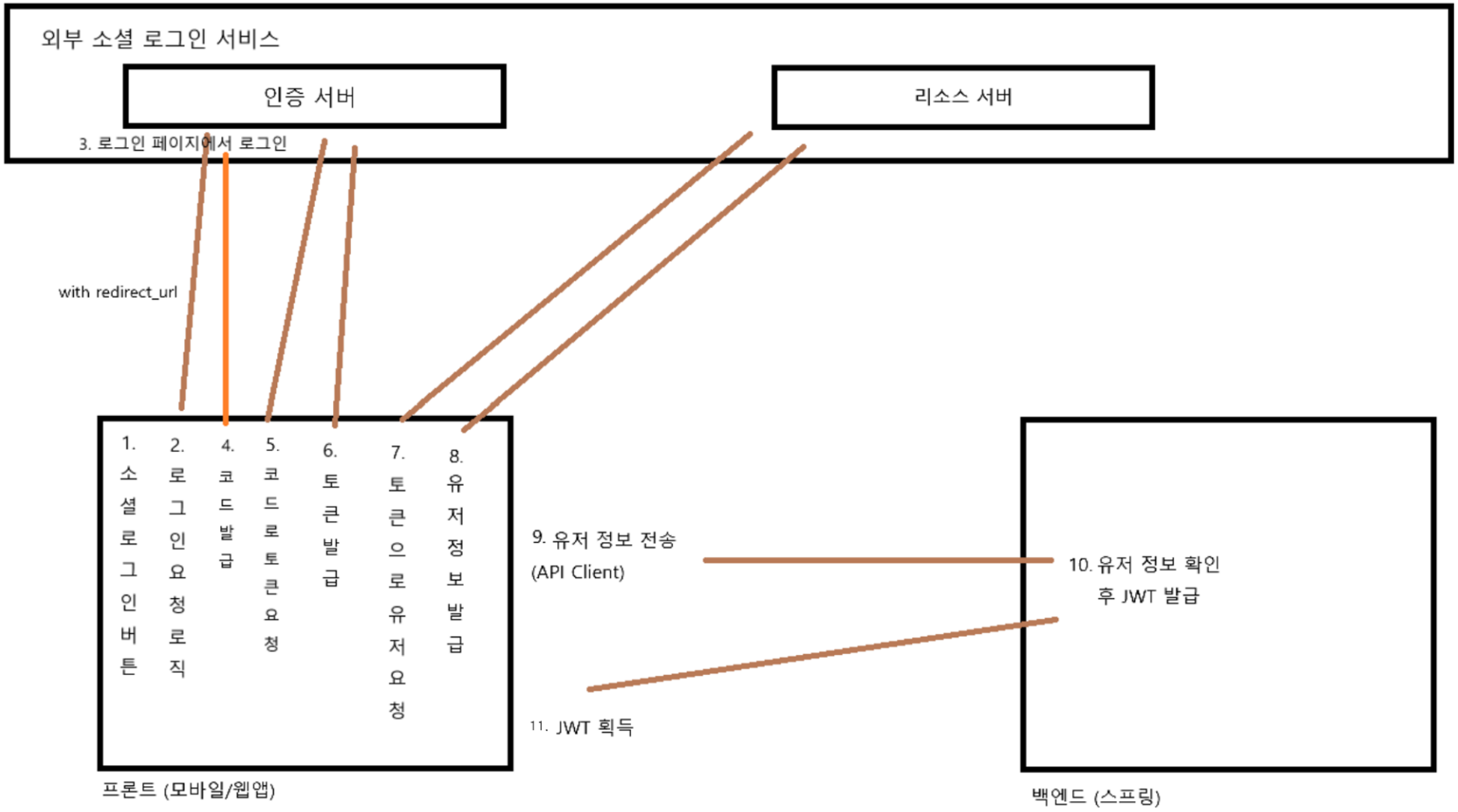
09

OAuth2 JWT

01. 기본 동작

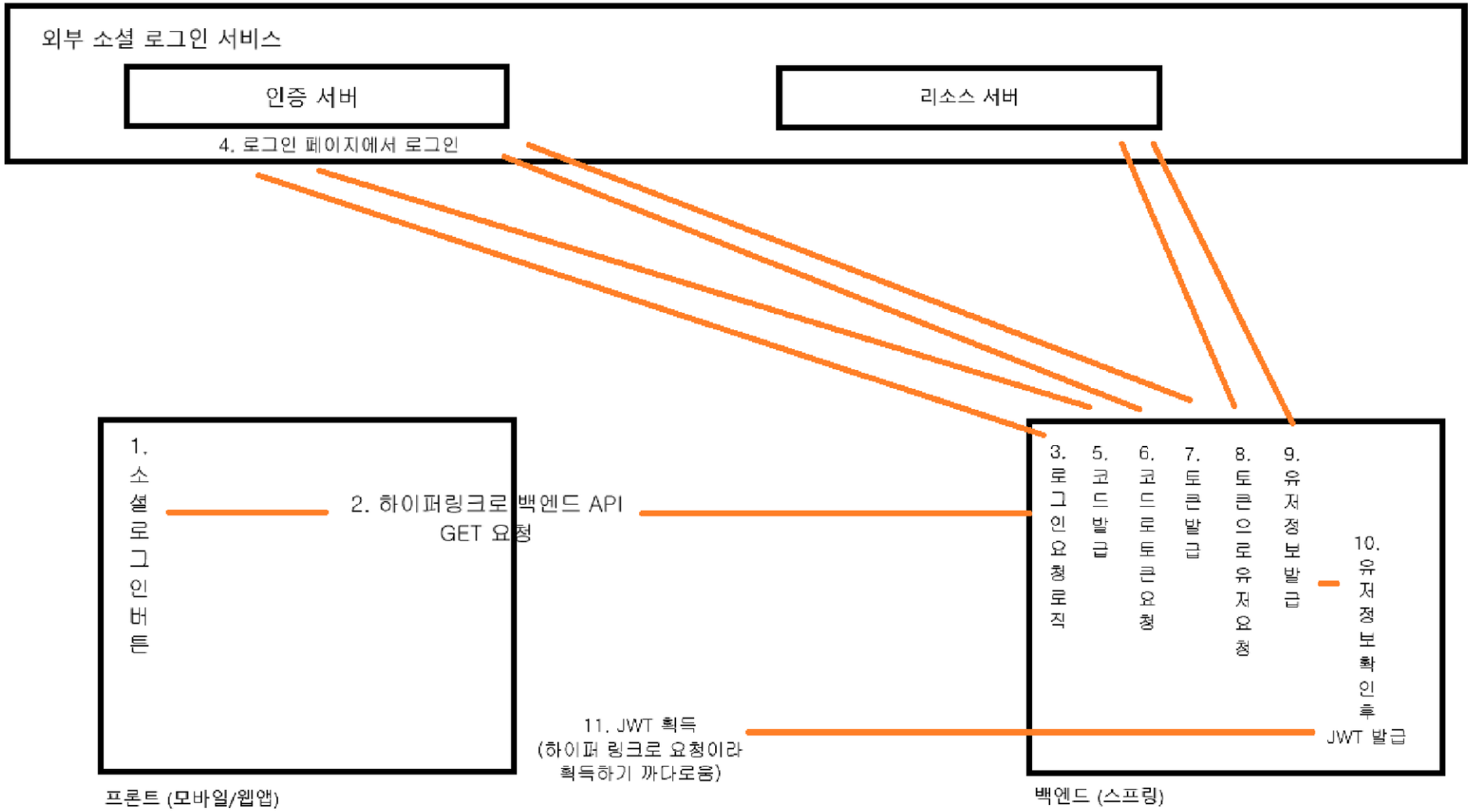
- 클라이언트 로그인 요청
- 소셜 인증 서버에 로그인 요청
- 소셜 인증 서버의 로그인 인증 완료 후 인가 코드 발급
- 인가 코드를 통해 다시 소셜 인증 서버에게 Access토큰 발급 요청
- 발급 받은 Access토큰을 통해 소셜 리소스 서버에 유저 정보 획득
- 획득한 유저정보로 JWT 클라이언트에게 발급

02. Front/Back 책임분배 (Frontend 중심방식)



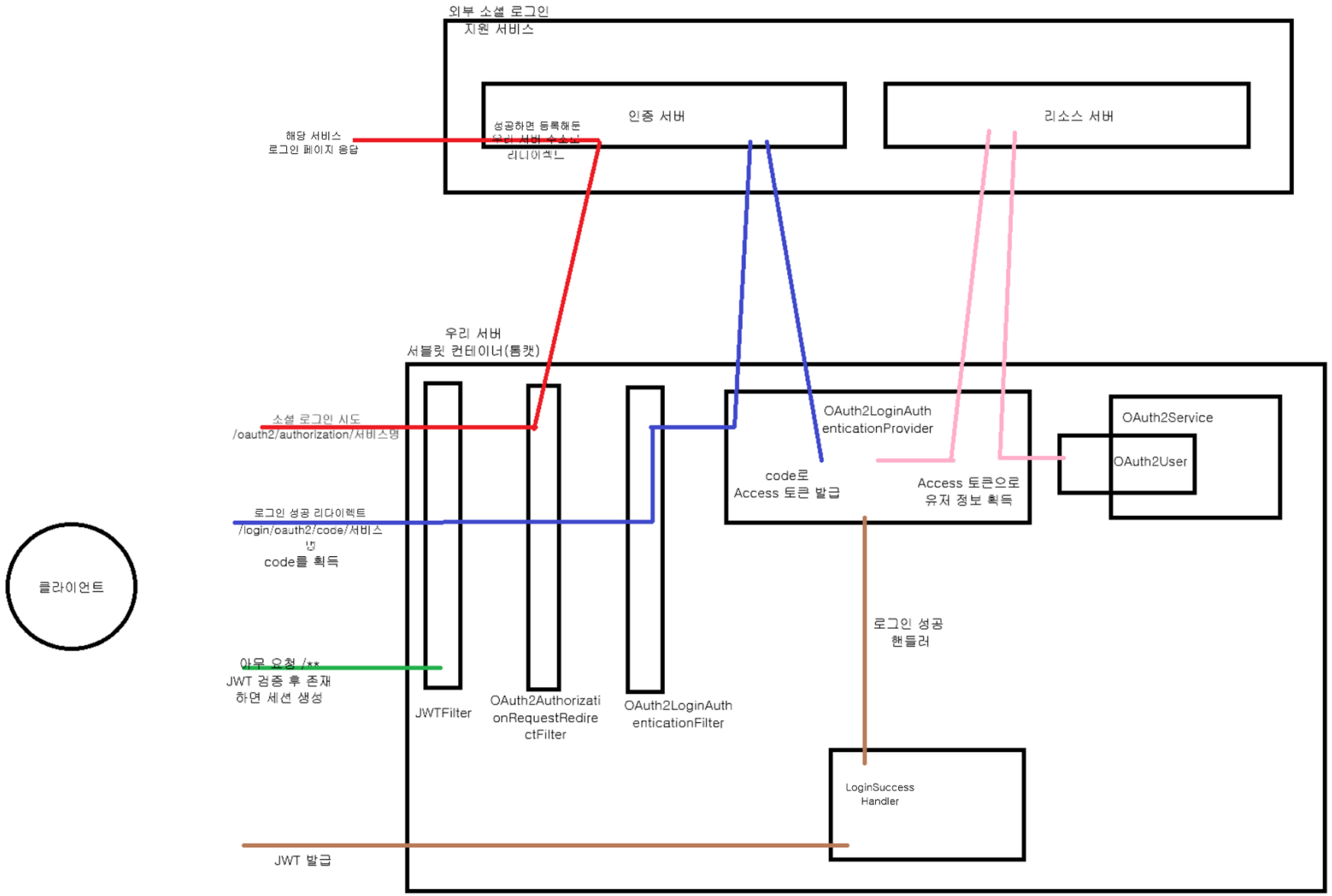
주로 네이티브 앱에서 사용하는 방식

02. Front/Back 책임분배 (BackEnd 중심방식)



웹앱/모바일앱 통합 환경 서버에서 사용하는 방식

03. 동작원리(브라우저 리다이렉트 방식)



04. 인증 흐름

1. 클라이언트 인증 요청(/oauth2/authorization/naver)
2. 애플리케이션 서버는 클라이언트를 소셜 인증 서버 로그인페이지로 리다이렉트
3. 소셜 인증 서버의 로그인 인증이 끝나면 클라이언트를 애플리케이션 서버가 미리 등록해둔 콜백 주소로 인가 코드와 함께 다시 리다이렉트
   (https://yourapp.com/callback?code=abcd1234&state=xyz)
4. 애플리케이션 서버는 인가 코드로 소셜 인증 서버에게 ACCESS 토큰을 발급받음
5. ACCESS 토큰으로 소셜 리소스 서버에게 유저정보를 획득
6. 획득한 유저정보를 이용하여 JWT 토큰을 클라이언트에게 발급

05. Github secrets변수 및 Springboot전역 변수 설정

- Github – Settings – Security – Secrets and Variables – actions –Repository secrets
- GOOGLE_CLIENT_ID, GOOGLE_CLIENT_SECRET, NAVER_CLIENT_ID, NAVER_CLIENT_SECRET, KAKAO_CLIENT_ID, KAKAO_CLIENT_SECRET 등록

application_properties

```
server.ip=http://3.39.100.13

google.client-id=${GOOGLE_CLIENT_ID}
google.client-secret=${GOOGLE_CLIENT_SECRET}
google.redirect-uri=${server.ip}/api/login/oauth2/code/google

naver.client-id=${NAVER_CLIENT_ID}
naver.client-secret=${NAVER_CLIENT_SECRET}
naver.redirect-uri=${server.ip}/api/login/oauth2/code/naver

kakao.client-id=${KAKAO_CLIENT_ID}
kakao.client-secret=${KAKAO_CLIENT_SECRET}

kakao.redirect-uri=${server.ip}/api/login/oauth2/code/kakao
```