



OAuth + OpenID Connect due sigle, un solo obiettivo

Se pensi sia solo un “login” cambierai idea



SPEAKERS:

CLAUDIO PIFFER

MAURIZIO DEL MAGNO





CLAUDIO PIFFER

DEVELOPER E BATTERISTA



claudio.piffer@gmail.com
info@pc-soft.it



<https://github.com/claudio-piffer>



<https://www.linkedin.com/in/piffer-claudio-3599a16a/>



<https://twitter.com/claudio130868>



OAuth + OpenID Connect
due sigle, un solo obiettivo

Se pensi sia solo un “login” cambierai idea



SPEAKERS:
CLAUDIO PIFFER
MAURIZIO DEL MAGNO



OAuth + OpenIDConnect due sigle, un solo obiettivo

Se pensi sia solo un “login” cambierai idea



SPEAKERS:
CLAUDIO PIFFER
MAURIZIO DEL MAGNO



MAURIZIO DEL MAGNO
DEVELOPER



Lev@nte software



i-ORM
DJSON

github.com/mauriziodm/iORM
github.com/mauriziodm/DJSON



mauriziodm@levantesw.it
mauriziodelmagno@gmail.com



facebook.com/maurizio.delmagno
iORM + DJSON (group)

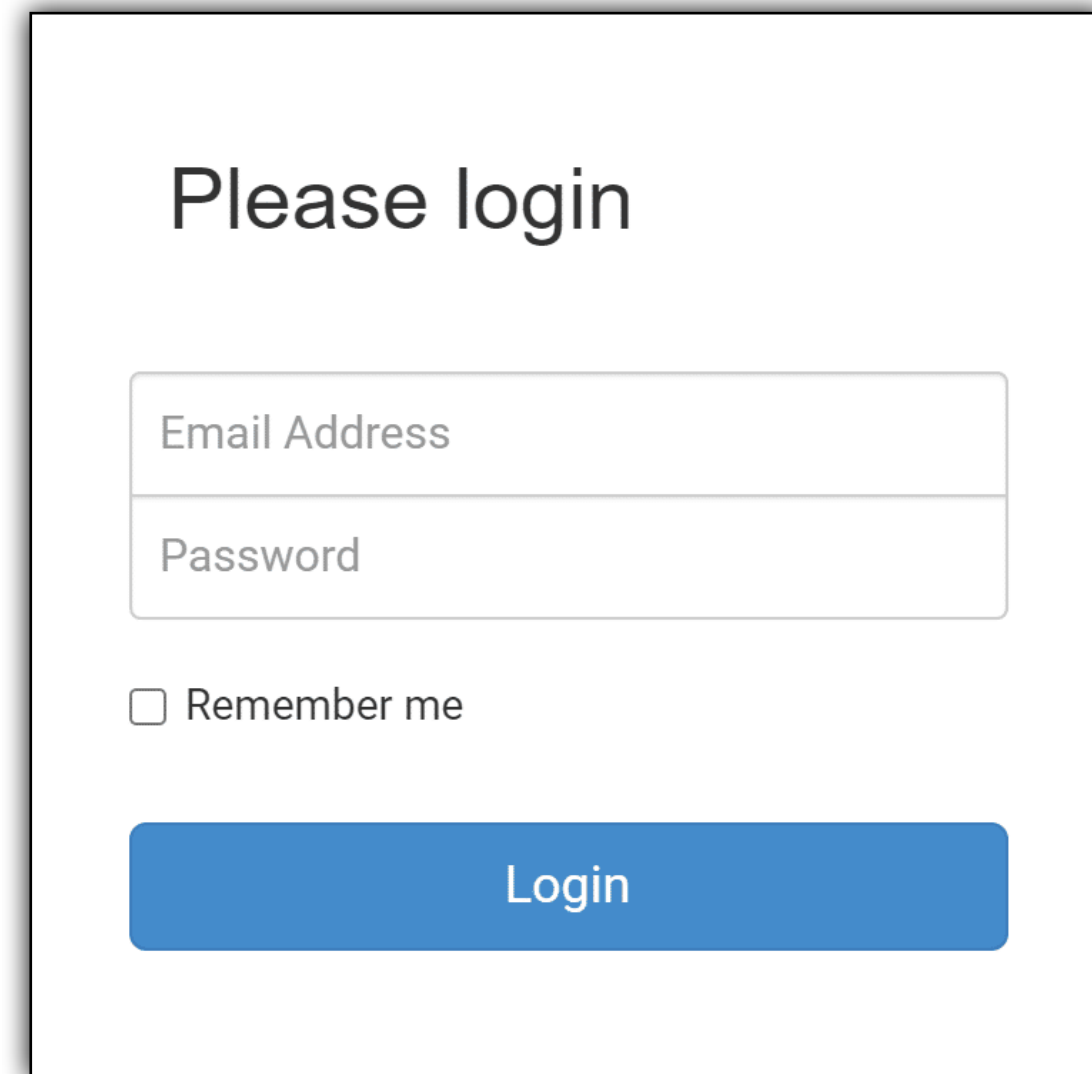


Membro fondatore
eInvoice4D
<https://github.com/delphiforce/eInvoice4D>

OAuth + OpenID Connect
due sigle, un solo obiettivo

OAuth + OpenID Connect

due sigle, un solo obiettivo



Please login

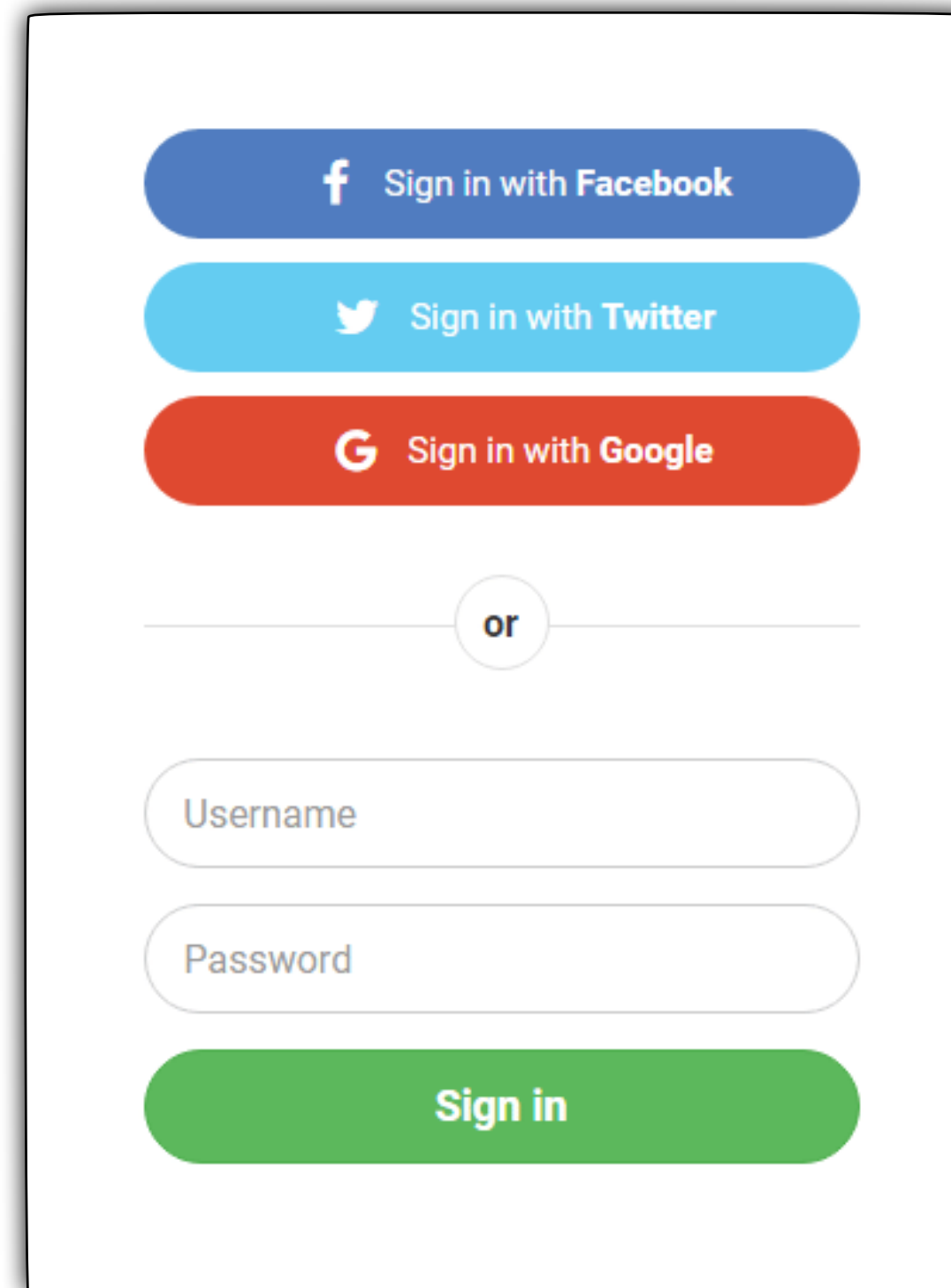
Email Address

Password

☐ Remember me

Login

Authentication



Sign in with Facebook

Sign in with Twitter

Sign in with Google

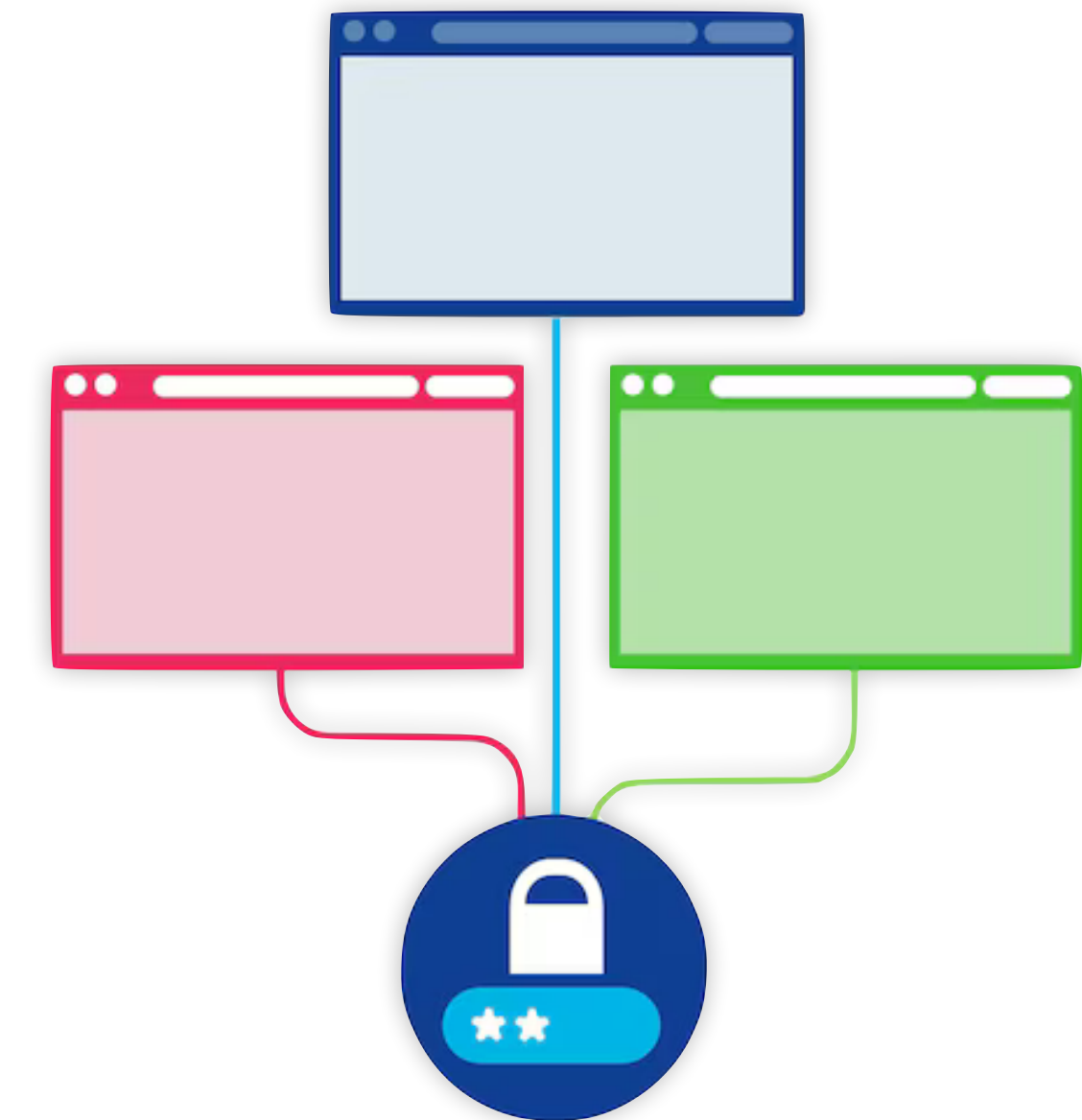
or

Username

Password

Sign in

Social authentication

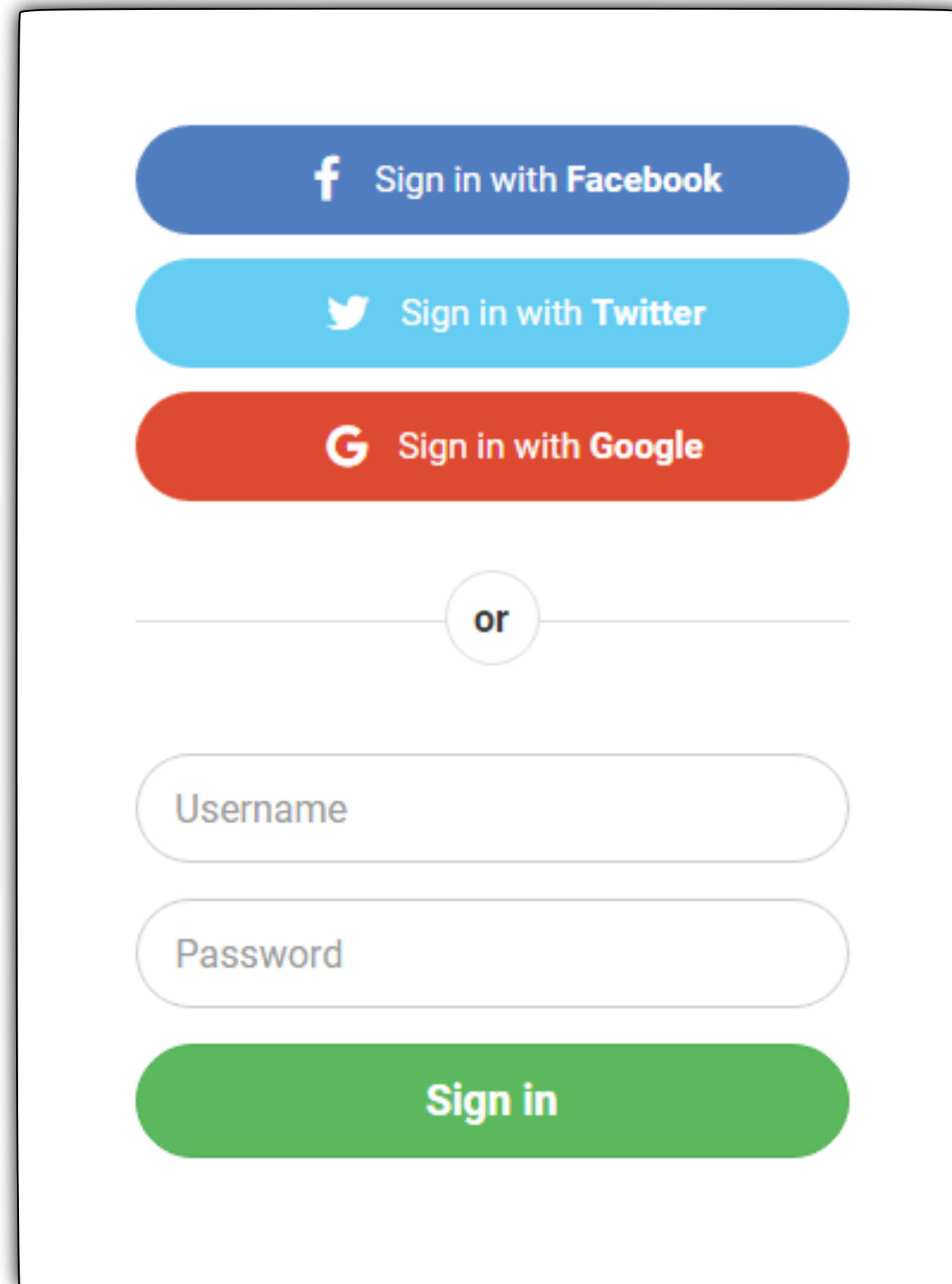


Single Sign-On (SSO)

NON FATELO!!!

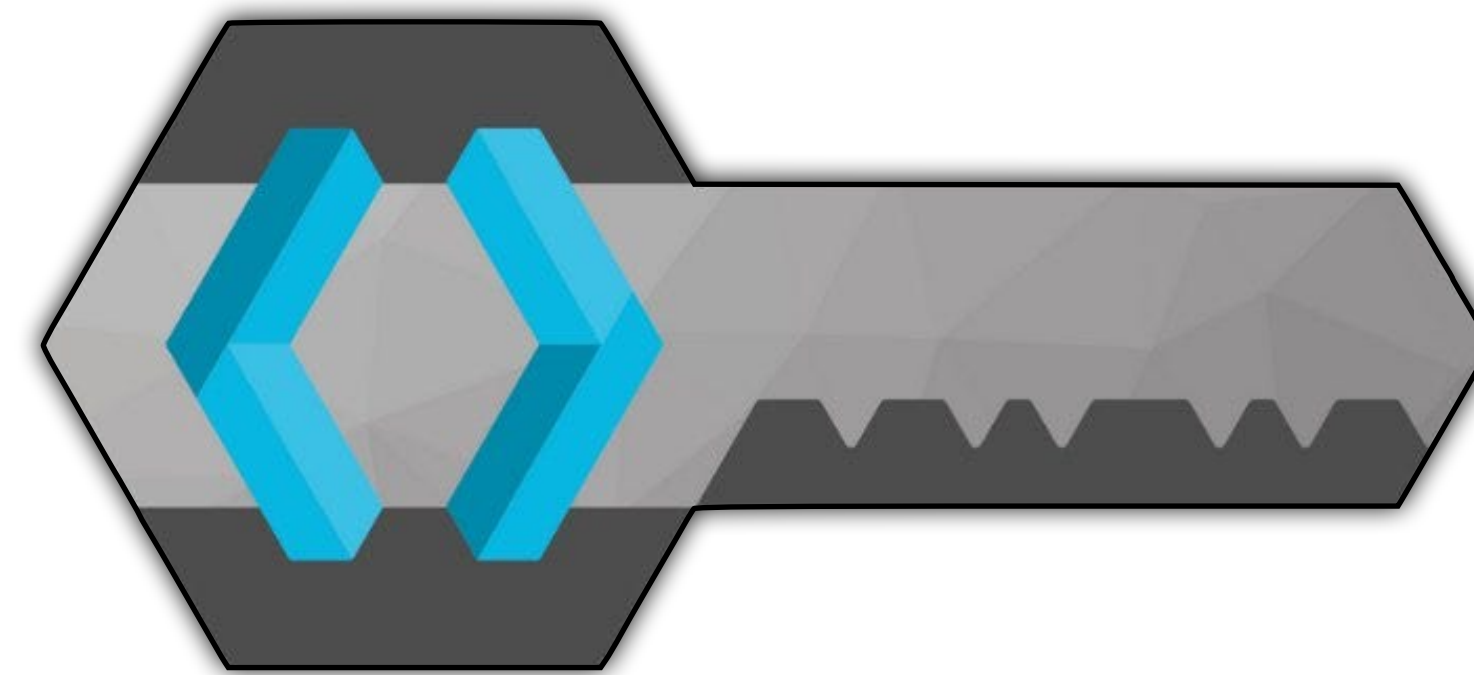
OAuth + OpenID Connect

due sigle, un solo obiettivo



Mockup of a login interface:

- Buttons for social login: "Sign in with Facebook", "Sign in with Twitter", "Sign in with Google".
- A separator with the word "or".
- Input fields for "Username" and "Password".
- A green "Sign in" button.



KEYCLOAK

IAM server
(Identity & Access Management)



a redhat sponsored open-source project

OAuth + OpenID Connect

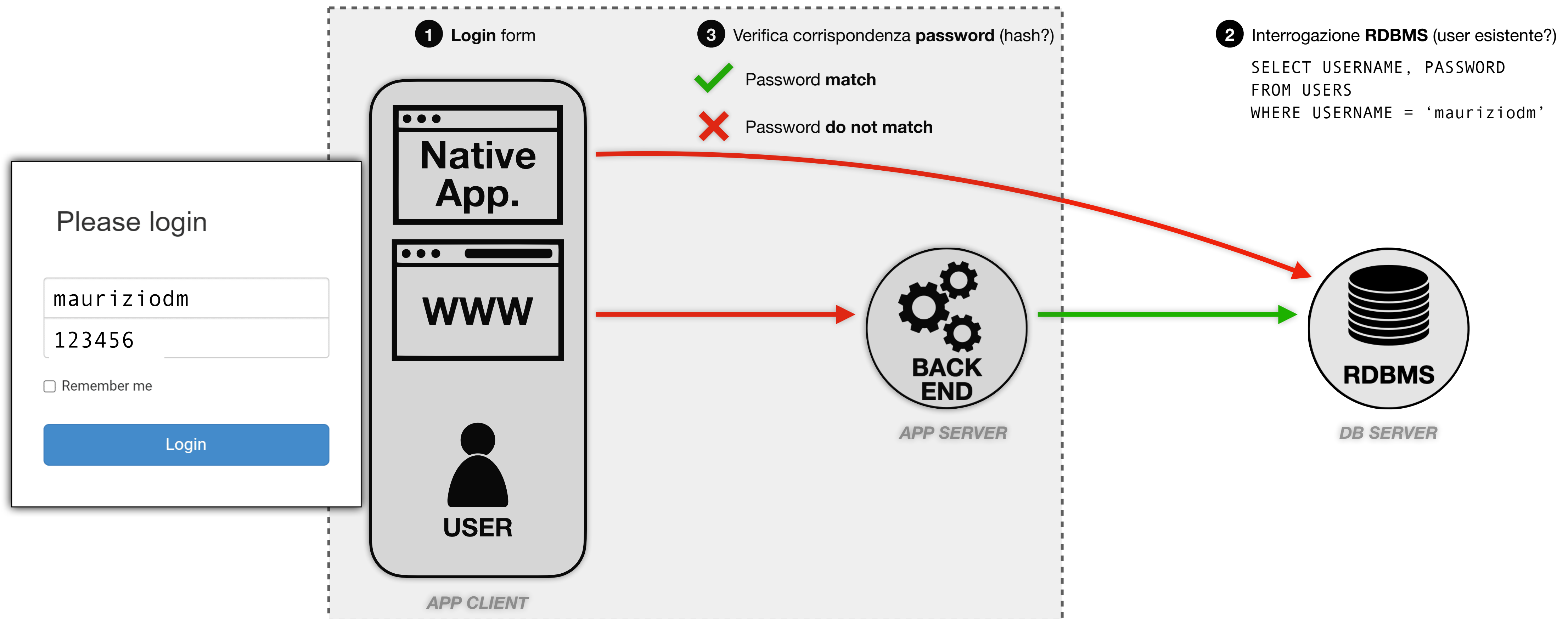
due sigle, un solo obiettivo

Scopo del seminario:

- Illustrare il funzionamento dei protocolli (Perchè sono così? Quali problemi risolvono?)
- Chiarire la **terminologia**
- Superare la **confusione** derivante dall'uso di termini diversi per gli stessi concetti
- OAuth non è un problema, è **una soluzione**

SimpleLogin

SimpleLogin



Perchè il settore si è allontanato da questo?

- Logiche di autenticazione e autorizzazione **tutto nell'app**
- Problemi di **sicurezza**
- Problemi di **manutenibilità**

Autenticazione

Autenticazione

- Il modo in cui dimostro di essere chi dichiaro di essere
- **3 modi** per dimostrarlo:
 - 1) **ciò che so** (password, pin, domande e risposte, codici di recupero)
 - 2) **ciò che possiedo** (devices, chiave fisica, OTP, documento di identità)
 - 3) **ciò che sono** (impronta digitale, riconoscimento facciale, analisi vocale, scansione retina)
- Nel tempo è diventata sempre più complessa

Autenticazione

- Sempre più complessa e articolata
 - **Multifactor** authentication
 - **Email address** verification
 - **Phone number** verification
 - **OTP** (token fisico, app authenticator)
 - Safe & registered **browsers** management
 - Safe & registered **devices** management
 - Notifica a un'**altro device** già registrato
 - **Authentication-delegation** (SPID, google, Facebook, LinkedIn...)
 - **SSO** (Single Sign On)
 - Domande e risposte di recupero
 - Codici di recupero

Autorizzazione

Autorizzazione

- Come assegno e verifico i permessi/autorizzazioni di una **identità**
- Definisce a quali informazioni una **identità** può accedere, quali azioni può compiere su di esse e quali processi, anche non legati ai dati, può avviare
- Esempi:
 - Un dipendente di un'azienda può essere autenticato tramite le proprie credenziali ma le sue autorizzazioni (grant) gli permetteranno di accedere solo ai file/dati e alle applicazioni in relazione al suo ruolo.
 - Un cliente di una banca potrebbe essere autenticato per accedere al suo conto (non al conto di altri) ma essere autorizzato ad eseguire solo le operazioni consentite per quel tipo di account.

Autorizzazione

Può essere organizzata in diversi modi:

- **RBAC** (Role-Based Access Control)
 - Utenti raggruppati in **ruoli**
 - Singoli **permessi** assegnati **ai ruoli**, non direttamente agli utenti
 - Vantaggi: **semplice** da implementare e gestire
 - Svantaggi: **minor granularità**
- **ABAC** (Attribute-Based Access Control)
 - Permessi definiti in base a una serie di **attributi dell'utente** (ruolo, posizione), della **risorsa** (tipo di file, sensibilità del dato) e dell'**ambiente** (orario, posizione geografica)
 - L'accesso viene concesso valutando la **combinazione degli attributi** dell'utente, risorsa, ambiente
 - Vantaggi: Più **granulare e flessibile**, consente di definire permessi più specifici e contestuali
 - Svantaggi: Più **complesso** da implementare e gestire, richiede una definizione accurata degli attributi e delle regole di accesso
- **Altri...** (non è argomento di oggi)

non**solo**persone

non**solo**persone

- Al giorno d'oggi **app e sistemi** diversi sono sempre più **interconnessi**
- **App/API** che accedono ad altre **App/API**
- Esempio: Levante può accedere o dare accesso
 - Google **calendar**
 - Google **tasks**
 - Invio **email/sms**
 - Invio/ricezione **fatture elettroniche**
 - **Levante4Friends**

non**solo**persone

- **Che si fa?**

- Inserisco le mie credenziali google su Levante?
- E se poi Google cambia il metodo di autenticazione? (es: a due fattori)
- Richiedo l'autenticazione per ogni applicazione? Ogni volta che si accede?
- E se un'app. ha un comportamento che non mi piace come posso revocarle l'accesso ai miei dati?
- Cancello o modifico le mie credenziali? (e le altre app?)
- Chiudo l'account? (e le altre app? E i miei dati?)

non**solo**persone

- **...serve un modo per:**
 - **Identificare una **app/API** terza** (es: Levante che accede a google calendar; un'app terza che vuole accedere a Levante; frontend —> backend)
 - **Autorizzare una **app/API** ad accedere a dati o azioni **per conto di una identità****
 - **Limitare l'accesso alle sole risorse necessarie**

OAuth

molta confusione, perchè?

OAuth

molta confusione, perchè?

- OAuth lascia **ampio margine di interpretazione**
- Le specifiche iniziali, nonostante il contributo di grandi aziende, **mancavano di precisione e coerenza**
- OAuth è una **raccolta di best practice**, non uno standard rigido
- Si è **evoluto nel tempo** per rispondere a nuove e mutevoli minacce
- L'avvento di nuove tecnologie (web app, SPA, app native, mobile...) ha richiesto **approcci differenti per garantire sicurezza**
- Tutto questo ha generato **implementazioni e terminologie diverse**
- Con **OAuth 2.1** e **Open ID Connect** è diventato molto più definito

prima di OAuth

prima di OAuth

...facciamo un salto nel 2006

- Smartphone non ancora diffusi (iPhone 2007)
- Accesso ad API terze non ancora comune
- Simple login form, standard per l'autenticazione (+ cookie per web)
- Single Sign-On (SSO)
 - Accesso a più sistemi con un'unica autenticazione
 - Protocollo SAML (2002)
 - Principalmente in contesti enterprise
 - Potente ma complesso



prima di OAuth

...2010, nuovi casi d'uso

- Mobile Apps

- *Esigenza:* gestire **sessioni persistenti** su app native
- ...ma i cookie non funzionano bene
- ...15 anni fa mancava una soluzione

- App/API che accedono ad altre App/API

- Facilmente oggi abbiamo **diverse app che accedono a nostri dati** su google/Facebook/LinkedIn...
- ...probabilmente non le ricordiamo nemmeno tutte
- ...molto comune oggi
- ...ma 15 anni fa mancava una soluzione



prima di OAuth



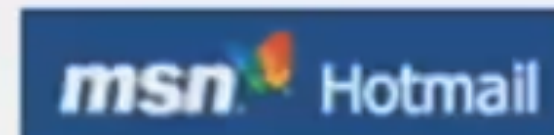
...2010, nuovi casi d'uso

...c'erano solo modi sbagliati

Are your friends already on Yelp?

Many of your friends may already be here, now you can find out. Just log in and we'll display all your contacts and you can select which ones to invite! And don't worry, we don't keep your email password or your friends' addresses. We loathe spam, too.

Your Email Service



Your Email Address

(e.g. bob@gmail.com)

Your Gmail Password

(The password you use to log into your Gmail email)

[Skip this step](#)

[Check Contacts](#)

prima di OAuth



...2010, nuovi casi d'uso

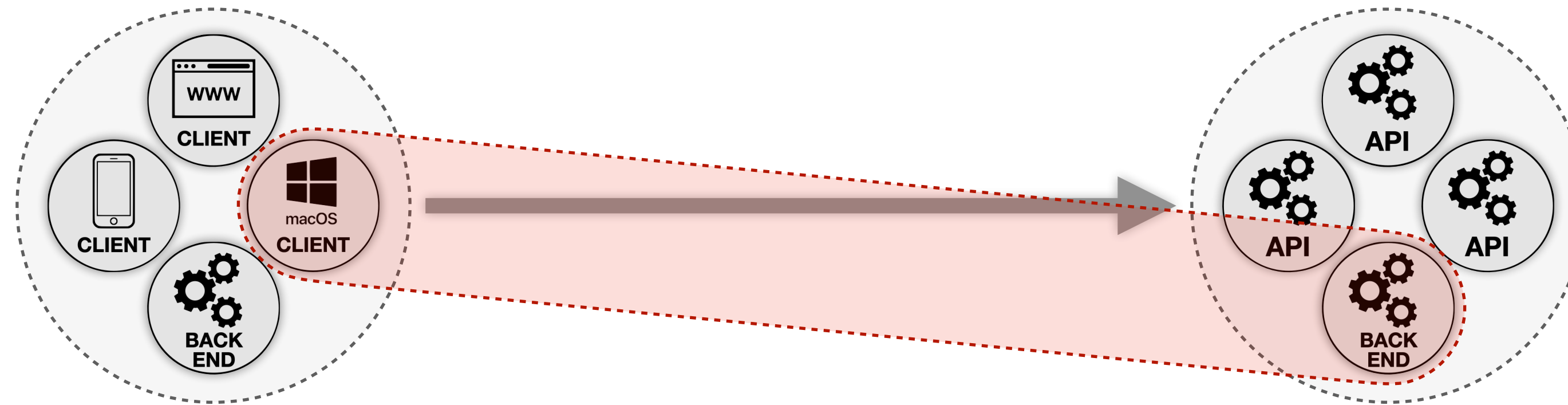
*“Come posso dare accesso ai miei dati **senza fornire la mia password?**”*

Authorization delegation

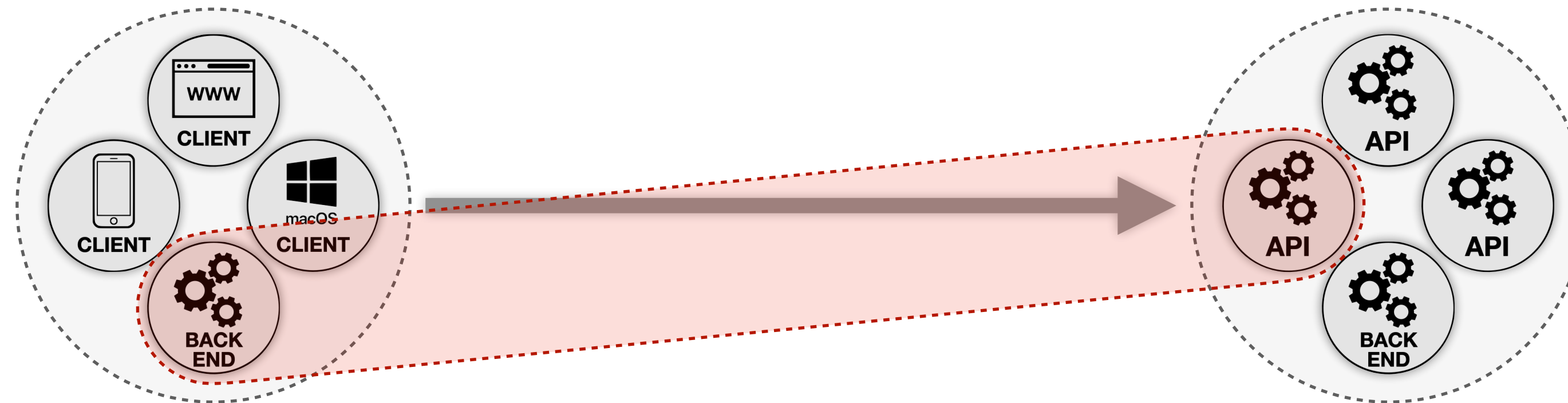
- E' il processo di **concessione di autorità** o permessi, **temporanea** o **permanente**, a un'altra parte (app, servizi...), consentendole di agire **per conto dell'utente originale** (come se fosse l'utente originale)
- **Scopo**: facilitare **collaborazione** e **interoperabilità**, semplificare i flussi di lavoro
- **Sicurezza**: deve essere gestito con attenzione per non compromettere la sicurezza o la privacy
- **Controllo**: deve consentire di **mantenere il controllo** (revocation)

OAuth

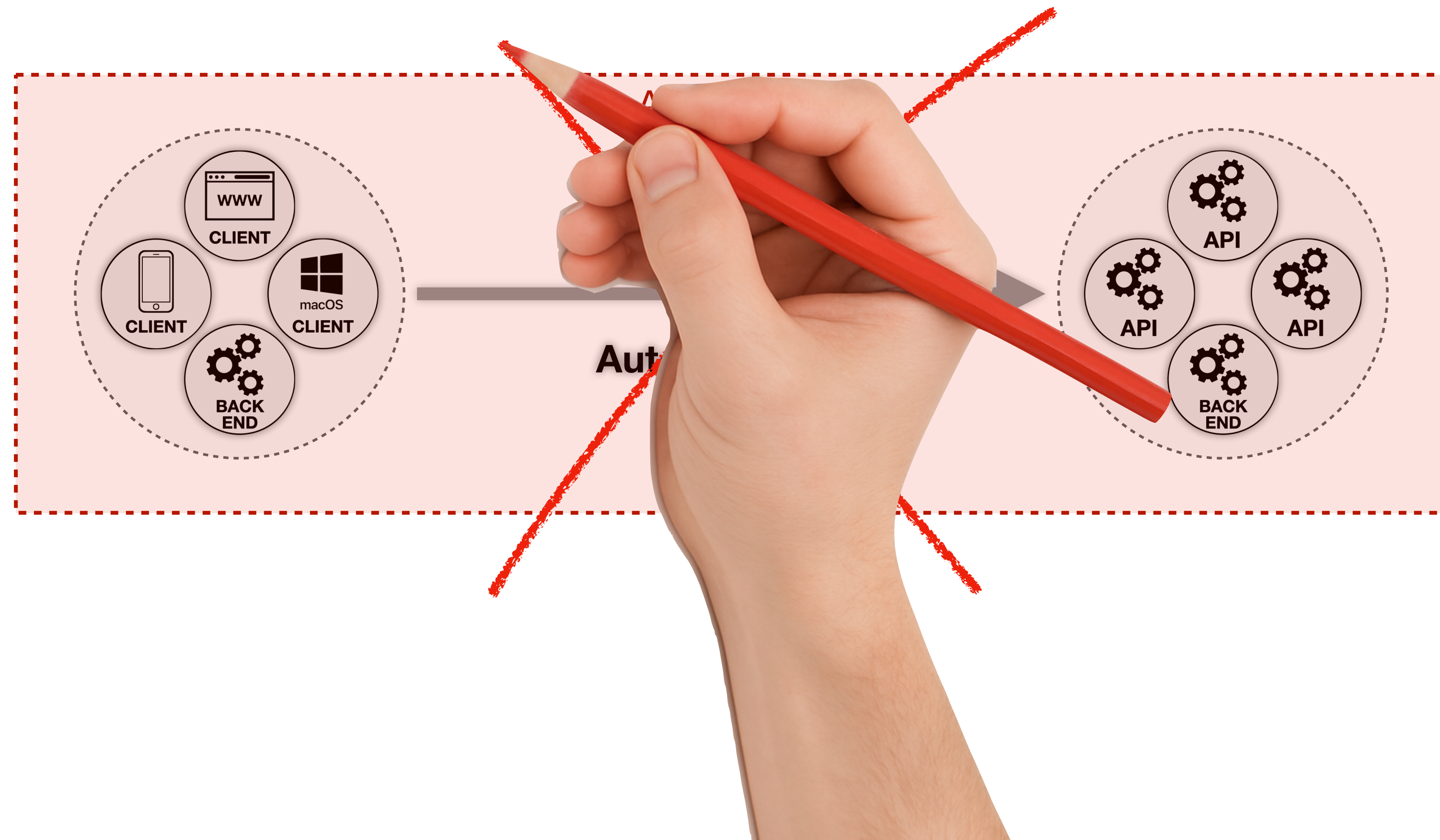
OAuth



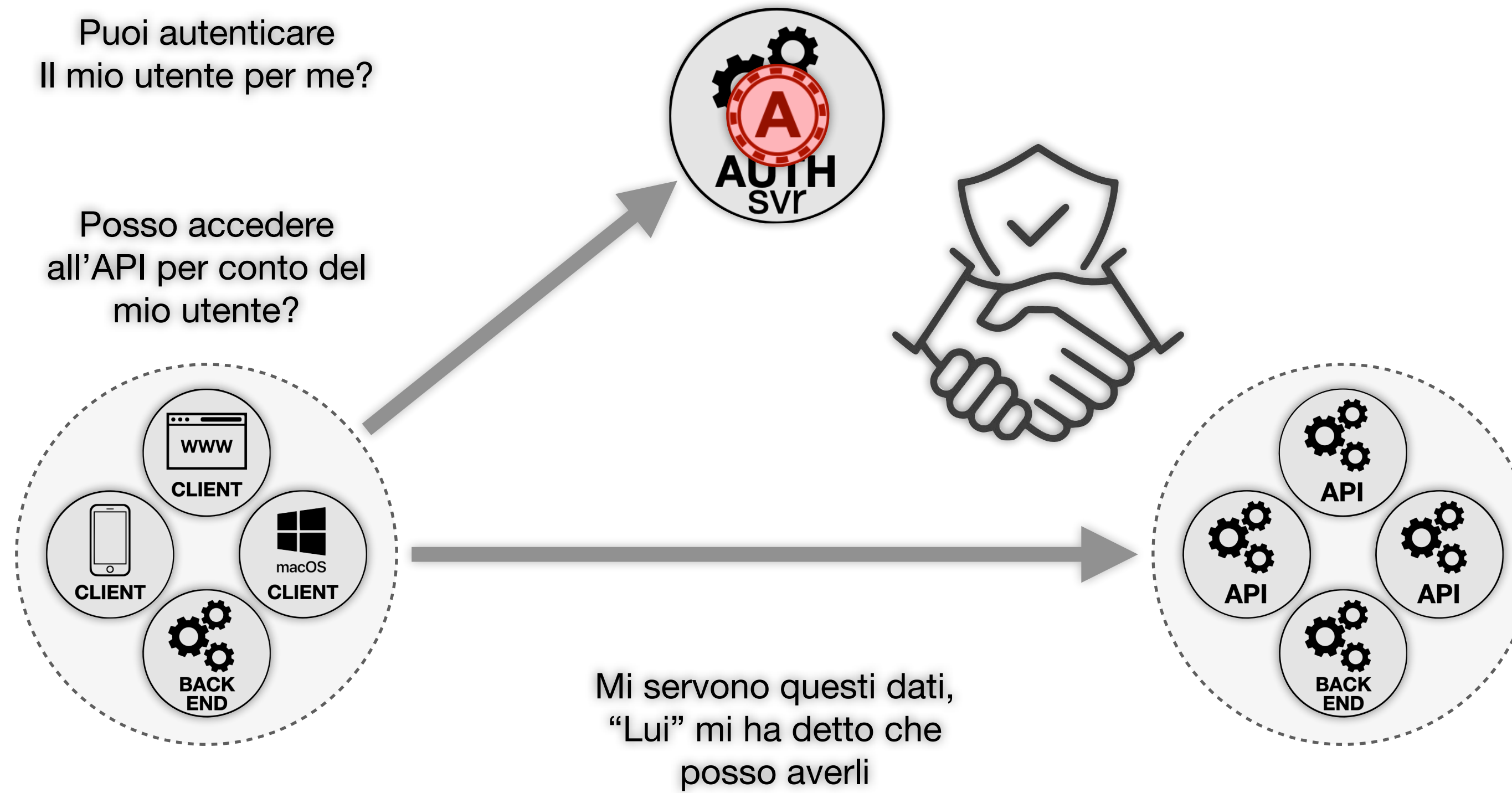
OAuth



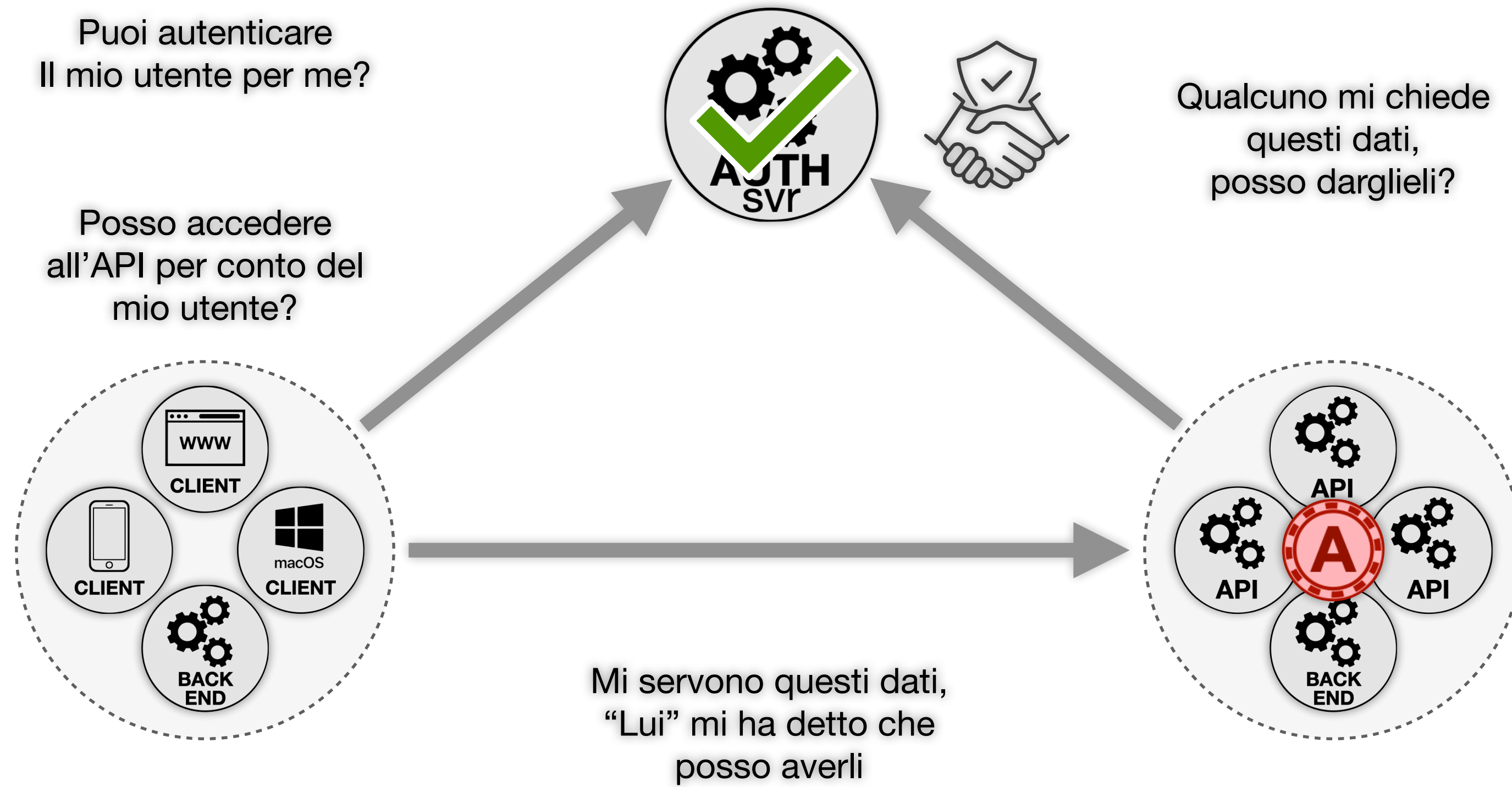
OAuth



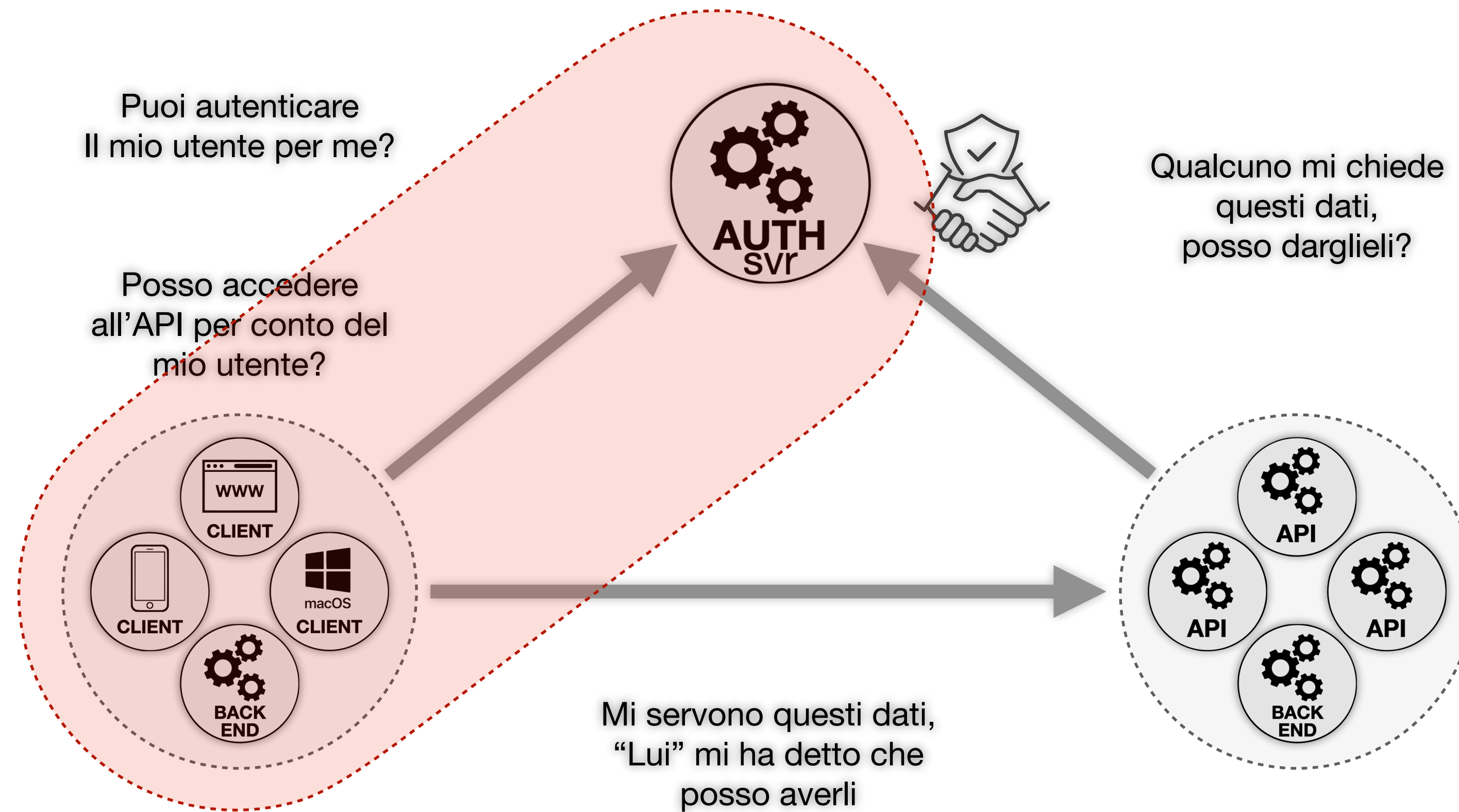
OAuth



OAuth



OAuth

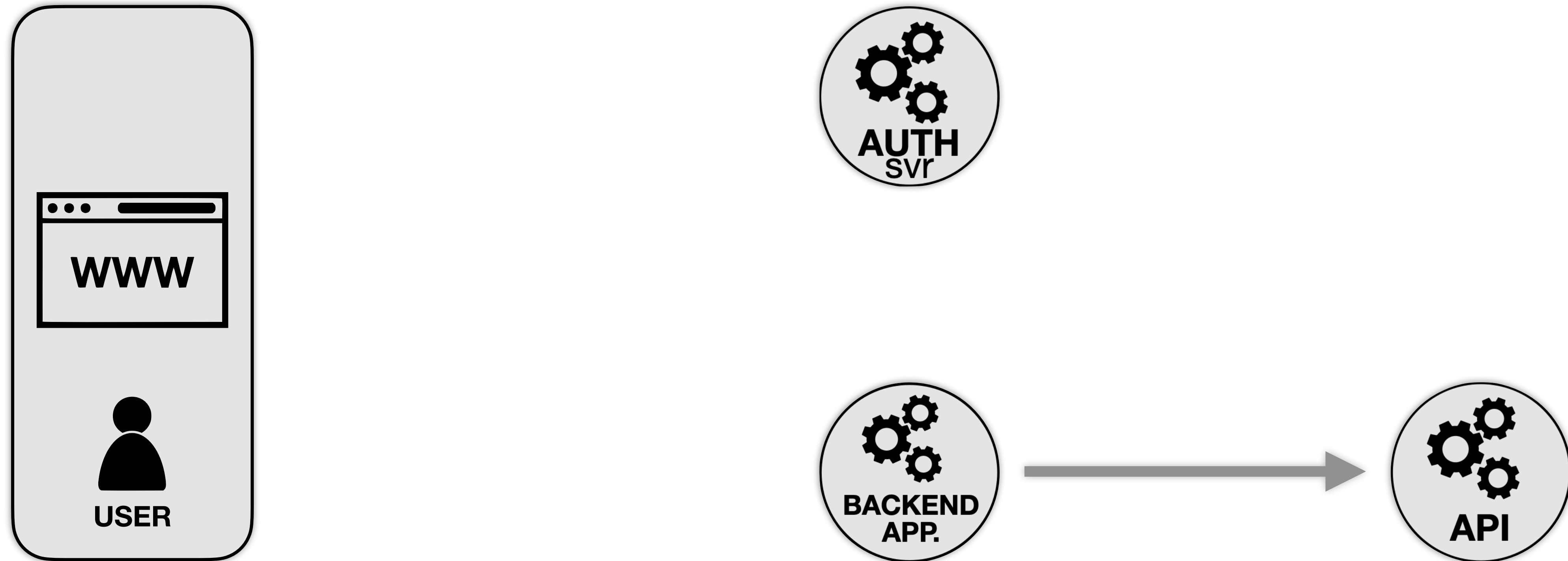


Flussi OAuth (OAuth flows):

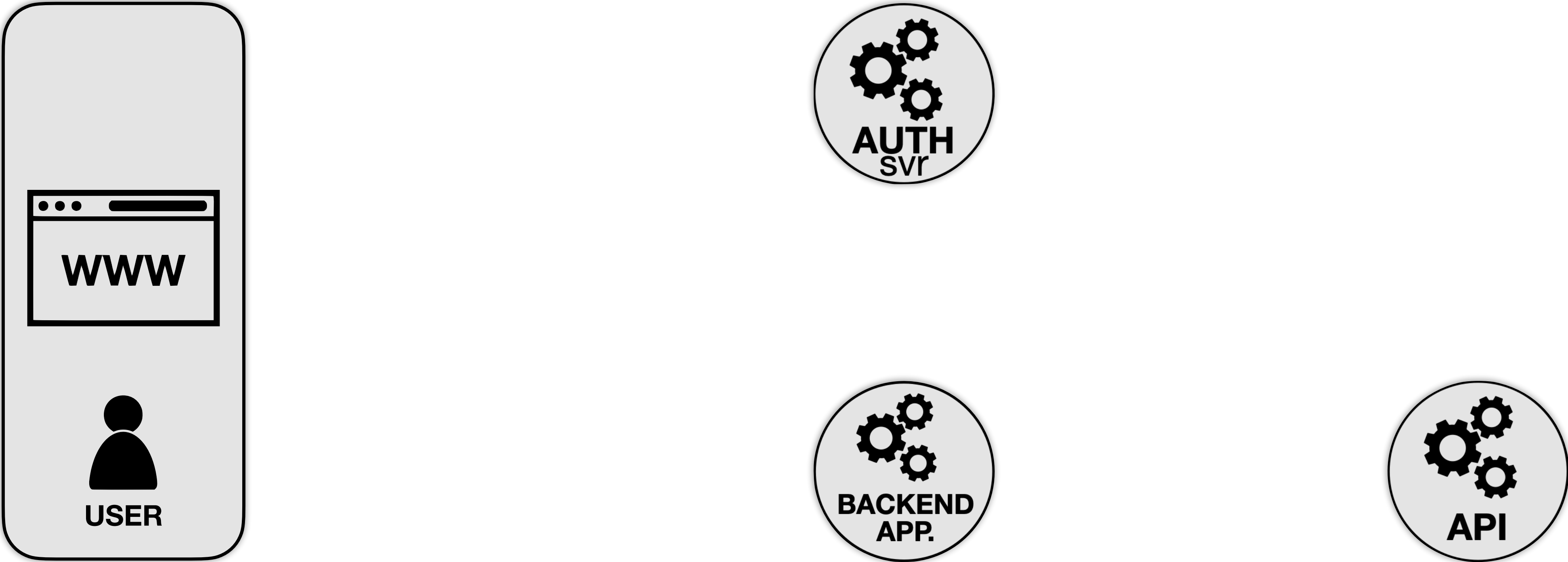
- Authorization-code flow
- Client-credentials flow
- Refresh-token flow
- Device-Authorization flow
- Implicit flow
- Resource-Owner flow

AuthorizationCodeFlow (backend app.)

AuthorizationCodeFlow (backend app.)



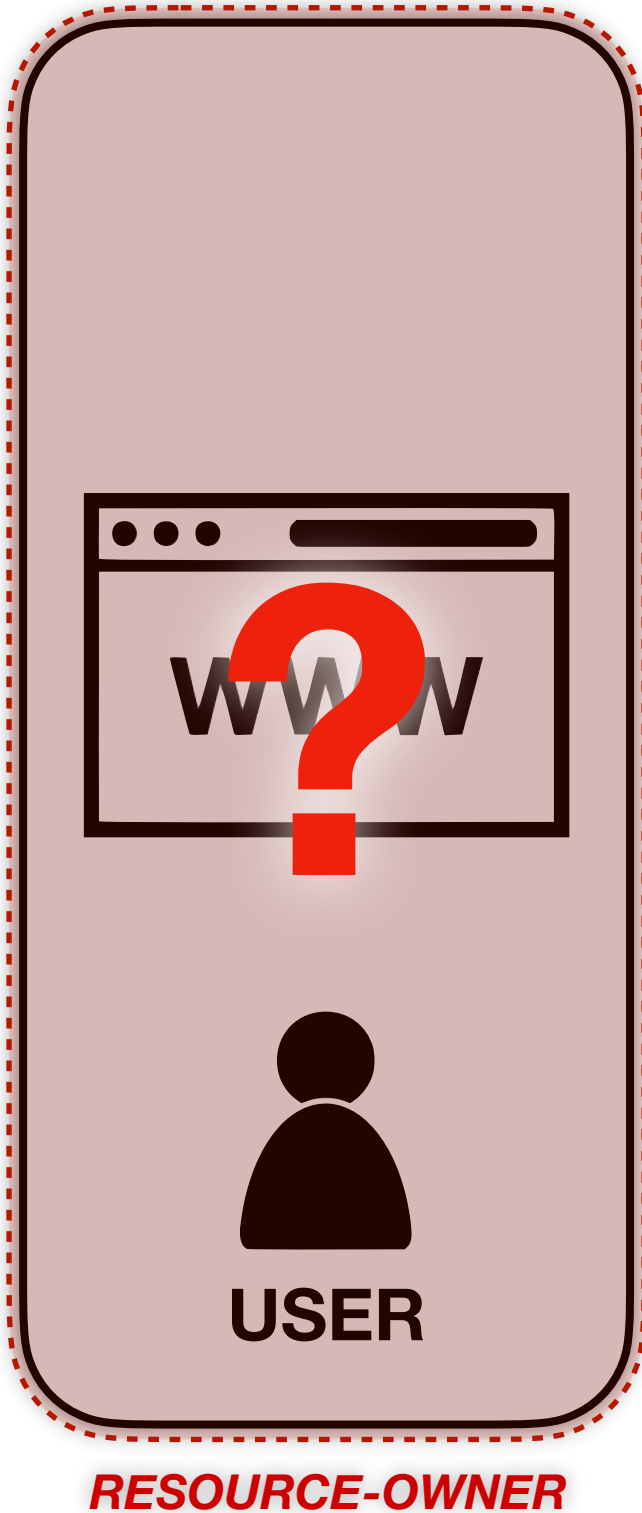
AuthorizationCodeFlow (backend app.)



Termini:

usuali	OAuth	Open ID Connect (OIDC)
User	Resource-Owner	End-User
API	Resource-Server	- - -
Security Token Service	Authorization-Server	Open ID Provider
Client	Client	Relying-Party

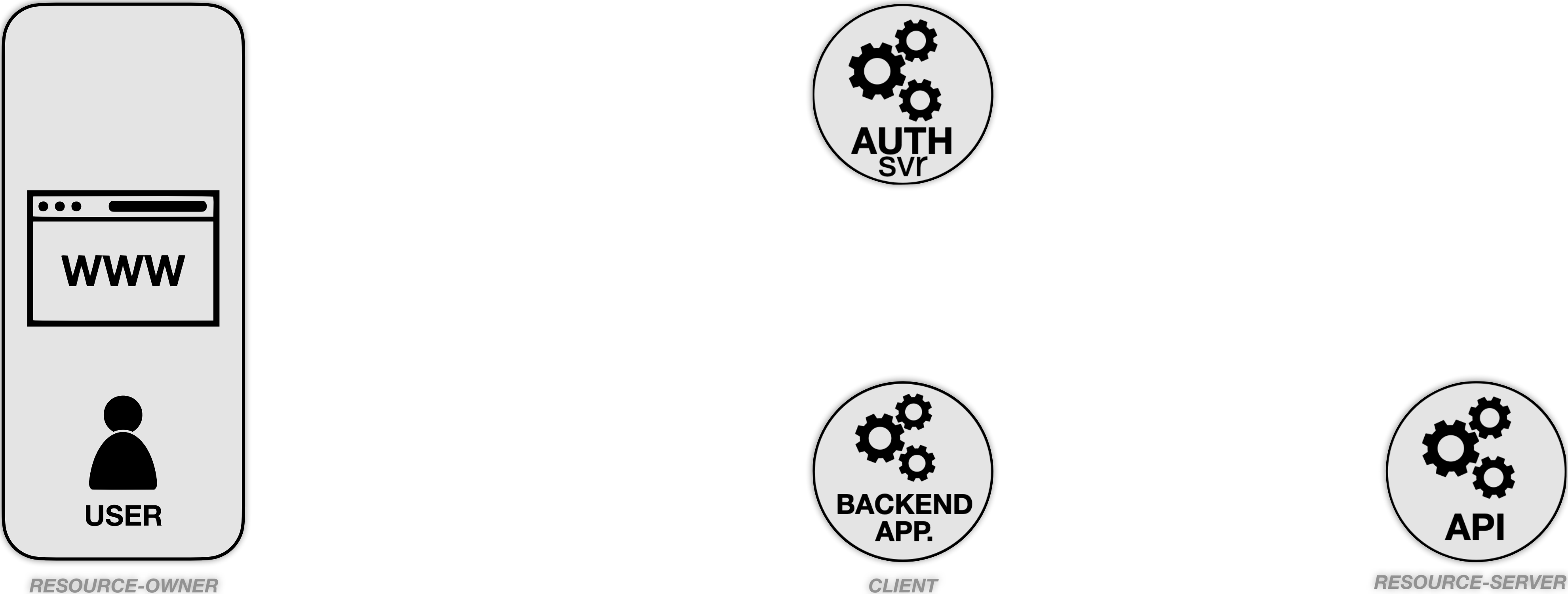
AuthorizationCodeFlow (backend app.)



Termini:

usuali	OAuth	Open ID Connect (OIDC)
User	Resource-Owner	End-User
API	Resource-Server	- - -
Security Token Service	Authorization-Server	Open ID Provider
Client	Client	Relying-Party

AuthorizationCodeFlow (backend app.)



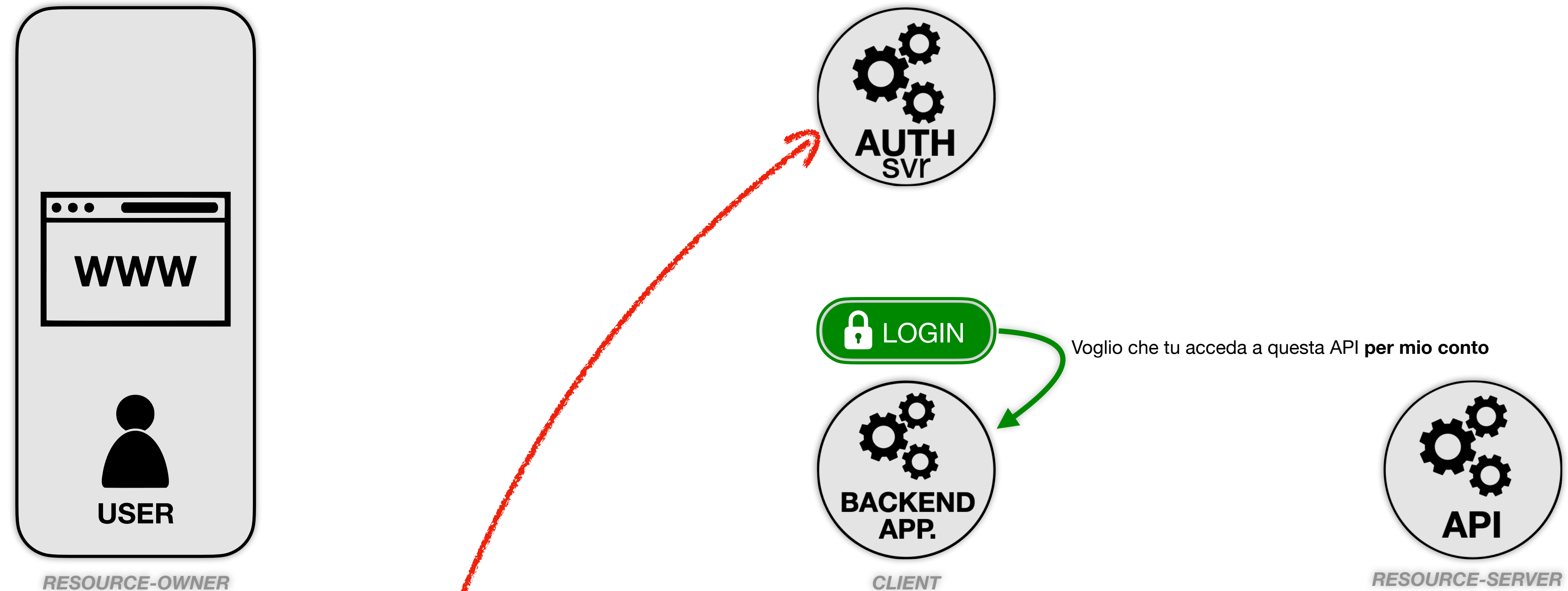
Termini:

usuali	OAuth	Open ID Connect (OIDC)
User	Resource-Owner	End-User
API	Resource-Server	- - -
Security Token Service	Authorization-Server	Open ID Provider
Client	Client	Relying-Party

AuthorizationCodeFlow (backend app.)

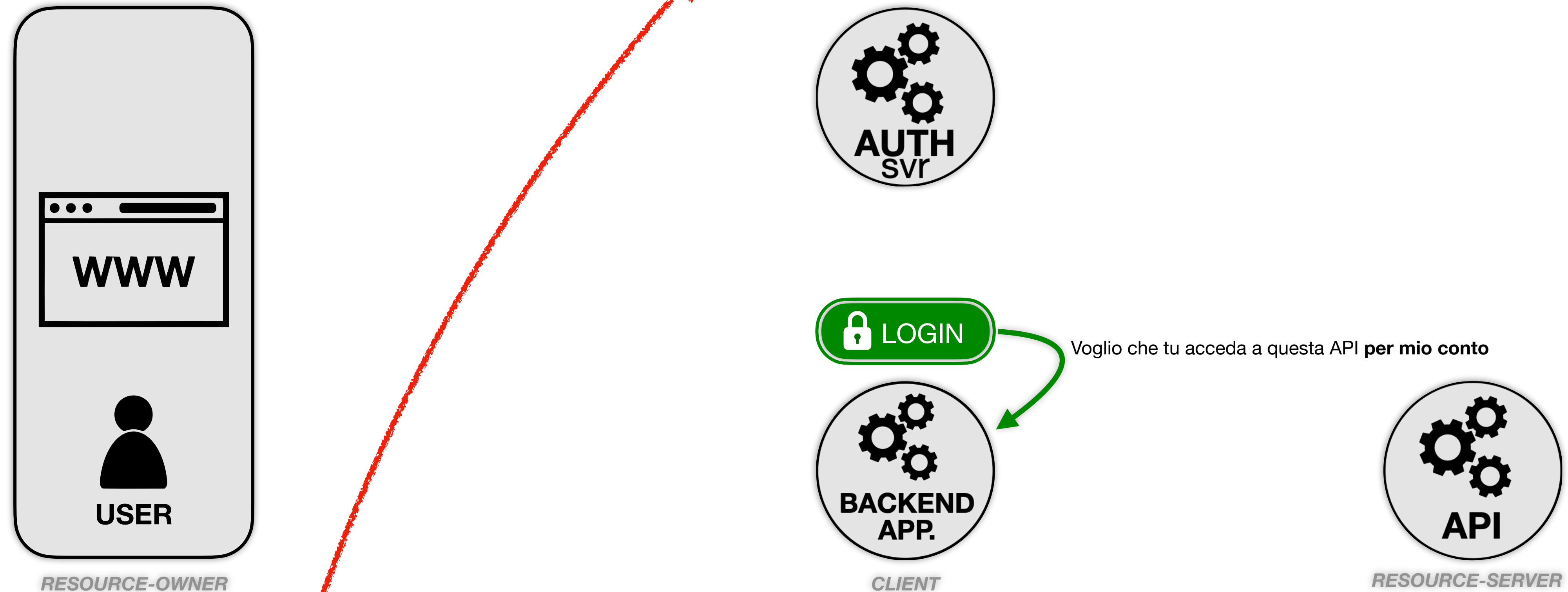


AuthorizationCodeFlow (backend app.)



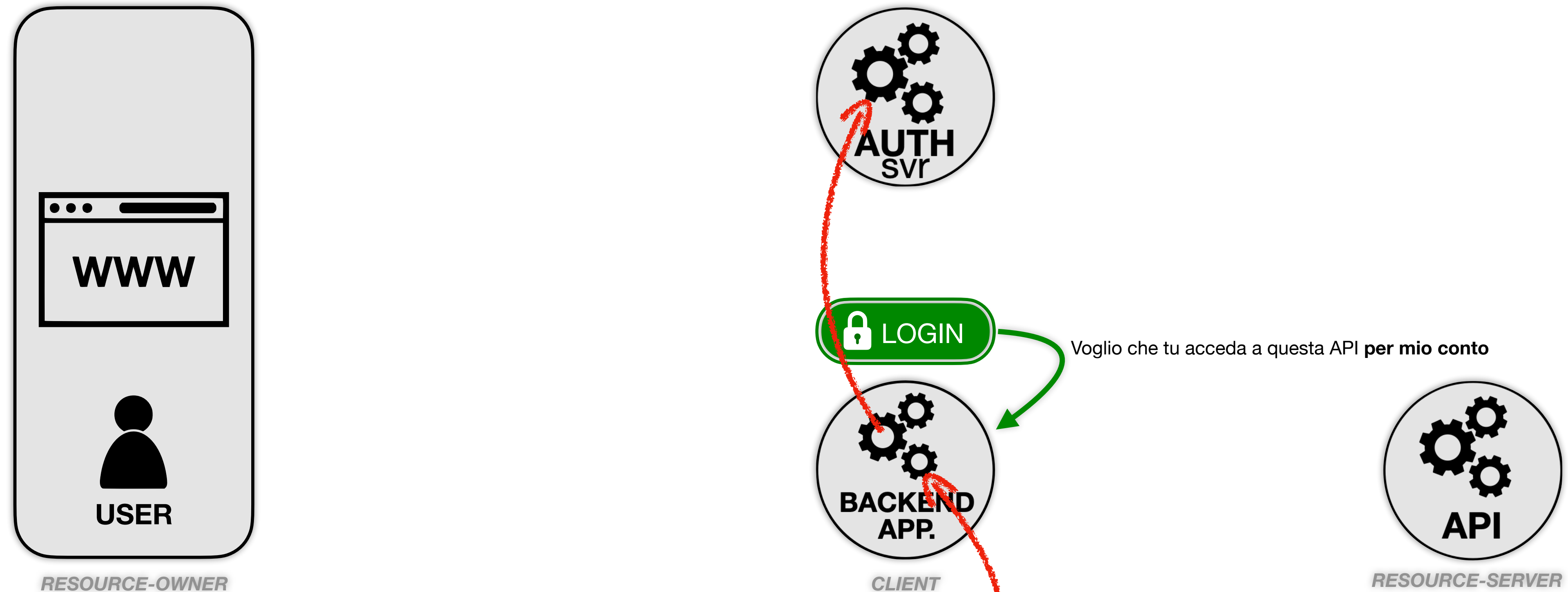
[https://accounts.google.com/o/oauth2/v2/auth?response_type=code&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42&redirect_uri=https://levante.com/callback&scope=profile contacts](https://accounts.google.com/o/oauth2/v2/auth?response_type=code&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42&redirect_uri=https://levante.com/callback&scope=profile%20contacts)

Authorization **Code** Flow (backend app.)



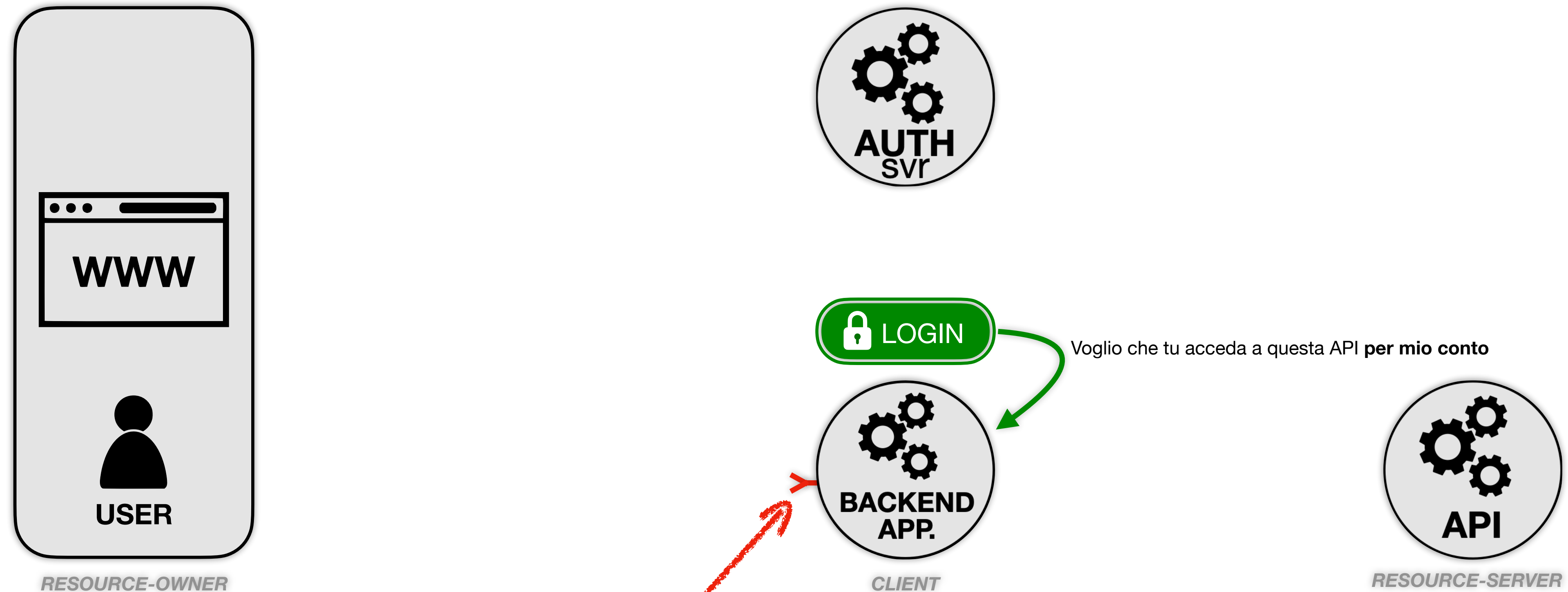
```
https://accounts.google.com/o/oauth2/v2/auth  
?response_type=code  
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42  
&redirect_uri=https://levante.com/callback  
&scope=profile contacts
```

AuthorizationCodeFlow (backend app.)



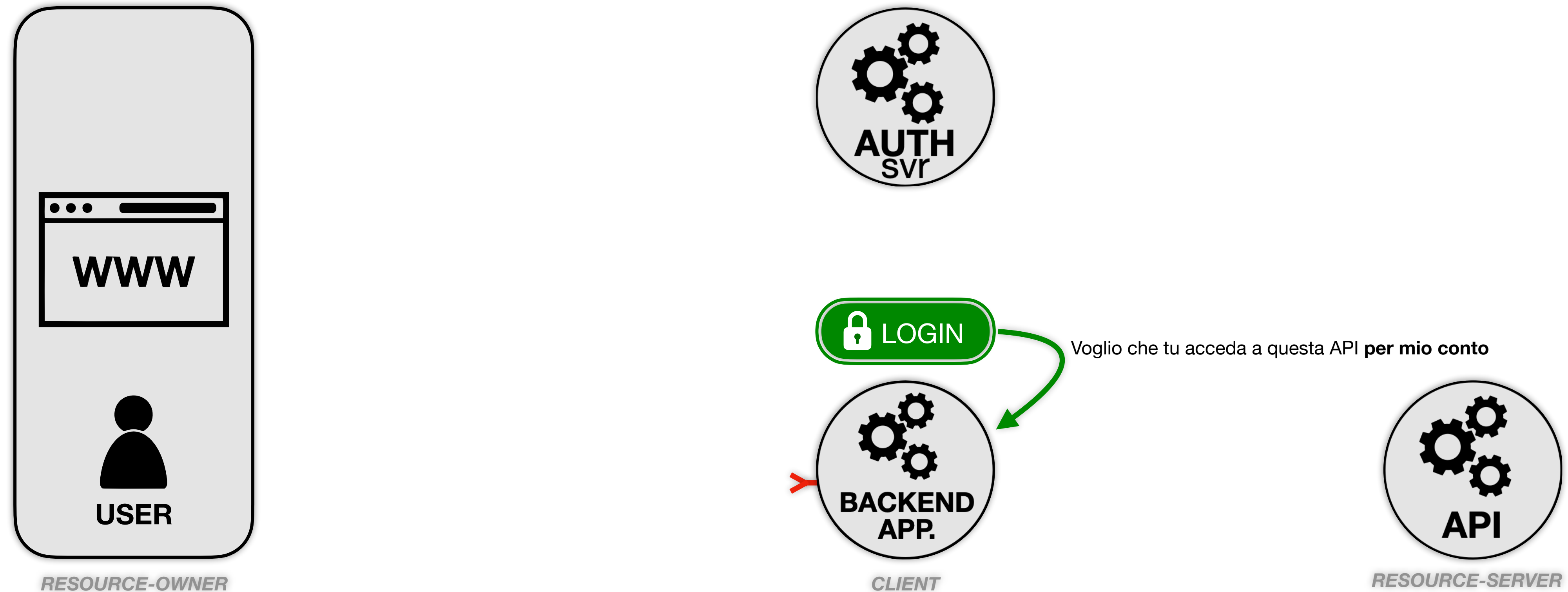
```
https://accounts.google.com/o/oauth2/v2/auth  
?response_type=code  
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42  
&redirect_uri=https://levante.com/callback  
&scope=profile contacts
```

AuthorizationCodeFlow (backend app.)



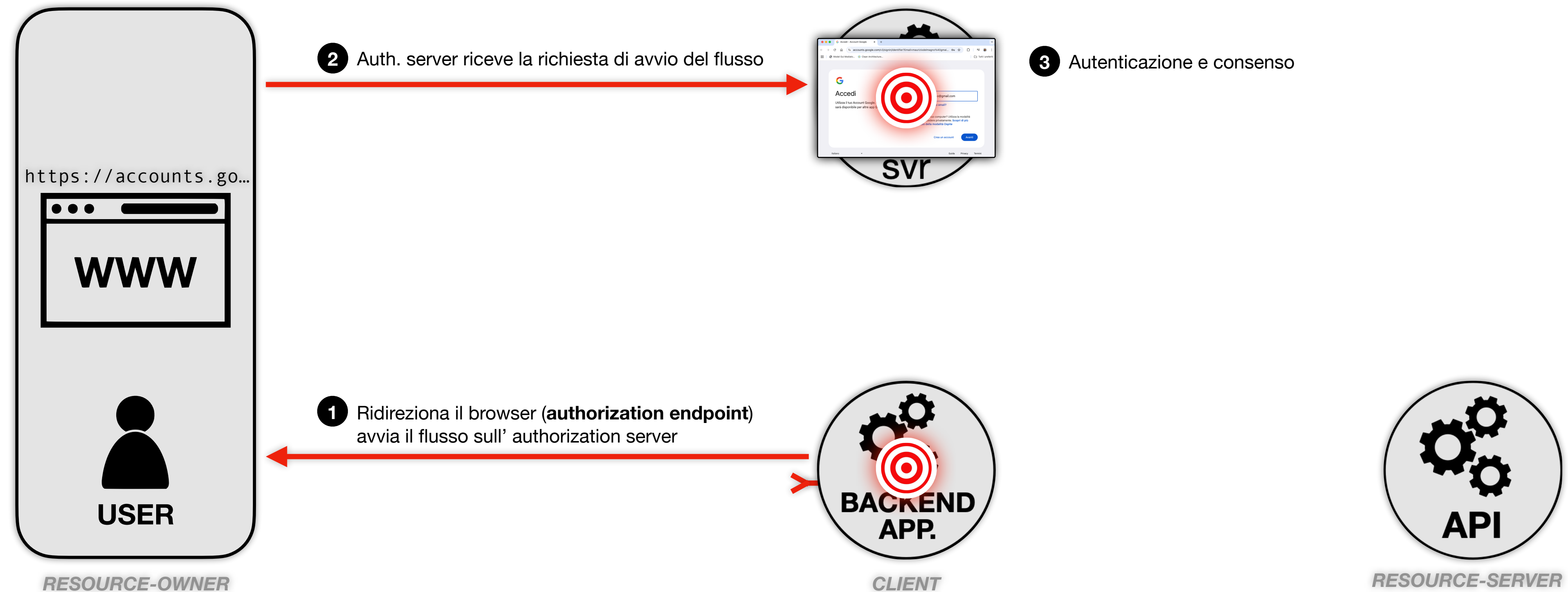
```
https://accounts.google.com/o/oauth2/v2/auth  
?response_type=code  
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42  
&redirect_uri=https://levante.com/callback  
&scope=profile contacts
```


AuthorizationCodeFlow (backend app.)



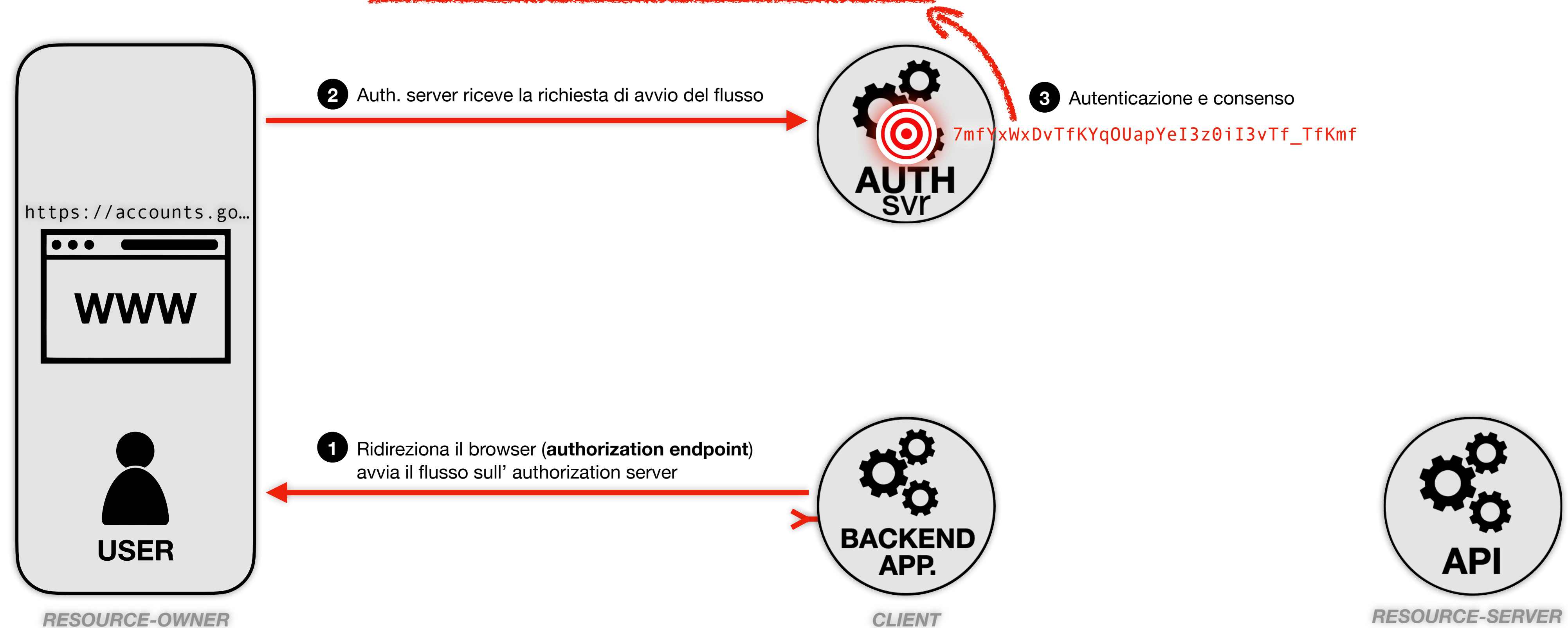
```
https://accounts.google.com/o/oauth2/v2/auth  
?response_type=code  
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42  
&redirect_uri=https://levante.com/callback  
&scope=profile contacts
```

AuthorizationCodeFlow (backend app.)



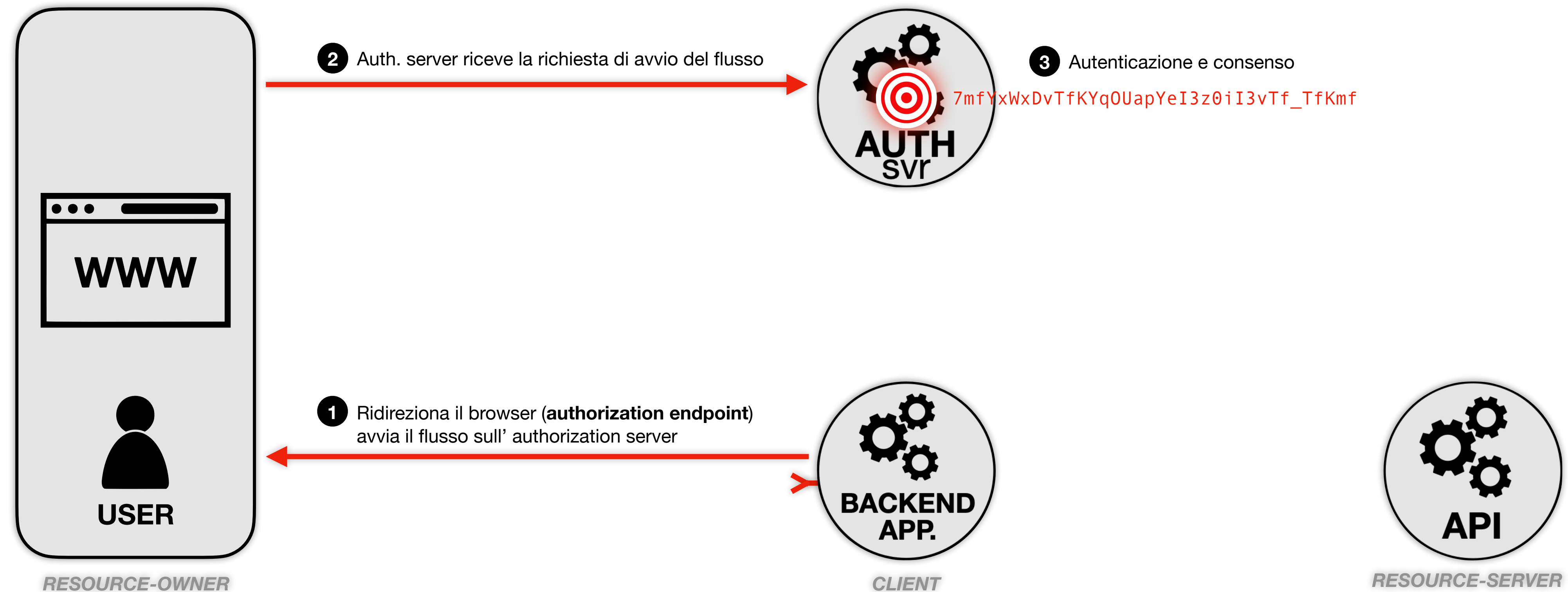
```
https://accounts.google.com/o/oauth2/v2/auth
?response_type=code
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
&redirect_uri=https://levante.com/callback
&scope=profile contacts
```

AuthorizationCodeFlow (backend app.)



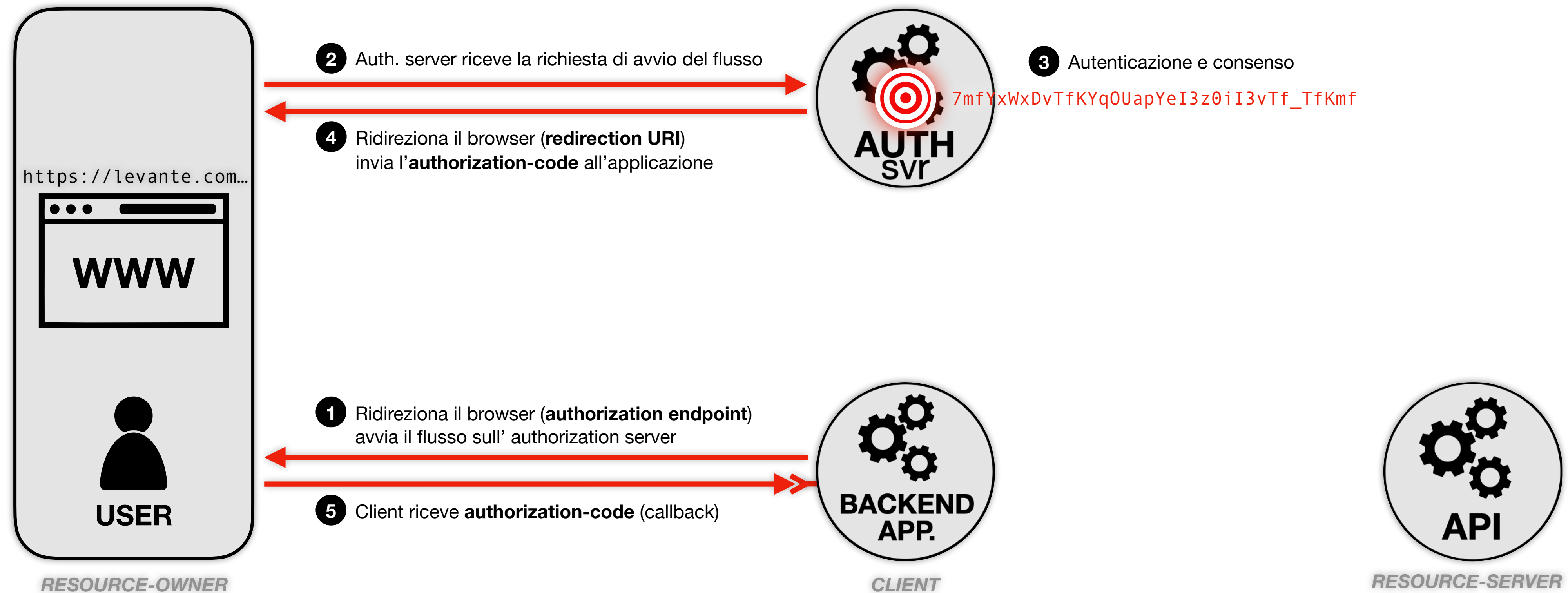
```
https://accounts.google.com/o/oauth2/v2/auth
?response_type=code
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
&redirect_uri=https://levante.com/callback
&scope=profile contacts
```

AuthorizationCodeFlow (backend app.)



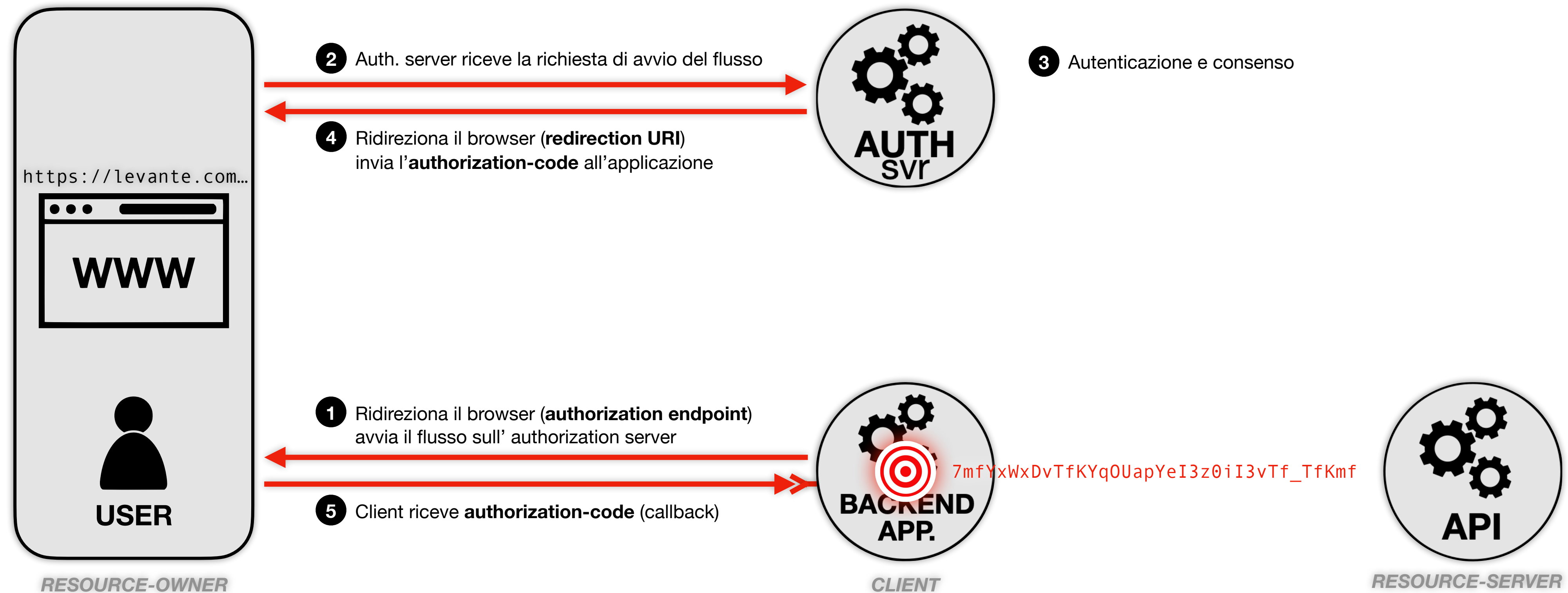
https://levante.com/callback?code=7mfYxWxDvTfKYq0UapYeI3z0iI3vTf_TfKmf

AuthorizationCodeFlow (backend app.)



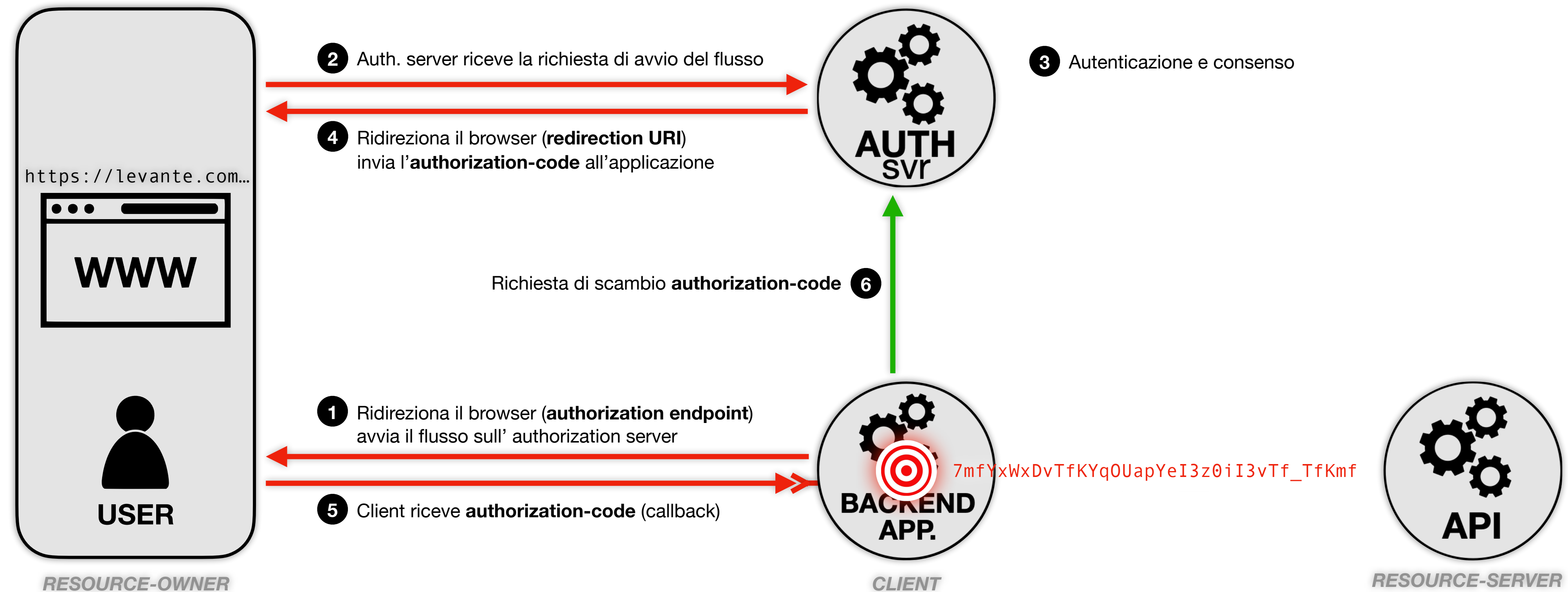
`https://levante.com/callback`
`?code=7mfYxWxDvTfKYq0UapYeI3z0iI3vTf_TfKmf`

AuthorizationCodeFlow (backend app.)



???

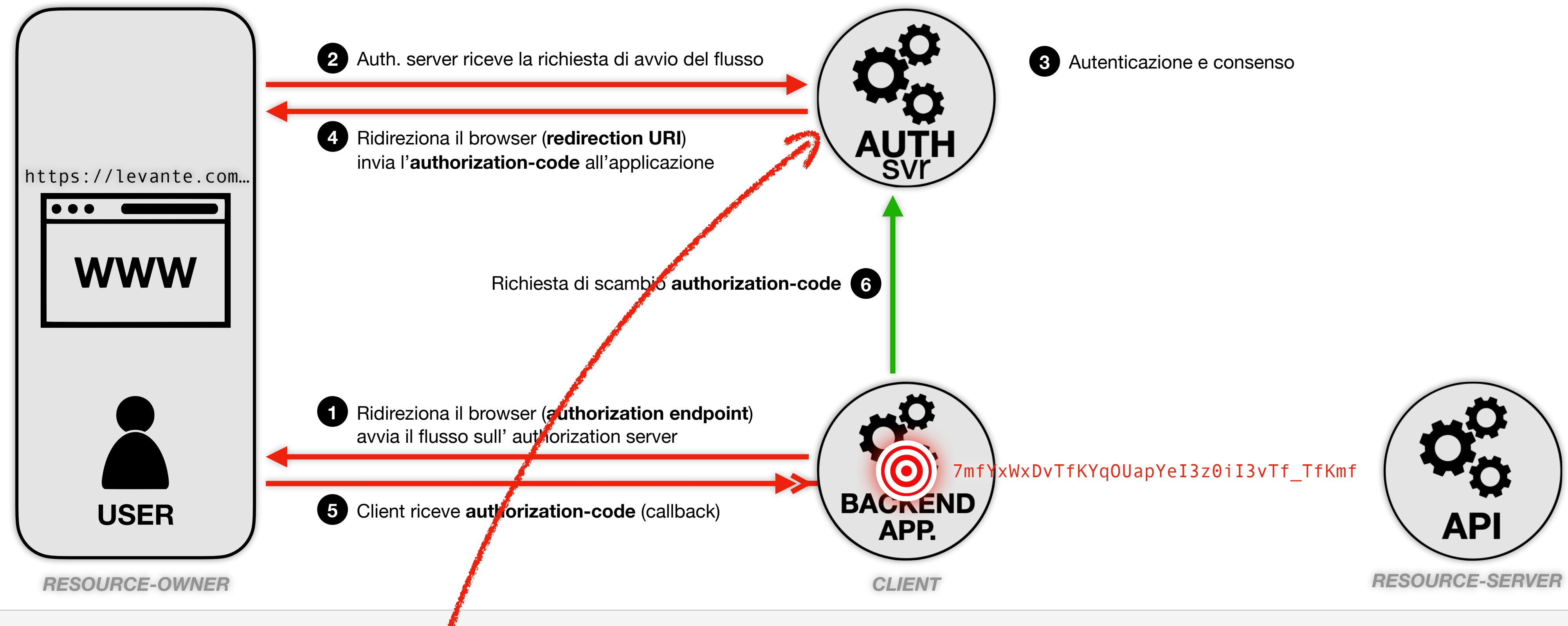
AuthorizationCodeFlow (backend app.)



Scambio **authorization-code** → **access-token**:

- *authorization-code*: sequenza di caratteri casuali (OAuth non specifica nulla al riguardo)
- *significato*: **ti autorizzo a richiedere l'autorizzazione** per gli scope che hai chiesto
- ...subito
- ...one time

AuthorizationCodeFlow (backend app.)



POST www.googleapis.com/oauth2/v4/token

Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code

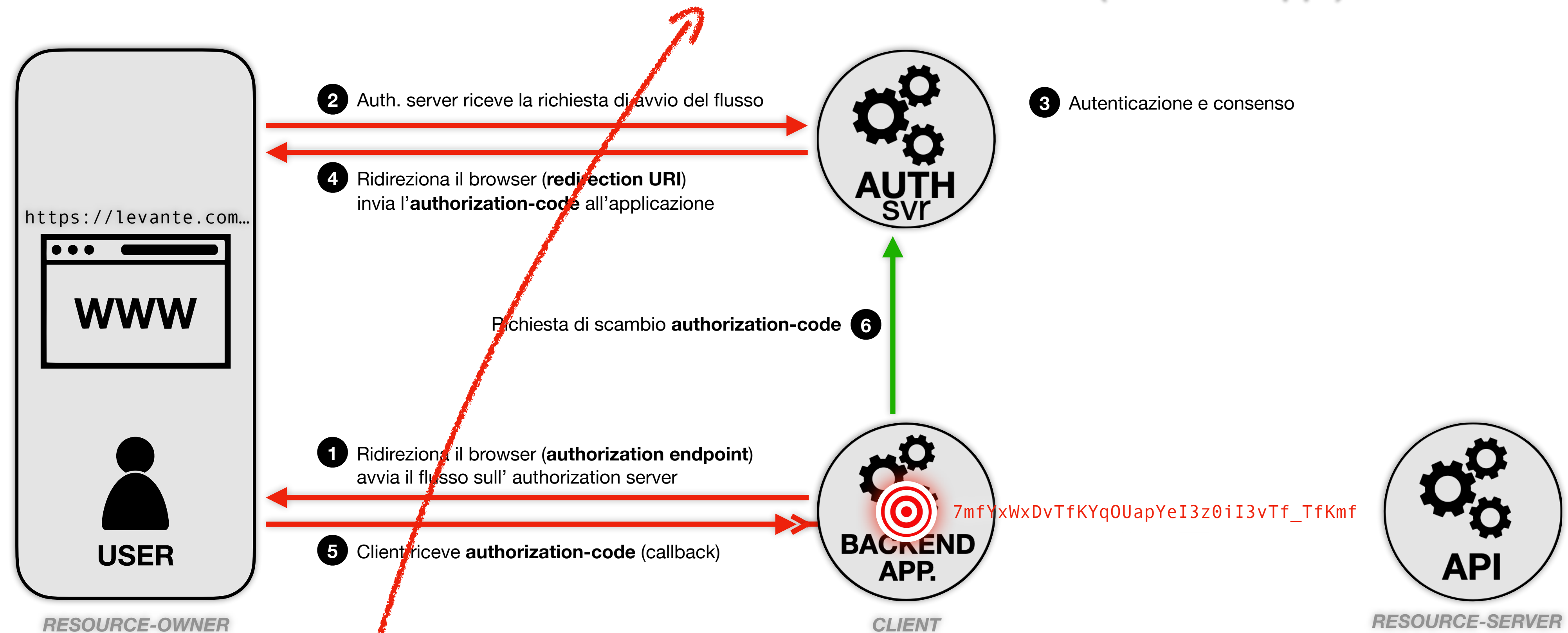
&code=7mfYxWxDvTfKYq0UapYeI3z0iI3vTf_TfKmf

&redirect_uri=https://levante.com/callback

&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42

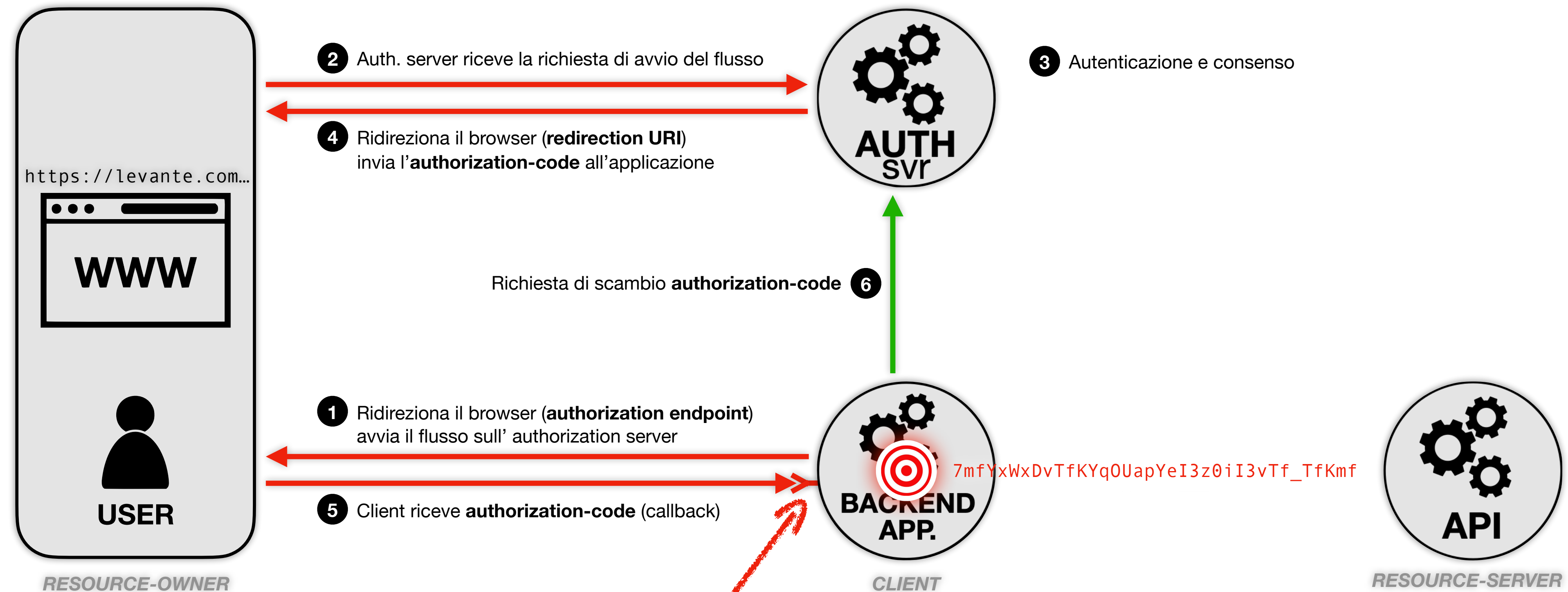
&client_secret=*****

AuthorizationCodeFlow (backend app.)



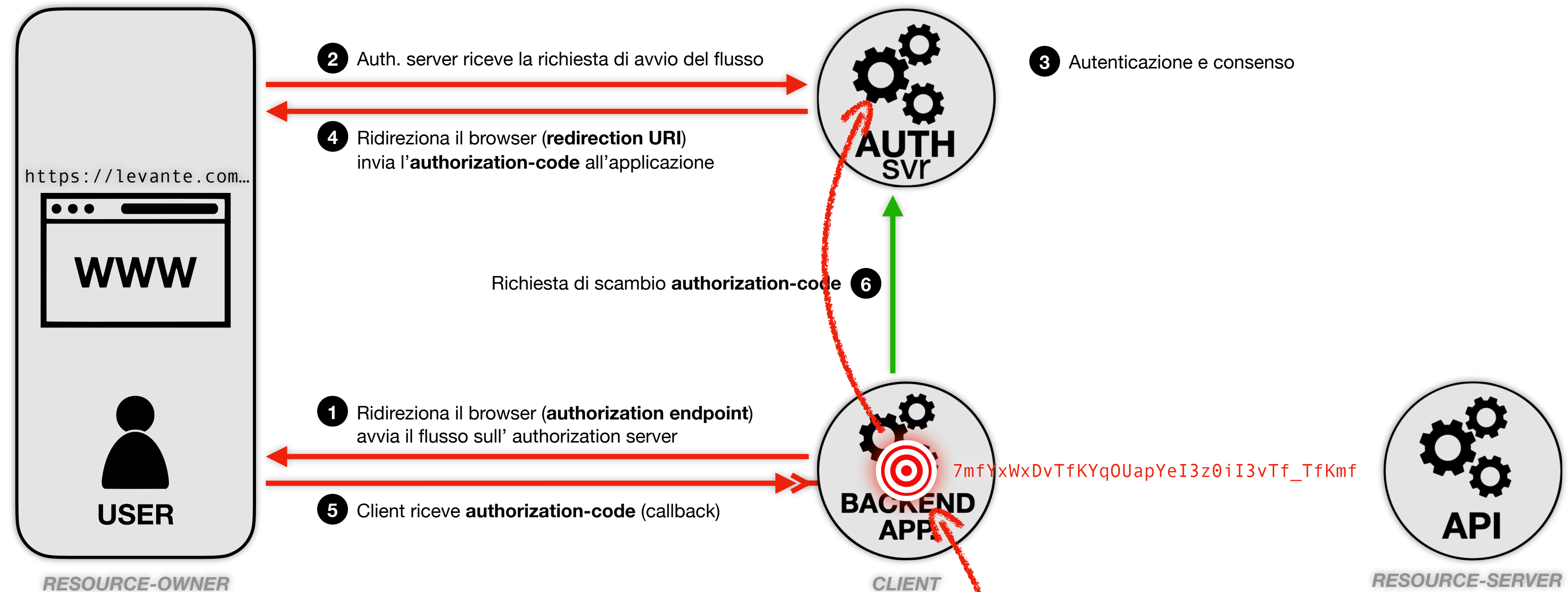
```
POST www.googleapis.com/oauth2/v4/token
Content-Type: application/x-www-form-urlencoded
grant_type=authorization_code
&code=7mfYxWxDvTfKYq0UapYeI3z0iI3vTf_TfKmf
&redirect_uri=https://levante.com/callback
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
&client_secret=*****
```

AuthorizationCodeFlow (backend app.)



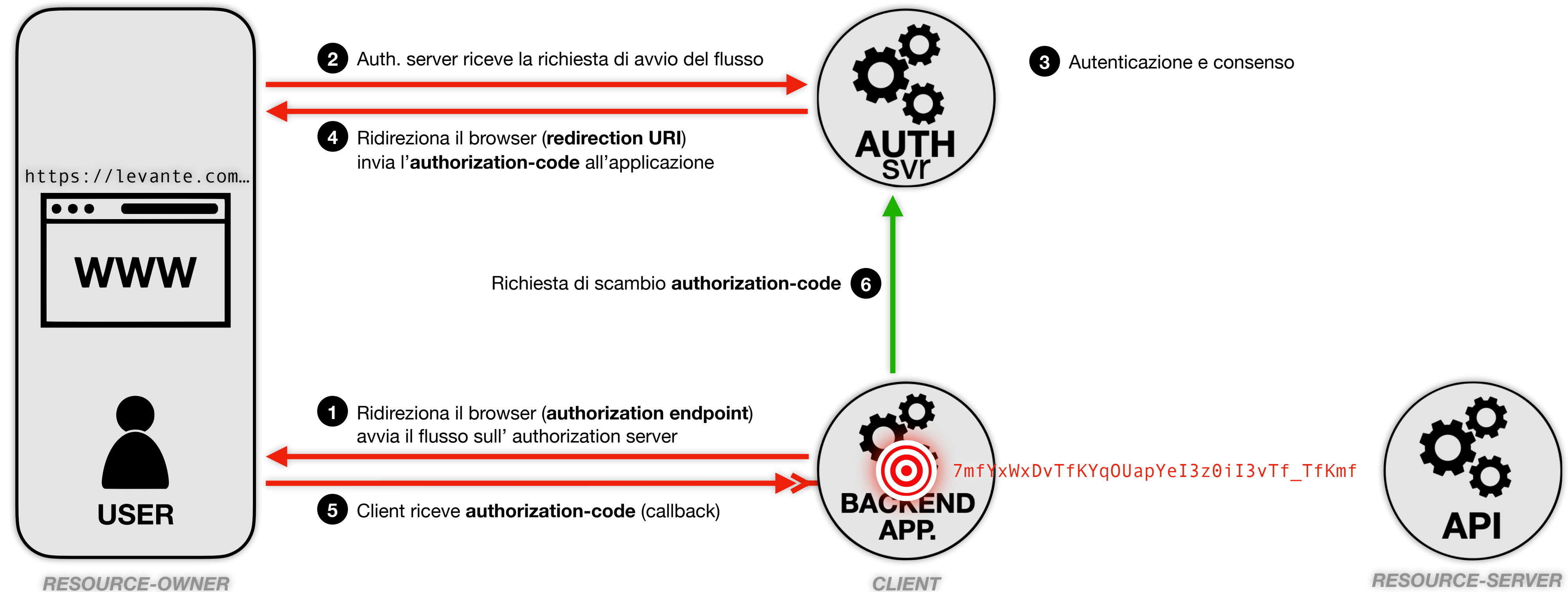
```
POST www.googleapis.com/oauth2/v4/token
Content-Type: application/x-www-form-urlencoded
grant_type=authorization_code
&code=7mfYxWxDvTfKYq0UapYeI3z0iI3vTf_TfKmf
&redirect_uri=https://levante.com/callback
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
&client_secret=*****
```

AuthorizationCodeFlow (backend app.)



```
POST www.googleapis.com/oauth2/v4/token
Content-Type: application/x-www-form-urlencoded
grant_type=authorization_code
&code=7mfYxWxDvTfKYq0UapYeI3z0iI3vTf_TfKmf
&redirect_uri=https://levante.com/callback
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
&client_secret=*****
```

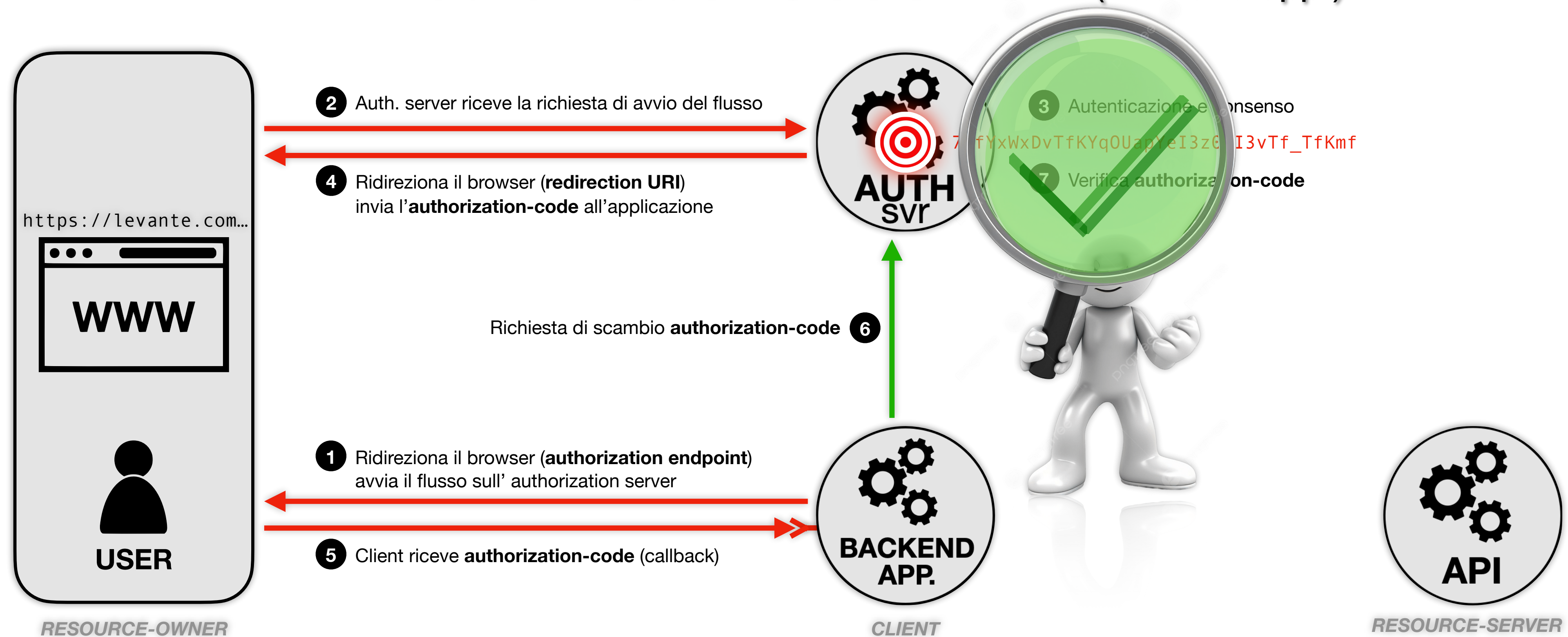

AuthorizationCodeFlow (backend app.)



```
POST www.googleapis.com/oauth2/v4/token
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code
&code=7mfYxWxDvTfKYq0UapYeI3z0iI3vTf_TfKmf
&redirect_uri=https://levante.com/callback
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
&client_secret=*****
```

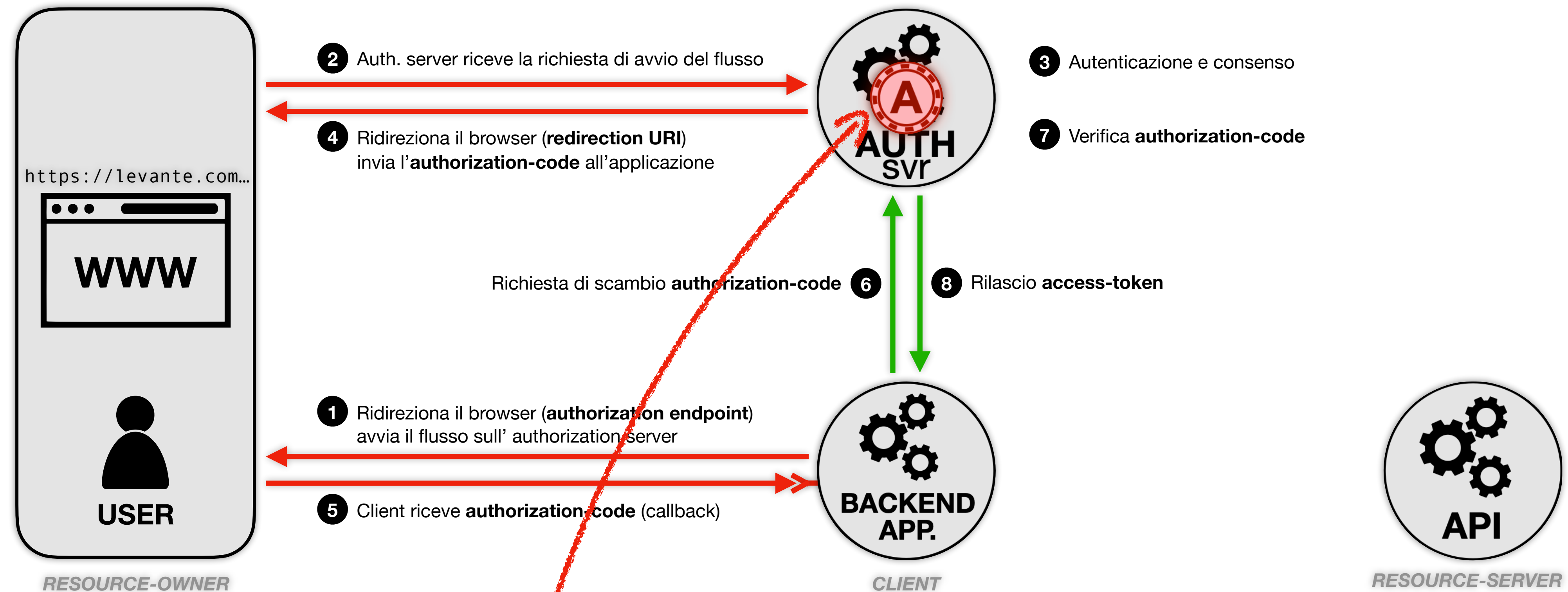

AuthorizationCodeFlow (backend app.)



```
POST www.googleapis.com/oauth2/v4/token
Content-Type: application/x-www-form-urlencoded

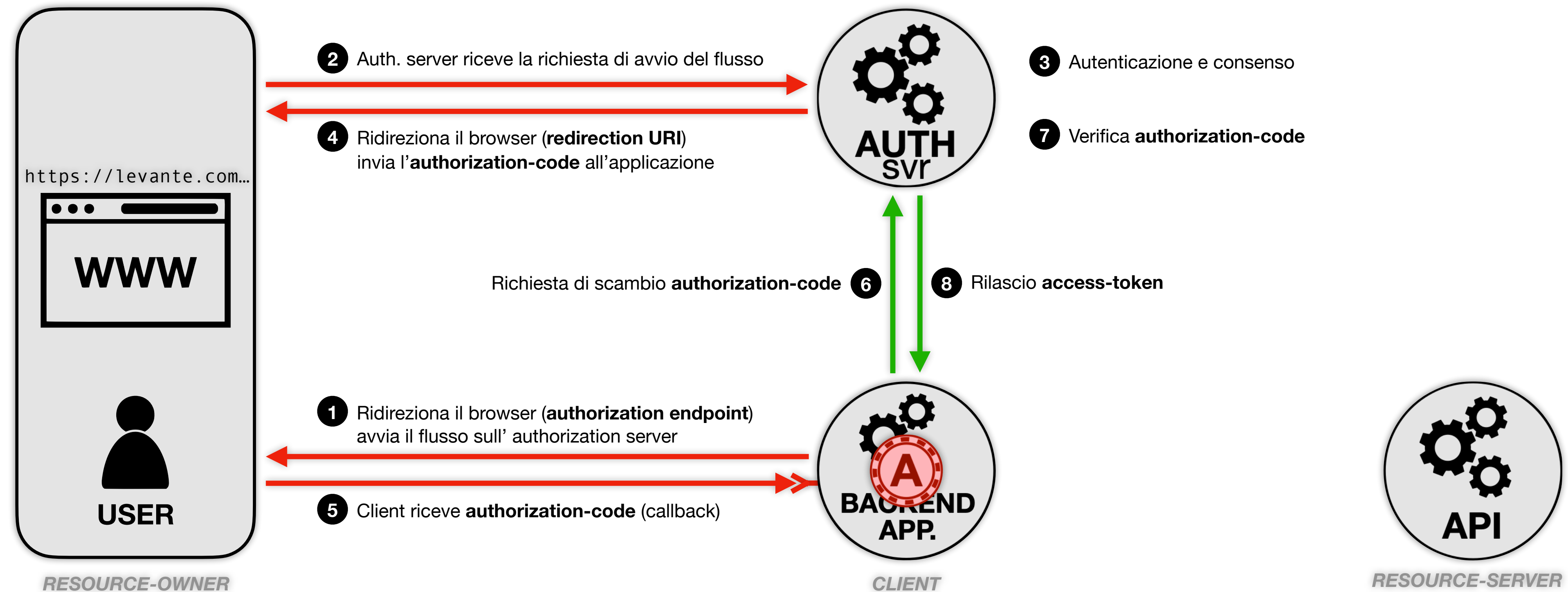
grant_type=authorization_code
&code=7mfYxWxDvTfKYq0UapYeI3z0iI3vTf_TfKmf
&redirect_uri=https://levante.com/callback
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
&client_secret=*****
```

AuthorizationCodeFlow (backend app.)



```
{  
  "access_token": "eyJhbGciOi0...du6TY9w",  
  "expires_in": 3600,  
  "token_type": "Bearer"  
}
```

AuthorizationCodeFlow (backend app.)



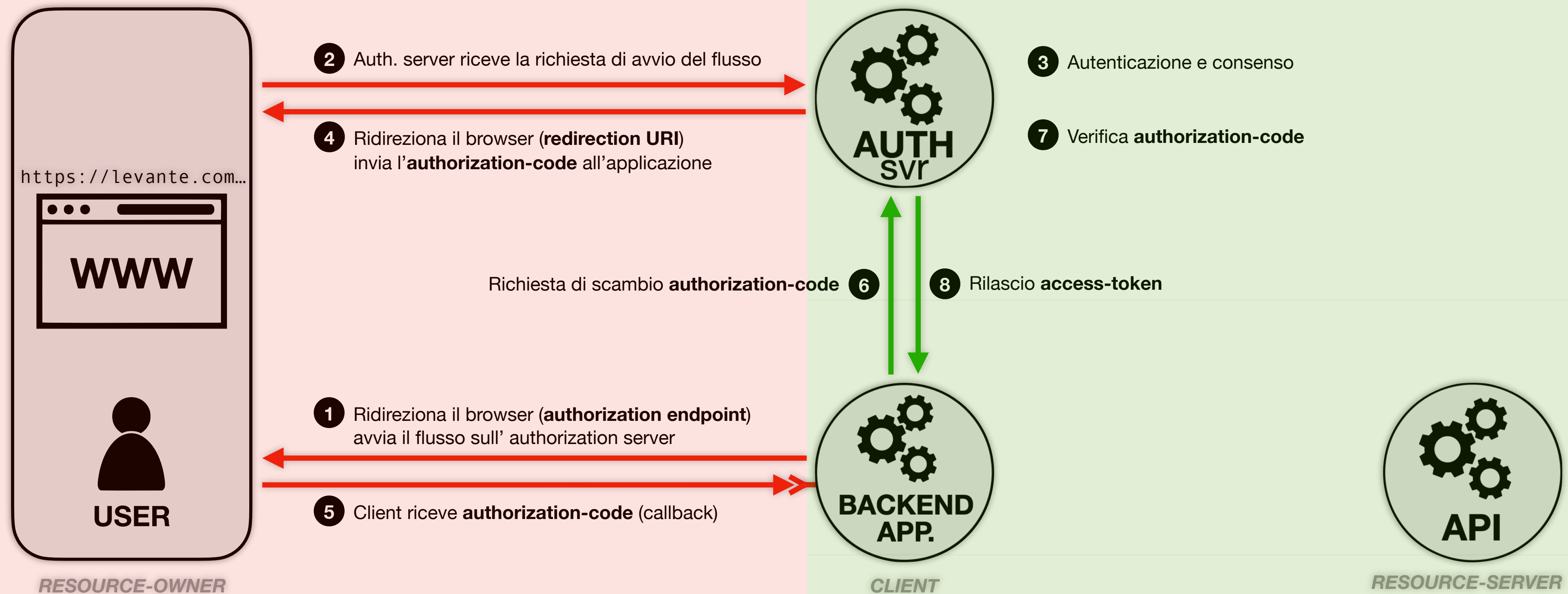
```
{  
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXZWQ",  
  "expires_in": 3600,  
  "token_type": "Bearer"  
}
```

???

Perchè questo doppio passaggio?

FrontChannel vs BackChannel

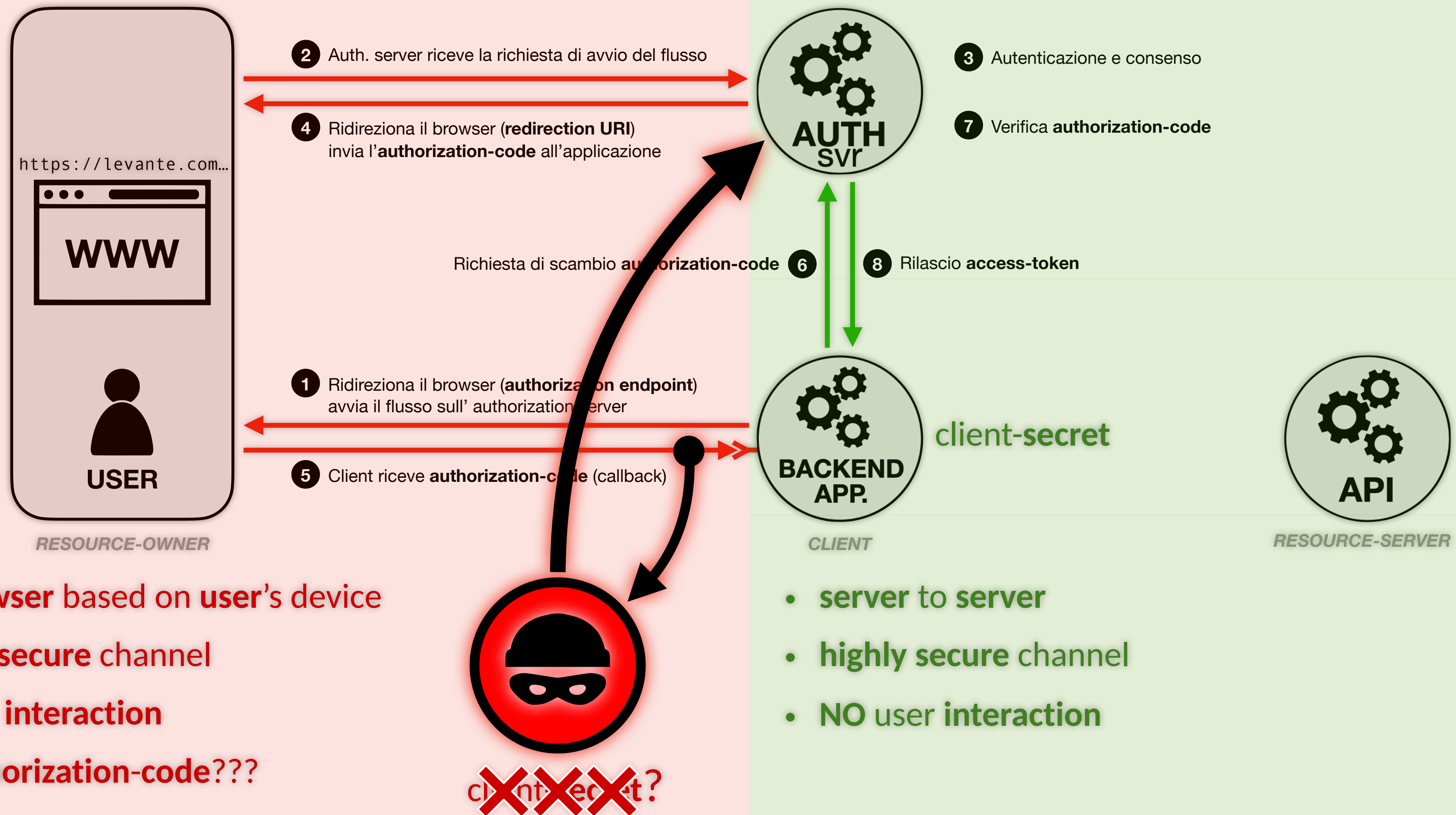
FrontChannel vs BackChannel



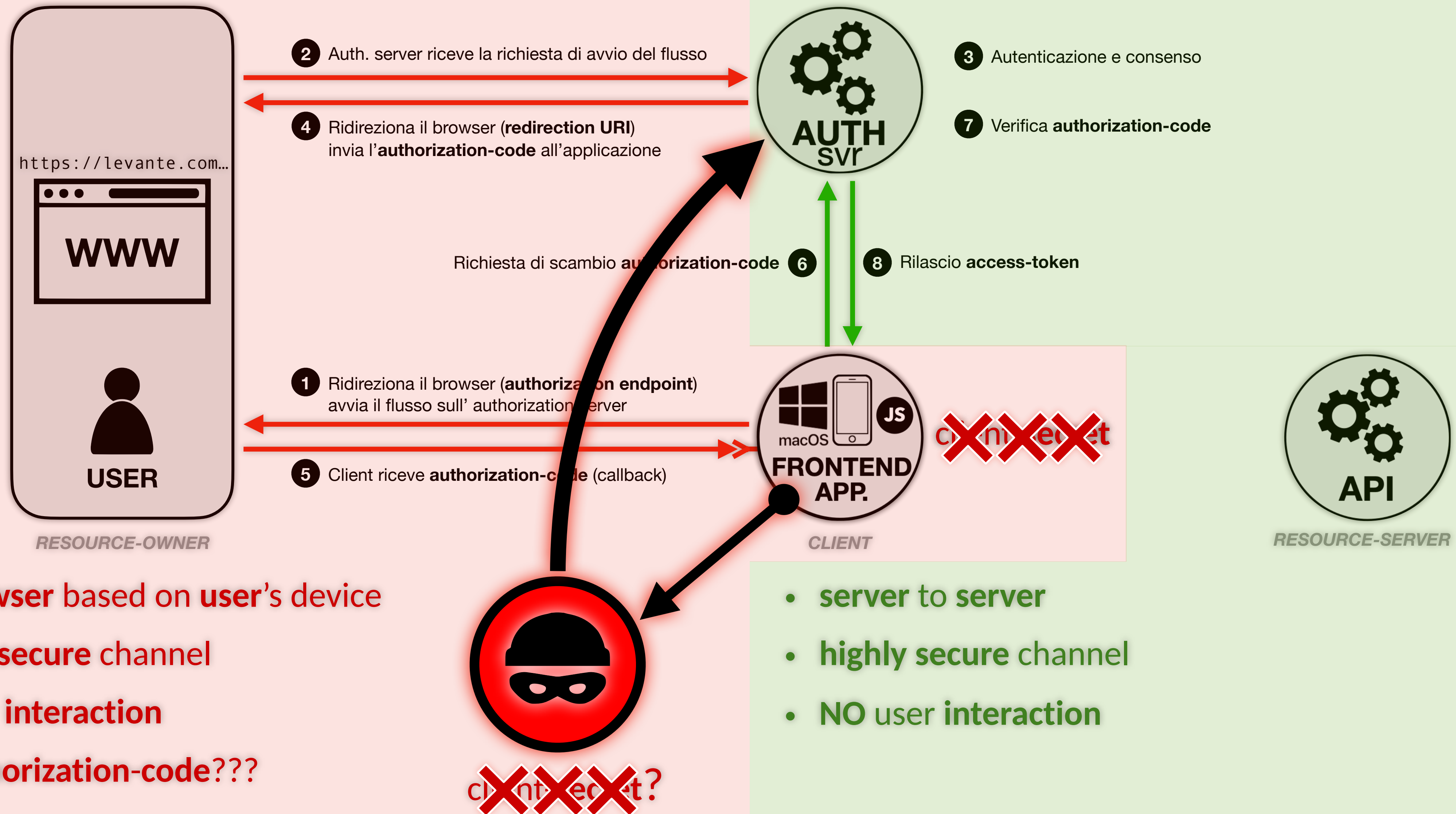
- **browser based on user's device**
- **less secure channel**
- **user interaction**
- **authorization-code???**

- **server to server**
- **highly secure channel**
- **NO user interaction**

FrontChannel vs BackChannel



FrontChannel vs BackChannel

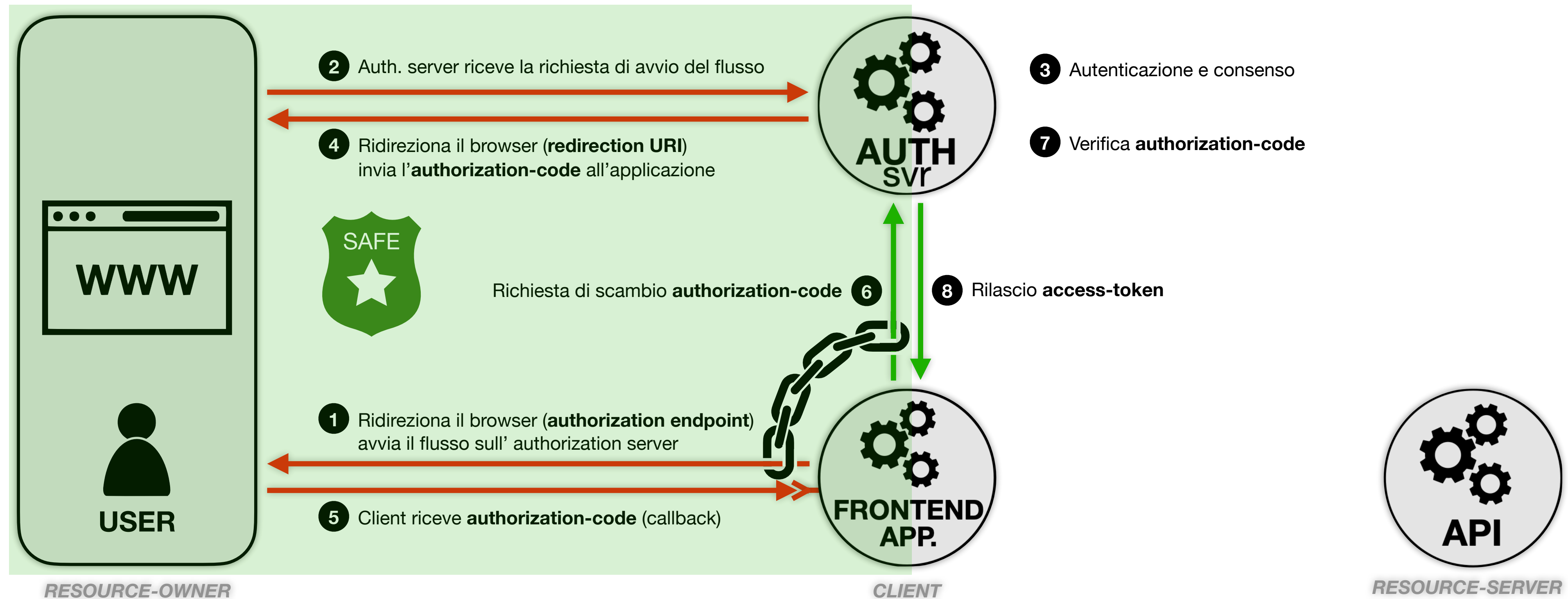


- browser based on user's device
- less secure channel
- user interaction
- authorization-code???

- server to server
- highly secure channel
- NO user interaction

PKCE (Proof Key for Code Exchange)

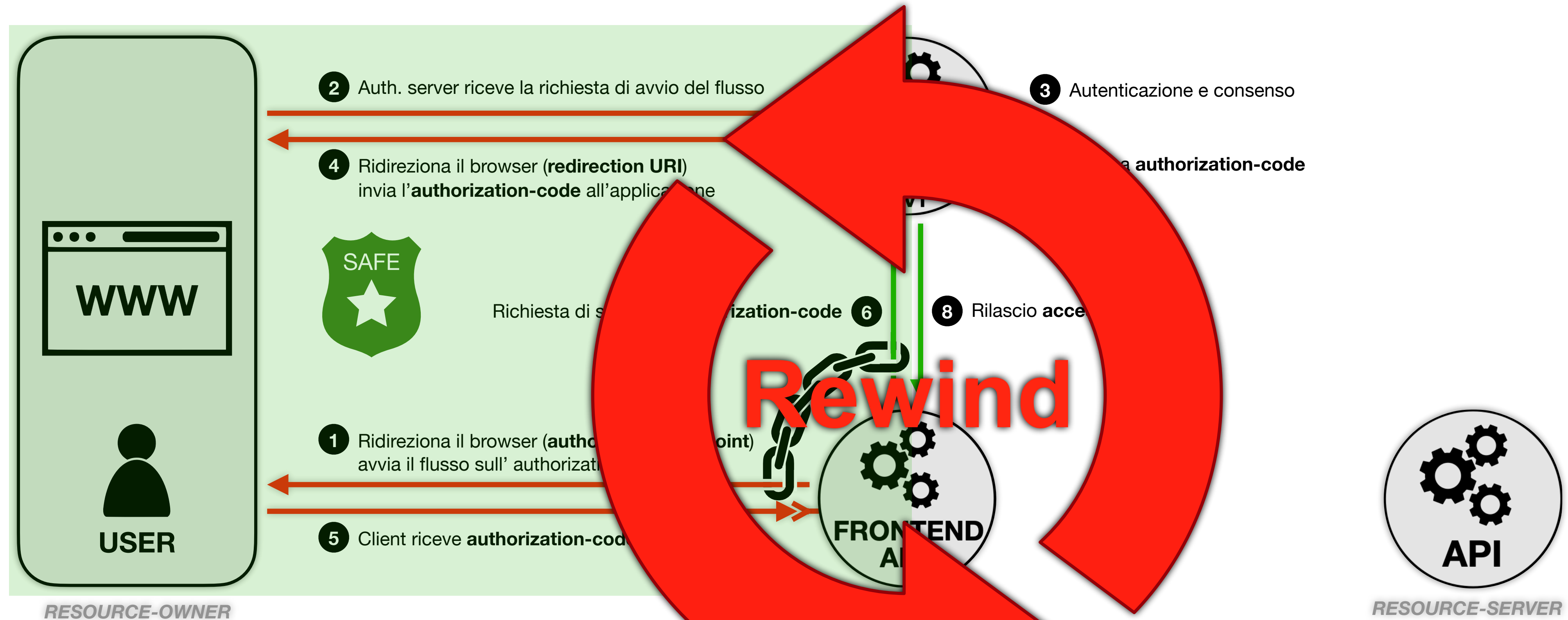
PKCE (Proof Key for Code Exchange)



NO CLIENT-SECRET!!!

- Assicurare che l'**istanza client** che ha avviato il flusso (step 1) **sia la stessa** che sta cercando di effettuare lo scambio (step 6)
- Preservare l'integrità dell'intero flusso **dallo step 1 al 6**
- Il primo passo sicuro (back-channel) **valida** tutti i precedenti meno sicuri (front-channel)

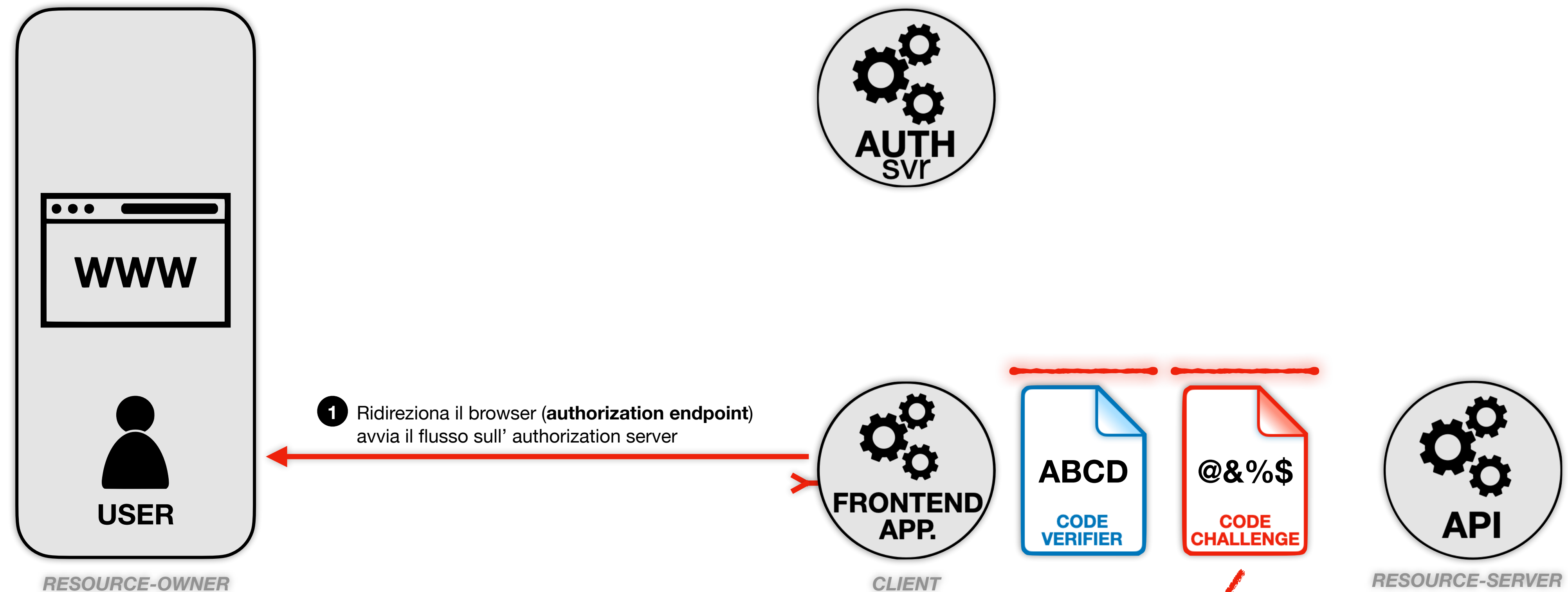
PKCE (Proof Key for Code Exchange)



NO CLIENT-SECRET!!!

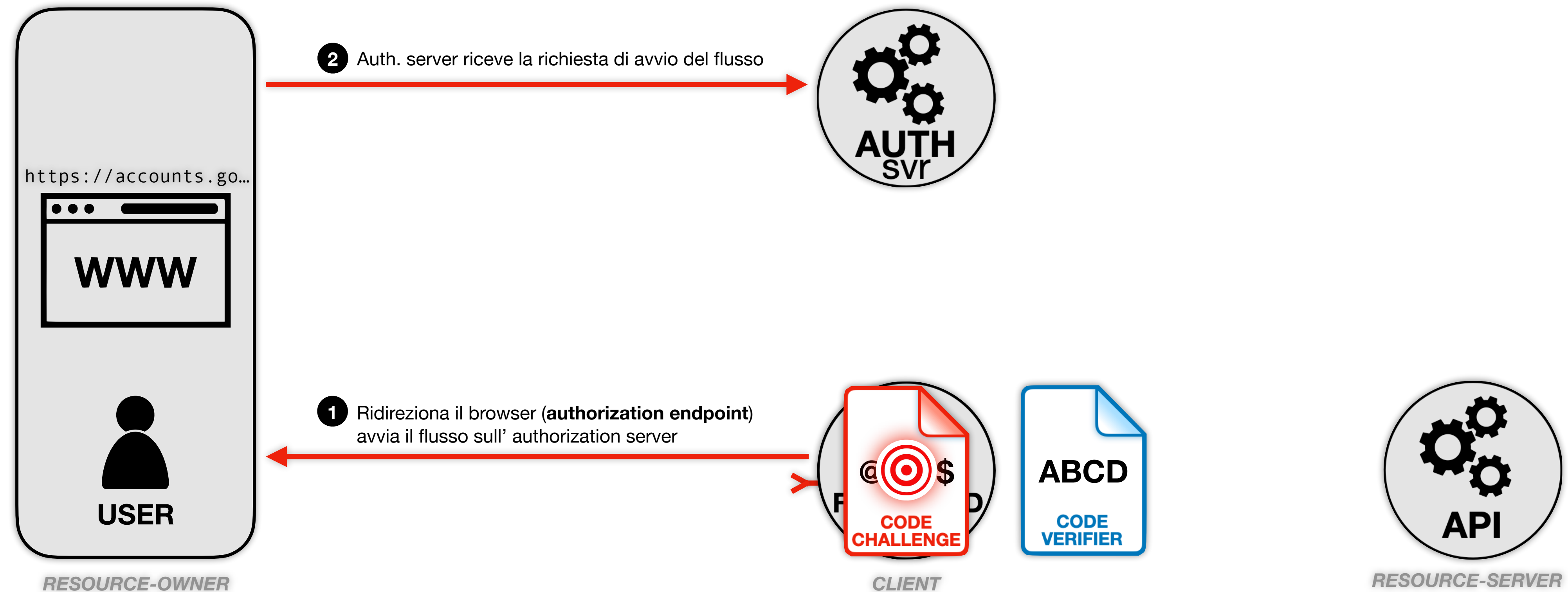
- Assicurare che l'istanza **client** che ha avviato il flusso (step 1) sia la stessa che sta cercando di effettuare lo scambio (step 6)
- Preservare l'integrità dell'intero flusso **dallo step 1 al 6**
- Il primo passo sicuro (back-channel) **valida** tutti i precedenti meno sicuri (front-channel)

PKCE (Proof Key for Code Exchange)



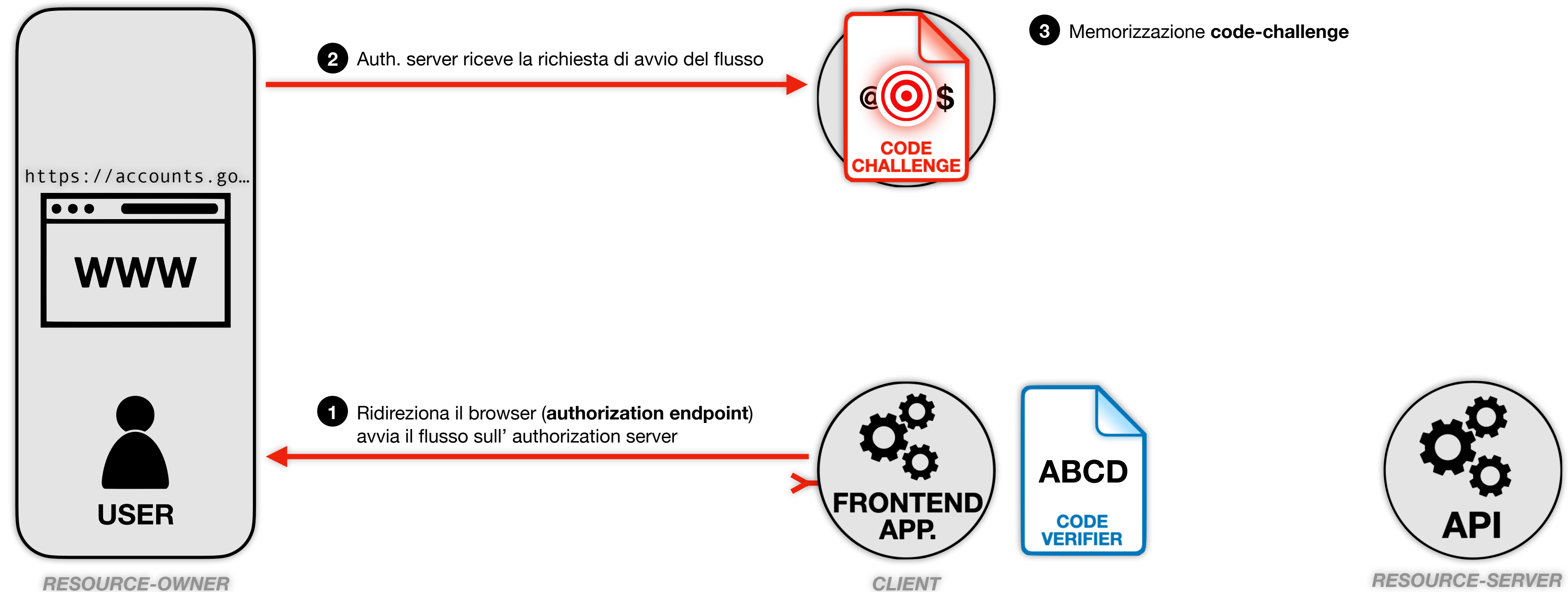
```
https://accounts.google.com/o/oauth2/v2/auth
?response_type=code
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
&redirect_uri=https://levante.com/callback
&scope=profile contacts
&code_challenge=E9Melhoa20wvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
&code_challenge_method=S256
&state=ABCD...
```


PKCE (Proof Key for Code Exchange)



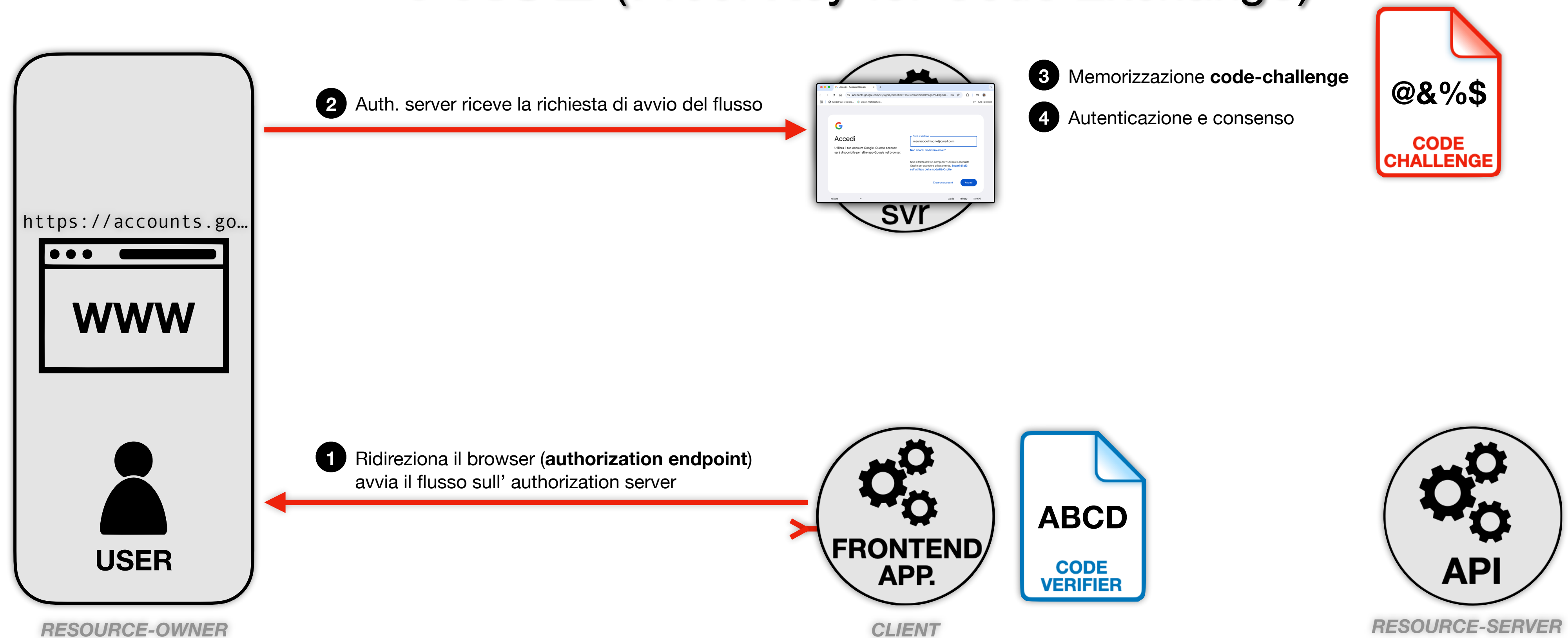
```
https://accounts.google.com/o/oauth2/v2/auth
?response_type=code
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
&redirect_uri=https://levante.com/callback
&scope=profile contacts
&code_challenge=E9Melhoa20wvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
&code_challenge_method=S256
&state=ABCD...&nonce=EFGH...
```


PKCE (Proof Key for Code Exchange)



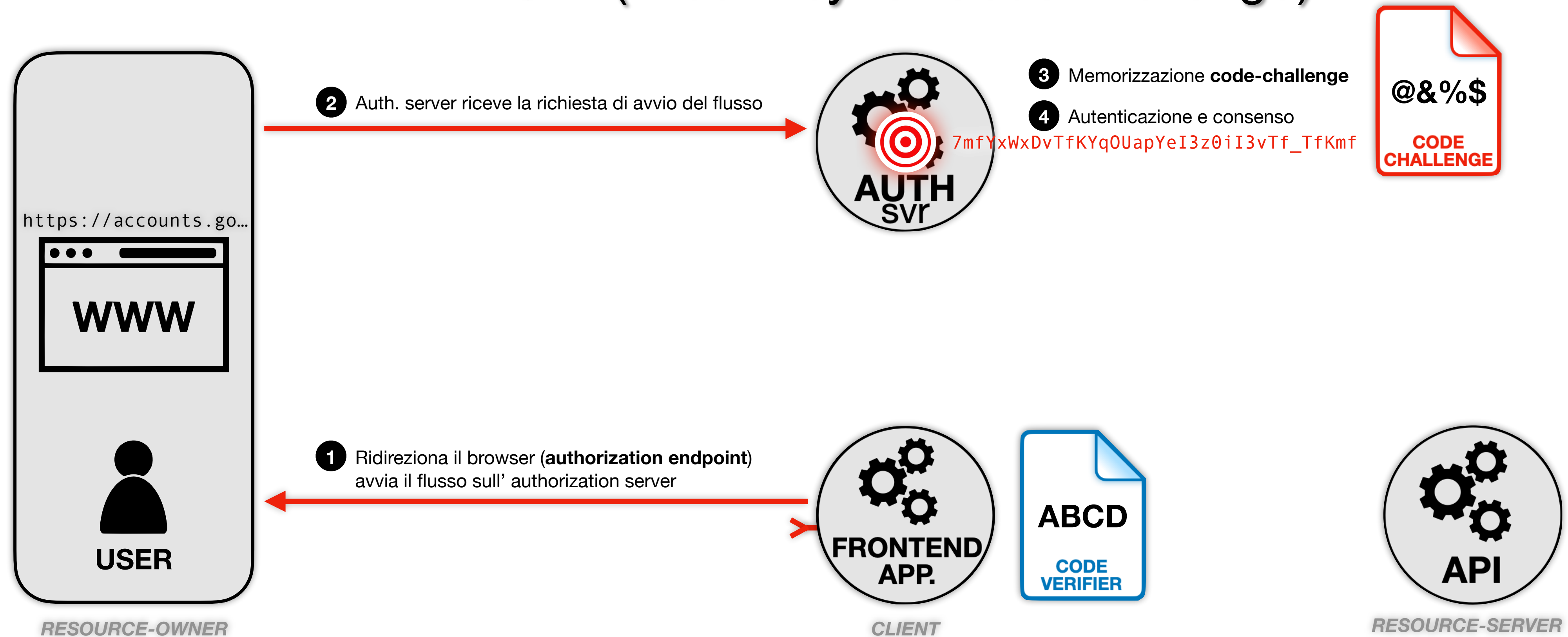
```
https://accounts.google.com/o/oauth2/v2/auth
?response_type=code
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
&redirect_uri=https://levante.com/callback
&scope=profile contacts
&code_challenge=E9Melhoa20wvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
&code_challenge_method=S256
&state=ABCD...&nonce=EFGH...
```

PKCE (Proof Key for Code Exchange)



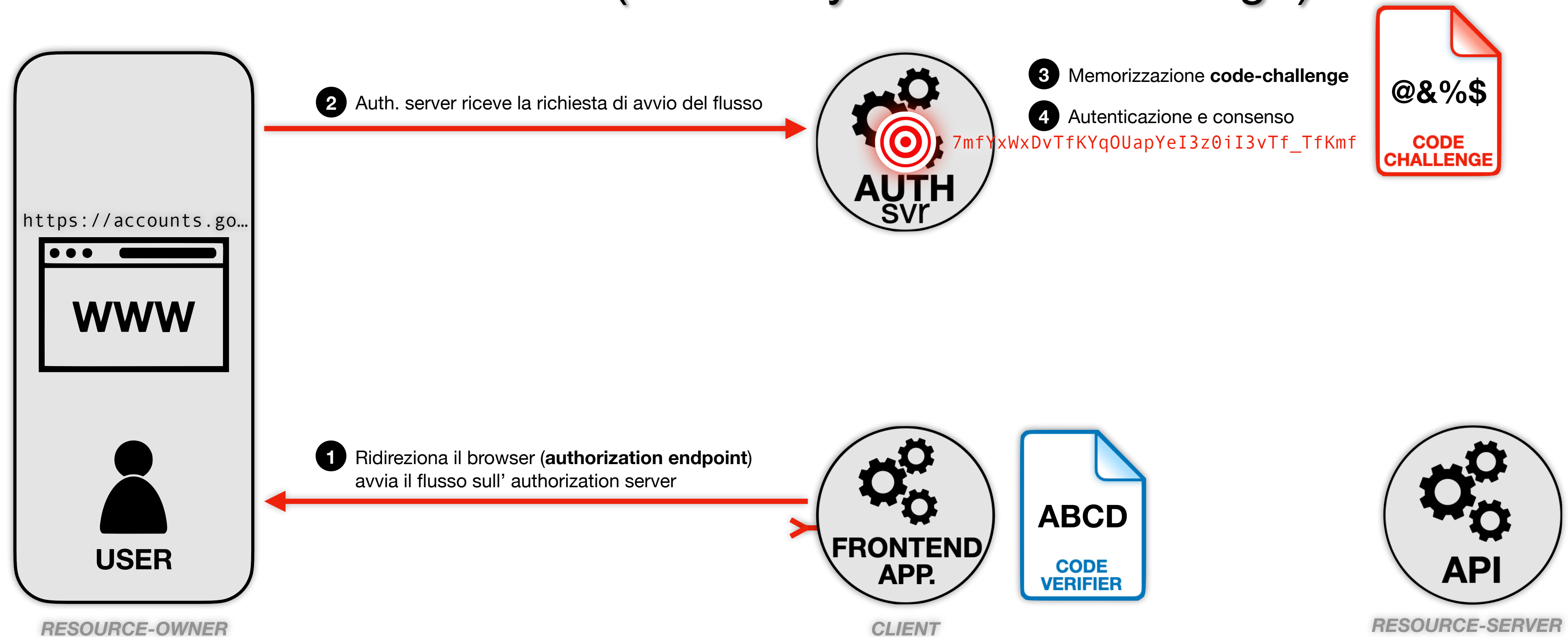
```
https://accounts.google.com/o/oauth2/v2/auth
?response_type=code
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
&redirect_uri=https://levante.com/callback
&scope=profile contacts
&code_challenge=E9Melhoa20wvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
&code_challenge_method=S256
&state=ABCD...&nonce=EFGH...
```

PKCE (Proof Key for Code Exchange)



```
https://accounts.google.com/o/oauth2/v2/auth
?response_type=code
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
&redirect_uri=https://levante.com/callback
&scope=profile contacts
&code_challenge=E9Melhoa20wvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
&code_challenge_method=S256
&state=ABCD...&nonce=EFGH...
```

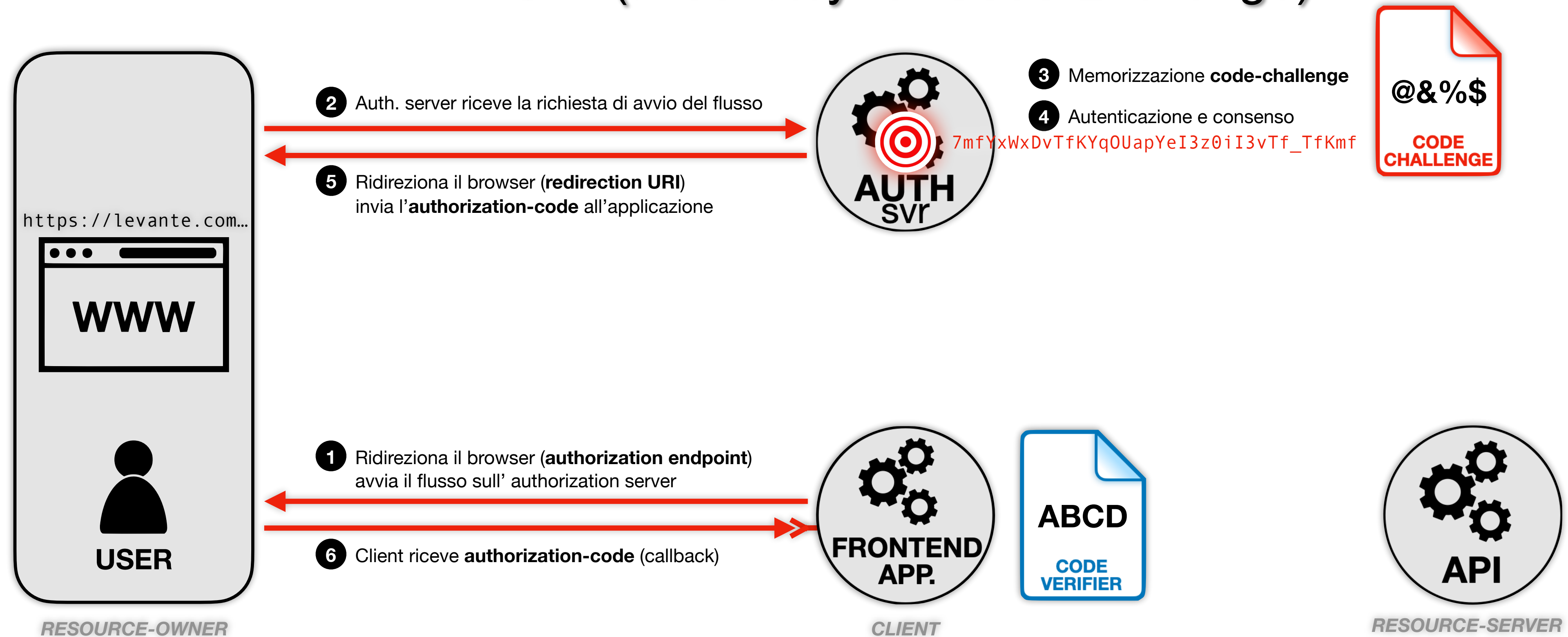

PKCE (Proof Key for Code Exchange)



`https://levante.com/callback`
`?code=7mfYxWxDvTfKYq0UapYeI3z0iI3vTf_TfKmf`
`state=ABCD...`

B

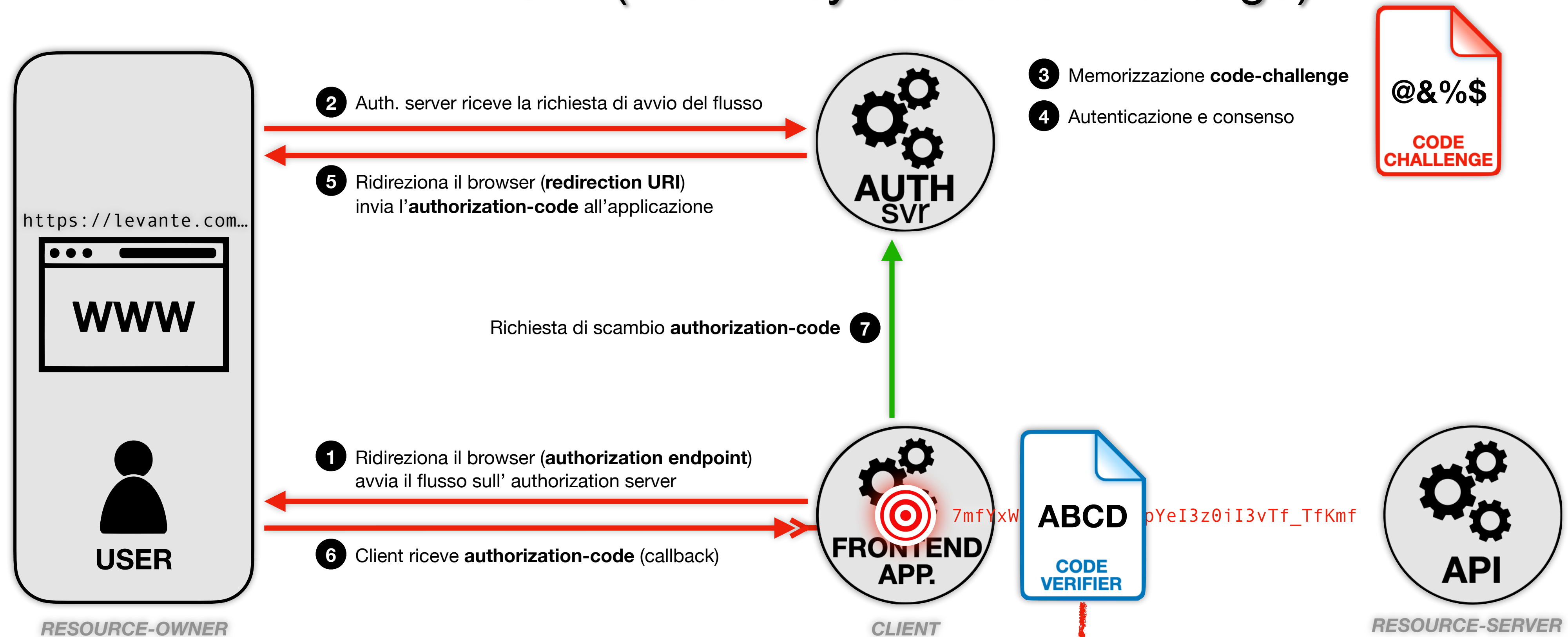
PKCE (Proof Key for Code Exchange)



`https://levante.com/callback`
`?code=7mfYxWxDvTfKYq0UapYeI3z0iI3vTf_TfKmf`
`state=ABCD...`

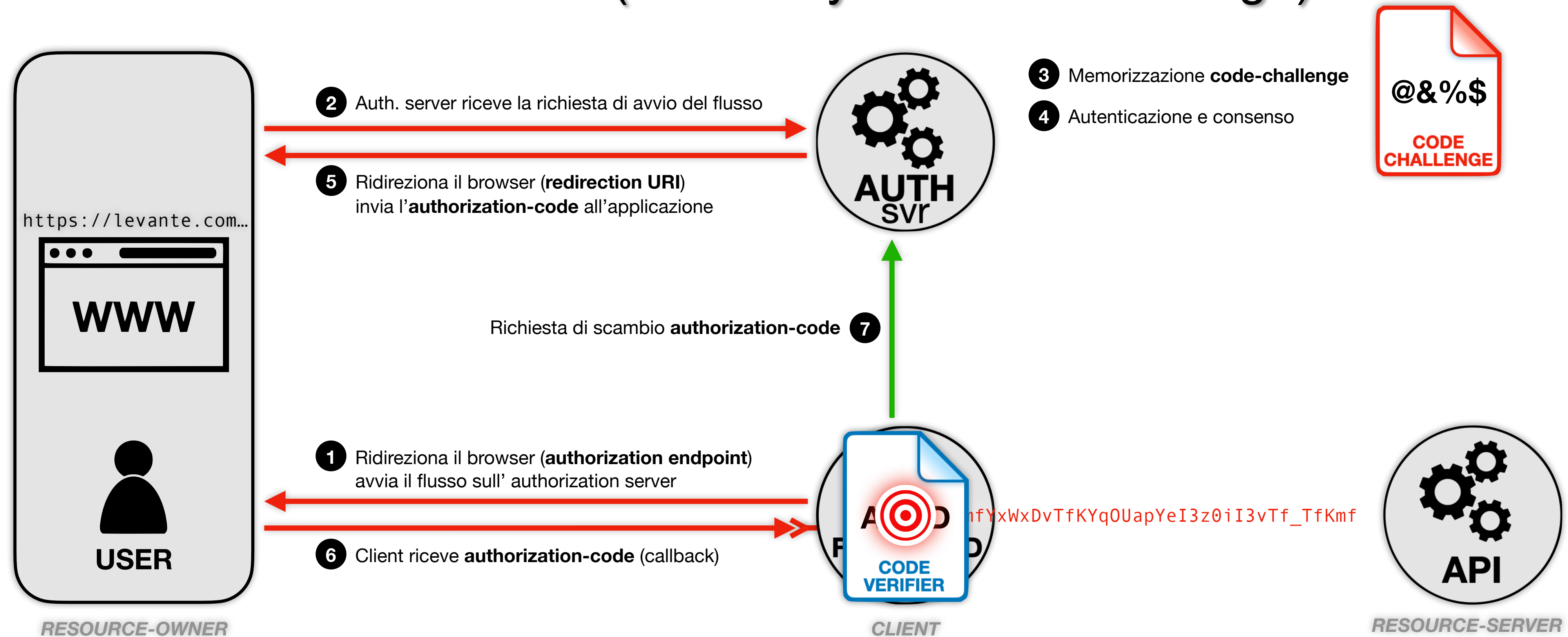
B

PKCE (Proof Key for Code Exchange)



```
POST www.googleapis.com/oauth2/v4/token
Content-Type: application/x-www-form-urlencoded
grant_type=authorization_code
&code=7mfYxWxDvTfKYq0UapYeI3z0iI3vTf_TfKmfB
&redirect_uri=https://levante.com/callback
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
&code_verifier=ABCD...
```

PKCE (Proof Key for Code Exchange)



```
POST www.googleapis.com/oauth2/v4/token
```

```
Content-Type: application/x-www-form-urlencoded
```

```
grant_type=authorization_code
```

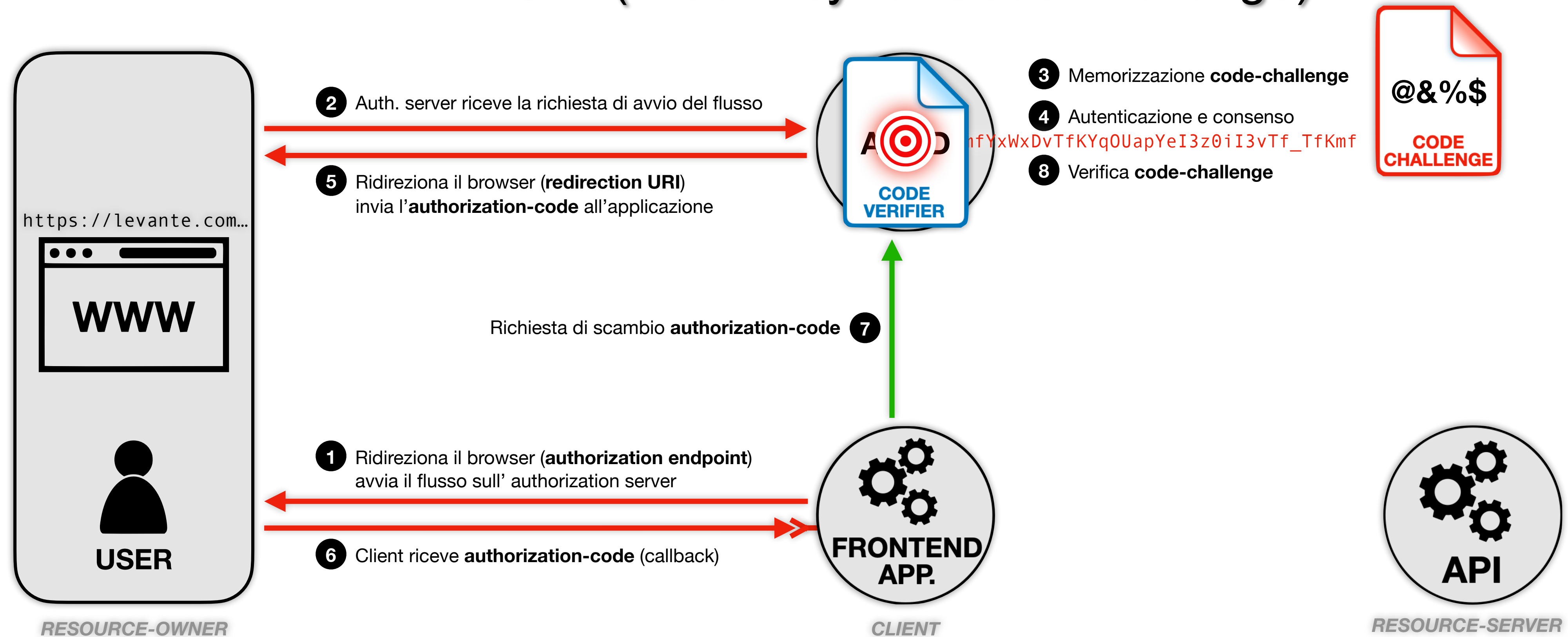
```
&code=7mfYxWxDvTfKYq0UapYeI3z0iI3vTf_TfKmfB
```

```
&redirect_uri=https://levante.com/callback
```

```
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
```

```
&code_verifier=ABCD...
```


PKCE (Proof Key for Code Exchange)



```
POST www.googleapis.com/oauth2/v4/token
```

```
Content-Type: application/x-www-form-urlencoded
```

```
grant_type=authorization_code
```

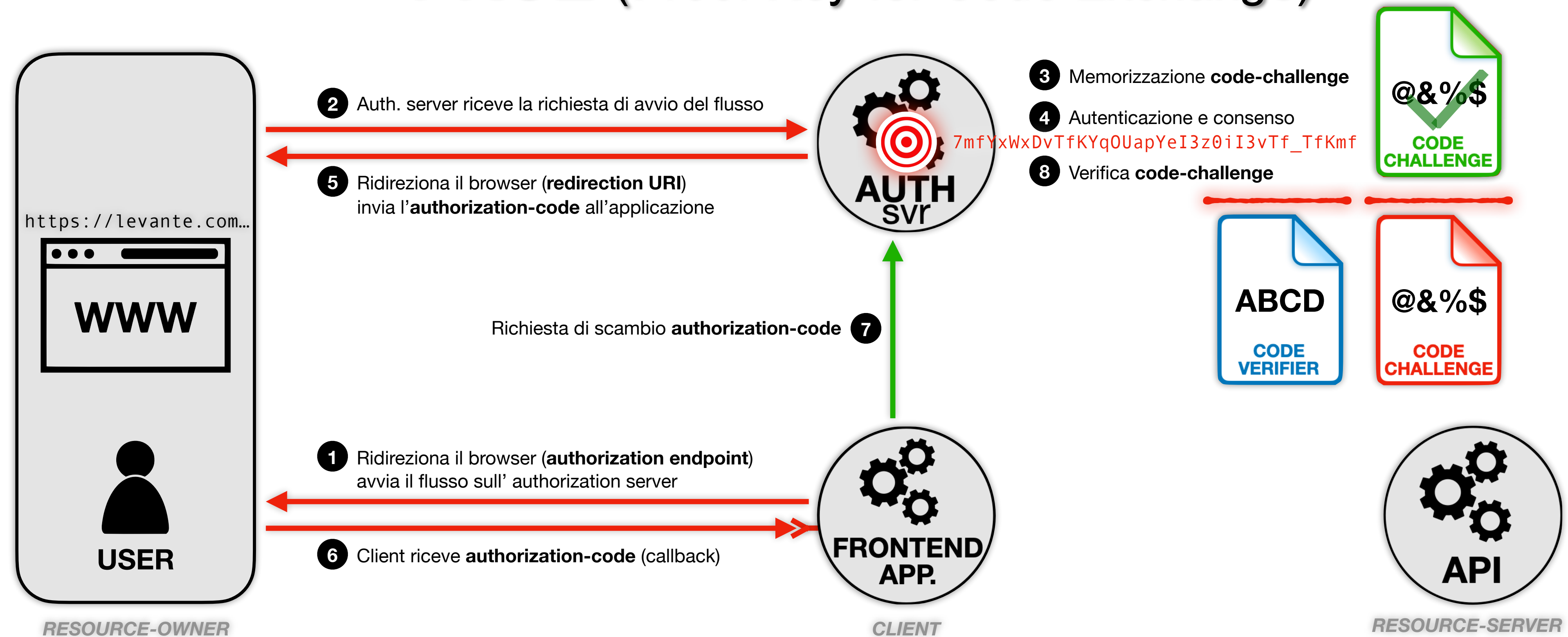
```
&code=7mfYxWxDvTfKYq0UapYeI3z0iI3vTf_TfKmfB
```

```
&redirect_uri=https://levante.com/callback
```

```
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
```

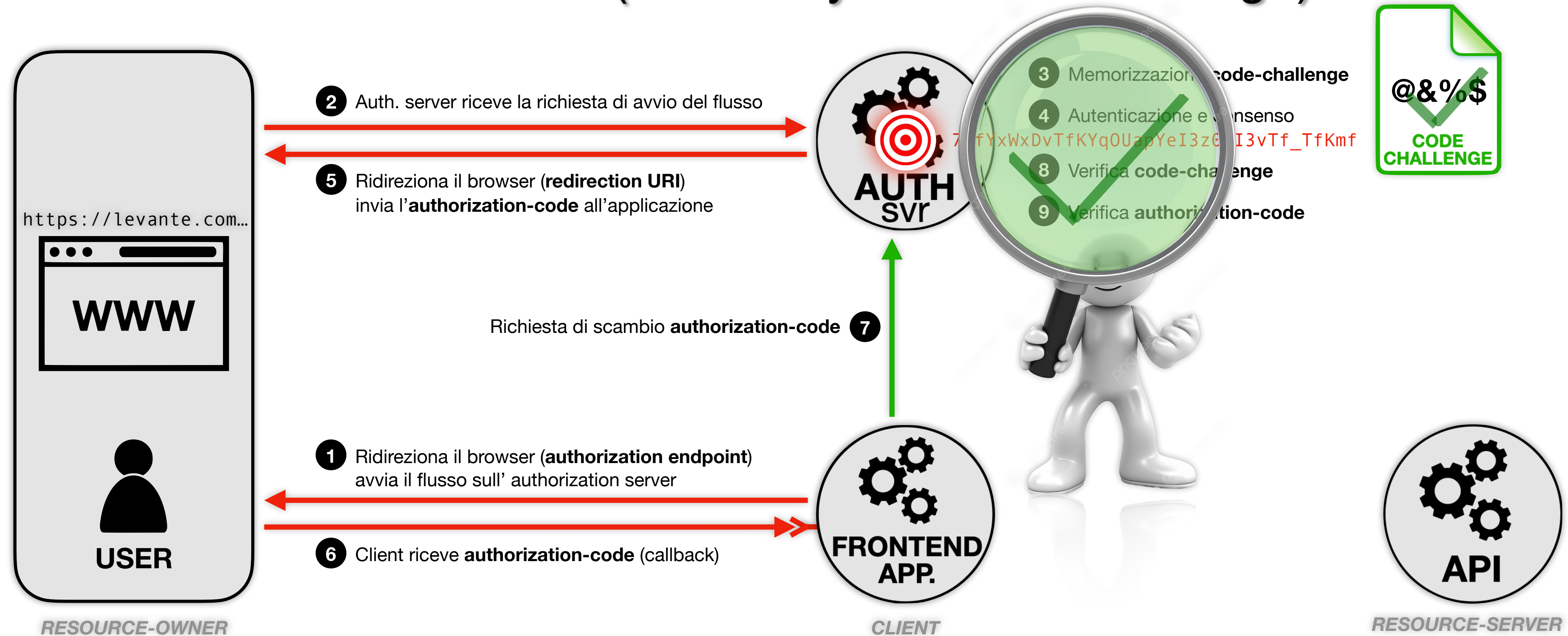
```
&code_verifier=ABCD...
```


PKCE (Proof Key for Code Exchange)



```
POST www.googleapis.com/oauth2/v4/token
Content-Type: application/x-www-form-urlencoded
grant_type=authorization_code
&code=7mfYxWxDvTfKYq0UapYeI3z0iI3vTf_TfKmfB
&redirect_uri=https://levante.com/callback
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
&code_verifier=ABCD...
```

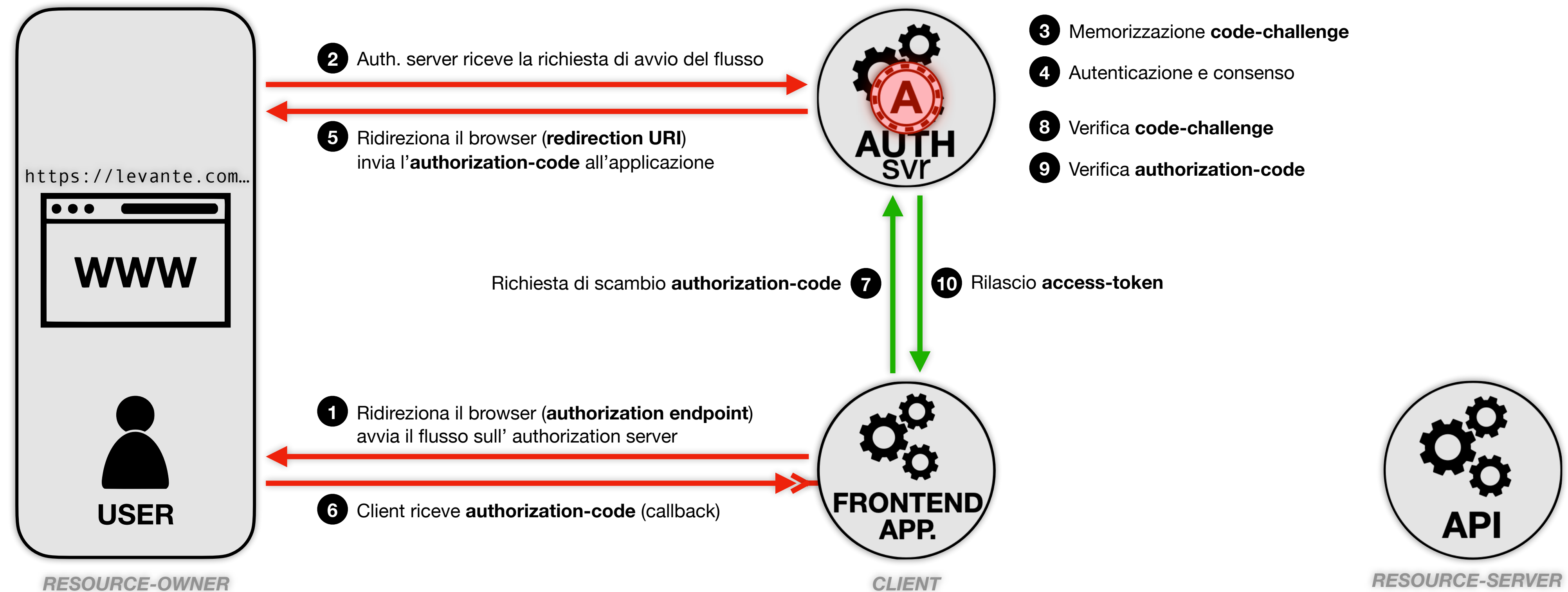
PKCE (Proof Key for Code Exchange)



```
POST www.googleapis.com/oauth2/v4/token
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code
&code=7mfYxWxDvTfKYq0UapYeI3z0iI3vTf_TfKmfB
&redirect_uri=https://levante.com/callback
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
&code_verifier=ABCD...
```

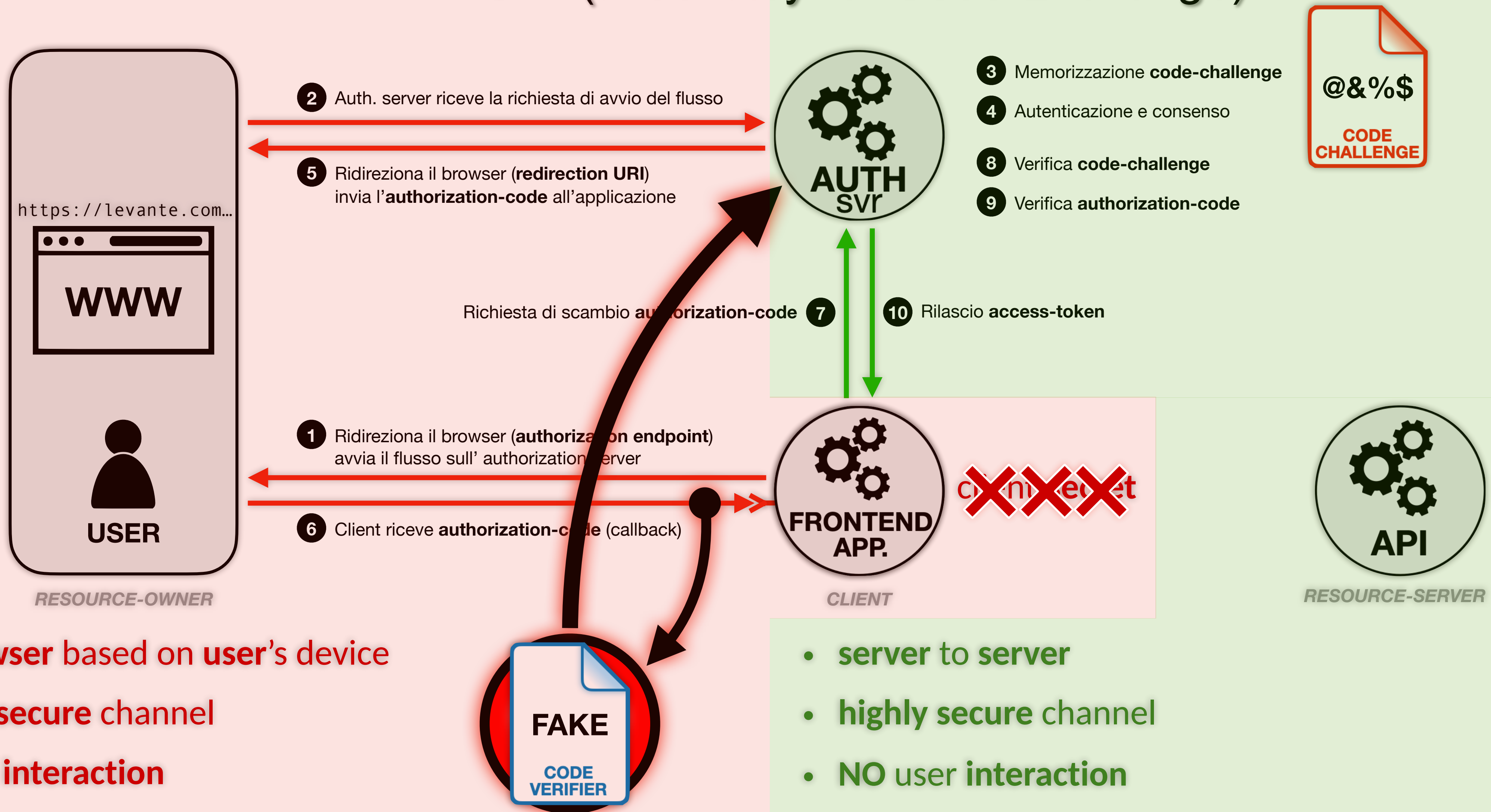

PKCE (Proof Key for Code Exchange)



```
{  
  "access_token": "eyJhbGciOi0...du6TY9w",  
  "expires_in": 3600,  
  "token_type": "Bearer"  
}
```

B

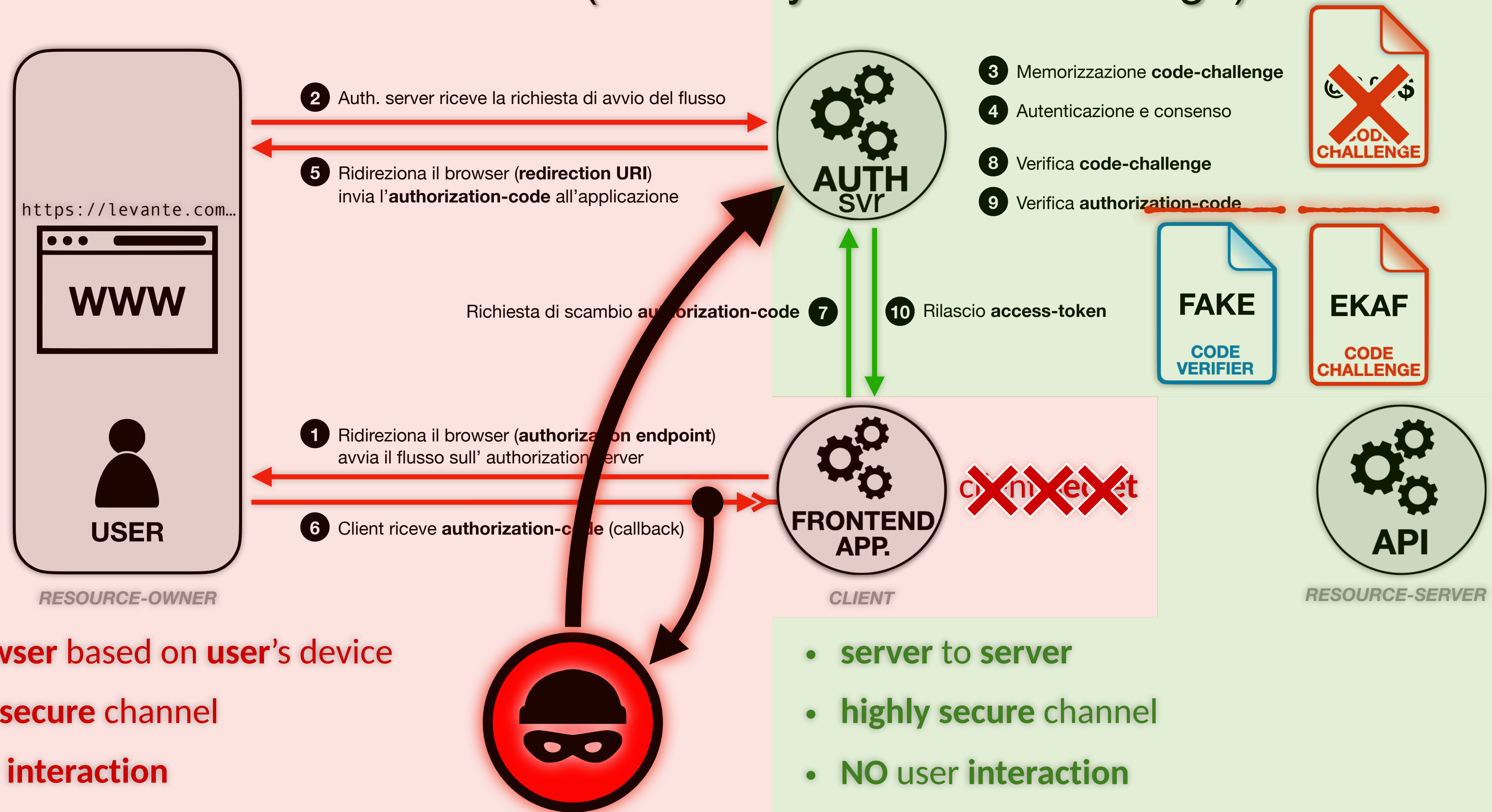
PKCE (Proof Key for Code Exchange)



- **browser** based on **user's** device
- **less secure** channel
- **user interaction**
- **authorization-code???**

- **server to server**
- **highly secure** channel
- **NO user interaction**

PKCE (Proof Key for Code Exchange)



- browser based on user's device
- less secure channel
- user interaction
- authorization-code???

- server to server
- highly secure channel
- NO user interaction

AccessToken



AccessToken



- **Autorizzazione temporanea** ad accedere a risorse protette **in nome e per conto dell'utente**
- **Allegato** ad ogni richiesta al resource-server
- **Nessuna specifica da OAuth**
- **NO info sull'utente (solo ID)**



OpenID Connect (OIDC)

OpenID Connect (OIDC)

- Piccola estensione di OAuth 2.0
- **OAuth** dice a una app/servizio **cosa può fare** (autorizzazione)
- **OIDC** dice anche **chi è l'utente autenticato** (info utente, autenticazione)

OpenID Connect (OIDC)

aggiunge 5 cose

1) ID-Token:

- veicola **informazioni** sull'**utente** (id, nome, email...)
- deve essere un **JWT**
- specifica quali **claims** devono/possono esserci e il loro nome ("sub" è obbligatorio)

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

{
  "sub": "1234567890",
  "name": "Mario Rossi",
  "email": "mario.rossi@example.com",
  "email_verified": true,
  "nonce": "abc123",
  "iat": 1682000000,
  "exp": 1682003600,
  "iss": "https://auth.example.com",
  "aud": "my-client-id",
  "auth_time": 1682000000
}
```

OpenID Connect (OIDC)

aggiunge 5 cose

1) ID-Token:

- veicola **informazioni** sull'**utente** (id, nome, email...)
- deve essere un **JWT**
- specifica quali **claims** devono/possono esserci e il loro nome ("sub" è obbligatorio)

2) standard set of scopes:

- "openid" (obbligatorio, senza è solo OAuth 2.0)
- "profile" (fornisce **info** base del **profilo** utente: nome, cognome, immagine...)
- "offline_access" (permette di ottenere un **refresh token**, per accesso persistente)
- "email", "address", "phone"

3) userinfo_endpoint: fornisce informazioni aggiuntive sull'utente autenticato

4) "prompt" parameter:

- si può inserire nella chiamata iniziale (step 1) (sempre query-params)
- imposta il livello di interazione con l'utente
- "none", "login", "consent", "select_account"

OpenIDConnect (OIDC)

aggiunge 5 cose

5) **well-known**: fornisce in modo **standard** tutte le info per interagire con un server OIDC

- URL per “authorization_endpoint”
- URL per “token_endpoint”
- URL per “userinfo_endpoint”
- “scopes_supported”, “response_types_supported”, “grant_types_supported”...

...esempio reale per google:

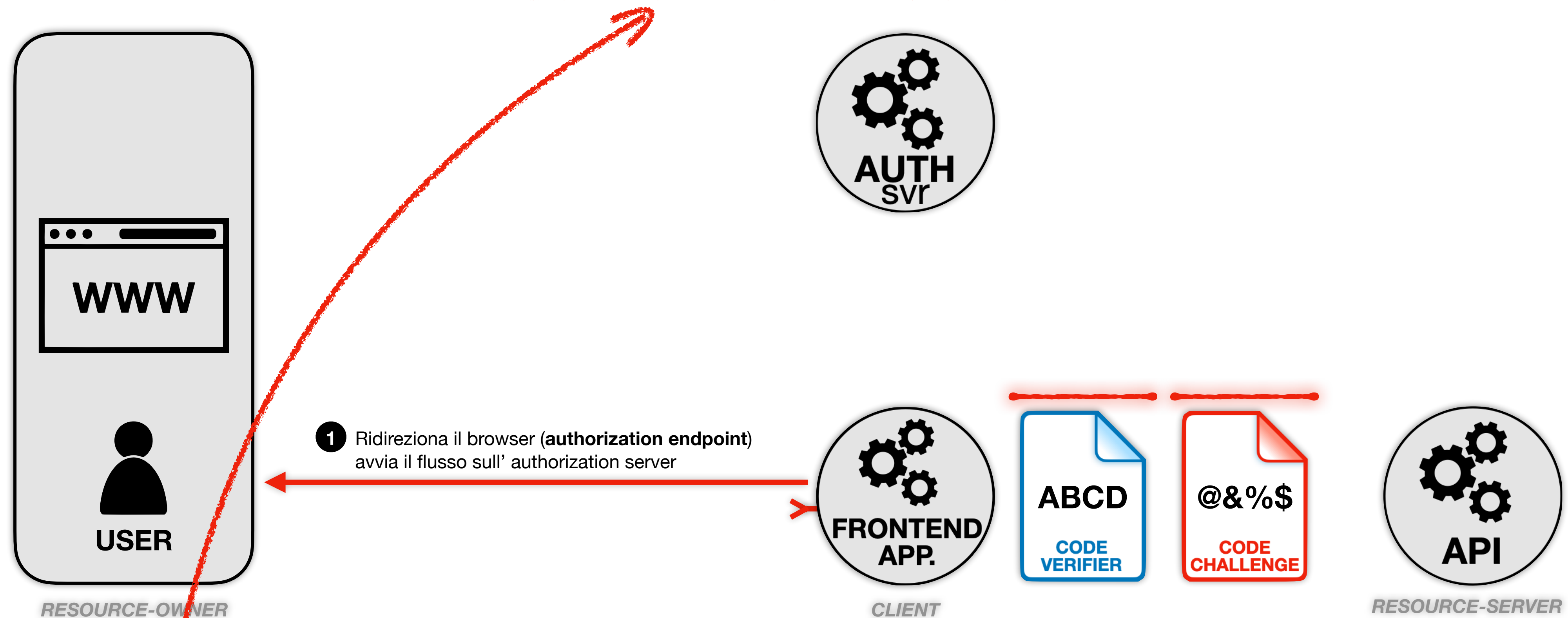
HTTPS://ACCOUNTS.GOOGLE.COM/.WELL-KNOWN/OPENID-CONFIGURATION

```
{
  "issuer": "https://accounts.google.com",
  "authorization_endpoint": "https://accounts.google.com/o/oauth2/v2/auth",
  "token_endpoint": "https://oauth2.googleapis.com/token",
  "userinfo_endpoint": "https://openidconnect.googleapis.com/v1/userinfo",
  "jwks_uri": "https://www.googleapis.com/oauth2/v3/certs",
  ...
}
```


OpenID Connect (OIDC)

```
{
  "issuer": "https://auth.example.com",
  "authorization_endpoint": "https://auth.example.com/oauth2/authorize",
  "token_endpoint": "https://auth.example.com/oauth2/token",
  "userinfo_endpoint": "https://auth.example.com/oauth2/userinfo",
  "jwks_uri": "https://auth.example.com/oauth2/jwks",
  "registration_endpoint": "https://auth.example.com/oauth2/register",
  "scopes_supported": ["openid", "profile", "email", "address", "phone", "offline_access"],
  "response_types_supported": ["code", "id_token", "token id_token", "code id_token"],
  "grant_types_supported": ["authorization_code", "implicit", "refresh_token", "client_credentials"],
  "subject_types_supported": ["public", "pairwise"],
  "id_token_signing_alg_values_supported": ["RS256", "ES256"],
  "id_token_encryption_alg_values_supported": ["RSA-OAEP", "A256KW"],
  "id_token_encryption_enc_values_supported": ["A128CBC-HS256", "A256GCM"],
  "token_endpoint_auth_methods_supported": [
    "client_secret_basic", "client_secret_post", "private_key_jwt"
  ],
  "token_endpoint_auth_signing_alg_values_supported": ["RS256", "HS256"],
  "claims_supported": [
    "sub", "iss", "aud", "exp", "iat", "auth_time",
    "name", "family_name", "given_name", "middle_name", "nickname",
    "email", "email_verified", "picture", "locale", "zoneinfo",
    "address", "phone_number"
  ],
  "code_challenge_methods_supported": ["S256", "plain"],
  "claims_parameter_supported": true,
  "request_parameter_supported": true,
  "request_uri_parameter_supported": true,
  "require_request_uri_registration": false
}
```

OpenIDConnect (OIDC)

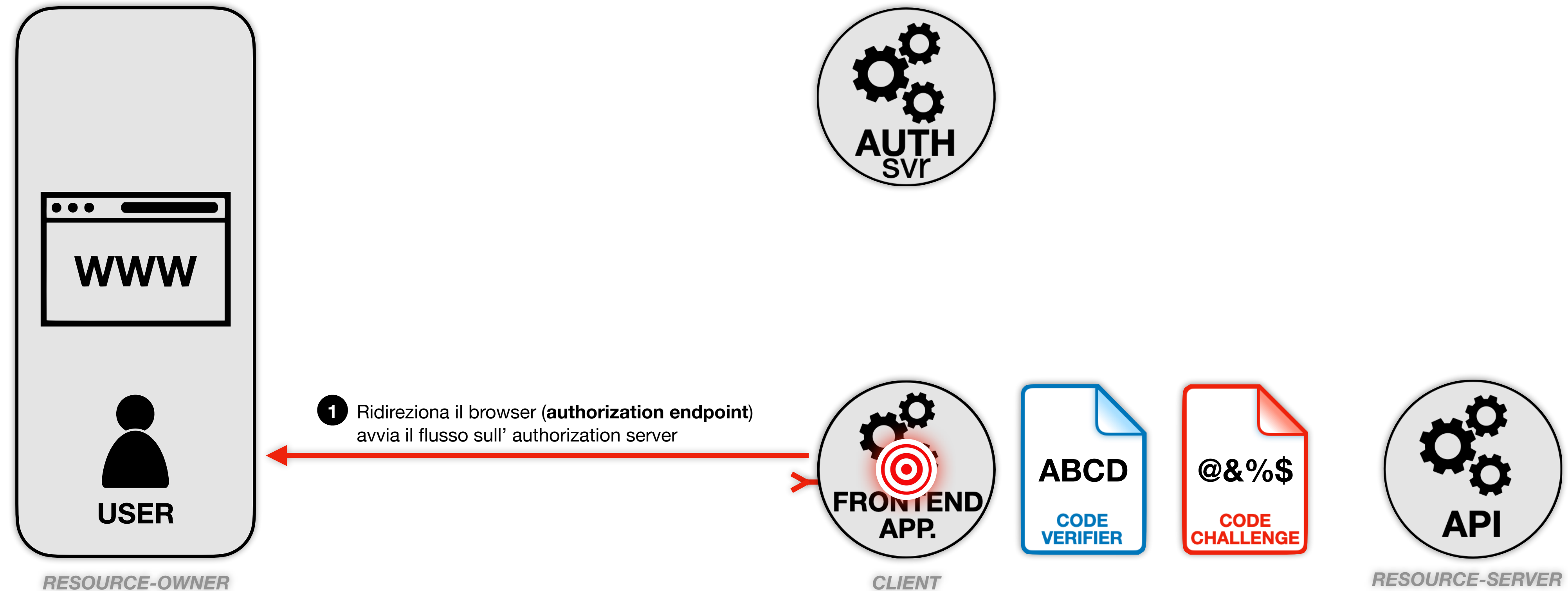


```
https://accounts.google.com/o/oauth2/v2/auth
?response_type=code
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
&redirect_uri=https://levante.com/callback
&scope=openid profile contacts offline_access
&code_challenge=E9Melhoa20wvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
&code_challenge_method=S256
&state=ABCD...&nonce=EFGH...
```



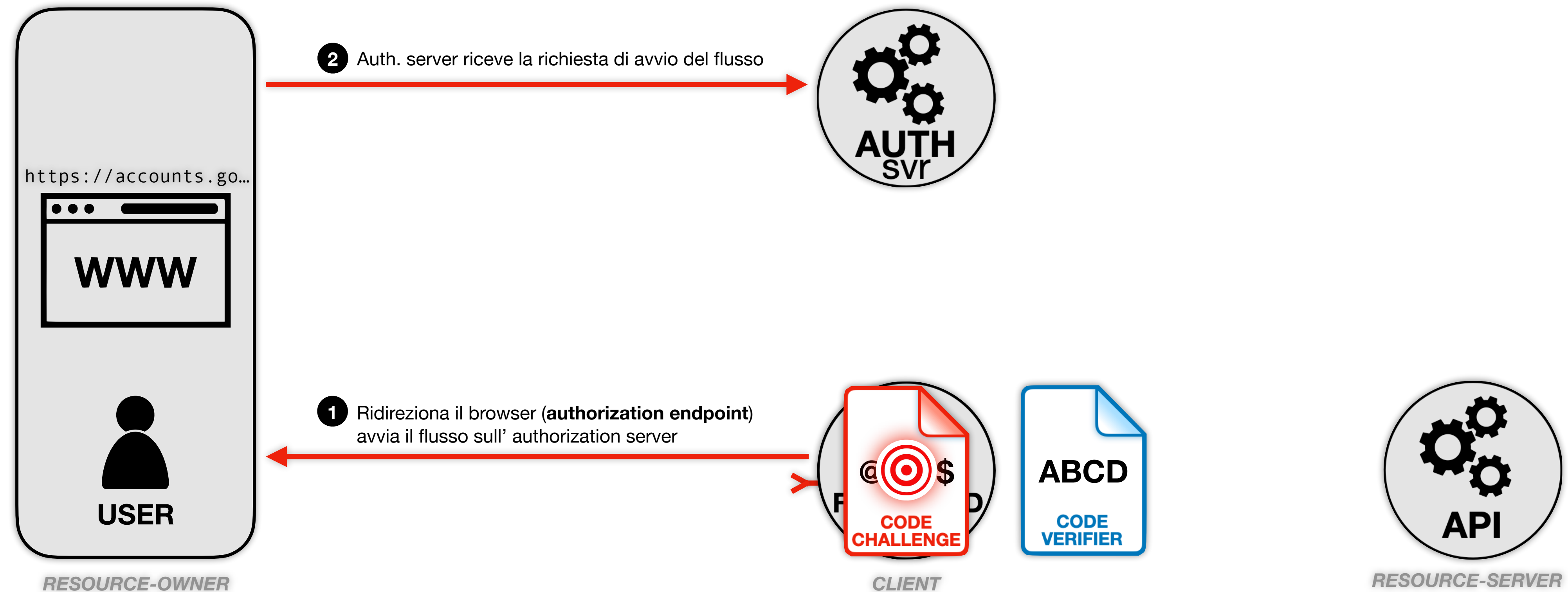
Refresh-token

OpenIDConnect (OIDC)



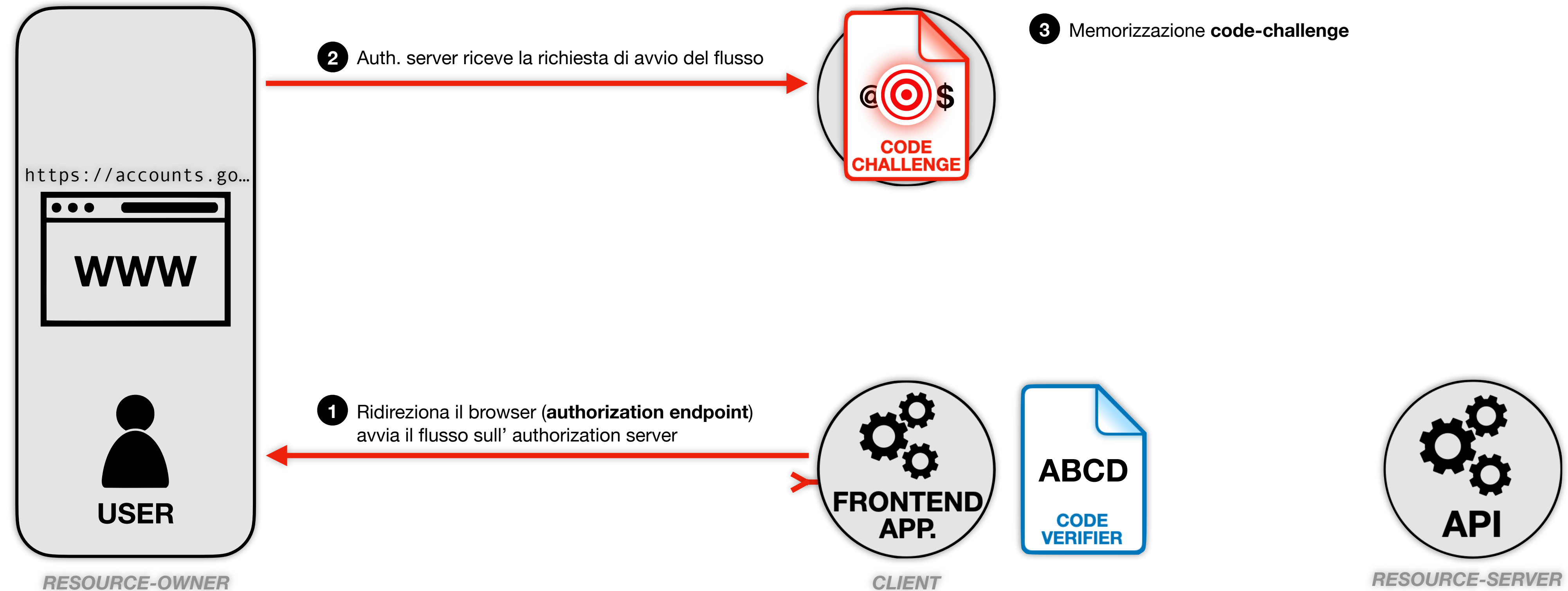
```
https://accounts.google.com/o/oauth2/v2/auth
?response_type=code
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
&redirect_uri=https://levante.com/callback
&scope=openid profile contacts offline_access
&code_challenge=E9Melhoa20wvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
&code_challenge_method=S256
&state=ABCD...&nonce=EFGH...
```


OpenIDConnect (OIDC)



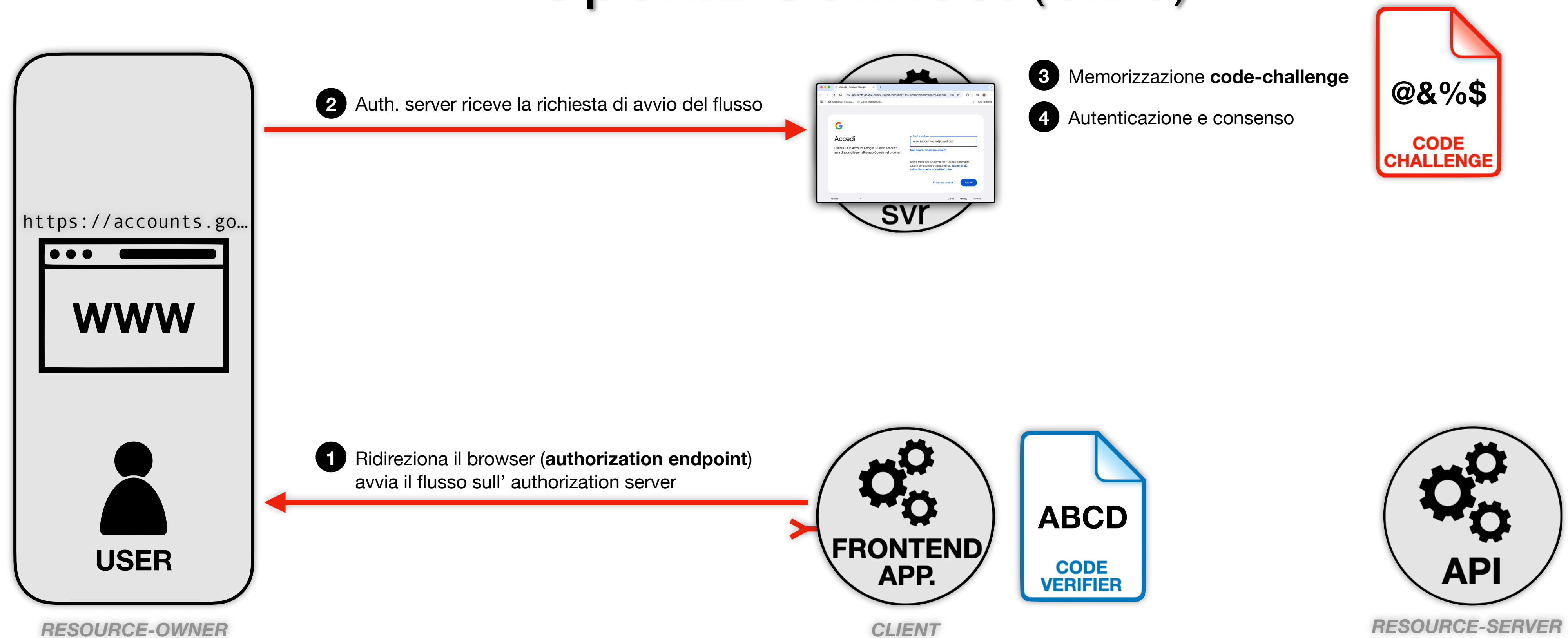
```
https://accounts.google.com/o/oauth2/v2/auth
?response_type=code
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
&redirect_uri=https://levante.com/callback
&scope=openid profile contacts offline_access
&code_challenge=E9Melhoa20wvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
&code_challenge_method=S256
&state=ABCD...&nonce=EFGH...
```


OpenIDConnect (OIDC)



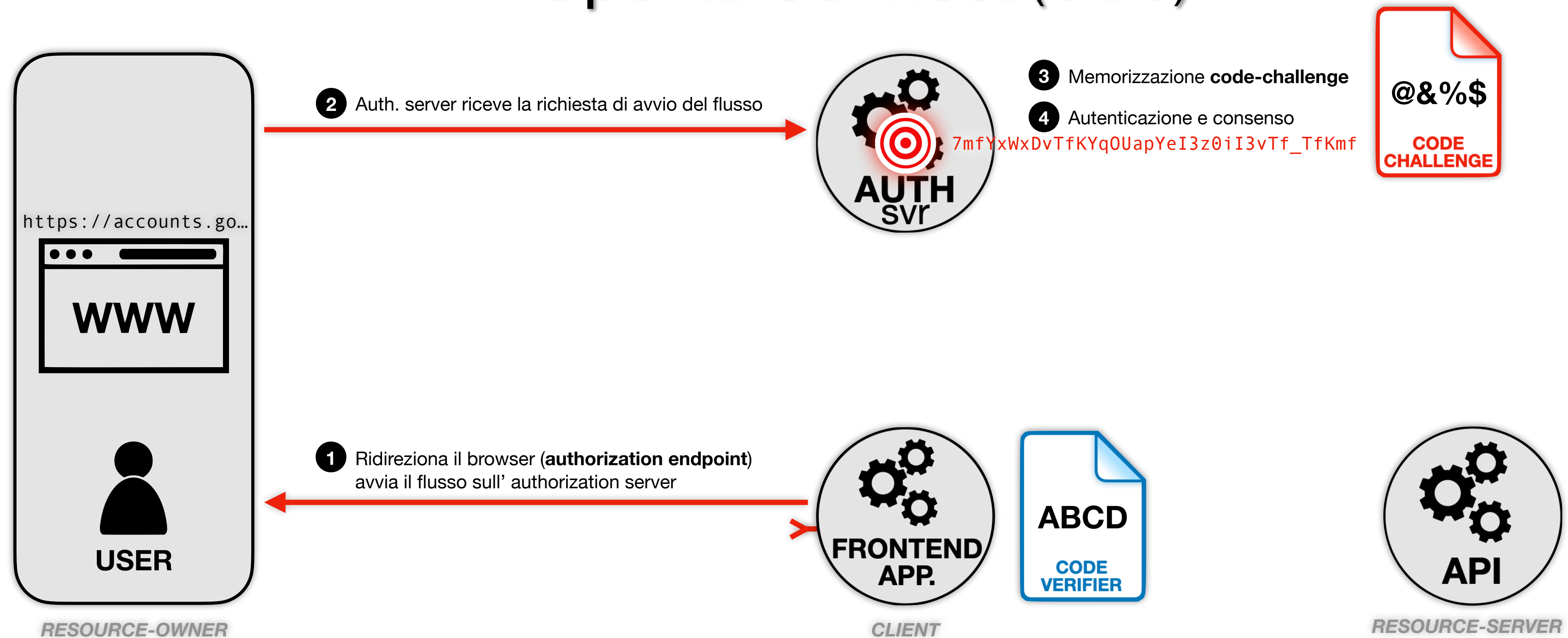
```
https://accounts.google.com/o/oauth2/v2/auth
?response_type=code
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
&redirect_uri=https://levante.com/callback
&scope=openid profile contacts offline_access
&code_challenge=E9Melhoa20wvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
&code_challenge_method=S256
&state=ABCD...&nonce=EFGH...
```

OpenIDConnect (OIDC)



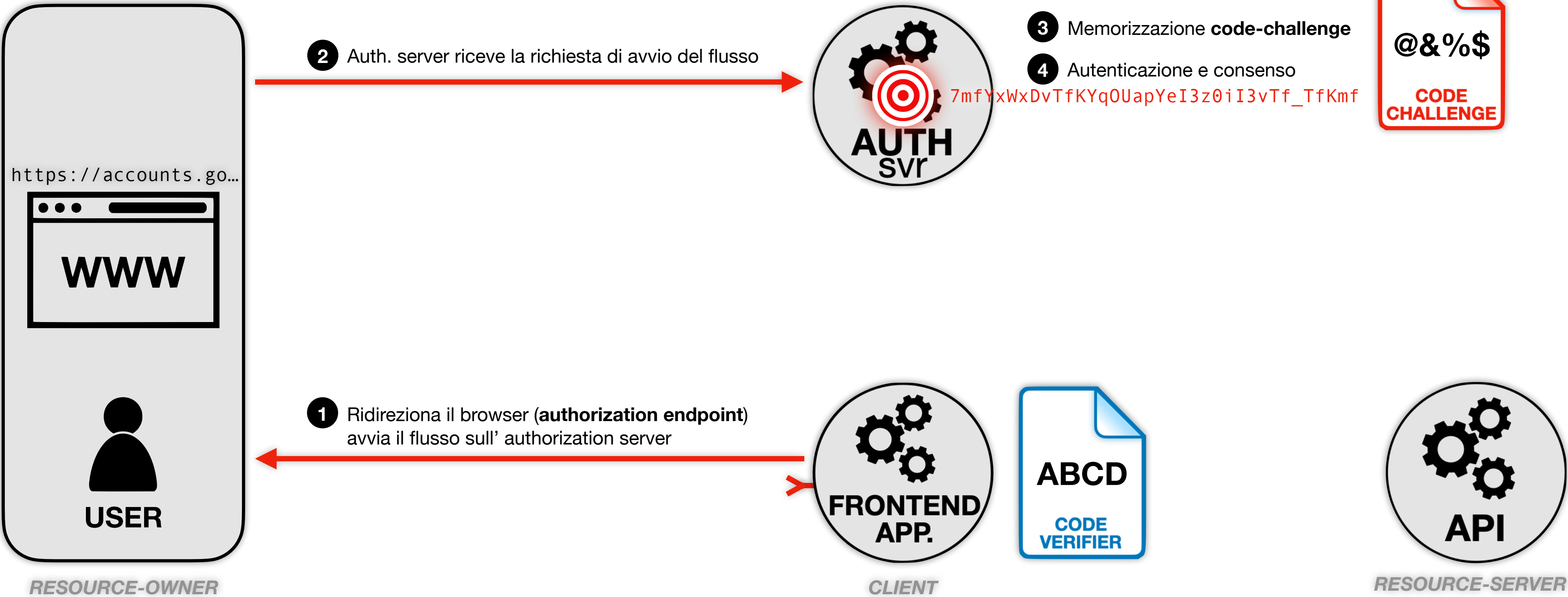
```
https://accounts.google.com/o/oauth2/v2/auth
?response_type=code
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
&redirect_uri=https://levante.com/callback
&scope=openid profile contacts offline_access
&code_challenge=E9Melhoa20wvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
&code_challenge_method=S256
&state=ABCD...&nonce=EFGH...
```

OpenIDConnect (OIDC)



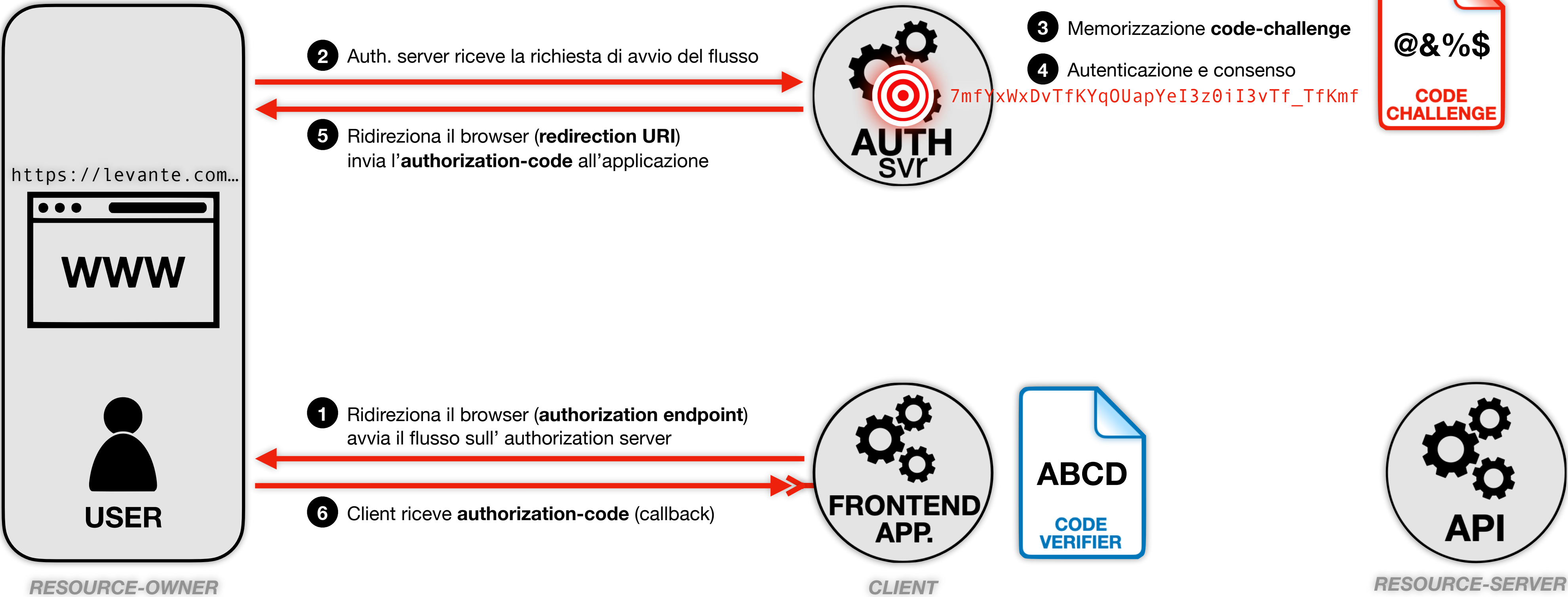
```
https://accounts.google.com/o/oauth2/v2/auth
?response_type=code
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
&redirect_uri=https://levante.com/callback
&scope=openid profile contacts offline_access
&code_challenge=E9Melhoa20wvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
&code_challenge_method=S256
&state=ABCD...&nonce=EFGH...
```


OpenIDConnect (OIDC)



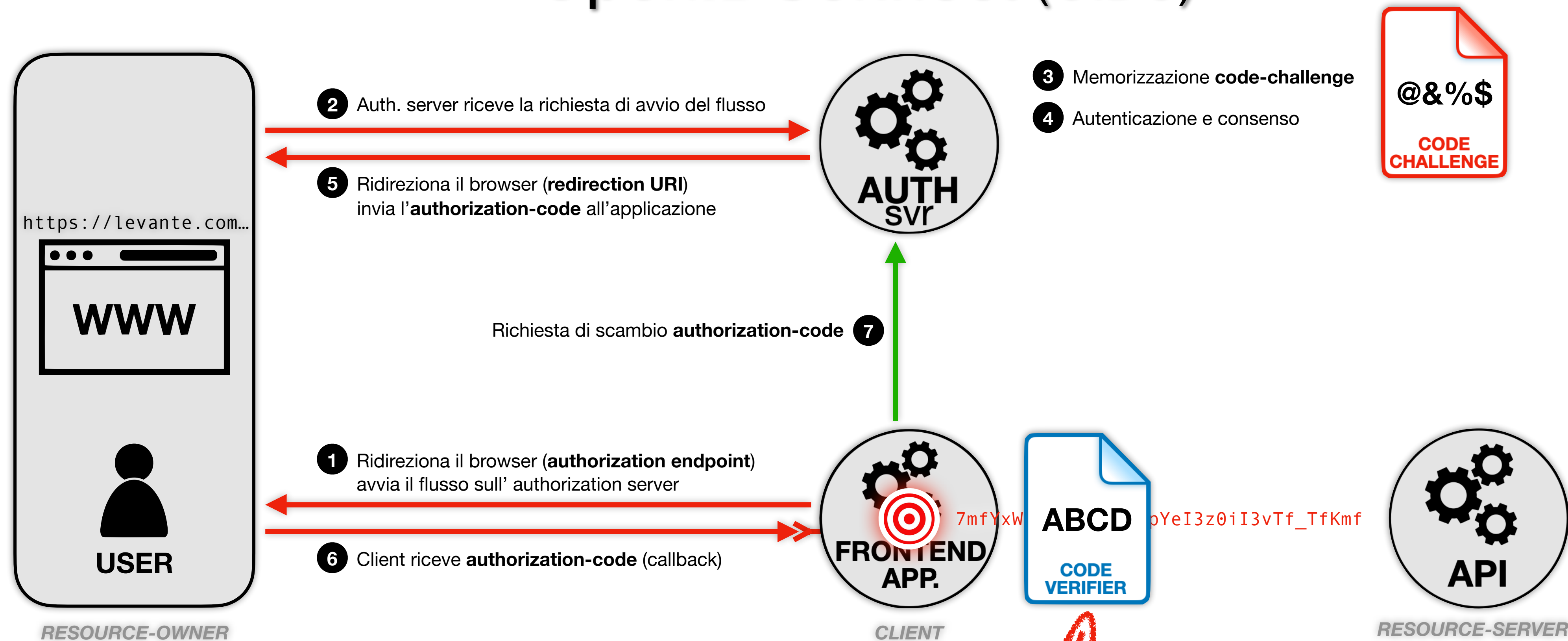
```
https://levante.com/callback
?code=7mfYxWxDvTfKYq0UapYeI3z0iI3vTf_TfKmf
&state=ABCD...
```

OpenIDConnect (OIDC)



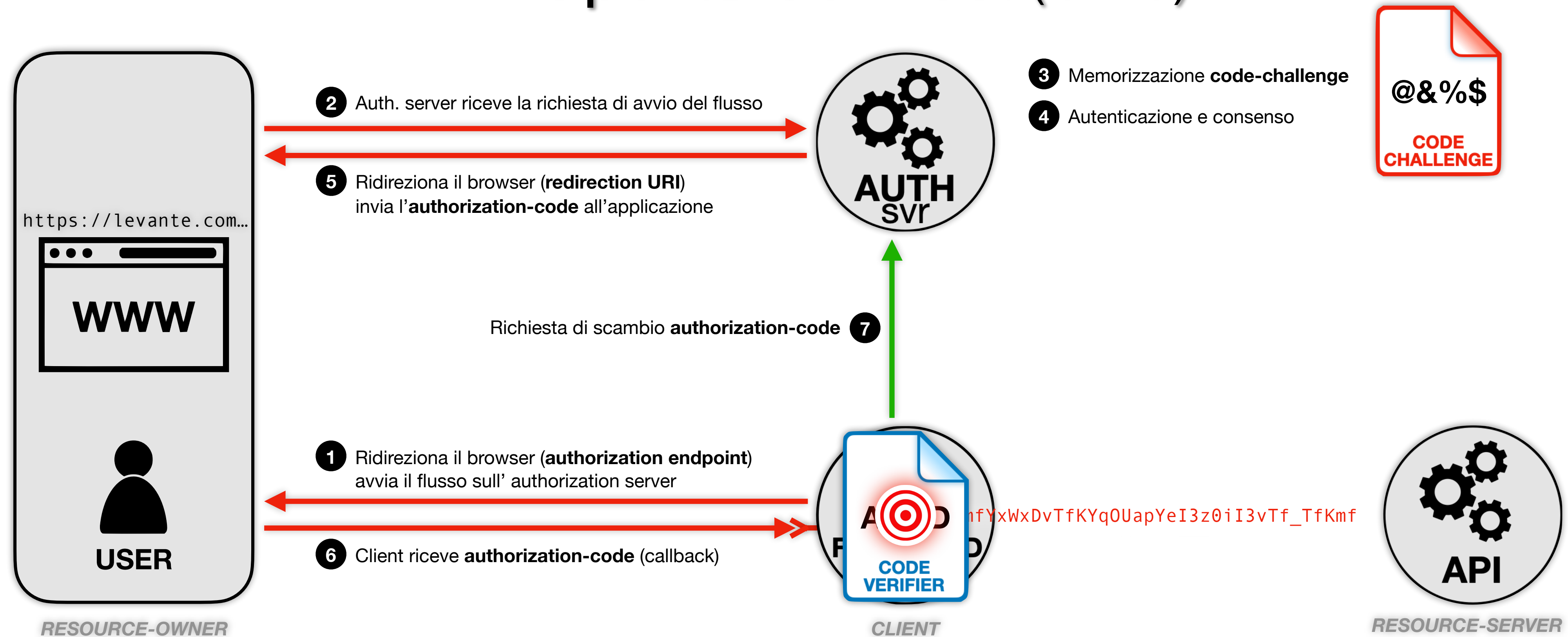
```
https://levante.com/callback
?code=7mfYxWxDvTfKYq0UapYeI3z0iI3vTf_TfKmf
&state=ABCD...
```

OpenIDConnect (OIDC)



```
POST www.googleapis.com/oauth2/v4/token
Content-Type: application/x-www-form-urlencoded
grant_type=authorization_code
&code=7mfYxWxDvTfKYq0UapYeI3z0iI3vTf_TfKmfB
&redirect_uri=https://levante.com/callback
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
&code_verifier=ABCD...
```


OpenIDConnect (OIDC)



POST `www.googleapis.com/oauth2/v4/token`

Content-Type: `application/x-www-form-urlencoded`

`grant_type=authorization_code`

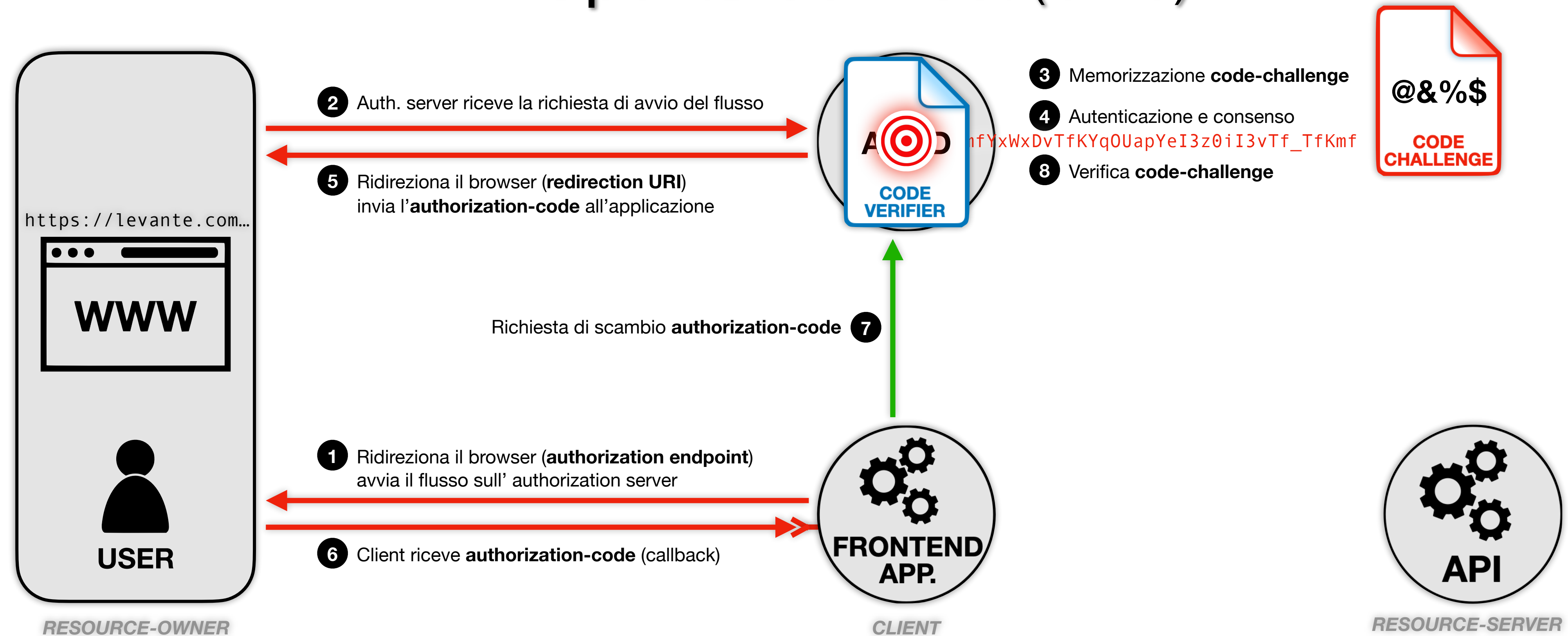
`&code=7mfYxWxDvTfKYq0UapYeI3z0iI3vTf_TfKmfB`

`&redirect_uri=https://levante.com/callback`

`&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42`

`&code_verifier=ABCD...`

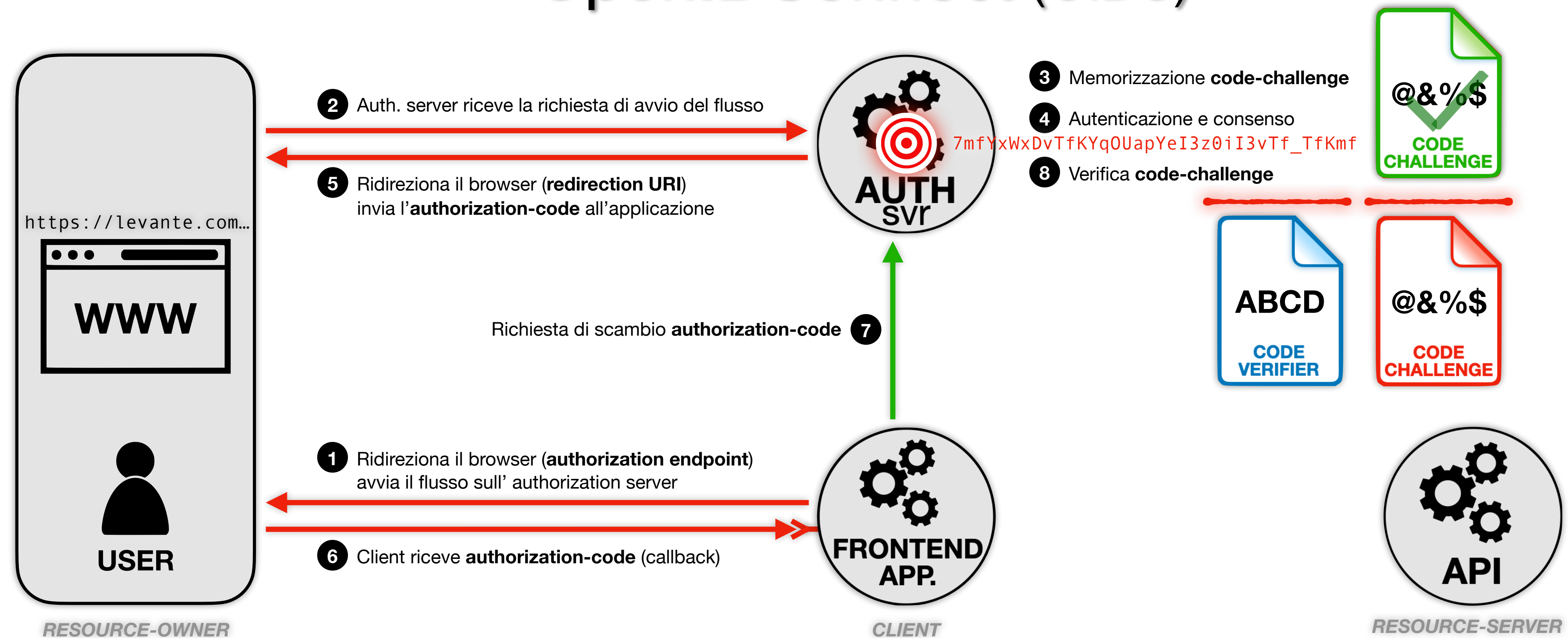
OpenID Connect (OIDC)



```
POST www.googleapis.com/oauth2/v4/token
Content-Type: application/x-www-form-urlencoded

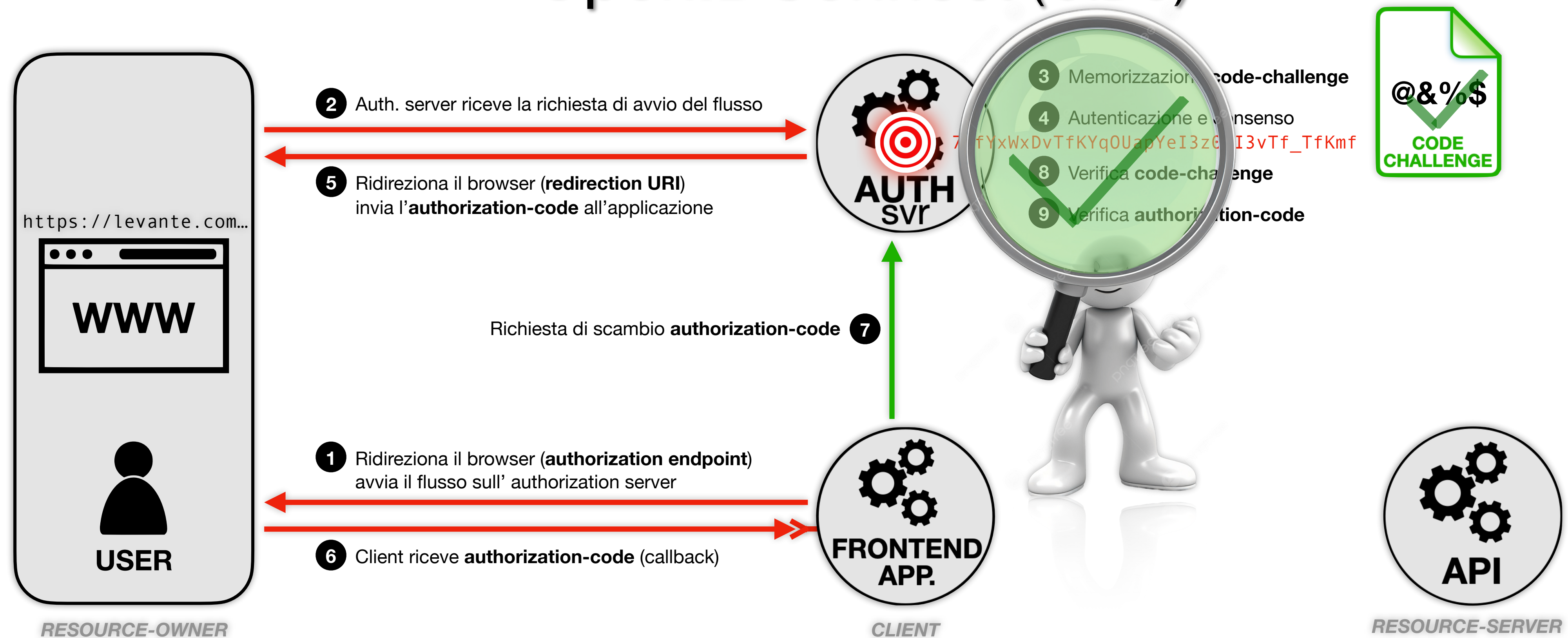
grant_type=authorization_code
&code=7mfYxWxDvTfKYq0UapYeI3z0iI3vTf_TfKmfB
&redirect_uri=https://levante.com/callback
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
&code_verifier=ABCD...
```

OpenIDConnect (OIDC)



```
POST www.googleapis.com/oauth2/v4/token
Content-Type: application/x-www-form-urlencoded
grant_type=authorization_code
&code=7mfYxWxDvTfKYq0UapYeI3z0iI3vTf_TfKmfB
&redirect_uri=https://levante.com/callback
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
&code_verifier=ABCD...
```


OpenIDConnect (OIDC)



```
POST www.googleapis.com/oauth2/v4/token
```

```
Content-Type: application/x-www-form-urlencoded
```

```
grant_type=authorization_code
```

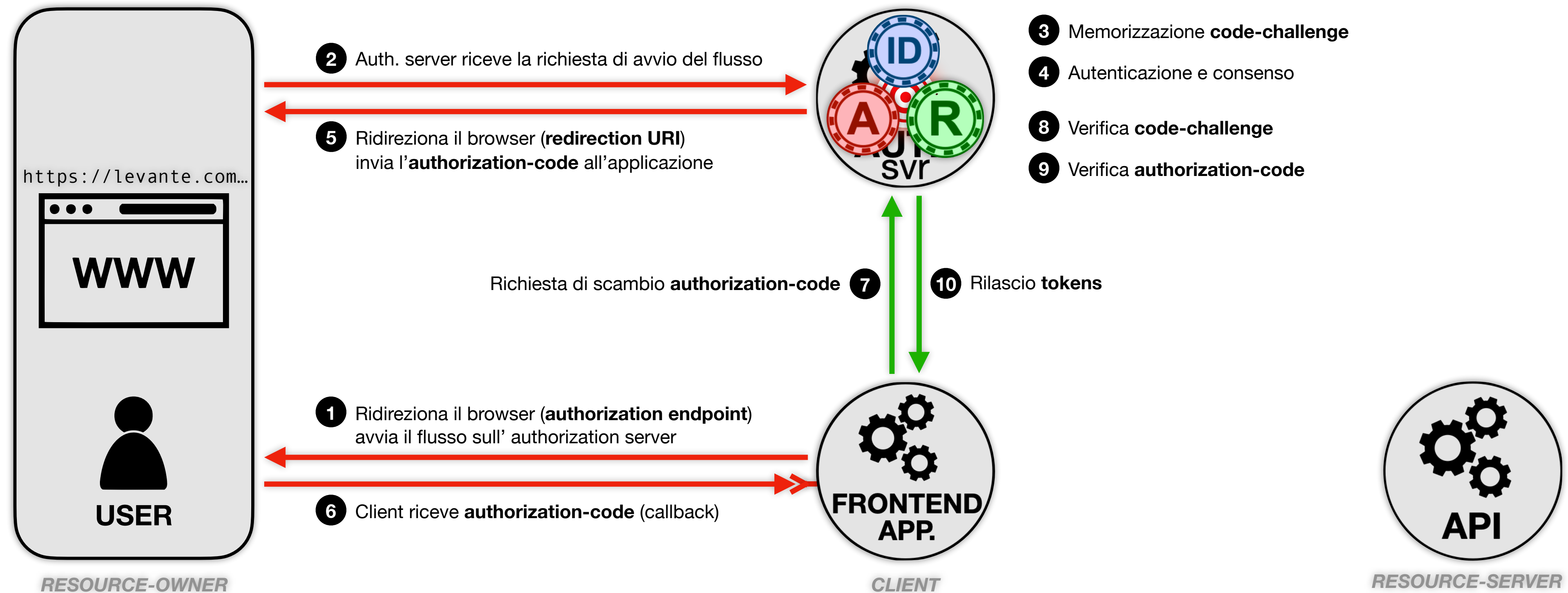
```
&code=7mfYxWxDvTfKYq0UapYeI3z0iI3vTf_TfKmfB
```

```
&redirect_uri=https://levante.com/callback
```

```
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
```

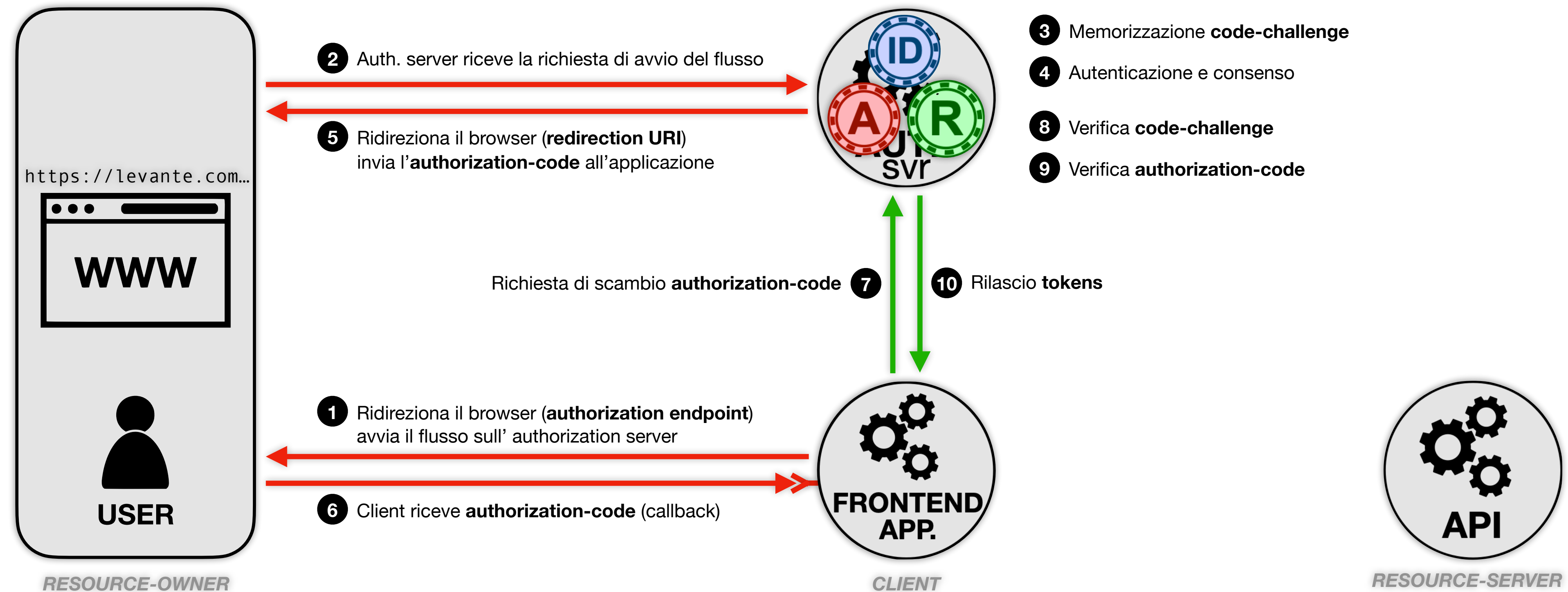
```
&code_verifier=ABCD...
```

OpenIDConnect (OIDC)



```
{  
  "access_token": "eyJhbGciOi0...du6TY9w",  
  "expires_in": 3600,  
  "token_type": "Bearer",  
  "refresh_token": "8xL0xBtZp8JSUzI1NiIs...",  
  "id_token": "eyJhbGciOi0iI6IkpXVCJ9..."  
}
```


OpenIDConnect (OIDC)



```
{  
  "access_token": "eyJhbGciOi0...du6TY9w",  
  "expires_in": 3600,  
  "token_type": "Bearer",  
  "refresh_token": "8xL0xBtZp8JSUzI1NiIs...",  
  "id_token": "eyJhbGciOi0iI6IkpXVCJ9..."  
}
```


Refresh**Token** (long term access)



RefreshToken (long term access)



Permette di ricevere un **nuovo access-token senza doversi ri-autenticare** (senza interazione utente)

Lungo termine:

- dura più a lungo dell'access-token (giorni, mesi, lifetime)
- la durata può dipendere anche dal tipo di utente o dal tipo di client
- **non è rinnovabile** ma quando richiediamo un nuovo access-token riceviamo anche un nuovo refresh-token
- ad alcuni utenti o client potrebbe non essere fornito affatto
- **nessuna specifica** da OAuth

RefreshToken (long term access)



Permette di ricevere un **nuovo access-token senza doversi ri-autenticare** (senza interazione utente)

Sensibile:

- se un malintenzionato riuscisse a rubarlo potrebbe accedere ai nostri dati **per un tempo indefinito**
- va protetto come una password (forse sarebbe meglio mi rubassero le credenziali se ho autenticazione a due fattori)
- ad alcuni utenti o client potrebbe non essere fornito affatto

Revoca:

- un refresh-token **può essere invalidato** prima della sua scadenza naturale
- l'authorization-server **traccia** i refresh-token, può mostrare all'utente quali sono le app. autorizzate (e per quanto tempo)
- anche l'app-client può chiedere la revoca quando l'accesso alle risorse non è più necessario (logout)
- possibile revocare l'accesso alla singola app. (mantenendone attive altre)

RefreshToken (long term access)



Permette di ricevere un **nuovo access-token senza doversi ri-autenticare** (senza interazione utente)

Dove conservarlo (web app):

- cookie + HttpOnly + SecureCookie + SameSite=strict/lax (invisibile a JavaScript, solo tuo dominio)
- solo ID sessione nel cookie, refresh-token sul server (reference-token)
- **cifratura** consigliata
- **MAI** su frontend JavaScript
- sull'**authorization-server**, associato al relativo access-token, refresh automatico quando riceve una richiesta di accesso con access-token scaduto o in scadenza, rotazione automatica

RefreshToken (long term access)



Permette di ricevere un **nuovo access-token senza doversi ri-autenticare** (senza interazione utente)

Dove conservarlo (native mobile app):

- iOS: **Keychain**
- Android: **Android Keystore System**
- ...sono entrambi sistemi di archiviazione **crittografati** per piccole quantità di dati sensibili (password, token, certificati)
- ...crittografia **hardware-based** protetti dal sistema operativo
- ...accesso protetto anche con **sensori biometrici**

RefreshToken (long term access)



Permette di ricevere un **nuovo access-token senza doversi ri-autenticare** (senza interazione utente)

Dove conservarlo (native desktop app):

- un pò dove si vuole, anche in un DB, purché **protetti e criptati**
- Windows: **Windows Credential Manager**, integrato nel OS, dati memorizzati nel profilo utente e crittografati
- MacOS: **Keychain**, integrato nel OS, crittografia hw, può richiedere autenticazione utente (face-ID, touch-ID)
- Linux: **Secret Service API**, servizi di gestione sicura dei segreti, crittografia (token, password, chiavi)

NB: Conservare in storage sicuro solo i token sensibili e persistenti (refresh-token, ID-token)

NB: Gli access-token, possono essere mantenuti anche in RAM e richiederli ad ogni avvio grazie al refresh-token

(tanto in caso di accessi sporadici ne richiederebbe uno nuovo comunque)

RefreshToken (long term access)



Permette di ricevere un **nuovo access-token senza doversi ri-autenticare** (senza interazione utente)

Come si ottiene:

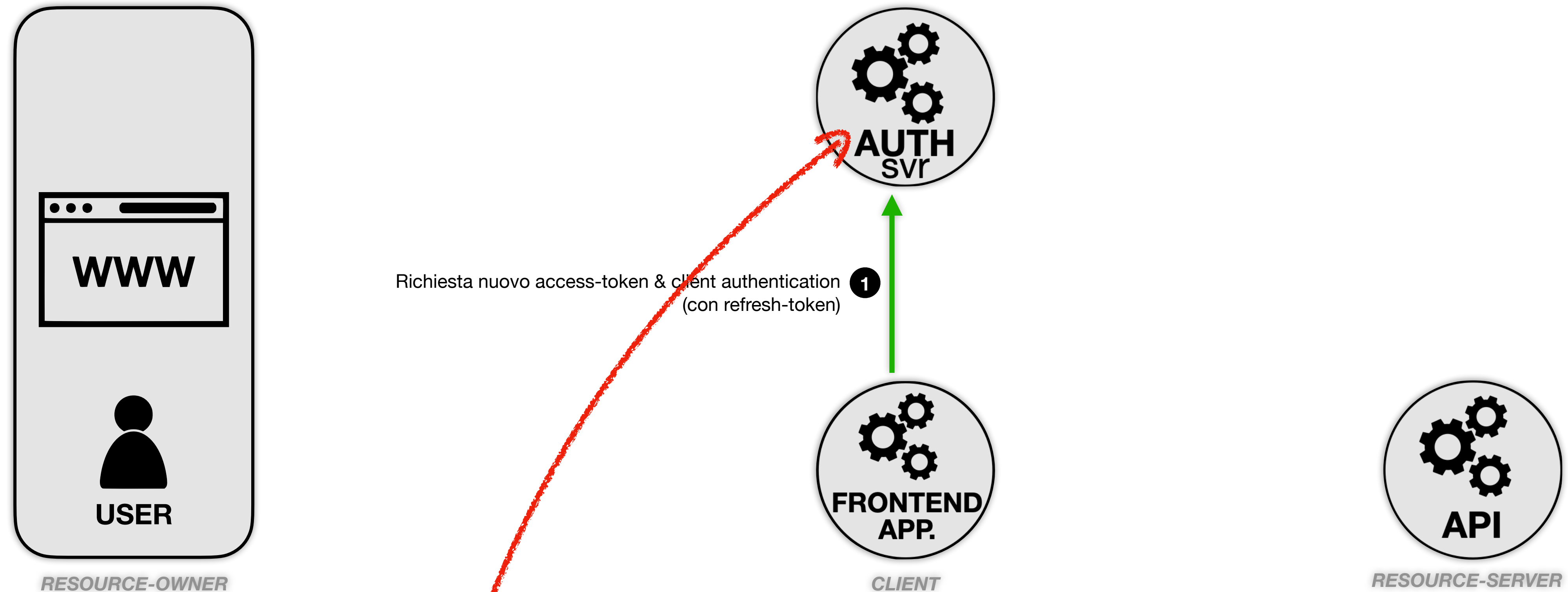
- automaticamente, **insieme all'access-token** (flussi authorization-code e refresh-token)
- se **OIDC** potrebbe essere necessario includere lo scope “**offline_access**”
- avviando il **refresh-token flow**

Quando usarlo:

- **dopo** accesso alle risorse rifiutato per access-token scaduto
- **prima** che l'access-token raggiunga la scadenza

RefreshTokenFlow

RefreshTokenFlow

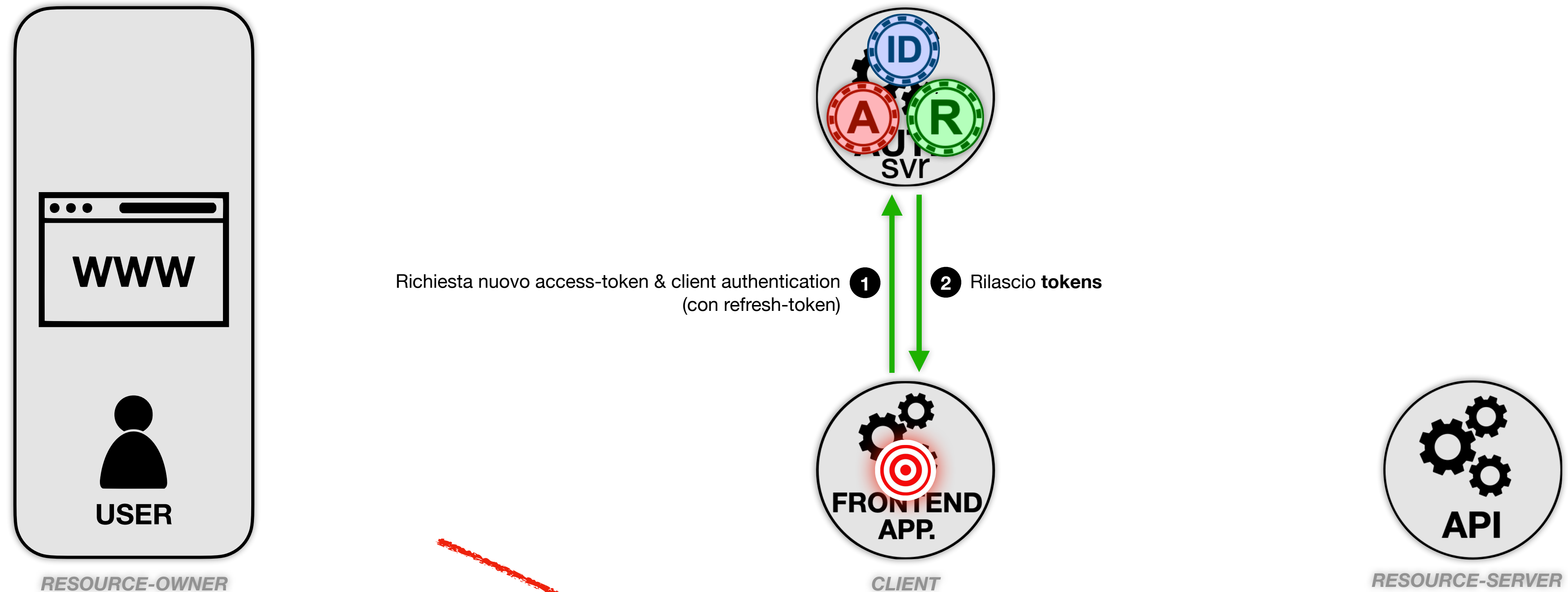


POST <https://oauth2.googleapis.com/token>
Content-Type: application/x-www-form-urlencoded

grant_type=refresh_token
&refresh_token=7mfYxWxDvTfKYq0UapYeI3z0iI3vTf_TfKmf
&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42
&client_secret=*****

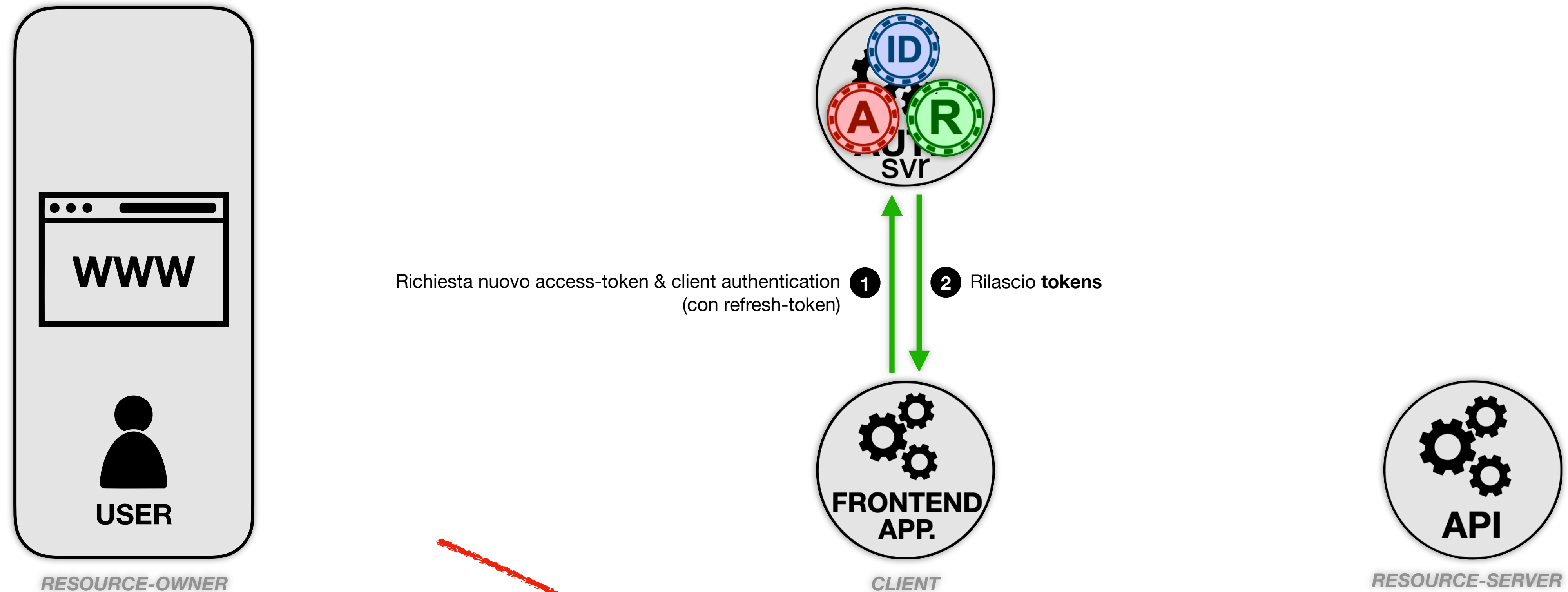
SE RICHIESTO (CONFIDENTIAL CLIENT)

RefreshTokenFlow



```
{  
  "access_token": "eyJhbGciOi0...du6TY9w",  
  "expires_in": 3600,  
  "token_type": "Bearer",  
  "refresh_token": "8xL0xBtZp8JSUzI1NiIs...",  
  "id_token": "eyJhbGciOi0iI6IkpXVCJ9..."  
}
```

RefreshTokenFlow

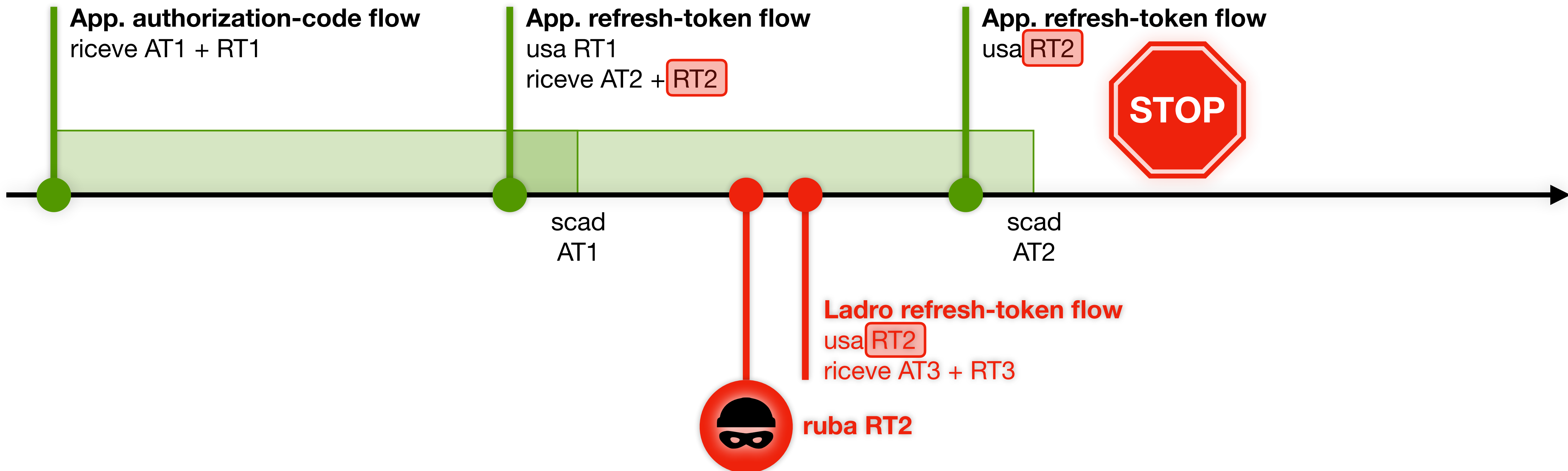


```
{  
  "access_token": "eyJhbGciOi0...du6TY9w",  
  "expires_in": 3600,  
  "token_type": "Bearer",  
  "refresh_token": "8xL0xBtZp8JSUzI1NiIs...",  
  "id_token": "eyJhbGciOi0iI6IkpXVCJ9..."  
}
```

RefreshTokenRotation (abuse detection)

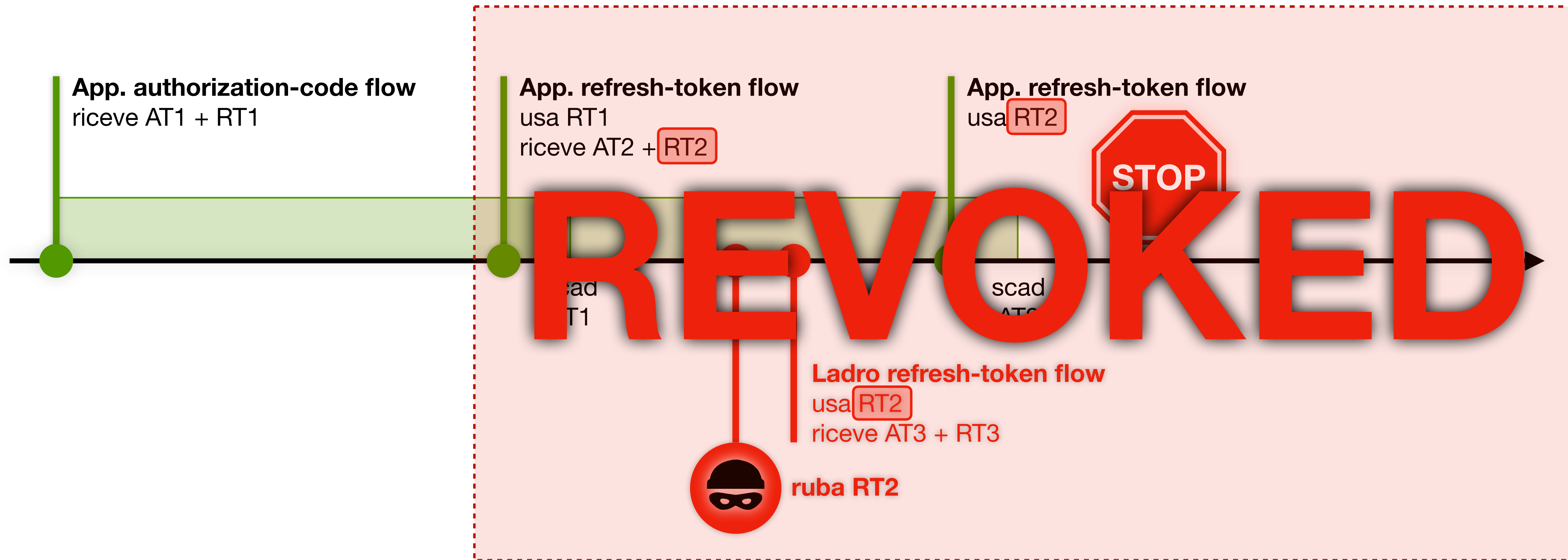
RefreshTokenRotation (abuse detection)

...se un malintenzionato rubasse il refresh-token?



RefreshTokenRotation (abuse detection)

...se un malintenzionato rubasse il refresh-token?



JWT (JSON Web Token)

JWT (JSON Web Token)

- è uno **standard** per rappresentare **informazioni (claim)** in modo **compatto**, **sicuro** e **trasportabile**
- composto da **3 parti** separate dal carattere “.” (punto)
 - **HEADER**: in chiaro, codificato **Base64url**
 - **PAYLOAD**: in chiaro, codificato **Base64url**
 - **SIGNATURE**: **SHA256** di “HEADER.PAYLOAD”, garantisce **integrità** e **autenticità** del token
- se serve più riservatezza usare JWE (JSON Web Encryption)

JWT (JSON Web Token)

esempio

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJEZWxwaGlEYXkgMjAyNSIsIm1hdCI6MTc1MDE1MjU3OSwiZXhwIjoxNzg4NTg4NTc1LCJhdWQiOiJTZW1pbmFyaW8gT0F1dGgrT0lEQyIsInN1YiI6Im1hdXJpemlvZGVsbWFnbm9AZ21haWwuY29tIiwiaWR2L2ZW50YW1lIjoiTWF1cm16aW8iLCJTdXJuYW1lIjoiaRGVsIE1hZ25vIiwiaWR2L2ZW50YW1lIjoiaRGVsIE1hZ25vQGdtYW1sLmNvbSJ9.fwmoEXqvavr7l46GV20_1IQdBKh056jkMJ0YVYw39zU

```
signature key = "1234"
```


JWT (JSON Web Token)

decodifica Base64url

```
{"typ": "JWT", "alg": "HS256"}
```

.

```
{"iss": "DelphiDay2025", "iat": 1750152579, "exp": 1781688579, "aud": "Seminario_0Auth+0IDC", "sub": "mauriziodelmagno@gmail.com", "GivenName": "Maurizio", "Surname": "DelMagno", "Email": "mauriziodelmagno@gmail.com", "nonce": "EFGH..."}
```

.

```
fwmoEXqvavr7l46GV20_1IQdBKh056jkMJ0YVYw39zU
```

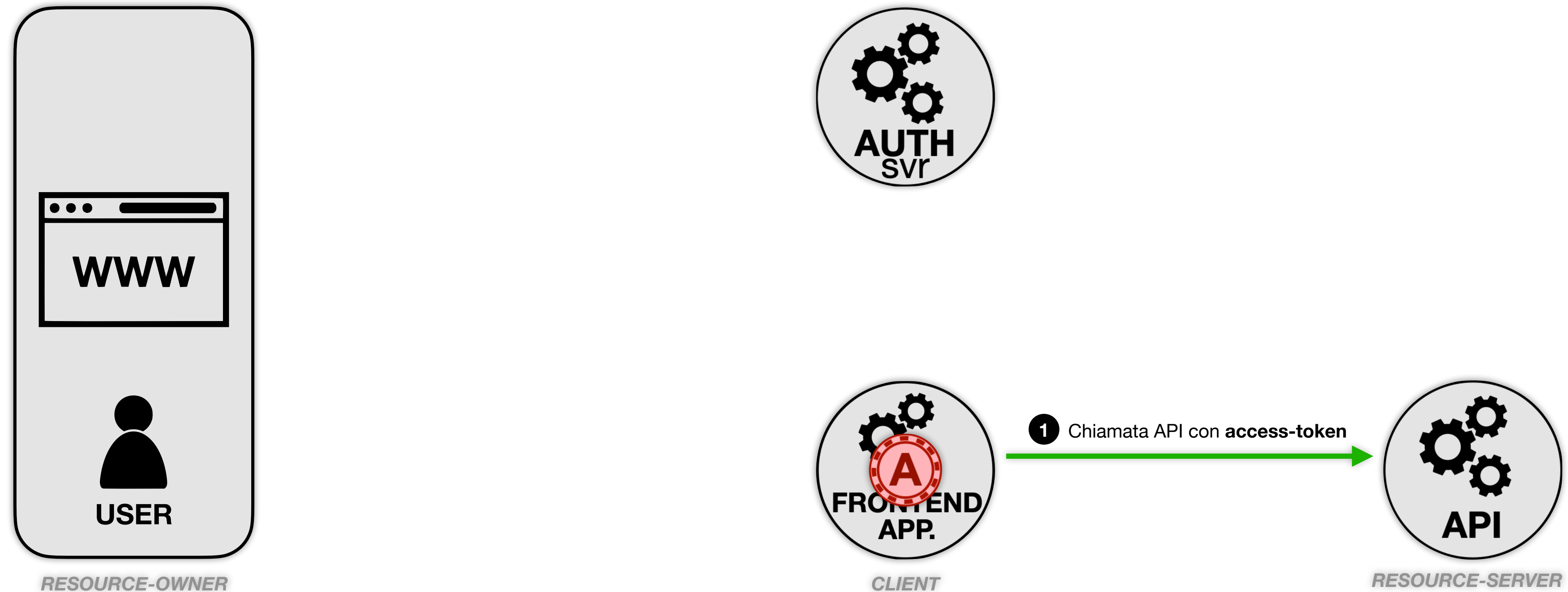
JWT (JSON Web Token)

```
{  
  "typ": "JWT",  
  "alg": "HS256"  
}  
.  
{  
  "iss": "DelphiDay2025",  
  "iat": 1750152579,  
  "exp": 1781688579,  
  "aud": "Seminario_0Auth+0IDC",  
  "sub": "mauriziodelmagno@gmail.com",  
  "GivenName": "Maurizio",  
  "Surname": "DelMagno",  
  "Email": "mauriziodelmagno@gmail.com",  
  "nonce": "EFGH..."  
}  
.  
fwmoEXqvavr7l46GV20_1IQdBKh056jkMJ0YVYw39zU
```

decodifica Base64url

AuthorizationDecision

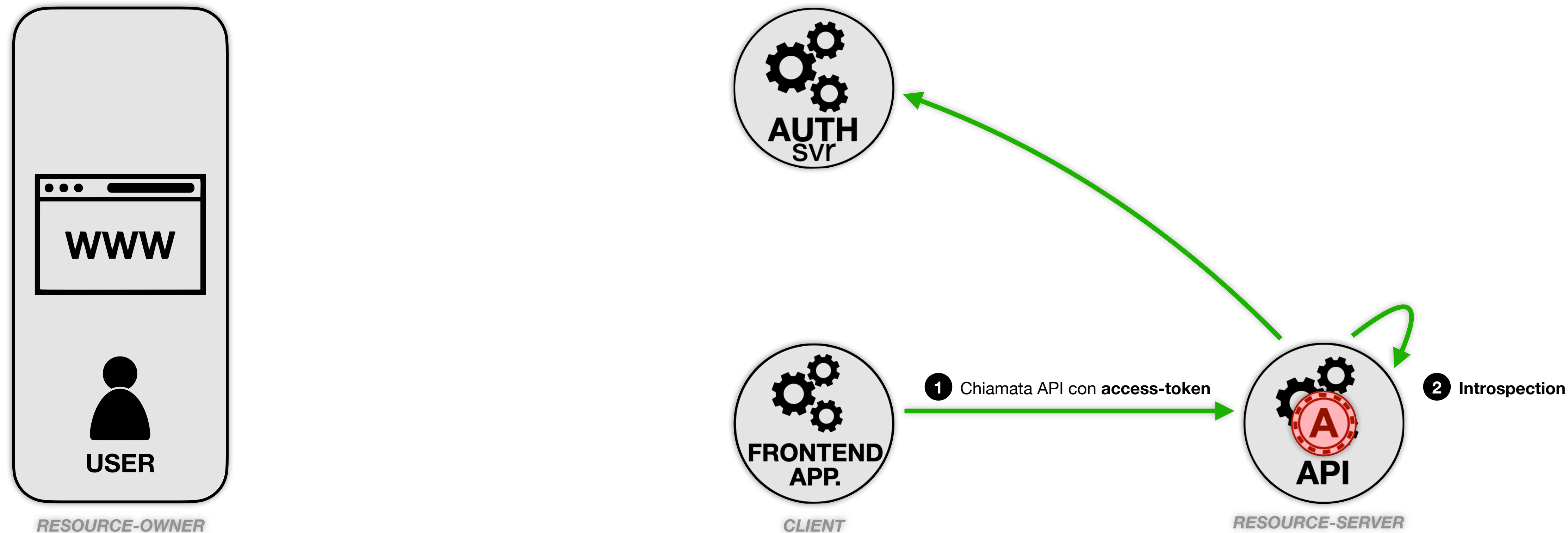
AuthorizationDecision



GET <https://openidconnect.googleapis.com/v1/userinfo>

Authorization: Bearer [eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...](#)
Accept: application/json

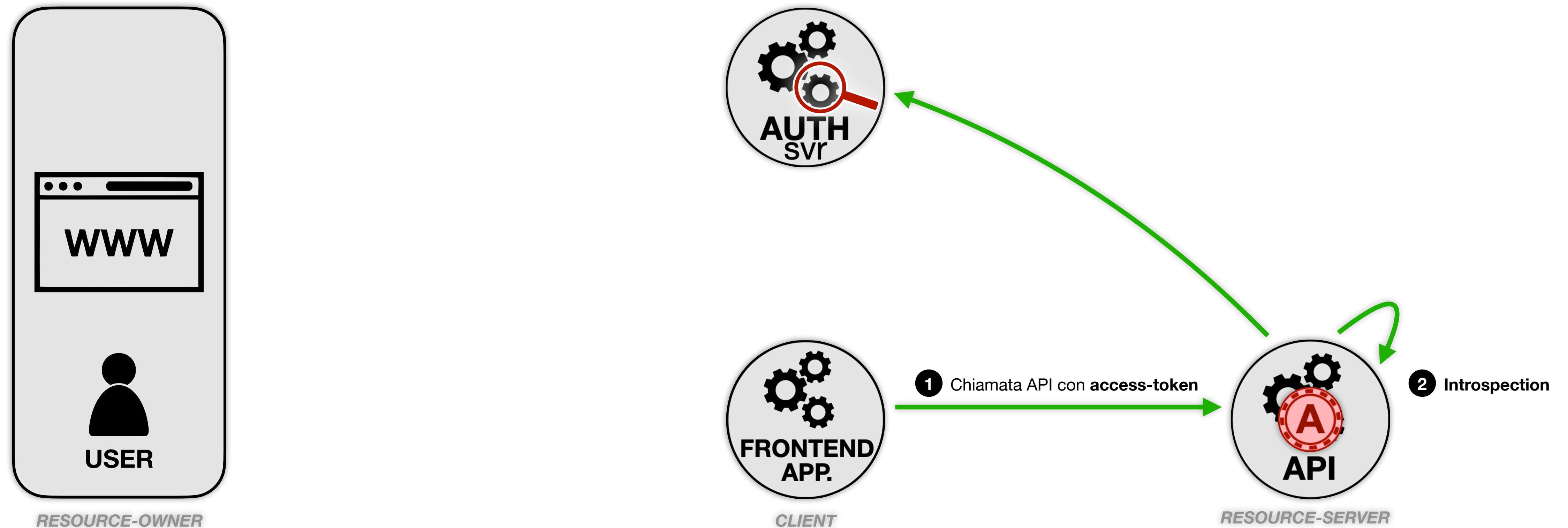
AuthorizationDecision



Introspection:

- Permette a un resource-server di verificare lo stato e i dettagli (claims) di un token
- **autonomamente** (self-contained-token trasparente)
- chiedendo all'**authorization-server**, introspection endpoint: (token opaco, reference-token)

AuthorizationDecision



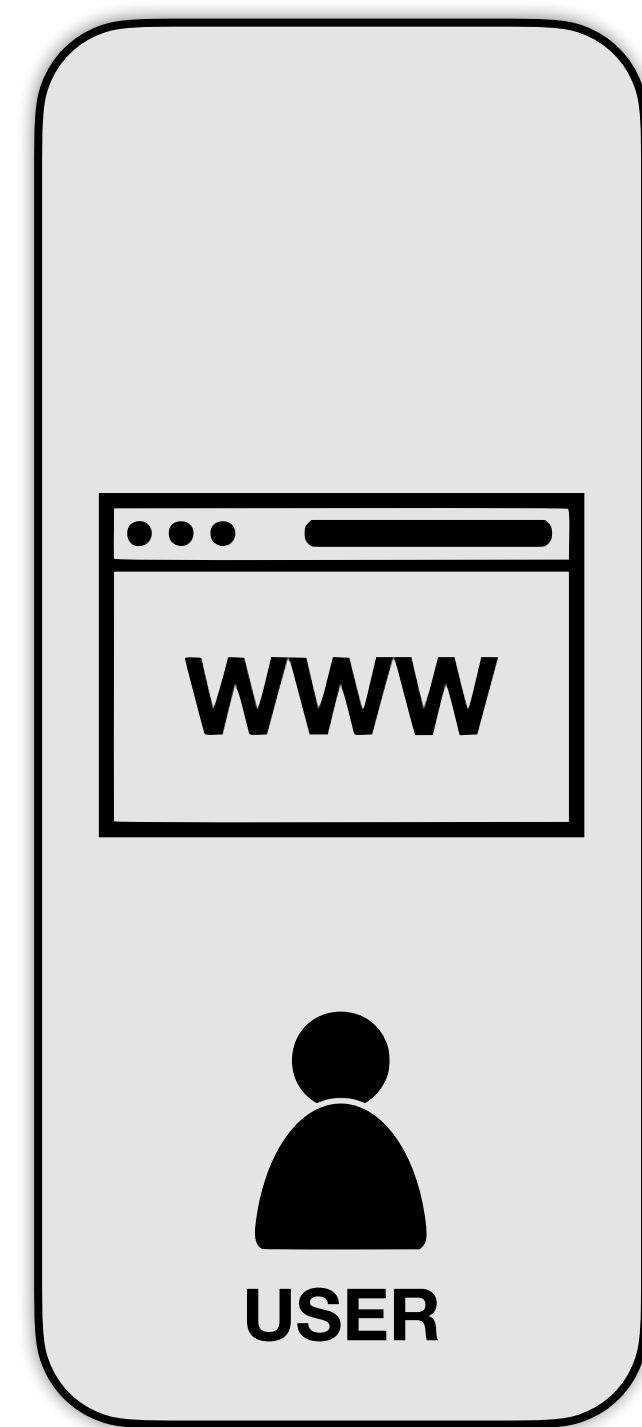
POST <https://example.com/oauth/introspect>

Content-Type: application/x-www-form-urlencoded

Authorization: Basic base64(client_id:client_secret)

token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...

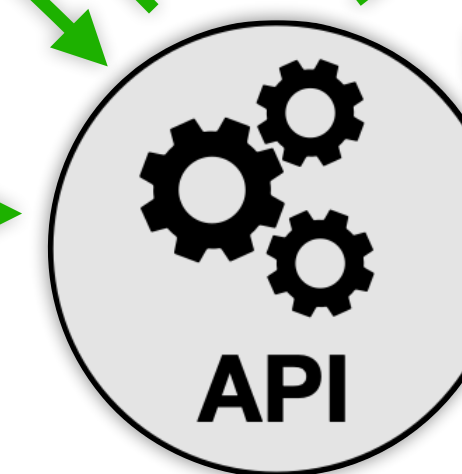
AuthorizationDecision



RESOURCE-OWNER



CLIENT



RESOURCE-SERVER

1 Chiamata API con access-token

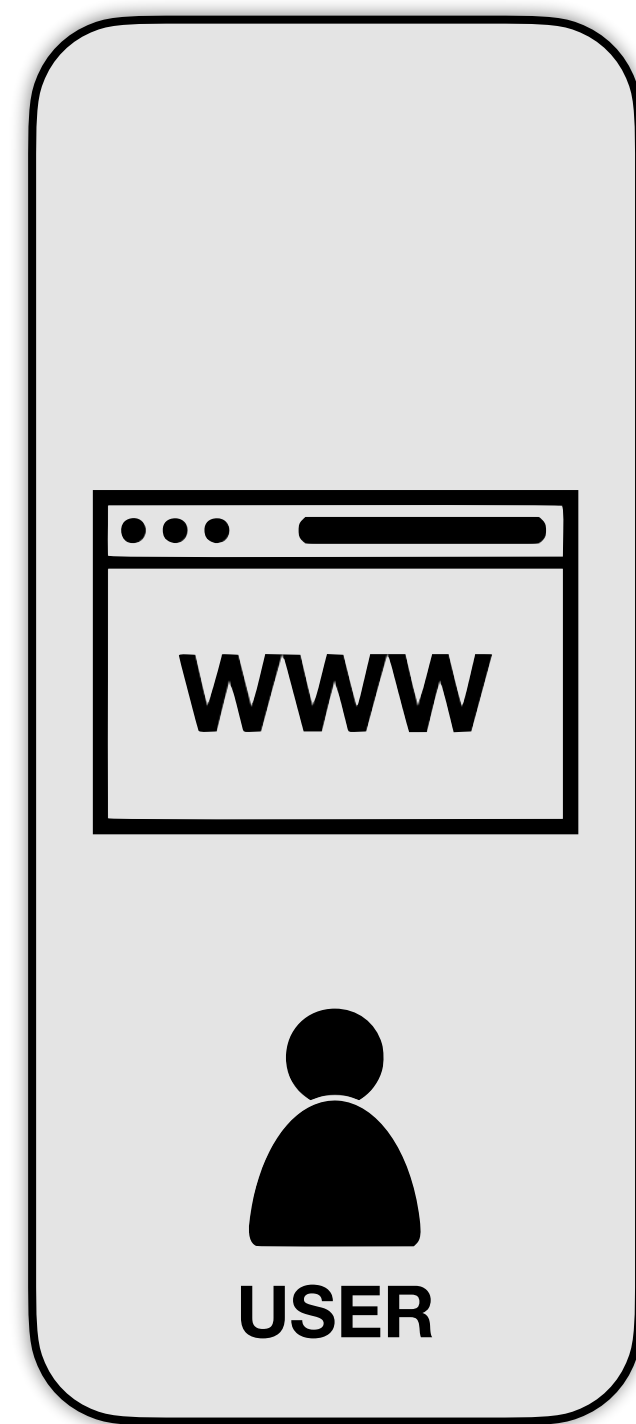
2 Introspection

```
{  
  "active": true,  
  "scope": "read write",  
  "client_id": "A123...",  
  "username": "delphiday@example.com",  
  "token_type": "access_token",  
  "exp": 1711234567,  
  "iat": 1711230000,  
  "sub": "user123",  
  "aud": "delphiday.example.com",  
  "iss": "https://auth.example.com"  
}
```

```
{  
  "active": false  
}
```

β

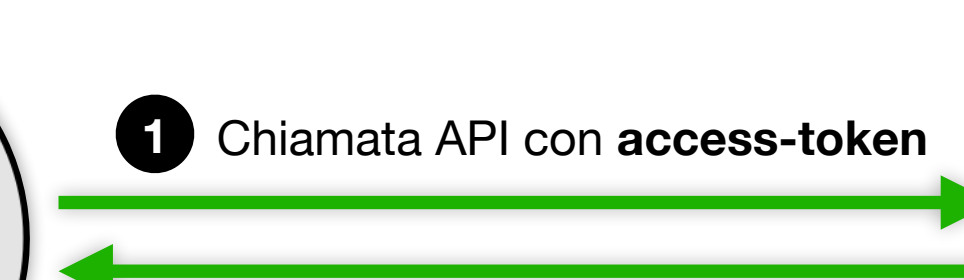
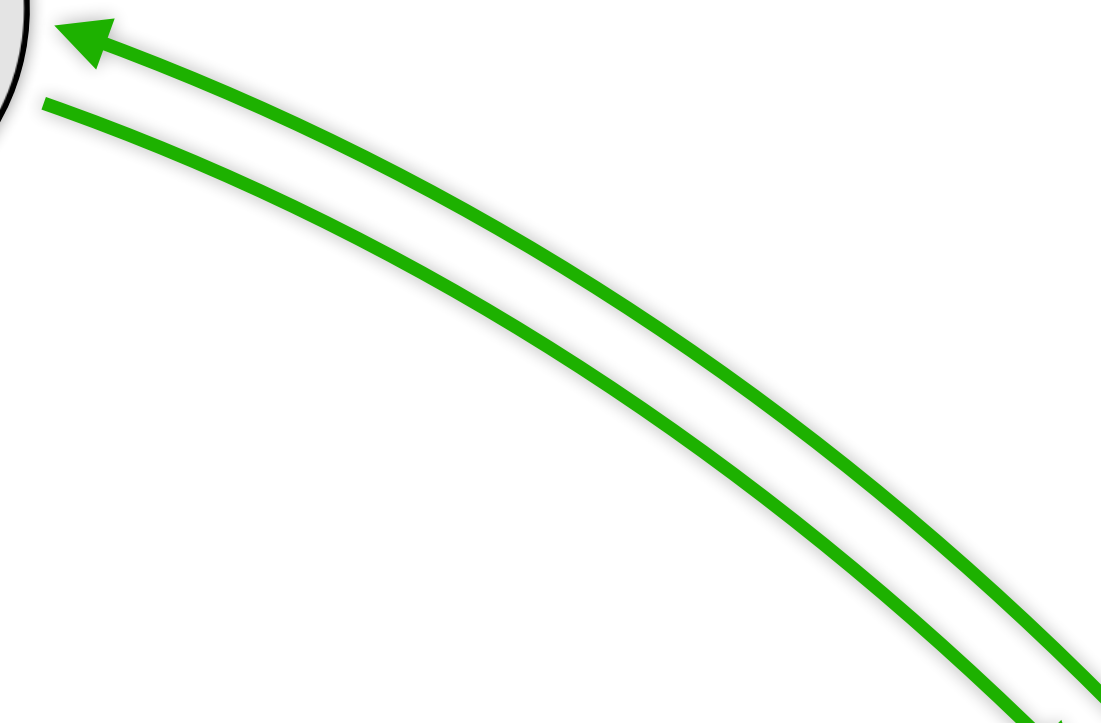
AuthorizationDecision



RESOURCE-OWNER



CLIENT



1 Chiamata API con **access-token**

4 Risposta con i dati richiesti



RESOURCE-SERVER

2 Introspection

3 Decisione di autorizzazione

```
{  
  "iss": "DelphiDay2025",  
  "iat": 1750152579,  
  "exp": 1781688579,  
  "aud": "Seminario_0Auth+OIDC",  
  "sub": "mauriziodelmagno@gmail.com",  
  "GivenName": "Maurizio",  
  "Surname": "DeLMagno",  
  "Email": "mauriziodelmagno@gmail.com"  
}
```

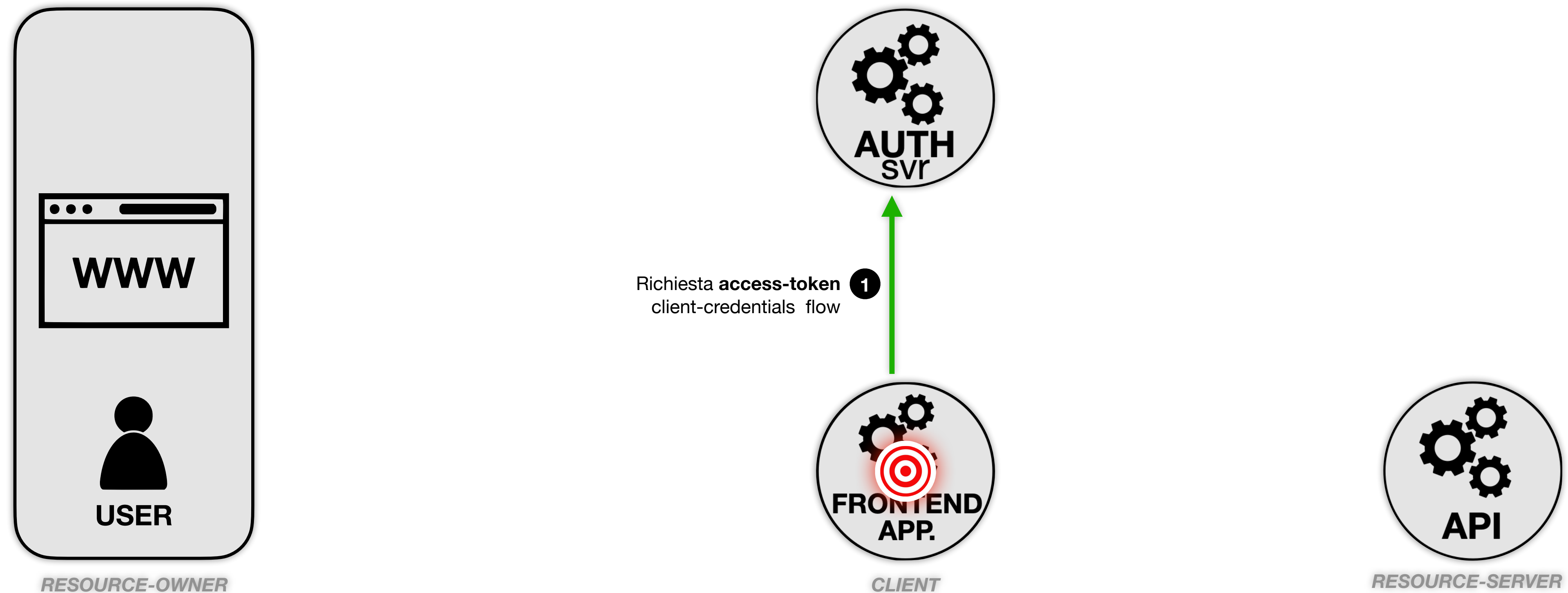
B

Client**Credentials**Flow

Client**Credentials**Flow

- ...e se non c'è alcun utente?
- ...non ci sarà nemmeno una UI!
- ...**nessuna interazione umana** (no browser)
- operazioni **machine to machine**, **NON** per conto di un utente (backup, configuration tools)
- comunicazione **back-channel** (molto sicura)
- abbiamo ugualmente bisogno di un **access-token**

ClientCredentialsFlow



POST www.googleapis.com/oauth2/v4/token

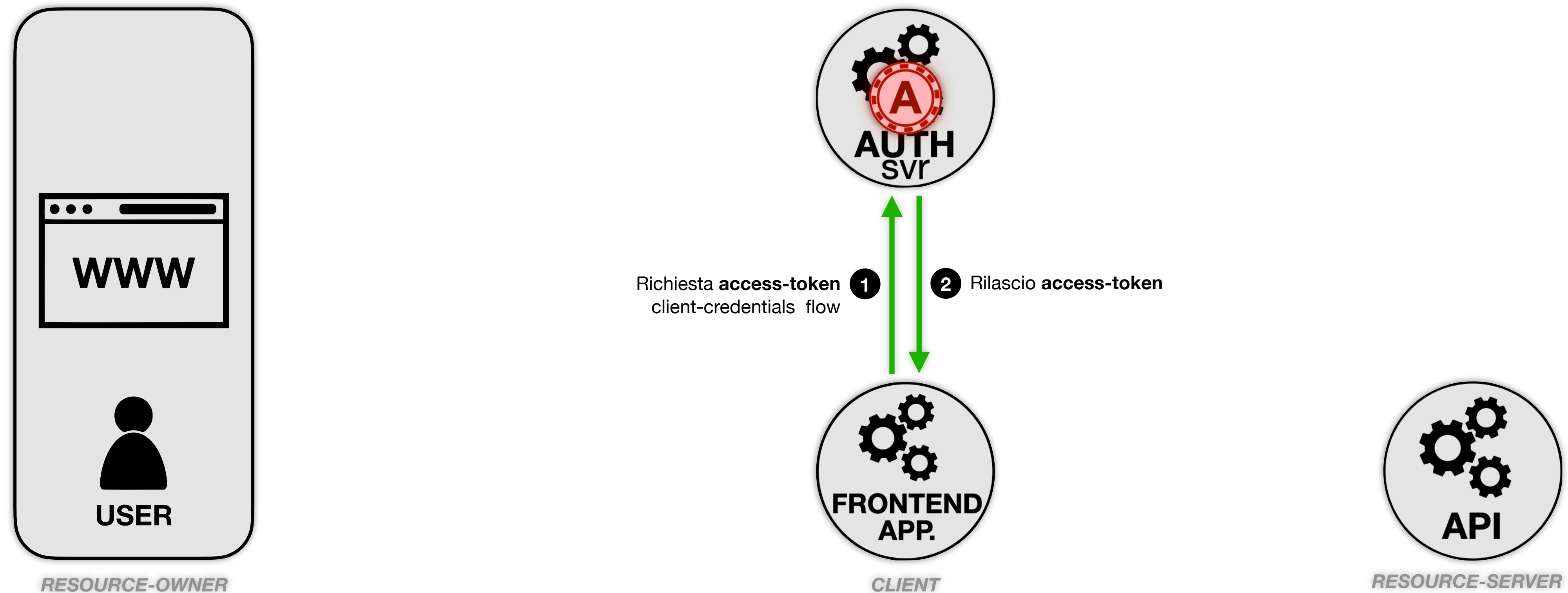
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials

&client_id=FN983CEYgx4mdUg3NKNKHjwfNAL5Fb42

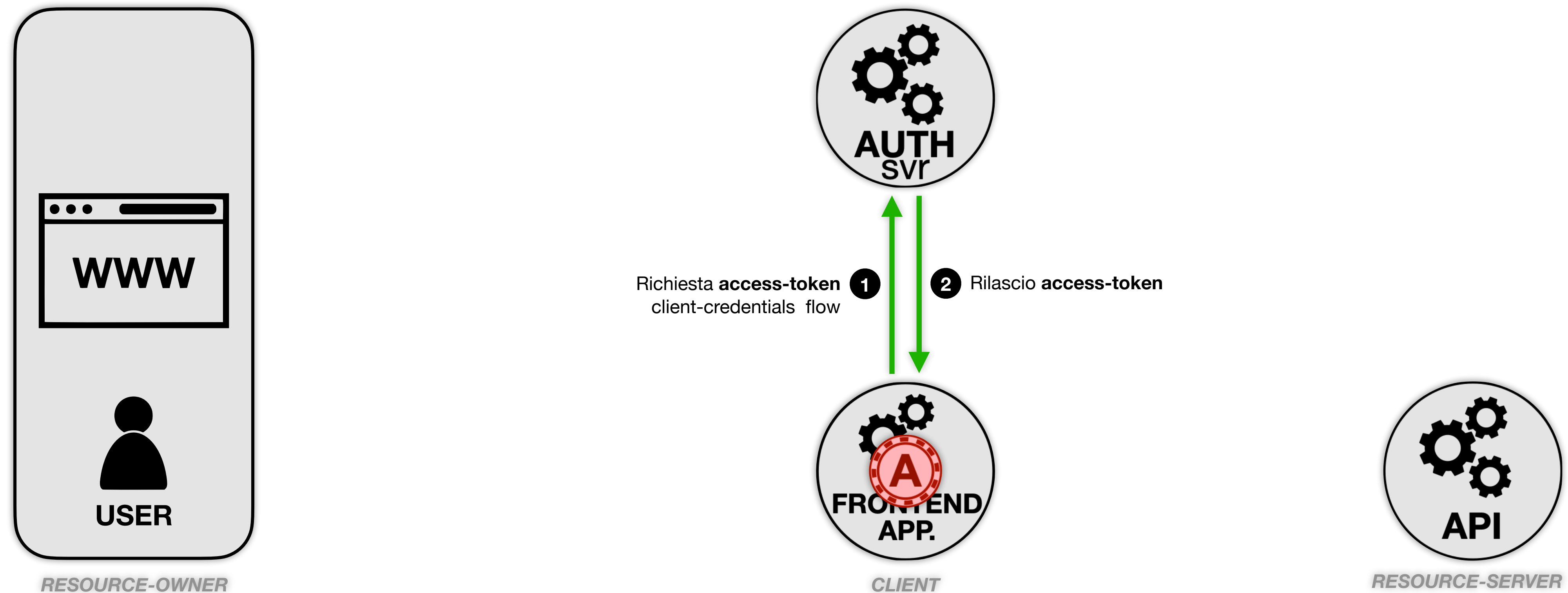
&client_secret=*****

ClientCredentialsFlow



```
{  
  "access_token": "eyJhbGciOi0...du6TY9w",  
  "expires_in": 3600,  
  "token_type": "Bearer",  
}
```

ClientCredentialsFlow



Vantaggi:

- **coerenza**: stesse chiamate per client e resource-server
- **NO refresh-token** (se serve un altro access-token basta chiedere con lo stesso flusso)
- non serve memorizzare l'**access-token** (quando ne servirà un altro basta chiedere con lo stesso flusso)
- solo **confidential-client** (client authentication con client-secret)

OAuth 2.1

~~OAuth 2.1~~

Flussi OAuth 2.0:

- Authorization-code flow
- Client-credentials flow
- Refresh-token flow
- Device Authorization flow
- Implicit flow
- Resource-Owner flow

Flussi OAuth 2.1:

- Authorization-code flow + PKCE (no client-secret)
- Client-credentials flow
- Refresh-token flow



OAuth + OpenID Connect due sigle, un solo obiettivo

Se pensi sia solo un “login” cambierai idea



SPEAKERS:
CLAUDIO PIFFER
MAURIZIO DEL MAGNO

