


# 원티드 백엔드 기업과제

|       |   |
|-------|---|
| 📅 날짜  | @2022년 6월 14일   |
| 👤 작성자 |  정이든 |
| 🕒 속성  | 진행중   |
| ☰ 유형  |   |

## 기능 명세

보스레이드 PVE 콘텐츠 관련 아래 6가지 라우터를 작성해주세요.

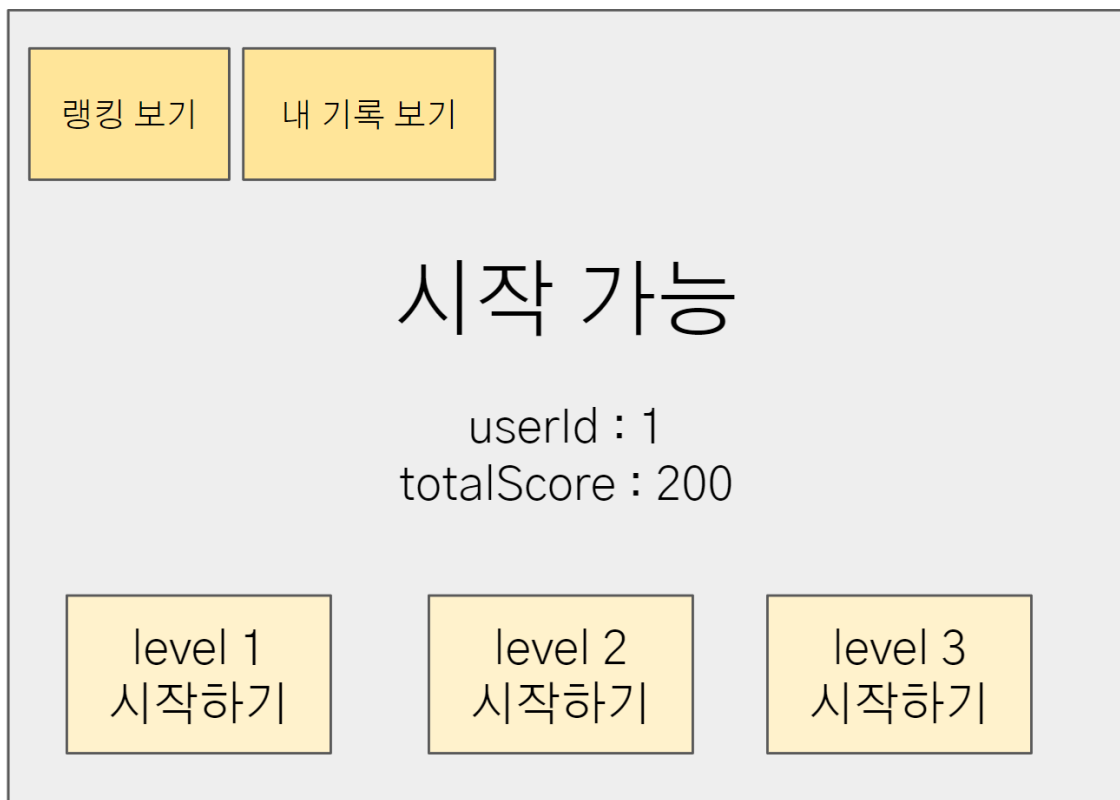
- 유저 생성
- 유저 조회
- 보스레이드 상태 조회
- 보스레이드 시작
- 보스레이드 종료
- 랭킹 조회

## 라우터 로직 설명

아래 REST API 명세서 규칙을 준수하여 구현해주세요.

- 유저 생성
  - 중복되지 않는 userId를 생성
  - 생성된 userId를 응답
- 유저 조회
  - 해당 유저의 보스레이드 총 점수와 참여기록 응답
- 보스레이드 상태 조회
  - 보스레이드 현재 상태 응답

- canEnter : 입장 가능한지
- enteredUserId : 현재 진행중인 유저가 있다면, 해당 유저의 id
- 입장 가능 조건 : 한번에 한 명의 유저만 보스레이드를 진행할 수 있습니다.
  - 아무도 보스레이드를 시작한 기록이 없다면 시작 가능합니다.
  - 시작한 기록이 있다면 마지막으로 시작한 유저가 보스레이드를 종료했거나, 시작한 시간으로부터 **레이드 제한시간**만큼 경과되었어야 합니다.



[클라이언트 화면 예시] 입장 가능



[클라이언트 화면 예시] 입장 불가 ( userId 2에 해당하는 유저가 시작 후 아직 종료하지 않았음 )

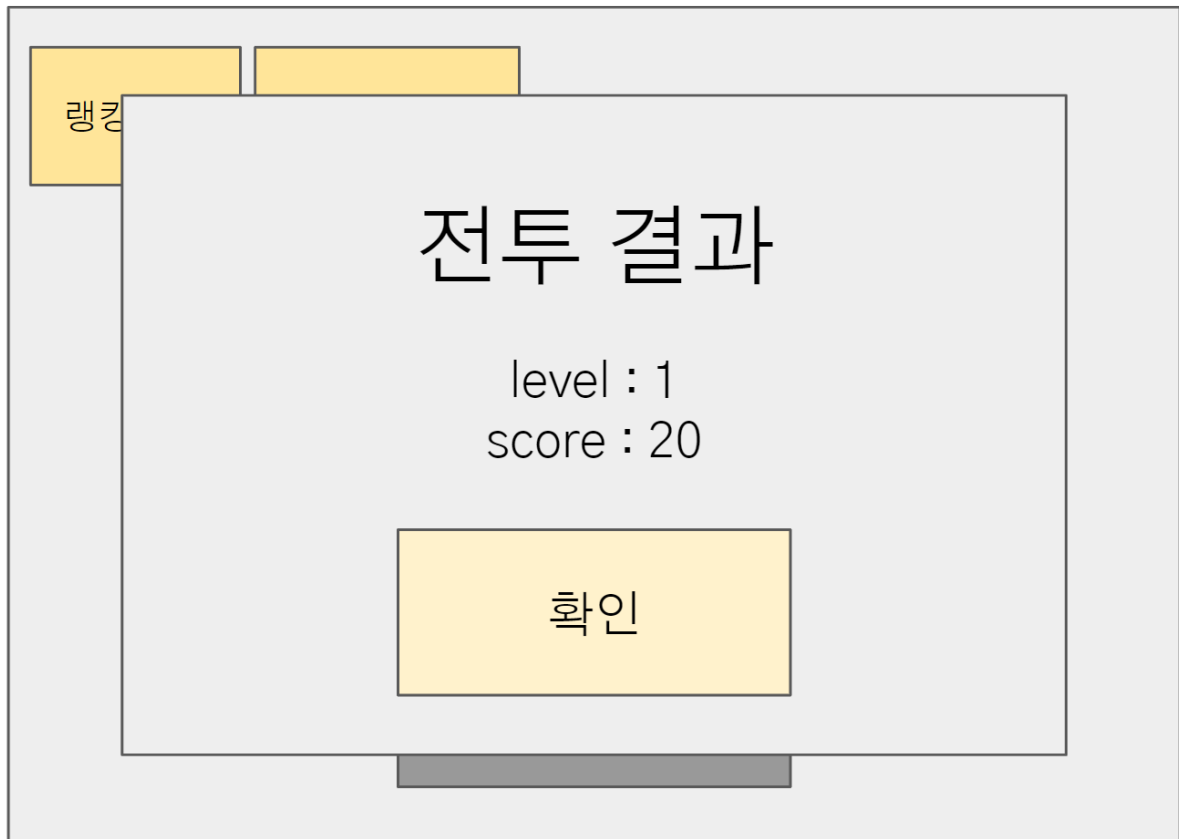
- **보스레이드 시작**

- 레이드 시작 가능하다면 중복되지 않는 raidRecordId를 생성하여 isEntered:true와 함께 응답
- 레이드 시작이 불가하다면 isEntered : false

- **보스레이드 종료**

- raidRecordId 종료 처리
  - 레이드 level에 따른 score 반영
- 유효성 검사
  - 저장된 userId와 raidRecordId 일치하지 않다면 예외 처리

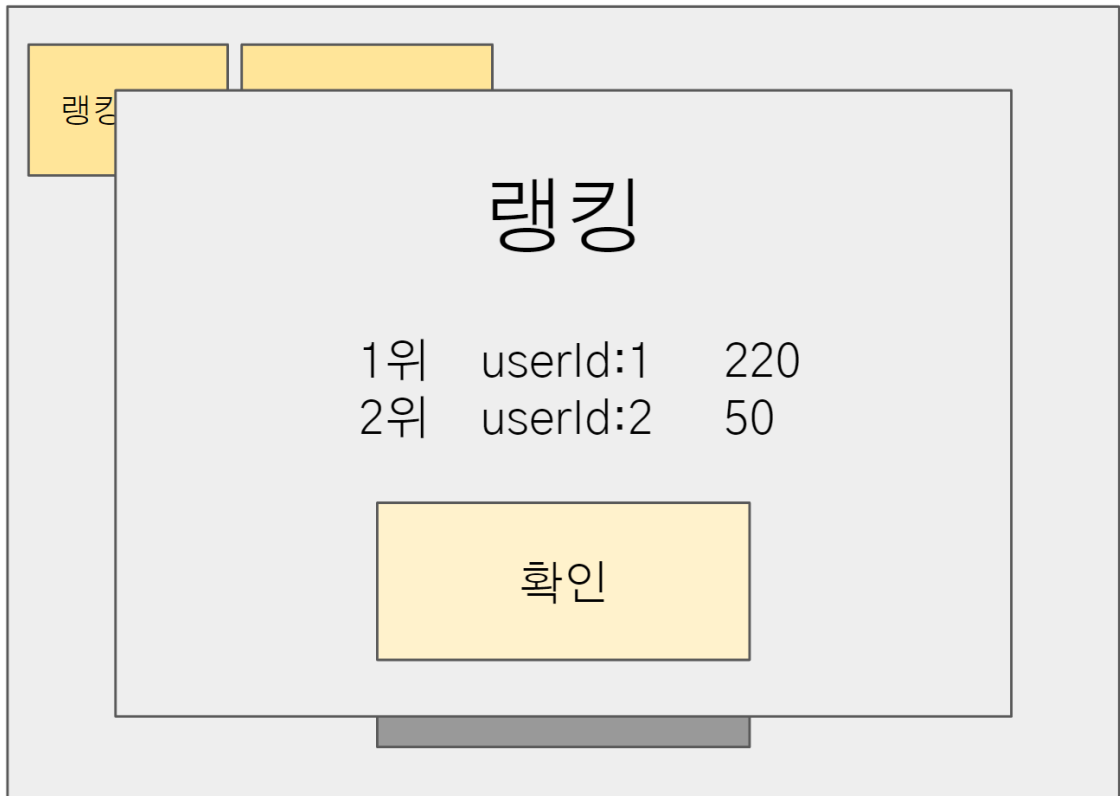
- 시작한 시간으로부터 **레이드 제한시간**이 지났다면 예외 처리



[클라이언트 화면 예시]

- **보스레이드 랭킹 조회**

- 보스레이드 **totalScore** 내림차순으로 랭킹을 조회합니다.



[클라이언트 화면 예시]

## REST API 명세서

### 유저 생성

[POST] {{BASE\_URL}}/user

```
request.body
{
}
```

```
response
{
  userId:number
}
```

## 유저 조회

```
[GET] {{BASE_URL}}/user/[userId]

request.body
{
}

response
{
  totalScore:number,
  bossRaidHistory: [
    { raidRecordId:number, score:number, enterTime:string, endTime:string },
    //..
  ]
}
```

## 보스레이드 상태 조회

```
[GET] {{BASE_URL}}/bossRaid

request.body
{
}

response
{
  canEnter:boolean,
  enteredUserId:number
}
```

## 보스레이드 시작

```
[POST] {{BASE_URL}}/bossRaid/enter

request.body
{
  userId: number;
  level: number;
}

response
```

```
{
  isEntered: boolean;
  raidRecordId:number;
}
```

## 보스레이드 종료

```
[PATCH] {{BASE_URL}}/bossRaid/end

request.body
{
  userId: number;
  raidRecordId:number;
}

response
{
}
```

## 보스레이드 랭킹 조회

```
{{BASE_URL}}/bossRaid/topRankerList

request.body
{
  userId: number;
}

response
{
  topRankerInfoList: RankingInfo[]
  myRankingInfo: RankingInfo
}
```

```
interface RankingInfo {
  ranking: number; // 랭킹 1위의 ranking 값은 0입니다.
  userId: number;
  totalScore: number;
}
```

### 고려 사항

- 랭킹 데이터는 웹 서버에 캐싱하거나 레디스에 캐싱하여 구현해주세요.

## \*Static Data 활용

<https://dmpilf5svl7rv.cloudfront.net/assignment/backend/bossRaidData.json>

보스레이드의 정적 데이터를 담고있는 S3 오브젝트의 주소입니다.

API 비즈니스 로직 내부에서 위 S3 오브젝트의 응답값을 활용하여 코드를 작성해주세요.

```
{
  "bossRaids": [
    {
      "bossRaidLimitSeconds": 180,
      "levels": [
        {
          "level": 0,
          "score": 20
        },
        {
          "level": 1,
          "score": 47
        },
        {
          "level": 2,
          "score": 85
        }
      ]
    }
  ]
}
```

### 데이터 설명

- **bossRaidLimitSeconds** : 제한시간입니다. 위의 경우 3분입니다.
- **levels** : 레벨 별 레이드 처치 점수 입니다.



## 선택 구현

- 레디스를 사용하여 랭킹 기능 구현 시 가산점
- staticData 웹서버 캐싱 고려 시 가산점
- NestJS 프레임워크 사용 시 가산점

## 평가 요소

- 요구한 기능들이 잘 작동하는가
- 동시성을 고려하는가
- 레이어 계층을 잘 분리하는가
- 발생할 수 있는 다양한 에러 상황을 잘 처리 했는가