instagram_API

API(Application Programming Interface)

- 응용 프로그램 프로그래밍 인터페이스

- 서로 다른 소프트웨어나 시스템이 데이터를 주고받고 기능을 사용할 수 있게 해주는 규칙이나 통로

🍔 음식점 비유

- 당신이 음식점에 갔어요. 메뉴를 보고 음식을 주문하죠.
- 이때, **당신(사용자)** ↔ **주방(서비스)** 사이를 연결해주는 게 **웨이터(API)**입니다.
- 웨이터는 당신이 원하는 걸 주방에 전달하고, 주방이 만든 음식을 다시 당신에게 가져다줘요.

양식

1. 제목
2. Endpoint
3. Request body (필드 설명 + JSON 예시)
4. Description (동작 설명 및 제약 조건)
5. Response body (성공/실패 구조와 예시 JSON)

- 사용자 계정
    - 사용자 생성
    - 사용자 인증 (로그인)
    - 사용자 정보 조회
    - 사용자 정보 수정
    - 사용자 삭제
- 포스팅
    - 포스트 올리기
    - 올라온 포스트 조회하기
    - 포스트의 커맨트 조회하기
    - 특정 포스트에 커맨트 달기
- 소셜
    - 다른 사용자 조회
    - 팔로우 신청
    - 팔로우한 목록을 조회
    - 자신에게 팔로우 요청한 목록을 조회
    - 팔로우를 수락/거절
- 메시지
    - DM을 보내기
    - DM 조회하기
    - DM 삭제하기

👤 사용자 계정

👤 사용자 생성

Endpoint : POST /users

Request body

필드	타입	필수	설명
nickname	string	✔	사용자 닉네임 (고유해야 함)
name	string	✔	사용자 이름
password	string	✔	사용자 비밀번호
age	int	✘	사용자 나이
email	string	✘	사용자 이메일 주소

```
{
  "nickname": "kevin",
  "name": "이승학",
  "password": "1234",
  "email": "kevin.spreatics@gmail.com"
}
```

Description

사용자 계정을 생성한다. nickname, name, password는 필수 입력값이다.

nickname은 고유한 값이며, 기존 사용자와 중복되면 생성이 실패한다.

Response body

필드	타입	설명
status	string	"created" 또는 "failed"
user_id	int	생성 성공 시 반환
reason	string	실패 시 실패 사유

성공 예시	실패 예시

{
"status": "created",
"user_id": 105
}
	{
	"status": "failed",
	"reason": "nickname, kevin is duplicated"
	}

사용자 인증 (로그인)

Endpoint : POST /auth/login

Request body

필드	타입	필수	설명
nickname	string	✔	사용자 닉네임
password	string	✔	사용자 비밀번호

```
{
  "nickname": "kevin",
  "password": "1234"
}
```

Description

사용자가 로그인 요청을 보낸다.

닉네임과 비밀번호가 일치하면 토큰을 반환한다.

Response body

필드	타입	설명
status	string	"authenticated" 또는 "failed"
token	string	인증 성공 시 반환되는 JWT
reason	string	실패 시 실패 사유

성공 예시	실패 예시

{
"status": "authenticated",
"token": "eyJhbGciOiJIUzI1NiIsInR..."
}
	{
	"status": "failed",
	"reason": "Invalid credentials"
	}

사용자 정보 조회

Endpoint : GET /users/{user_id}

Request params

이름	타입	필수	설명
user_id	int	✔	조회할 사용자 ID

Description : 특정 사용자 ID로 사용자 정보를 조회한다.

Response body

```
{
  "user_id": 105,
  "nickname": "kevin",
  "name": "이승학",
  "email": "kevin.spreatics@gmail.com",
  "age": 25
}
```

👤 사용자 정보 수정

Endpoint : PUT /users/{user_id}

🔰 Path Parameter

이름	타입	필수	설명
user_id	int	✔	수정할 사용자 ID

Request body

필드	타입	필수	설명
name	string	✖	사용자 이름 수정
password	string	✖	비밀번호 수정
age	int	✖	나이 수정
email	string	✖	이메일 주소 수정

```
{
  "email": "newemail@example.com",
  "age": 26
}
```

Description : 사용자의 정보를 수정한다. 수정할 항목만 포함하면 된다.

Response body

필드	타입	설명
status	string	요청이 성공했는지 여부 ("updated")

```
{
  "status": "updated"
}
```

👤 사용자 삭제

HTTP 메서드 : DELETE

Endpoint (요청 주소) : /users/{user_id}

Description : 해당 사용자를 시스템에서 완전히 삭제한다.

📩 Path Parameter

이름	타입	필수	설명
user_id	int	✔	삭제할 사용자 ID

Response body

필드	타입	설명
status	string	"deleted" – 삭제 성공 여부 표시

```
{
  "status": "deleted"
}
```

🖼 포스팅

- 포스트 올리기

Endpoint : POST /posts

Request body

필드	타입	필수	설명
user_id	int	✔	게시물 작성자 ID
image_url	string	✔	업로드할 이미지 URL
caption	string	✘	게시물에 첨부할 설명

```
{
  "user_id": 105,
  "image_url": "https://cdn.example.com/image1.jpg",
  "caption": "오늘 날씨 완전 최고!"
}
```

Description

사용자가 새로운 게시물을 업로드한다.

이미지는 필수이며, 캡션은 선택 사항이다.

Response body

필드	타입	설명
status	string	created 또는 failed
post_id	int	생성된 포스트 ID
reason	string	실패 사유 (옵션)

성공 예시	실패 예시
{ "status": "created", "post_id": 3001 }	{ "status": "failed", "reason": "Missing image_url" }

- 올라온 포스트 조회하기

Endpoint : GET /posts?user_id={user_id}

Request params

이름	타입	필수	설명
user_id	int	✔	해당 사용자의 ID

Description : 특정 사용자가 올린 게시물 목록을 조회한다.

Response body


```
[
  {
    "post_id": 3001,
    "image_url": "https://cdn.example.com/image1.jpg",
    "caption": "오늘 날씨 완전 최고!",
    "created_at": "2025-06-13T15:20:00"
  },
  {
    "post_id": 3002,
    "image_url": "https://cdn.example.com/image2.jpg",
    "caption": "점심은 김치찌개",
    "created_at": "2025-06-13T18:10:00"
  }
]
```

- 포스트의 커맨트 조회하기

Endpoint : GET /posts/{post_id}/comments

Request params

이름	타입	필수	설명
post_id	int	✔	댓글을 볼 포스트 ID

Description : 특정 포스트에 달린 모든 댓글을 조회한다.

Response body

```
[
  {
    "comment_id": 501,
    "user_id": 108,
    "nickname": "jane",
    "comment": "와 진짜 예뻐요!",
    "created_at": "2025-06-13T16:00:00"
  },
  {
    "comment_id": 502,
    "user_id": 110,
    "nickname": "minsu",
    "comment": "어디예요 여기?",
    "created_at": "2025-06-13T16:01:30"
  }
]
```

- 특정 포스트에 커맨트 달기

Endpoint : POST /posts/{post_id}/comment

Request body

필드	타입	필수	설명
user_id	int	✔	댓글 작성자 ID
comment	string	✔	댓글 내용

```
{
  "user_id": 110,
  "comment": "와 여기 분위기 좋아보여요!"
}
```

Description : 사용자가 특정 포스트에 댓글을 작성한다.

Response body

필드	타입	설명
status	string	"created" 또는 "failed"
comment_id	int	생성된 댓글 ID (성공 시)
reason	string	실패 시 오류 메시지

성공 예시	실패 예시

{
"status": "created",
"comment_id": 503
}
	{
	"status": "failed",
	"reason": "Comment cannot be empty"
	}

소셜

다른 사용자 조회

Endpoint : GET /users?keyword={keyword}

Request Params

이름	타입	필수	설명
keyword	string	✔	사용자 닉네임 또는 이름 검색 키워드

Description : 사용자를 닉네임이나 이름 기준으로 검색합니다.

Response Body

```
[
  {
    "user_id": 110,
    "nickname": "minsu",
    "profile_image": "https://cdn.example.com/profiles/minsu.jpg"
  },
  ...
]
```

팔로우 신청

Endpoint : POST /follow/request

Request Body

```
{
  "from_user_id": 105,
  "to_user_id": 110
}
```

필드	타입	필수	설명
from_user_id	int	✔	팔로우 요청을 보낸 사용자 ID
to_user_id	int	✔	팔로우 대상 사용자 ID

Response Body

```
{
  "status": "requested"   // 또는 "already_following", "failed"
}
```

내가 팔로우한 목록 조회

Endpoint : GET /followings?user_id={user_id}

Request Params

이름	타입	필수	설명
user_id	int	✔	조회할 사용자 ID

Response Body

```
[
  {
    "user_id": 110,
    "nickname": "minsu",
    "profile_image": "https://cdn.example.com/profiles/minsu.jpg"
  },
  ...
]
```

📥 나에게 팔로우 요청한 목록 조회

Endpoint : GET /follow/requests?user_id={user_id}

Response Body

```
[
  {
    "request_id": 2001,
    "from_user_id": 108,
    "nickname": "jane",
    "profile_image": "https://cdn.example.com/profiles/jane.jpg"
  },
  ...
]
```

✅ 팔로우 수락 / ❌ 거절

Endpoint : POST /follow/respond

Request Body

```
{
  "request_id": 2001,
  "action": "accept" // 또는 "reject"
}
```

Response Body

```
{
  "status": "accepted" // 또는 "rejected", "failed"
}
```

메시지

💬 DM 보내기

Endpoint : POST /dm

Request Body

```
{
  "from_user_id": 105,
  "to_user_id": 110,
  "message": "안녕하세요! 반가워요"
}
```

Response Body

```
{
  "status": "sent",
  "dm_id": 9001
}
```

📨 DM 조회하기

Endpoint : GET /dm/conversation?user1_id={id1}&user2_id={id2}

Response Body

```
[
  {
    "dm_id": 9001,
    "from_user_id": 105,
    "message": "안녕하세요! 반가워요",
    "sent_at": "2025-06-13T16:30:00"
  },
  ...
]
```

❌ DM 삭제하기

Endpoint : DELETE /dm/{dm_id}

Response Body

```
{
   "status": "deleted"
}
```