


액세스 토큰 생성과 검증

JWT를 구현하면 애플리케이션의 보안을 한 단계 더 강화할 수 있다. [그림 7-4]와 같이 토큰의 페이로드는 사용자 ID와 만료 시간으로 구성되며 하나의 긴 문자열로 인코딩된다.

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOi..."
}
```



사용자 ID	62655d4b52b6386b8b11b5fb
만료 시간	datetime.datetime(2022, 4, 24, 15, 39, 16, 630218)

그림 7-4 JWT 구조

JWT는 서버와 클라이언트만 아는 비밀키 secret key를 사용해 사인된다. `database/connection.py` 파일의 `Settings` 클래스와 환경 파일인 `.env`에 비밀키를 저장할 `SECRET_KEY` 변수를 설정해보자. `database/connection.py`에 다음과 같이 `SECRET_KEY` 변수를 추가한다.

```
class Settings(BaseSettings):
    SECRET_KEY: Optional[str] = None
```

`.env` 파일에 다음과 같이 `SECRET_KEY` 변수를 추가한다.

```
SECRET_KEY=Hl5HL3V3L$3CR3T
```

이번에는 `jwt_handler.py` 파일을 변경해보자.

```
import time
from datetime import datetime

from fastapi import HTTPException, status
from jose import jwt, JWTError
from database.connection import Settings
```