Live-coding SSO

(from first principles!)

Daniel Garnier-Moiroux

Voxxed Zürich, 2025-03-25

Daniel Garnier-Moiroux

Software Engineer

- Spring
- @garnier.wf
- https://garnier.wf/
- github.com/Kehrlann/
- contact@garnier.wf

What we'll talk about

1. OAuth2, OpenID: What's that?
    1. OAuth2
    2. OpenID Connect
2. How and why, with images
3. Live-coding

OAuth2, OpenID

What even is this?

OAuth 2 & 2.1

- An *Authorization* framework
  - Goal: Grant **applications** the **permission** to access **ressources** through **HTTP**.
- Using *tokens*, in this case `access_token`
- A long list of specs
  - **https://oauth.net/specs/** (sometimes a bit ... dry ...)

OAuth 2 & 2.1

For example:

👦 **Daniel**

authorizes

💻 **my-photo-book.example.com**

to access his pictures hosted on

📸 **Google Photos**

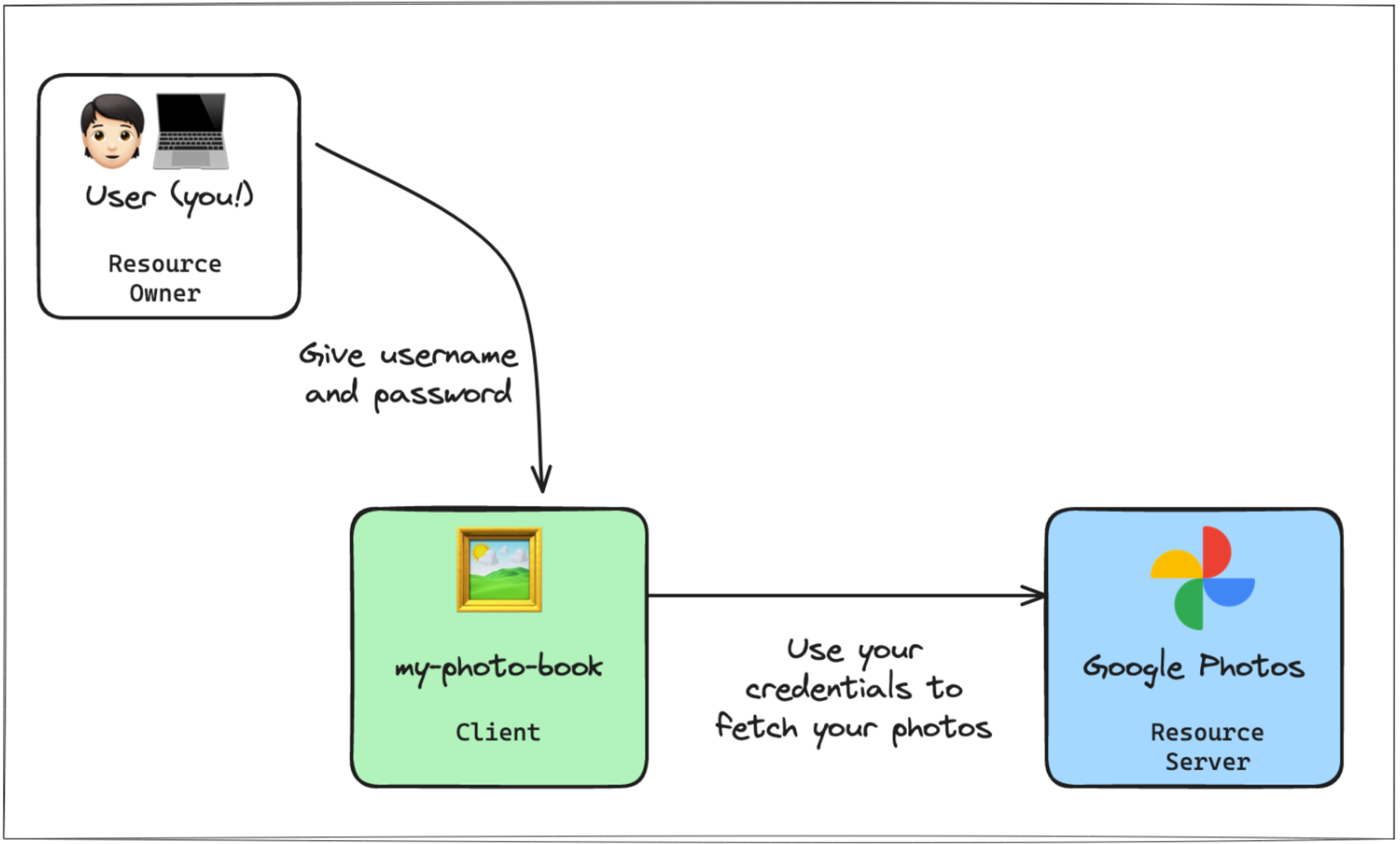
(without sharing his 🤥 Google password)

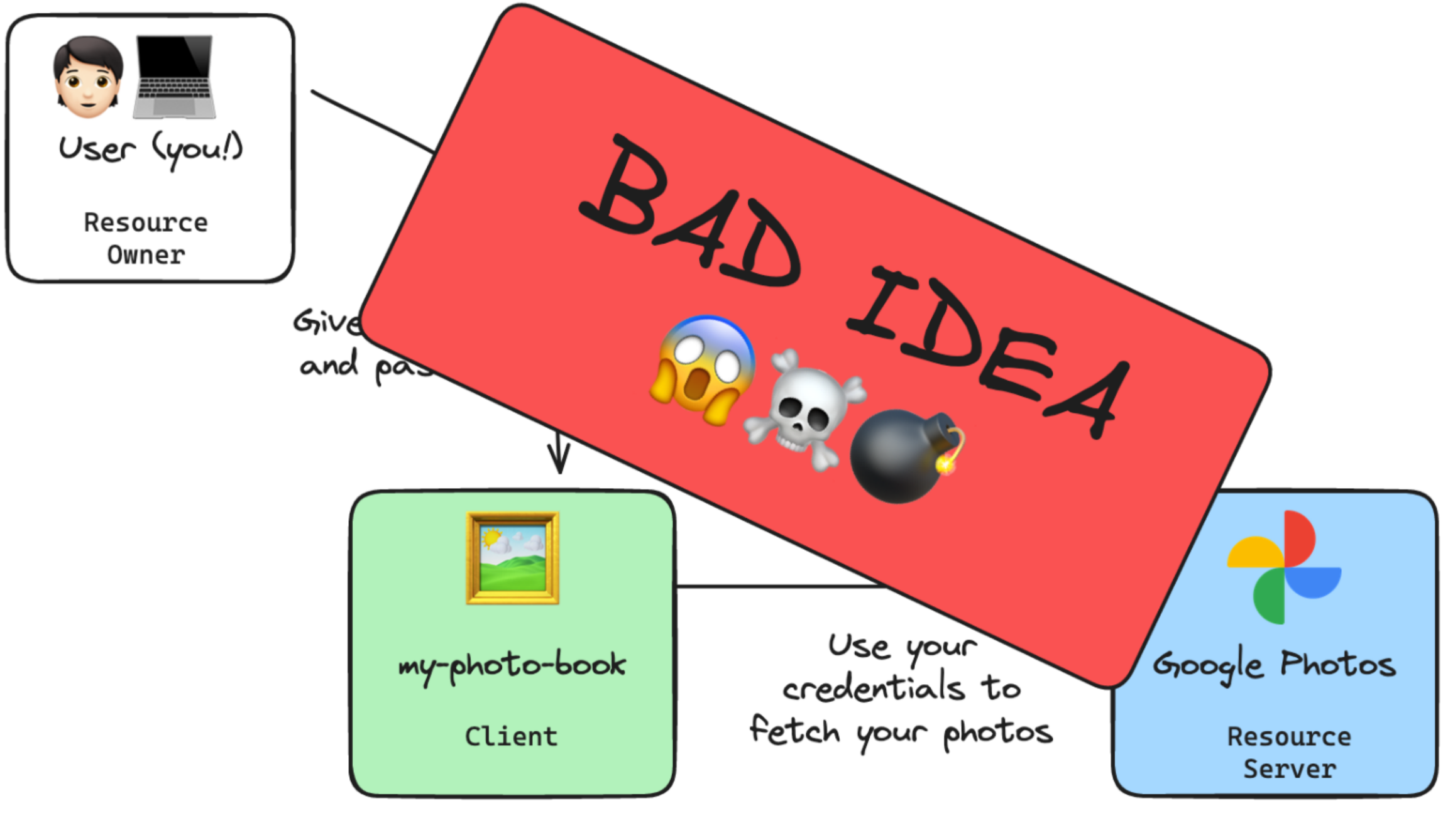
Notice: we’re not saying anything about **identity**…

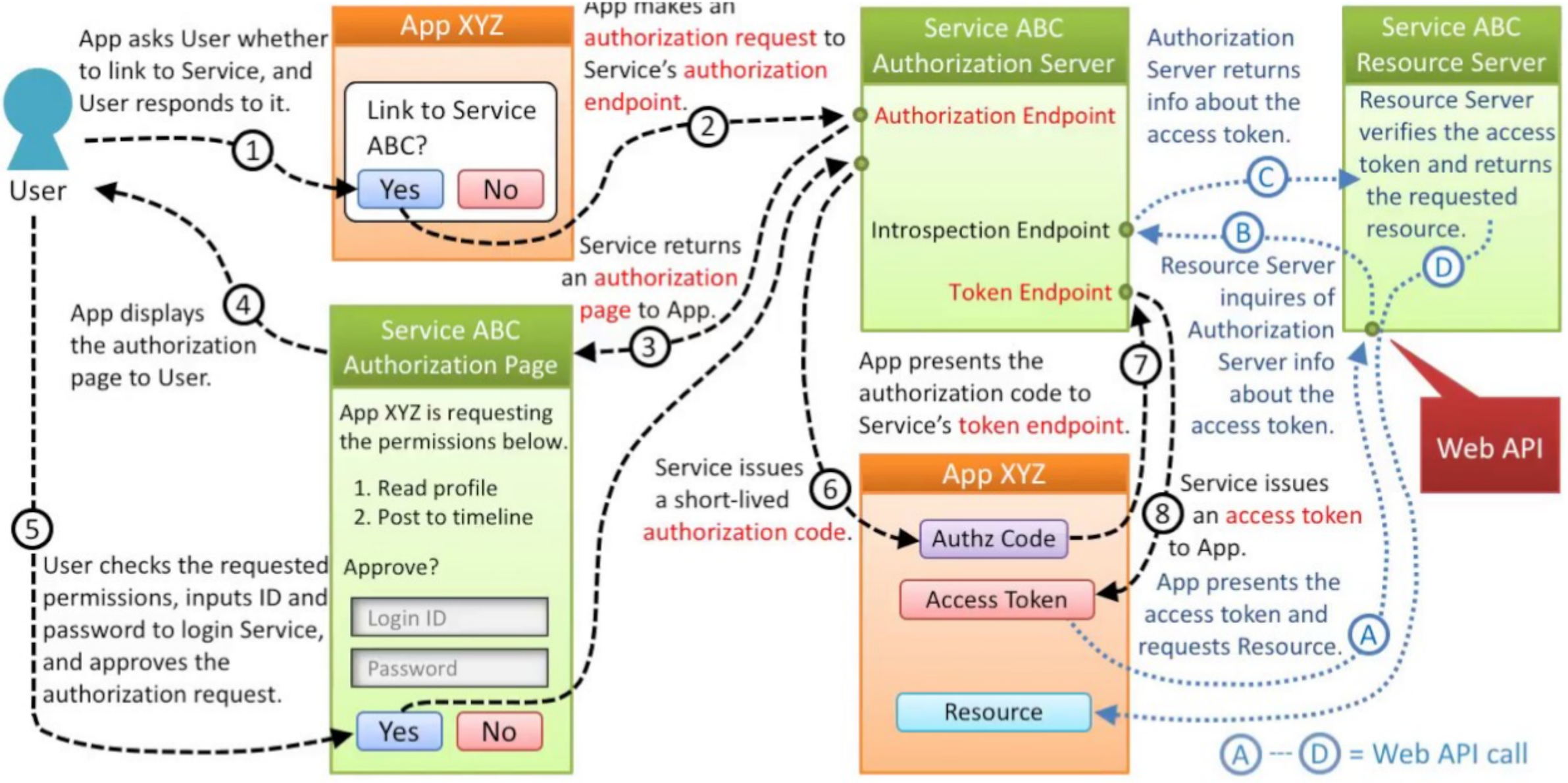
OpenID Connect

- An *Authentication* framework
  - Goal: give third-party **applications** identity data managed by an **identity provider**
  - Therefore enabling **Single-Sign-On** (SSO)
- Based on OAuth2, this time using `id_token`
- And of course... specs!
  - **https://openid.net/developers/specs/**

How and why?


User (you!)
Resource Owner
Give username and password
my-photo-book
Client
Use your credentials to fetch your photos
Google Photos
Resource Server


User (you!)
Resource Owner
BAD IDEA
Give
and pas
my-photo-book
Client
Use your credentials to fetch your photos
Google Photos
Resource Server



App asks User whether to link to Service, and User responds to it.

1

User

App displays the authorization page to User.

4

5

User checks the requested permissions, inputs ID and password to login Service, and approves the authorization request.

zation
eturns
ut the
oken.

C

B

rce Server
nquires of
horization
erver info
about the
ess token.

ice issues
access token
p.
esents the
oken and
Resource.

A

Service ABC
Resource Server

Resource Server verifies the access token and returns the requested resource.

D

Web API

A --- D = Web API call

c. https://www.authlete.com/

User (you!)

Resource Owner

1. Prove identity (log in)
2. Obtain token(s)

Google Accounts

Authorization Server

3. Give token(s)

my-photo-book

Client

4. Echange token for resources (photos)

Google Photos

Resource Server

User (you!)

Resource Owner

1. Prove identity (log in)
2. Obtain token(s)

id_token

access_token

Google Accounts

Authorization Server

3. Give token(s)

id_token

access_token

my-photo-book

Client

4. Echange token for resources (photos)

access_token

Google Photos

Resource Server

User (you!)

Resource Owner

1. Prove identity (log in)
2. Obtain token(s)

Google Accounts

Authorization Server

3. Give token(s)

my-photo-book

Client

4. Echange token for resources (photos)

Google Photos

Resource Server

User (you!)

Resource Owner

1. Prove identity (log in)
2. Obtain ~~token(s)~~ code

Google Accounts

Authorization Server

4. Exchange code for token(s)

code

3. Give ~~token(s)~~

my-photo-book

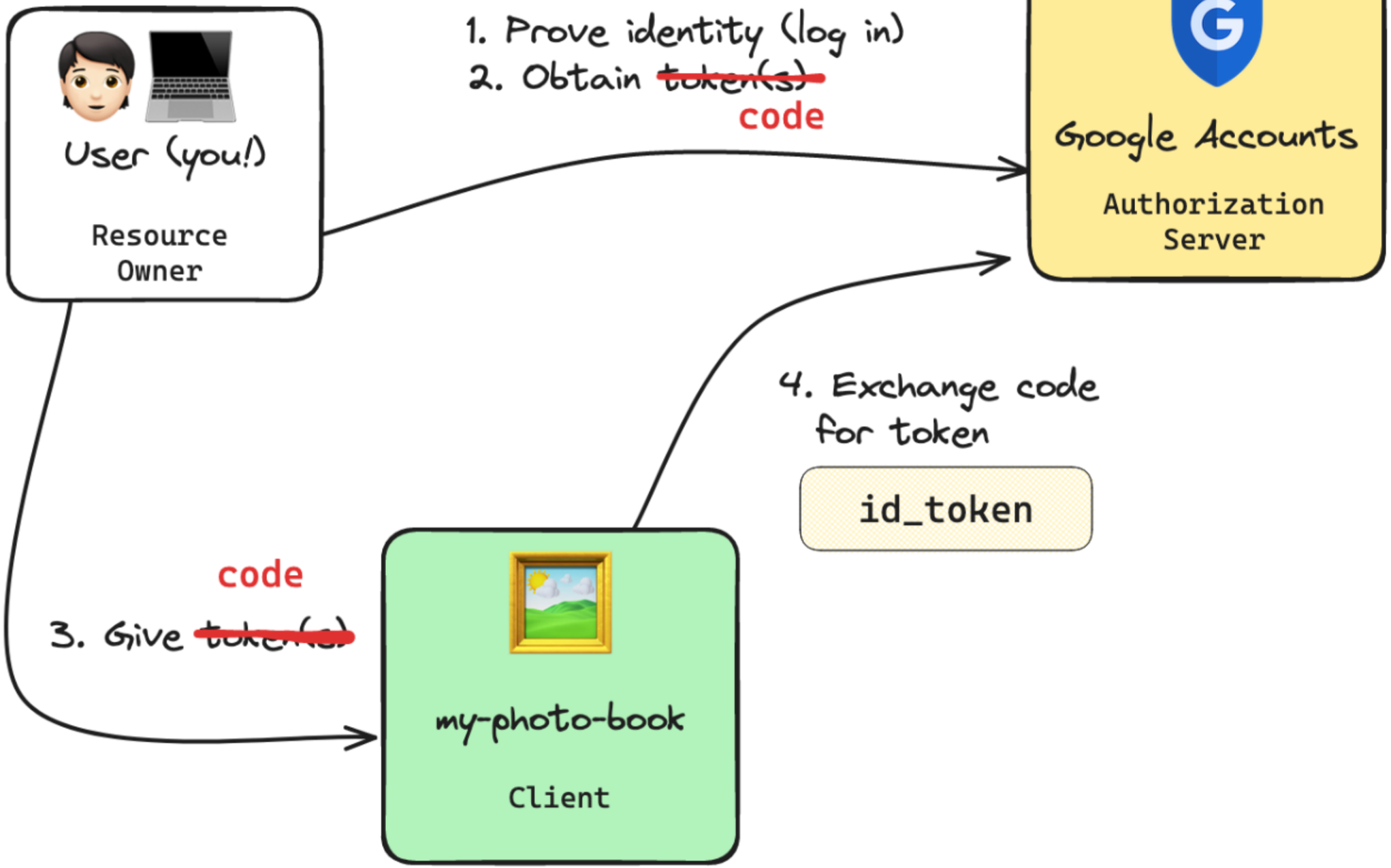
Client

5

~~4.~~ Echange token for resources (photos)

Google Photos

Resource Server


User (you!)
Resource Owner
1. Prove identity (log in)
2. Obtain ~~token(s)~~ code
Google Accounts
Authorization Server
4. Exchange code for token
id_token
code
3. Give ~~token(s)~~
my-photo-book
Client

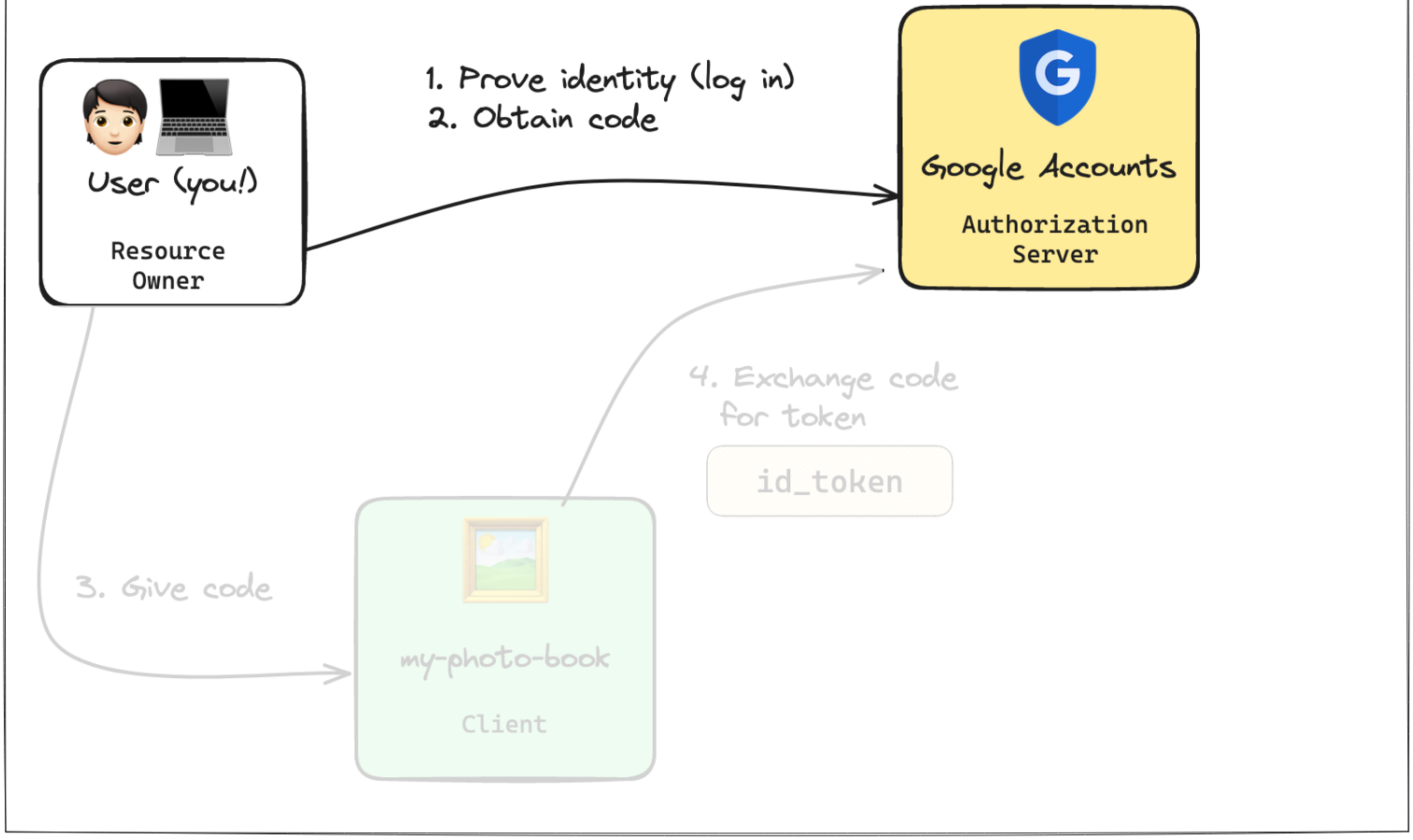
Live-coding

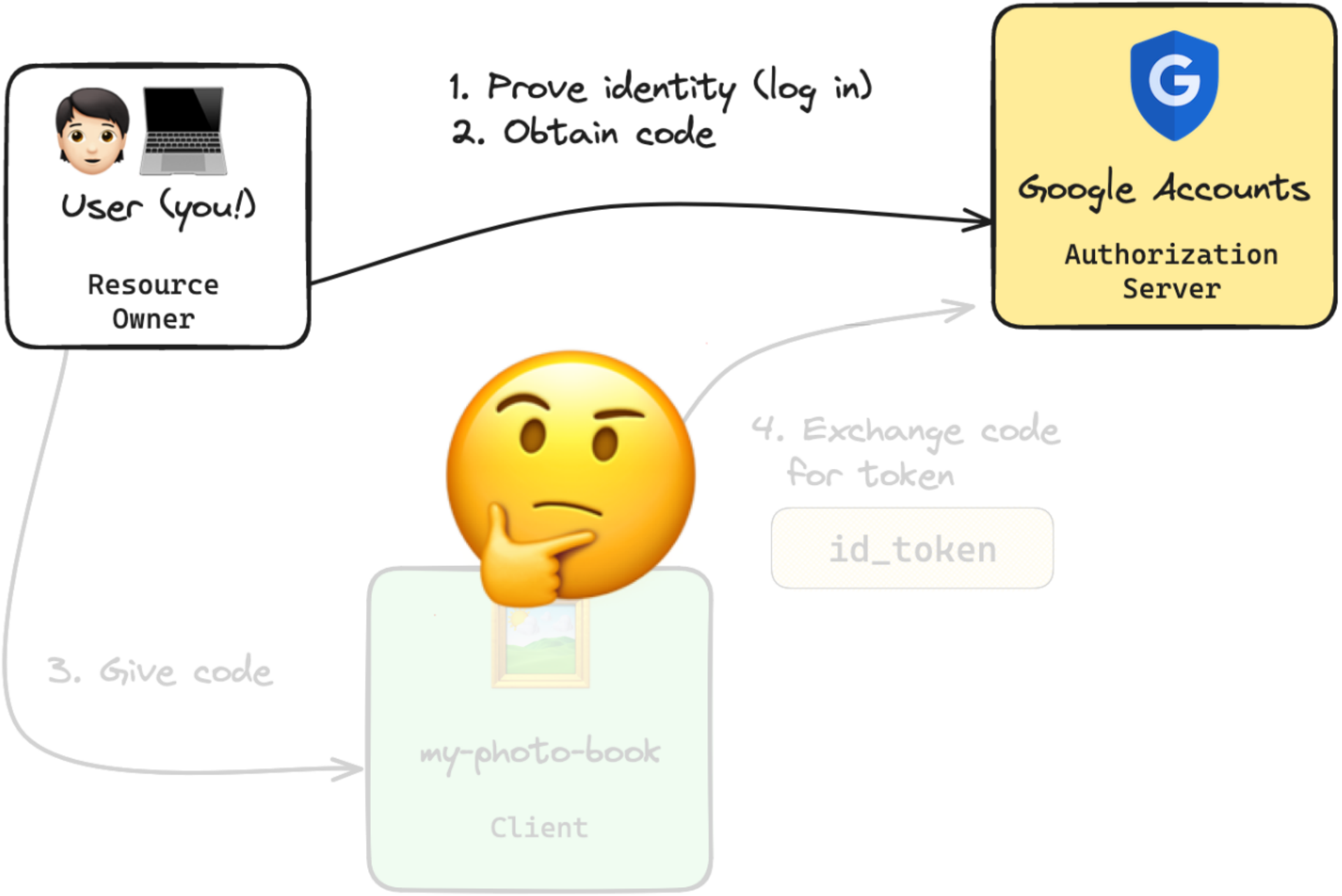
~~~ Let's code!

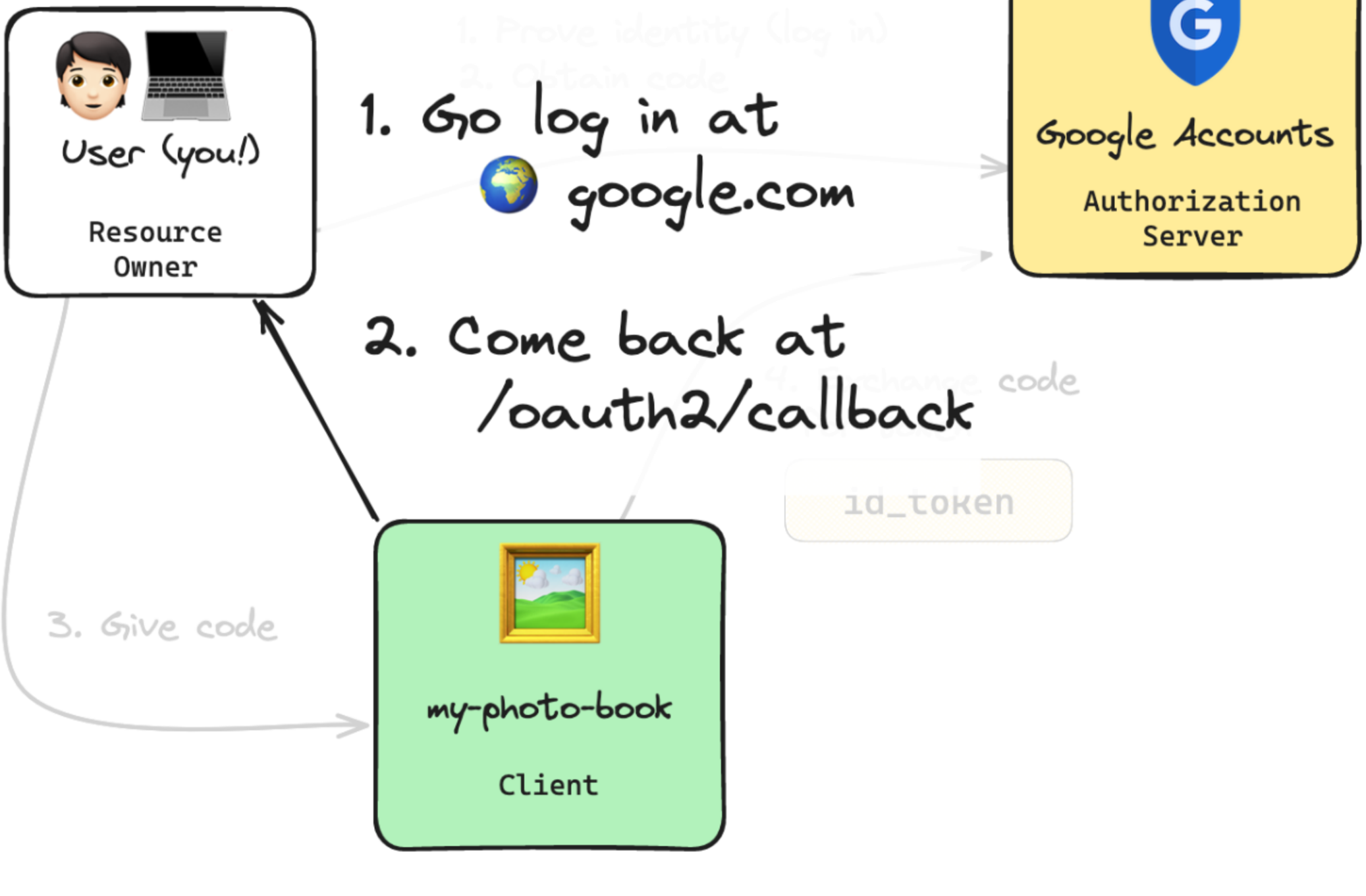
🚨 WARNING 🚨

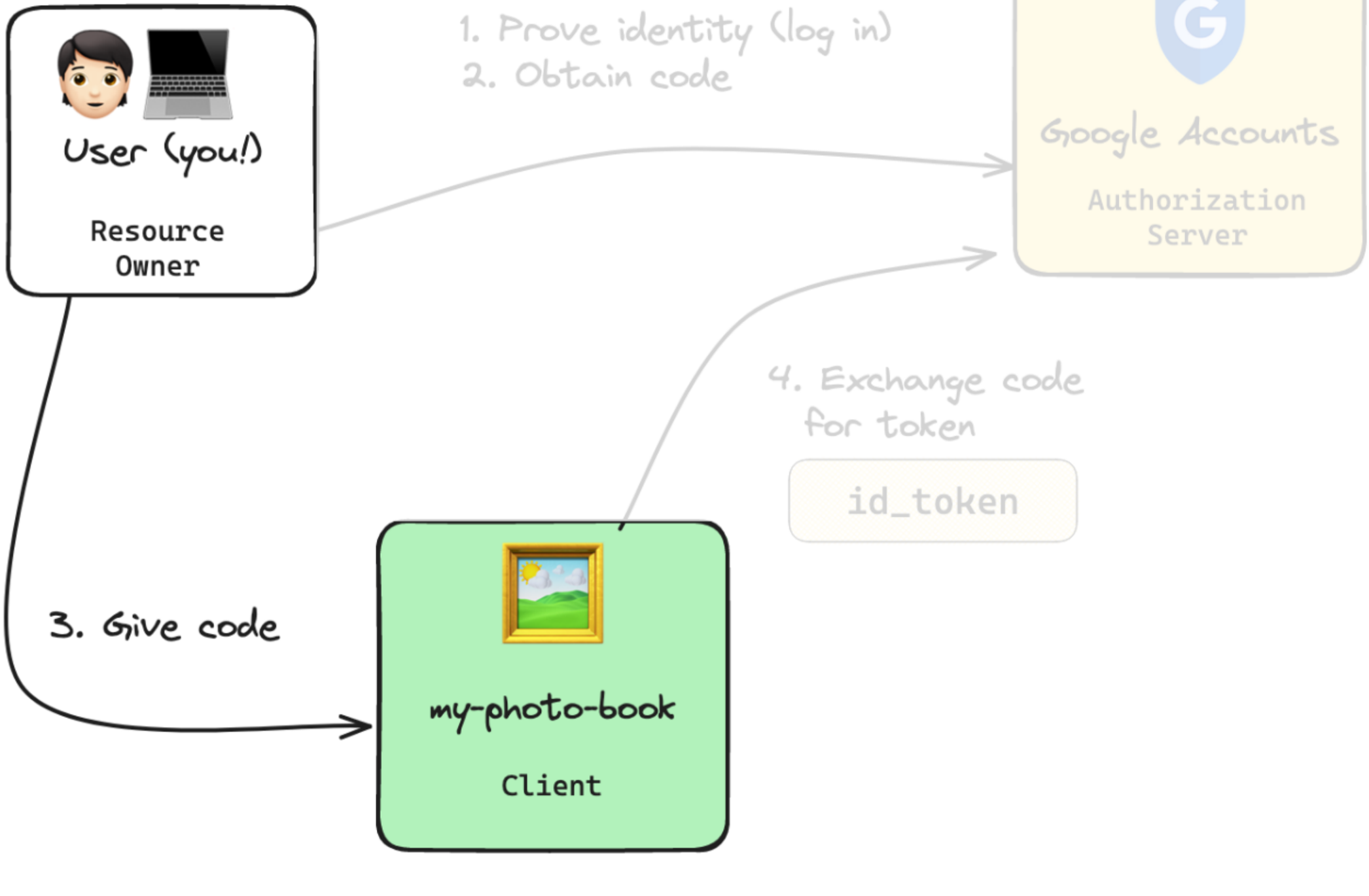
The stunts in this live-coding are performed by a professional[1]. Do **NOT** push any of this code to production. **EVER**. Use a library.

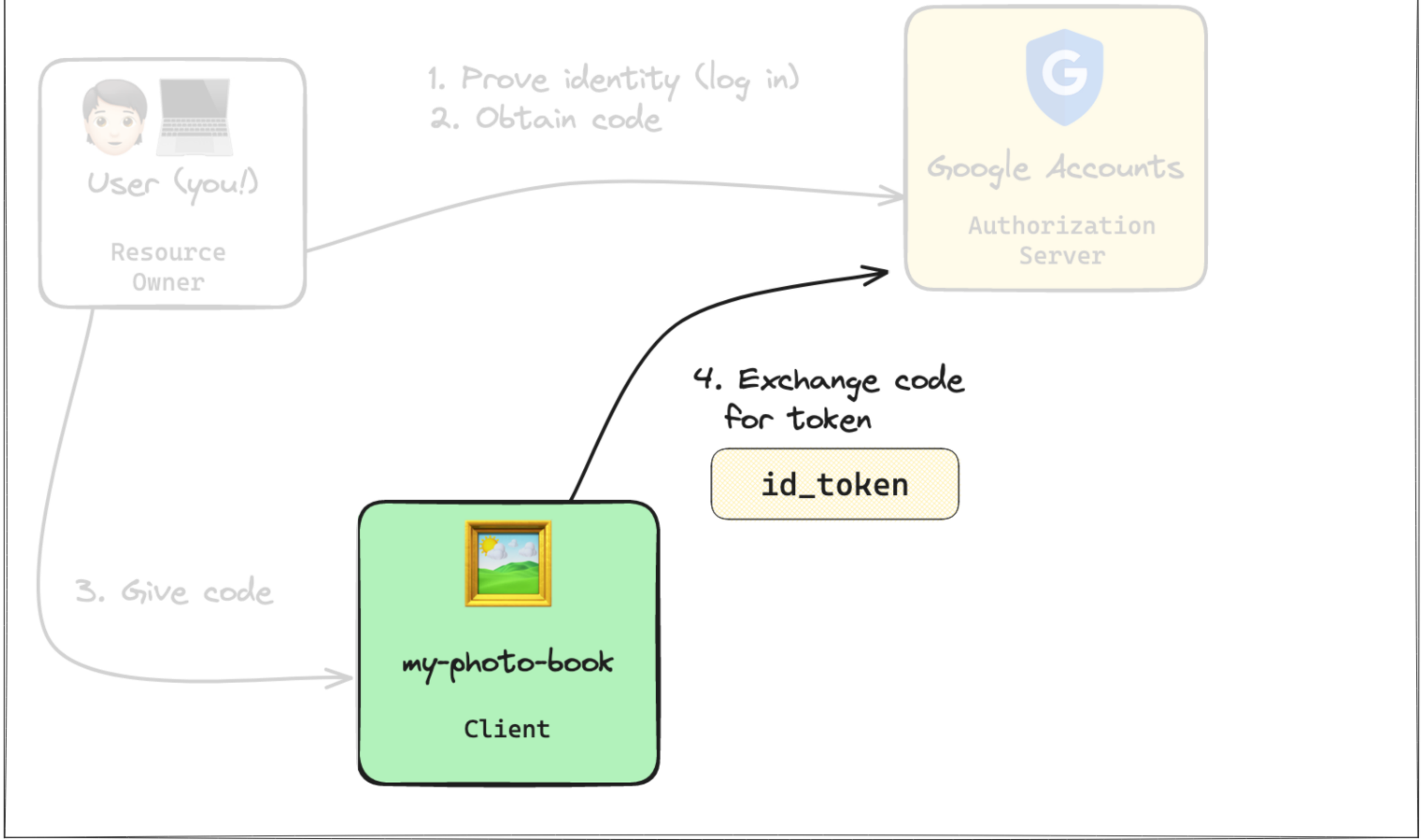
[1] *dubious claim*


User (you!)
Resource Owner
1. Prove identity (log in)
2. Obtain code
Google Accounts
Authorization Server
4. Exchange code for token
id_token
3. Give code
my-photo-book
Client


User (you!)
Resource Owner
1. Prove identity (log in)
2. Obtain code
Google Accounts
Authorization Server
4. Exchange code for token
id_token
3. Give code
my-photo-book
Client


User (you!)
Resource Owner
1. Go log in at
google.com
Google Accounts
Authorization Server
2. Come back at
/oauth2/callback
3. Give code
my-photo-book
Client


User (you!)
Resource Owner
1. Prove identity (log in)
2. Obtain code
Google Accounts
Authorization Server
4. Exchange code for token
id_token
3. Give code
my-photo-book
Client


User (you!)
Resource Owner
1. Prove identity (log in)
2. Obtain code
Google Accounts
Authorization Server
4. Exchange code for token
id_token
3. Give code
my-photo-book
Client

Header

{
"alg": "RS256",
"typ": "JWT"
}

eyJhbGciOiJSUzI1NiIsInR5
cCI6IkpXVCJ9 . eyJzdWIi
OiIxMjM0NTY3ODkwIiwibm
FtZSI6IkpvaG4gRG9lIiwi
YWRtaW4iOnRydWUsImlhd
CI6MTUxNjIzOTAyMn0 .
NHVaYe26Mbt0YhSKkoKYd
FVomg4i8ZJd8_-RU8VNbf
[...]

Payload

{
"sub": "1234567890",
"name": "John Doe",
"admin": true,
"iat": 1516239022
}

Signature

Remember!

Don’t do this. Use a library.

References

https://github.com/Kehrlann/sso-live-coding

- @garnier.wf
- https://garnier.wf/
- contact@garnier.wf

Thank you 😊


VMware Tanzu
VOXXEDDAYS
ZURICH
NEED THE WI-FI
RATE THE TALKS!
App Store