d.getElementsByTagName("script")[0];  
10             w[t] =  
11                 w[t] ||  
12                     function() {  
13                         (w["_rgea"] = w["_rgea"] || [{"uts": new Date()}]).push(  
14                             Array.prototype.slice.call(arguments)  
15                         );  
16                         w.RGEA && w.RGEA.p();  
17                     };  
18             s.type = "text/javascript";  
19             s.async = true;  
20             s.defer = true;  
21             s.src = l;  
22             f.parentNode.insertBefore(s, f);  
23  
24             // 되는 파라미터  
25             rgea("propertyId", "RGEA0002-32de3446-a2a6-4f9a-a387-f40138212b2b");  
26             rgea("lolpid", "kr");  
27             rgea("anonymous", false);  
28             } catch (error) {  
29                 console.error(error);  
30             }  
31             })(("rgea", "https://static.developer.riotgames.com/js/rgea.min.js");  
32         </script><script>  
33             window.googletag = window.googletag || { cmd: [] };  
34         </script><script>  
35             window.dataLayer = window.dataLayer || [];  
36             window.gtag = function () {  
37                 dataLayer.push(arguments);  
38             }
```

[문제!] 크롤링 불가능

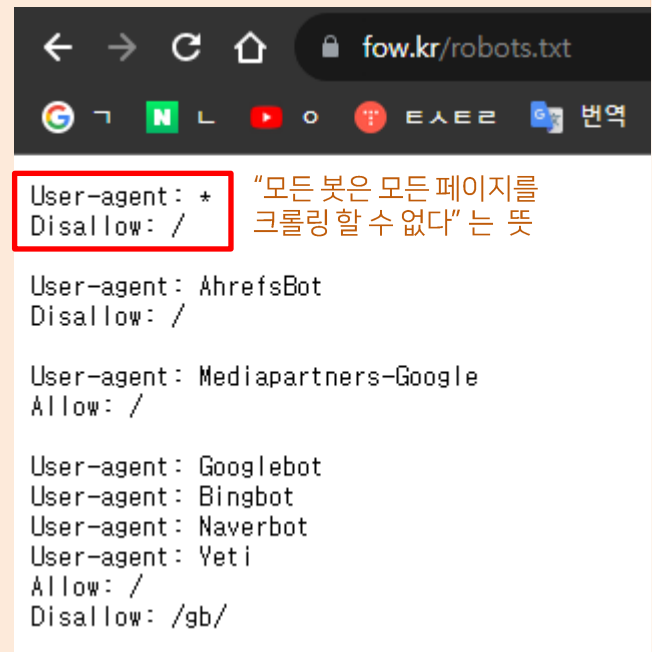
FOW는 특정 크롤링 봇을 제외한 모든 봇에게 크롤링을 허용하지 않았어요.
기술적으로는 가능했지만, 정책적으로 불가능했죠.

<페이지 소스보기> 후 보여지는
HTML에 접근해서 크롤링할 수 있는 사이트
(기술적으로는 가능했음)

```
<div style="display: block;" class="champ_tr" href="/stats/nacl0x/top">
  <div></div>
  <span>아트록스</span>
</div>
</td>
<td class="td_pos">
  <a style="display: block;" href="/stats/Aatrox/top">
     랭
  </a>
</td>
<td class="td_rate">49.7%</td>
<td class="td_rate">17.0%</td>
<td class="td_rate">14.7%</td>
<td class="td_rate">20,060</td>
<td class="td_rate">?</td>
<td class="td_rate"></td>
</tr><tr class="champ_one champ_tr even" position="jungle" rname="그레이브즈">
  <td class="td_champ">
```

<페이지 소스보기>에서
DOM 요소에 CSS 셀렉터 문법으로 접근

FOW의 크롤링 봇 허용 여부 불가 확인



```
User-agent: * "모든 봇은 모든 페이지를
Disallow: / 크롤링 할 수 없다" 는 뜻

User-agent: AhrefsBot
Disallow: /

User-agent: Mediapartners-Google
Allow: /

User-agent: Googlebot
User-agent: Bingbot
User-agent: Naverbot
User-agent: Yeti
Allow: /
Disallow: /gb/
```

크롤링 시도

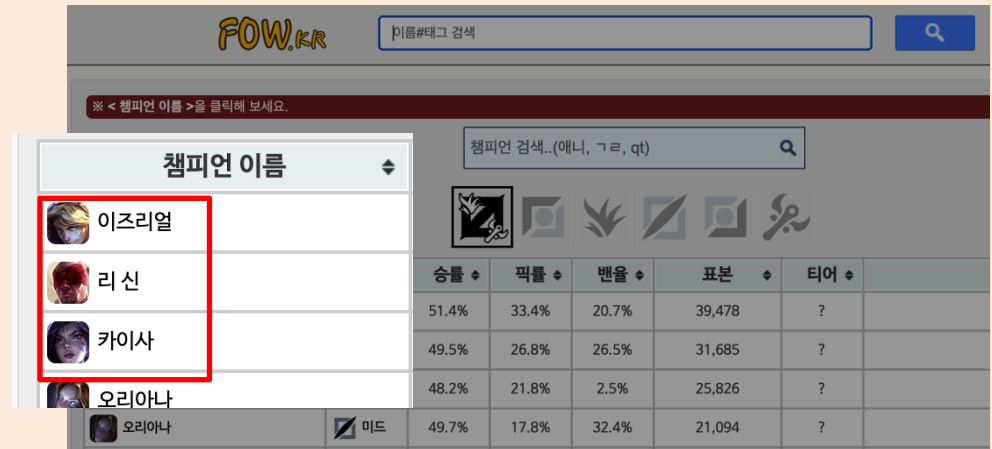
FOW는 기술적으로는 가능했기 때문에 크롤링을 해봤어요.

데이터 추출은 성공했지만 이 데이터들을 어떻게 정제하지? 라는 것에서 막혔어요.

크롤링한 결과 데이터

```
championName: '\n' +  
  '\n' +  
  '\n' +  
  이즈리얼 '\n' +  
  '\n' +  
  '\n' +  
  리 신 '\n' +  
  '\n' +  
  '\n' +  
  카이사 '\n' +
```

실제 웹 페이지



승률	픽률	밴율	표본	티어
51.4%	33.4%	20.7%	39,478	?
49.5%	26.8%	26.5%	31,685	?
48.2%	21.8%	2.5%	25,826	?
49.7%	17.8%	32.4%	21,094	?

크롤러

API 서버

MySQL

라이엇 API

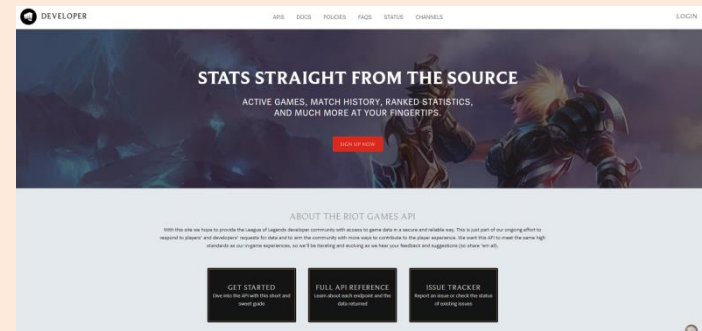
[문제!] 크롤링 하기 힘든 데이터

OP.GG , FOW 같은 전적검색 사이트에서도
Riot(게임사) API를 사용해서 데이터를 가져왔기 때문에
아래 그림처럼 갱신기간이 비교적 짧은 데이터는 크롤링하기가 힘들었어요.
그래서 우리도 Riot API를 사용하기로 합니다.

외부 api를 사용해서 크롤링이 힘든 데이터

무작위 총력전 한 시간 전		킬관여 59% CS 56 (2.8) Silver 4	분당CS1... 은신후회피 DTCW 나는요를... 죽전등 기...	see bara... 홍릉이와... 어셈블 골통상프... 하늘소여...
무작위 총력전 한 시간 전		킬관여 46% CS 35 (1.9) Gold 4	라디오티비 민정이와... 홍집에는 어셈블 엑날리야킹	피치떡콩... 말기행민이 어피지 eejejeje BLACKPL...
무작위 총력전 4시간 전		킬관여 63% CS 60 (2.7) Silver 2	pyopyopy... IAMATIO... 평방불명 어효소... 10만소녀...	장복례 리저드슨... 안길호 검곡 아자... 어셈블

라이엇 공식 홈페이지



API 서버 는 크롤러가 수집한 데이터를
My SQL에 보관했다가 요청할 때 건네주는 역할을 해요

API 서버는 익스프레스, 시퀄라이즈 ORM, MySQL 커넥터, node js로 구성돼요.

익스프레스 는 라우팅(페이지 전환) 역할을 해요

시퀄라이즈 ORM 은 챔피언 데이터, 소환사 데이터를 저장해요

My SQL 커넥터 는 챔피언 데이터, 소환사 데이터를 저장해요

NODE JS 는 데이터 베이스에 필요한 정보를 업데이트하고 원하는 형태로 조회할 수 있게 해요

My SQL 는 크롤러가 수집한 데이터를
보관하는 역할을 해요

API 서버는 익스프레스, 시퀄라이즈 ORM, MySQL 커넥터, node js로 구성돼요.

계정을 만들었어요

Lolify_admin 계정

데이터 베이스를 만들었어요

Lolify 데이터 베이스

```
(base) → ~ mysql -u lolify_admin -p
Enter password:
Welcome to the MySQL monitor.  Commands end
Your MySQL connection id is 8
Server version: 8.1.0 Homebrew

Copyright (c) 2000, 2023, Oracle and/or its
affiliates. Other names may be trademarks of
owners.

Type 'help;' or '\h' for help. Type '\c' to
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| lolify |
| performance_schema |
+-----+
3 rows in set (0.05 sec)

mysql> |
```

객체 모델 설계 했어요

ORM을 사용하지 않았다면 테이블을 설계해야 하지만
ORM을 사용하기 때문에 자바스크립트 코드로 사용할 객체모델만 설계하면 돼요.

객체 모델 코드

```
const { DataTypes } = require('sequelize');

module.exports = (sequelize) => {
  return sequelize.define(
    'Stat',
    {
      id: {
        autoIncrease: false,
        type: DataTypes.INTEGER.UNSIGNED,
        allowNull: false,
        primaryKey: true,
      },
      level: {
        type: DataTypes.INTEGER.UNSIGNED,
        allowNull: false,
      },
    },
  );
};
```

객체의 속성 정보 예시

용도	속성명	자료형	빈값허용
챔피언 코드	ChampionId	문자	X
이름	name	정수	X
체력	hp	정수	X
공격력	energy	정수	0
분노	rage	정수	0

데이터베이스와 ORM을 연결했어요

앞에서 준비한 객체 모델을 MySQL 서버와 연결시키면 돼요.

ORM을 사용하기 때문에 자바스크립트 코드로 사용할 객체모델만 설계하면 돼요.

연결하는 코드

```
const config = {
  host: process.env.LOLIFY_MYSQL_HOST || '127.0.0.1',
  port: 3306,
  database: 'lolify',
  user: 'lolify_admin',
  password: process.env.LOLIFY_MYSQL_PASSWORD || 'Wkd!!51349',
};

const sequelize = new Sequelize(config.database, config.user, config.password, {
  host: config.host,
  dialect: 'mysql',
});

module.exports = {
  sequelize,
  Champion: require('./champion.model')(sequelize),
  Item: require('./item.model')(sequelize),
  Stat: require('./stat.model')(sequelize),
  Summary: require('./summary.model')(sequelize),
};
```

객체 모델이 테이블이 된 모습

```
Database changed
mysql> show tables;
+-----+
| Tables_in_lolify |
+-----+
| Champion         |
| Item              |
| Stat              |
| Summary           |
+-----+
4 rows in set (0.00 sec)
```

```
mysql> desc champion;
```

Field	Type	Null	Key	Default	Extra
id	int unsigned	NO	PRI	NULL	auto_increment
championName	char(10)	NO		NULL	
runes	char(10)	NO		NULL	
skill	char(10)	NO		NULL	
spell	char(10)	NO		NULL	
item	char(10)	YES		NULL	
stat	int unsigned	NO		NULL	

```
7 rows in set (0.00 sec)
```

API 를 만들었어요

이제 서버에서 정의된 API를 호출해서 원하는 데이터를 저장하거나 조회할 수 있도록 연결해줬어요.
‘컨트롤러’란 “요청을 받고 요청에 맞게 로직을 수행한 후에 다시 응답을 돌려주는 역할을 하는 코드”예요.

데이터를 조회하는 컨트롤러 구현 코드 (GET)

```
const { Champion } = require('../models');

async function getAll(req, res) {
  const result = await Champion.findAll();
  res.status(200).json({ result });
}

module.exports = {
  getAll,
  insertOrUpdate,
  remove,
};
```

컨트롤러와 라우팅을 연결하는 코드

```
async function launchServer() {
  const app = express();
  app.use(bodyParser.json());

  app.get('/Champion', championController.getAll);
  app.get('/Item', itemController.getAll);

  try {
    await sequelize.sync();
    console.log('Database is ready!');
  } catch (error) {
    console.log('Unable to connect to the databases: ');
    console.log(error);
    process.exit(1);
  }

  const port = process.env.PORT || 8080;
  app.listen(port, () => {
    console.log(`${port} 번 포트에서 대기 중`);
  });
}

launchServer();
```

크롤러

API 서버

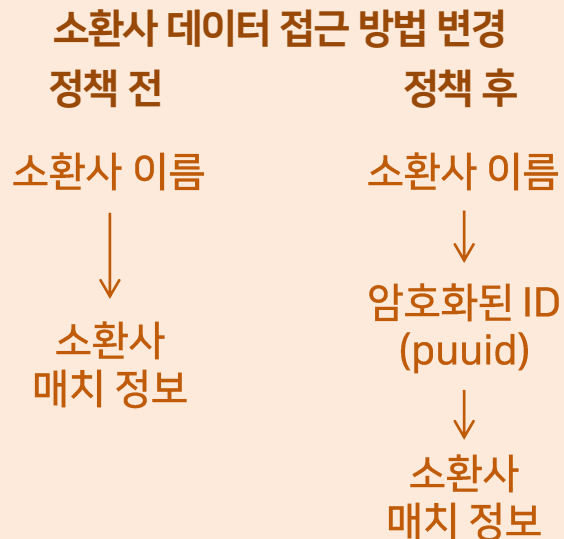
MySQL

라이엇 API

라이엇 API 공식문서

프로젝트 기간 도중에 소환사 이름 폐지 정책이 시행되었어요.. (11. 21)

그에 따라서 공식문서를 사용하는 방법도 복잡해졌어요.



소환사 이름 변경권 판매 중단 안내 및 Riot ID 관련 FAQ

11월 21일 13:21 폐지가 진행되면 소환사 이름 변경권 판매가 중단됩니다.

2개월 전

공식사항



플레이어 여러분,

2023년 11월 21일부로 소환사 이름의 사용을 단계적으로 폐지하고 라이엇의 모든 게임에 걸친 경험의 일관성을 제고하고자 모든 플레이어의 이름을 Riot ID만 사용하는 체제로 전환합니다.

이러한 변화에 따라 소환사 이름 변경권 판매도 중단될 예정입니다. 소환사 이름 변경권은 11월 21일부터 판매가 중단될 예정이기 때문에, 혹시 소환사 이름 변경을 고민하고 계셨던 분들은 해당 내용을 꼭 숙지하시고 신중하게 구매하시는 것을 권장드립니다.

♦ 소환사 이름 변경권 판매 중단 일시: **2023년 11월 21일 점검 종료 후**

더불어 향후 변경될 정책에 따라 플레이어 여러분들이 궁금해하시는 점들을 아래에 정리해두었으니 꼭 확인해보세요. 여전히 해결되지 않는 궁금증이 있다면 언제든 [고객지원](#)을 찾아주시기 바랍니다!

라이엇 API 노선정리

API 종류가 너무 많았기 때문에 필요한 데이터만 가져오기 위해서 공식문서 자체를 정리했어요.

어떤 API를 사용해야 원하는 데이터를 가져올 수 있는지 정리했어요.

The screenshot displays the Riot Games API documentation interface. On the left, a sidebar lists various API endpoints with their data types. The main content area shows the details for the **MATCH-V5** endpoint, including its parameters and a sample JSON response.

API 공식문서 ...

자료형 + 필터 추가

API 이름 (요약)	자료형
CHAMPION-V3 (공짜 챔피언 정보, 초보자 레벨 제한)	List[int] int
SUMMONER-V4 (소환사의 계정정보)	long string int
CHAMPION-MASTERY-V4 (챔피언 숙련도 정보)	string long boolean int
SPECTATOR-V4(플레이한 게임 정보)	long string List[...]
LEAGUE-V4 (유저의 티어 등수 정보)	string boolean int
LOR-RANKED-V1(마스터 티어의 플레이어 리스트 정보)	
CLASH-V1 (현재 활성화된 플레이 정보)	
MATCH-V5(유저 matchid, matchData)	string long boolean int List[...]

MATCH-V5(유저 matchid, matchData)

자료형 string long boolean int List[...]

+ 속성 추가

댓글 추가

👉 유저의 puuid를 입력받아서 해당 유저가 플레이한 판id를 가져오네요. 이거 써야 하네요!!

/lol/match/v5/matches/by-puuid/{puuid}/ids

결과

```
[
  "KR_6804240897",
  "KR_6804173281",
  "KR_6804125628",
  "KR_6804045290",
  "KR_6802296849",
  "KR_6802256864",
  "KR_6802154986"
]
```


프론트엔드 구현에는 시간이 부족

백엔드에 상대적으로 시간을 쏟다보니, 프론트엔드 구현은 시간이 부족했어요.
어려웠던 점, 힘들었던 점에 대해서 말씀을 드릴게요.

리액트는 싱글페이지앱인데,
여러 개의 페이지를 만든다고 생각하고
라우터로 처리해도 되는건가?
된다면 어떻게 코드를 작성해야 하지?
안 된다면 어떻게 코드를 수정해야 하지?

MySQL에 저장한 데이터는 어떻게
리액트에서 불러와서 렌더링하지?

챔피언별 승률, 픽률, 밴율 데이터(크롤링 -> MySQL)
챔피언별 정보(HP, MP..), 이미지(스킬이미지, 챔피언썸네일..) 파일(서버 public 폴더)
소환사별 puuid, 매치정보(라이엇 API)를 클라이언트에서 Fetch 해야 하는데
라이엇 API는 "puuid -> 매치ID -> 매치 정보" 로 순서가 있는 api인데 어떻게 코드를
짜야 하지?

리액트

정적파일

MySQL

라이엇 API

현재는 정적파일인 챔피언 정보(챔피언ID, HP, MP, 공격력 등), 챔피언 이미지는 화면에 렌더링 가능해요.

그런데 크롤링한 데이터, 라이엇 API는 렌더링을 어떻게 해야 하는지 모르겠어요.

각각 MySQL 데이터에 접근하는 것, 하나의 url의 response.body가 다시 다른 url에 request.body가 되어야 하는데 어떻게 해야 하는지 모르겠어요.