

# RESTFul

## API란

- 약속, 규약으로 통신에 참여하는 애플리케이션(백, 프론트) 간의 계약
- 종류
  - REST API (일반적인 소통)
  - WebSocket API (실시간 소통)
  - GraphQL API (다계층적인 데이터 ex) 로그, 시청자 통계 등)

## RESTful (Represtatinal State Transfer)

- REST 아키텍처 스타일의 규칙을 지켜 디자인한 API

## RESTful API

- REST 아키텍처의 규칙들을 정확하게 지켜서 만든 API
- 너무 빡빡해서 다 지키기 어렵고, 오히려 개발 생산성이나 보안에 취약해질 수 있다

## REST API

- 다 지키는 경우에 오히려 비효율적일 수 있으니 restful에서 주장하는 몇 가지만 알아서 쓰고 규칙을 좀 줄여서 적당히 융통성있게 만든 API

## REST 원칙

1. client - server로 나누어 구성
  - 프론트, 백엔드 구분
2. Stateless 하게 설계
  - 백엔드 세션 사용 X
  - 백엔드 서버를 여러 개 확장 가능 (Scale-out)
3. Cache 사용

#### 4. Layered Systemm

- 각 로직 별 서버 분리

#### 5. uniform interface

- 일괄적인 규약을 쓰라는 원칙
- URI 엔드 포인트 생성
- 기능을 메소드로 구분

### 세부규칙

#### 1. identification of resource

- 모든 자원은 고유한 식별자(id)를 가지고 있어야 한다.

#### 2. Resource manipulation through representation

- 정보의 원천과 표현은 달라도 된다
- API를 DB에 종속적으로 구현 X

/find/password	POST	header: {}  body: { name: String, email: String }	{ password: String }	비밀번호 찾기
----------------	------	--	----------------------------	---------

#### 3. self-descriptive messages

- 응답에 필요한 내용을 header에 포함
- 상태 코드를 너무 구체적으로 쓰면 보안에 취약해짐
- 실패했는데 200 주는 행동 하지 마라!

#### 4. heateoas

- 추가적으로 접근할 수 있는 엔드 포인트도 함께 보내는 규칙
- Github API 가 이런 식으로 제공함

#### 6. code-on-demand (선택 사항, 거의 안 지킴)

- 클라이언트에 코드 전송해서 기능 확장, 수정 가능

- 클라이언트가 백엔드에 의존하게 되고 보안 이슈 발생