



# **Blockchain, Criptomoedas & Tecnologias Descentralizadas**

## **Tecnologias descentralizadas: Delegação de acesso e Identidades Federadas**

**Prof. Dr. Marcos A. Simplicio Jr. – [mjunior@larc.usp.br](mailto:mjunior@larc.usp.br)  
Escola Politécnica, Universidade de São Paulo**

# Objetivos

- Discutir duas tecnologias para delegação de acesso a dados
  - OAuth 2.0: dados quaisquer
  - OpenID Connect (OIDC): identidades de usuários
- Nota: **não** foram projetados como **tecnologias P2P entre usuários finais**
  - Mas têm cenários com dados distribuídos como base: podem ser usados para construir sistemas com diferentes graus de descentralização!

# Cenário: dados de usuários

- Dados de usuários espalhados em diversos locais



- **Dados médicos:** bases de hospitais e laboratórios
- **Informações financeiras:** bancos, seguradoras, operadores da bolsa de valores, receita federal



- **Fotos:** redes sociais, backups em nuvem
- **Registros profissionais:** servidor dentro da empresa



- **Registro de nascimento e bens:** cartórios
- **Jogos, filmes:** plataformas de entretenimento
- ...



- Já se trata de um ambiente descentralizado... mas é pouco integrado: **silos de informação**
  - Como **integrar** esses silos c/ **segurança** (acesso controlado)?

# Delegação: OAuth

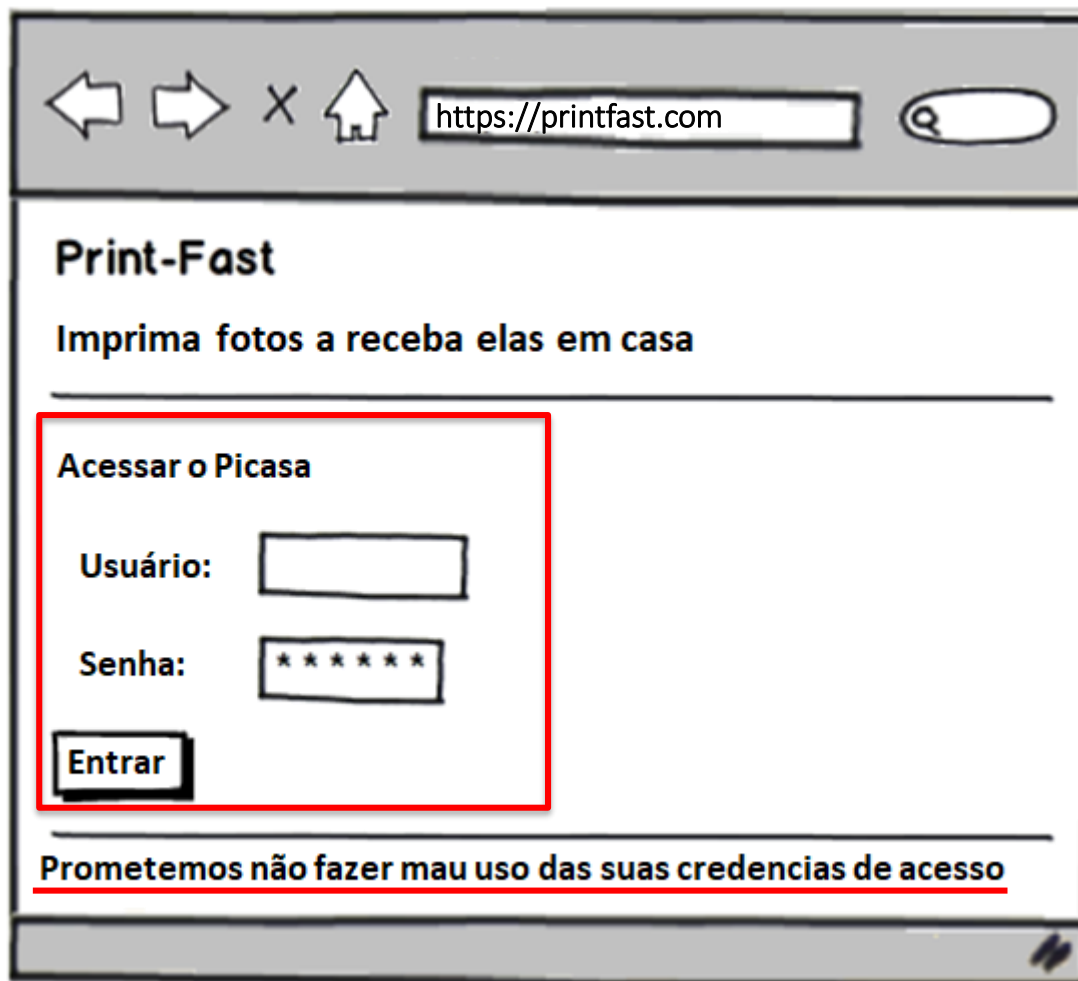


- OAuth (*Open Authorization*): protocolo padronizado para controle de acesso, com foco em **delegação**
  - Usuário pode **permitir que sistema R acesse seus dados** armazenados em sistema S sem expor suas credenciais
  - Baseado em sistema de "delegação", com **tokens que autorizam acesso** a determinado conjunto de recursos sob a anuência do usuário dono desses recursos
  - Projetado para uso com HTTP
    - Exige TLS em todas as comunicações envolvendo tokens
  - Versão atual: 2.0 (<http://oauth.net/2/>)
    - Comum o uso do formato JSON Web Token (JWT) -- RFC 7519
- **Especificação oficial:** RFC 6749
- **Vídeo introdutório:** [www.youtube.com/watch?v=XGmUlyggXVo](http://www.youtube.com/watch?v=XGmUlyggXVo)

# Exemplo: Acesso sem OAuth -- 1



# Exemplo: Acesso sem OAuth -- 2



Print-Fast

Imprima fotos a receba elas em casa

**Acessar o Picasa**

Usuário:

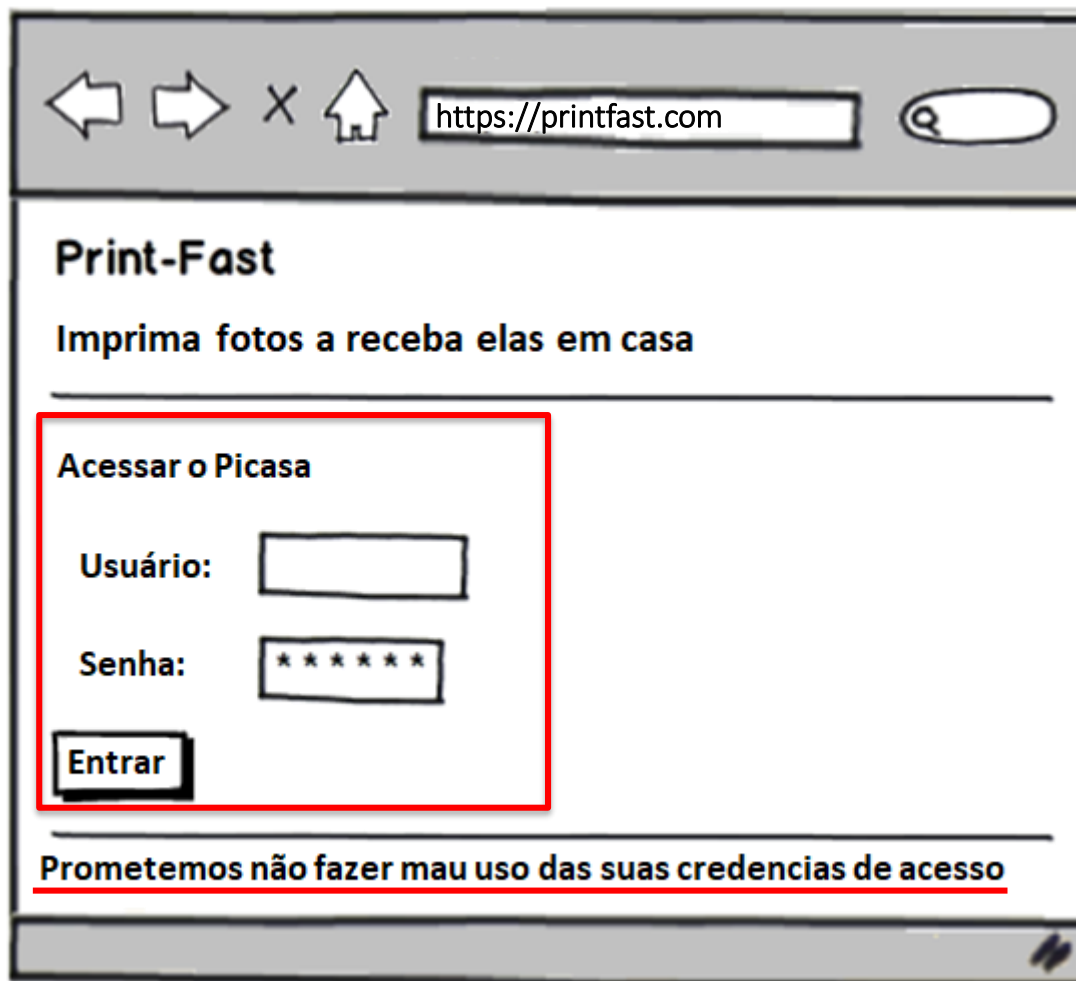
Senha:

**Entrar**

Prometemos não fazer mau uso das suas credencias de acesso



# Exemplo: Acesso sem OAuth -- 2



Print-Fast

Imprima fotos a receba elas em casa

**Acessar o Picasa**

Usuário:

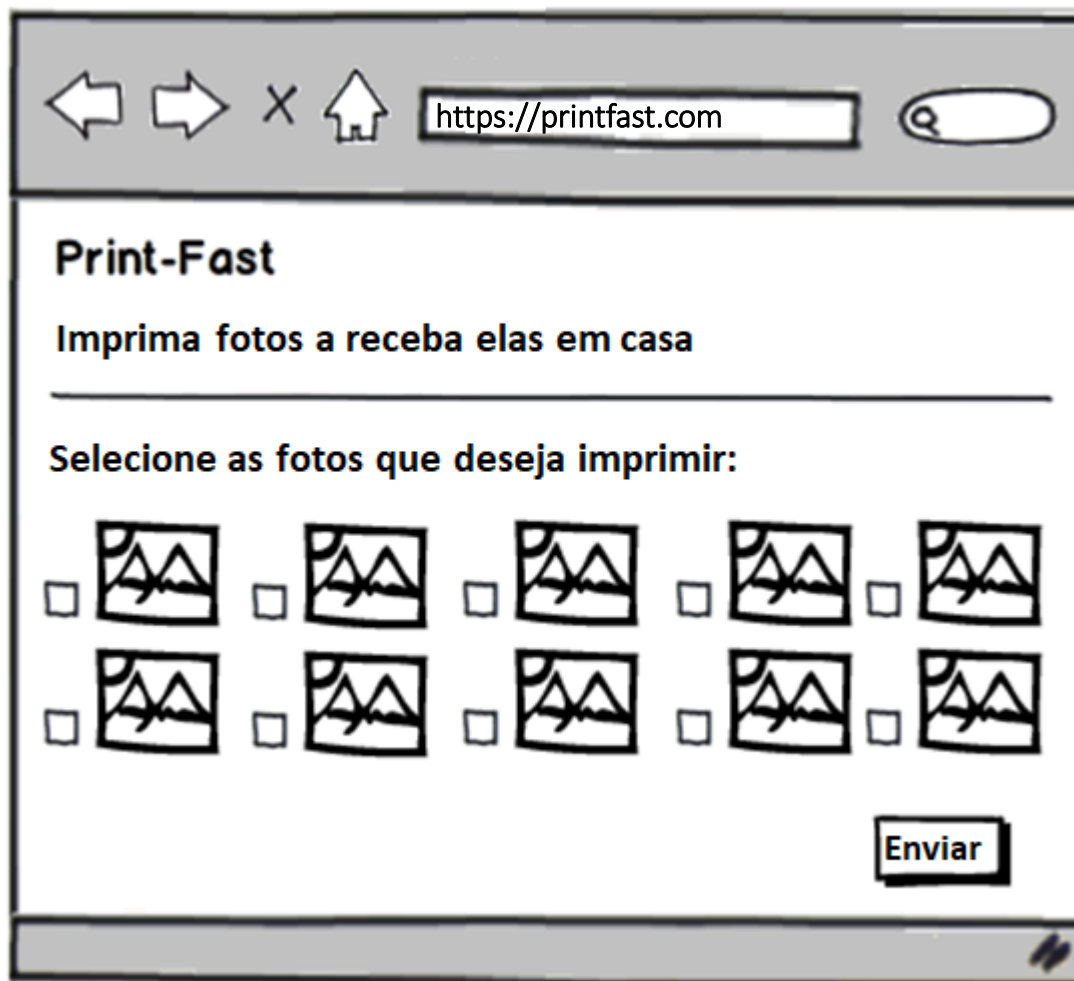
Senha:

**Entrar**

Prometemos não fazer mau uso das suas credencias de acesso



# Exemplo: Acesso sem OAuth -- 3





# Exemplo: Acesso com OAuth



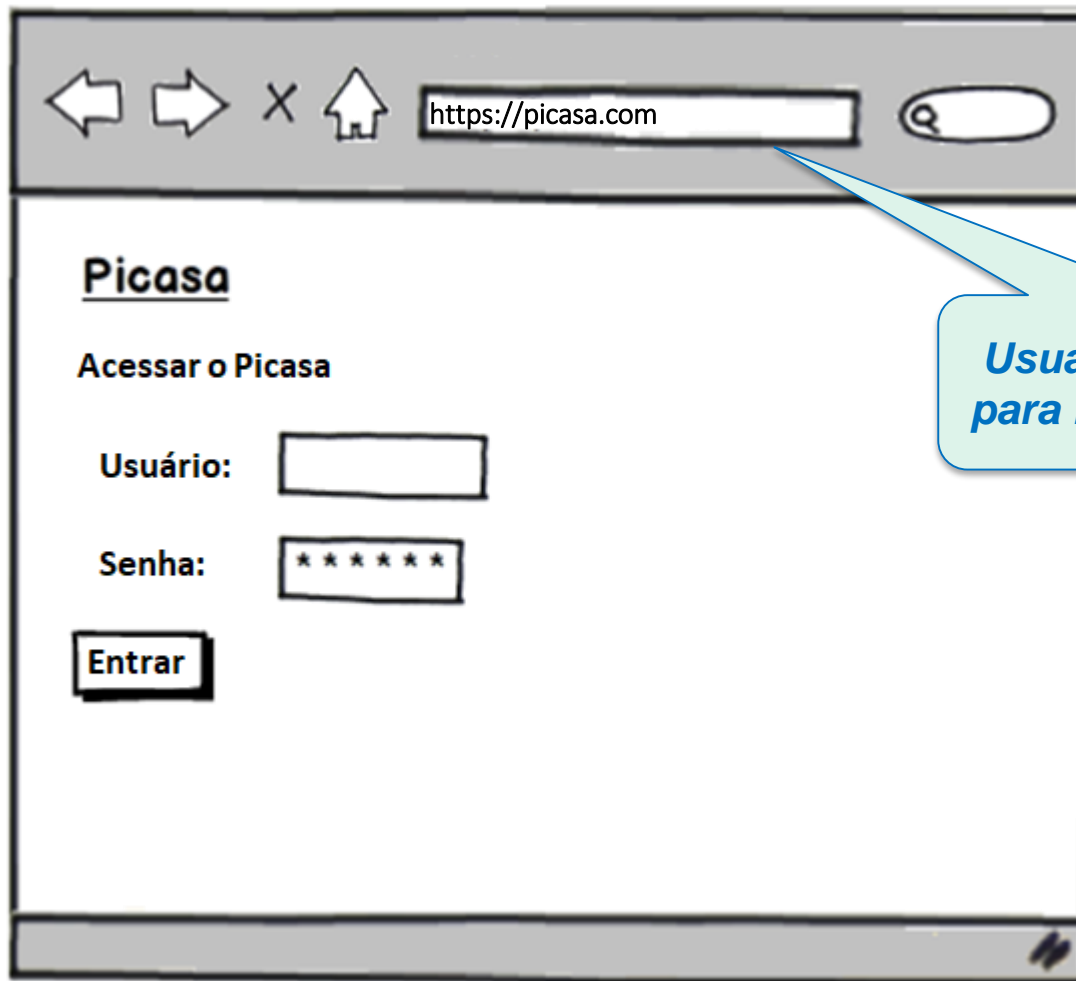
*melhor que*



# Exemplo: Acesso com OAuth -- 1



# Exemplo: Acesso com OAuth -- 2



← → × 🏠  🔍

**Picasa**

Acessar o Picasa

Usuário:

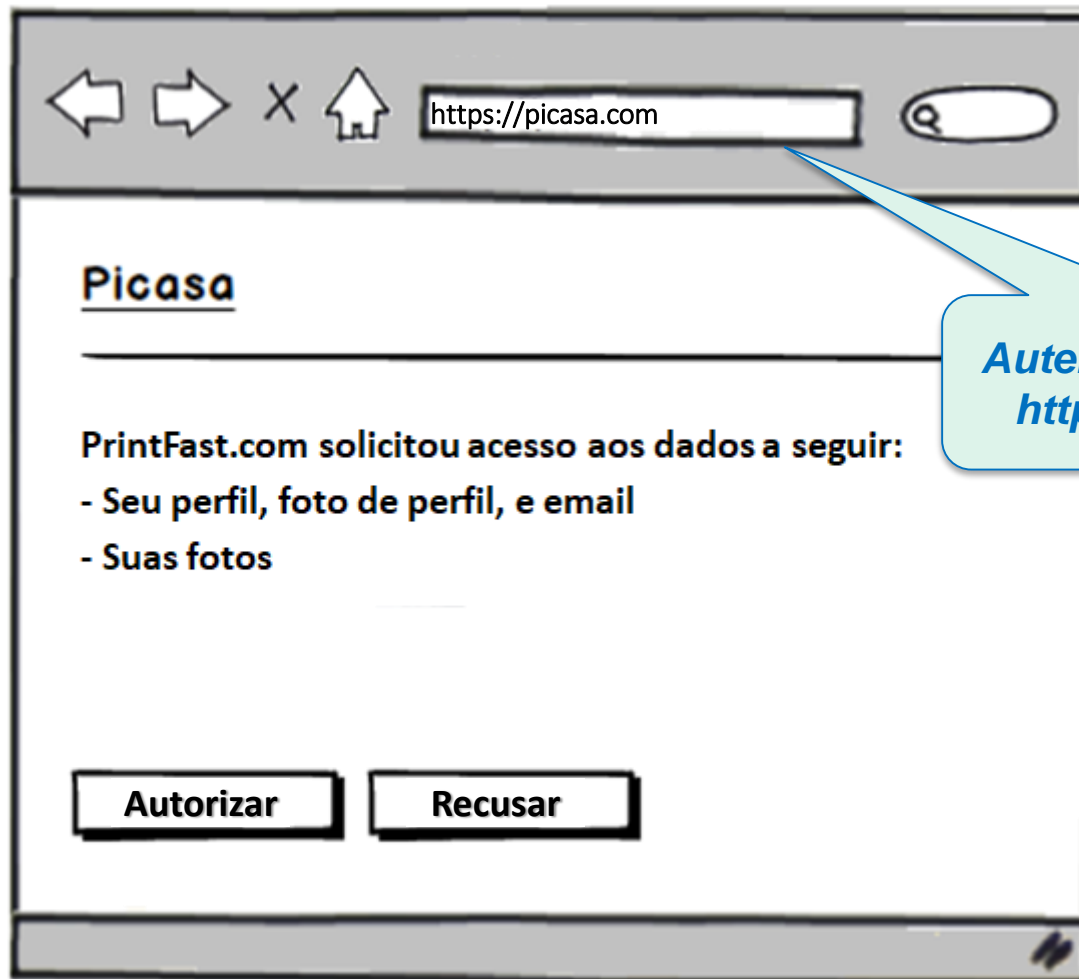
Senha:

**Entrar**

*Usuário redirecionado para https://picasa.com*



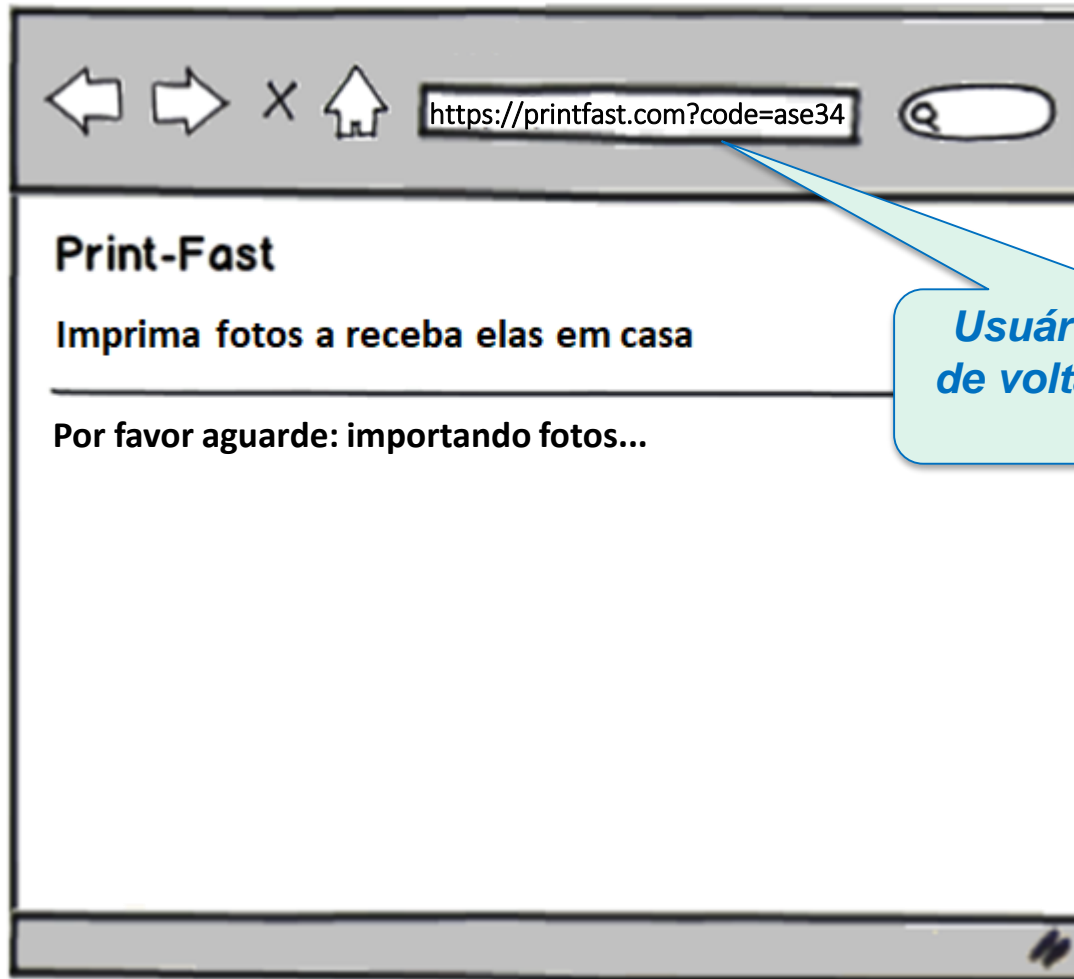
# Exemplo: Acesso com OAuth -- 3



*Autenticação feita em  
<https://picasa.com>*



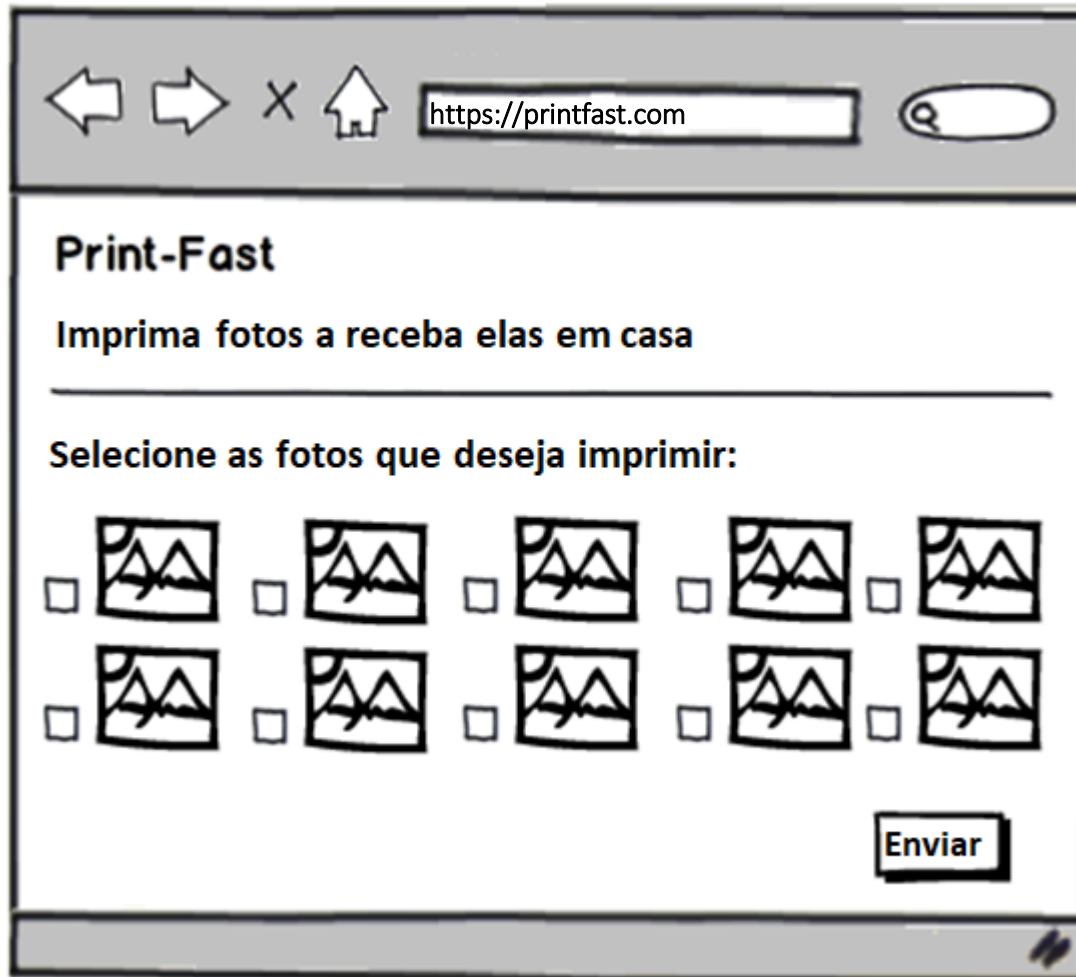
# Exemplo: Acesso com OAuth -- 4



*Usuário redirecionado  
de volta, com código de  
acesso*

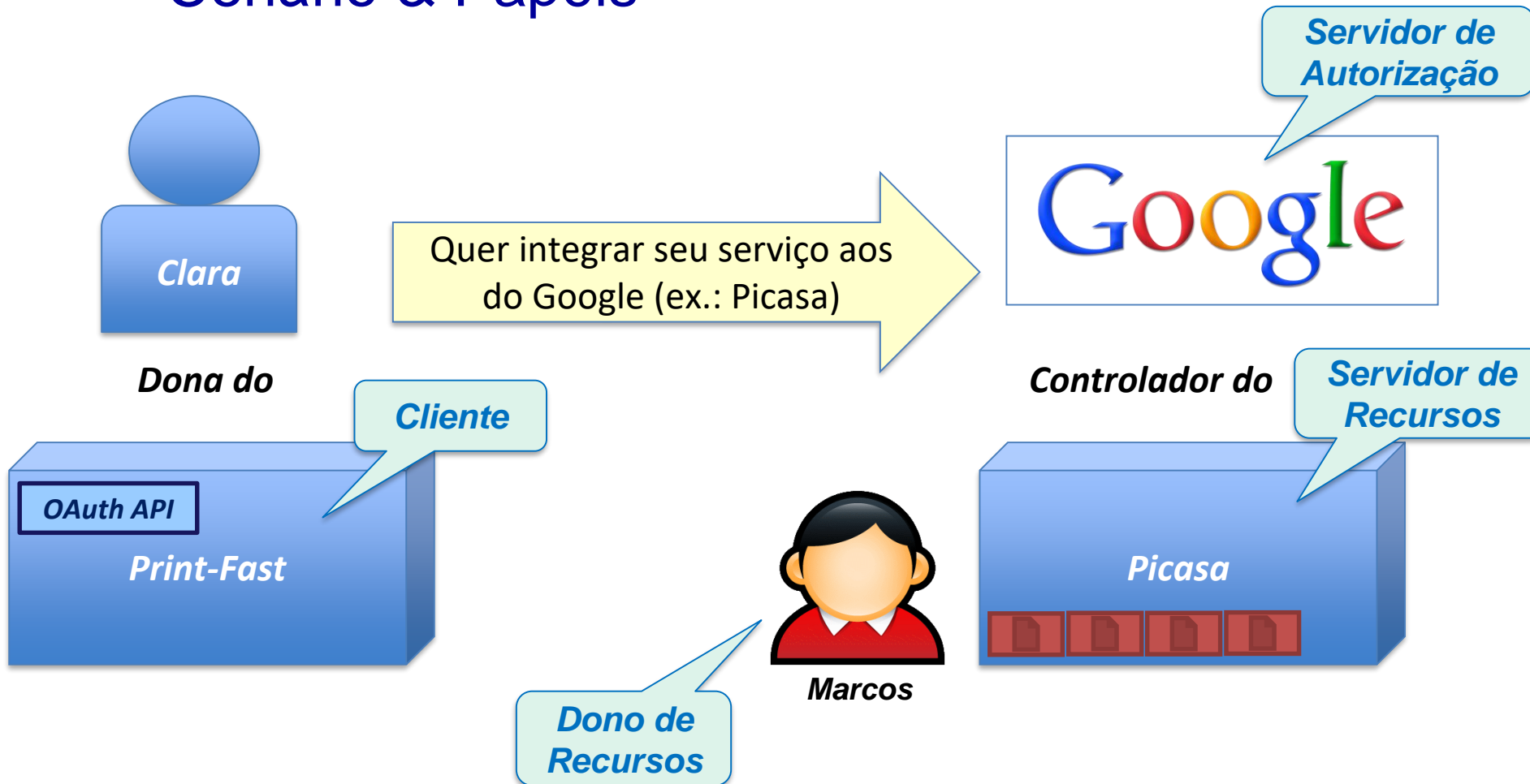


# Exemplo: Acesso com OAuth -- 5



# Protocolo OAuth: funcionamento

- Cenário & Papéis



# Protocolo OAuth: funcionamento

- Passo 0: Registro do cliente
  - Cliente: servidor, plugin de browser, aplicação nativa, ...





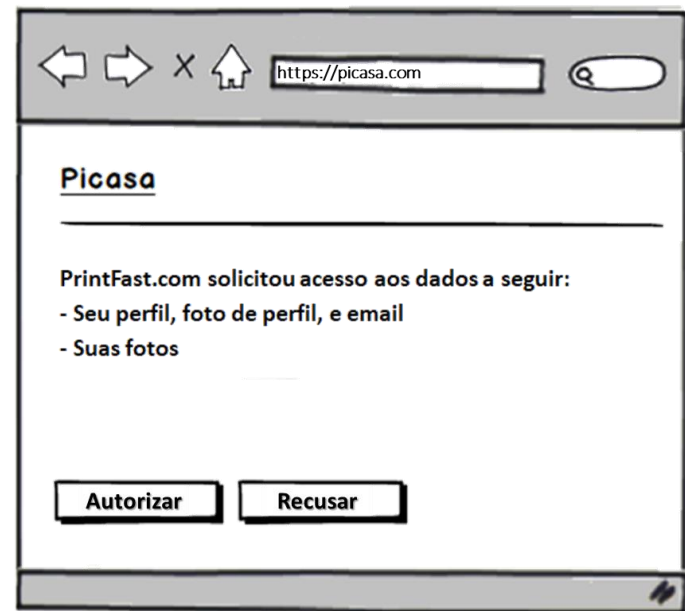
# Protocolo OAuth: funcionamento

- Passo 1: Obter código de autorização

## *Requisição de autorização*



## *Concessão de autorização*



**URL de redirecionamento:** `https://picasa.com/?client_id=print-fast`  
&`scope=profile,email,photos` &`redirect_uri=http://printfast.com`&`response_type=code`

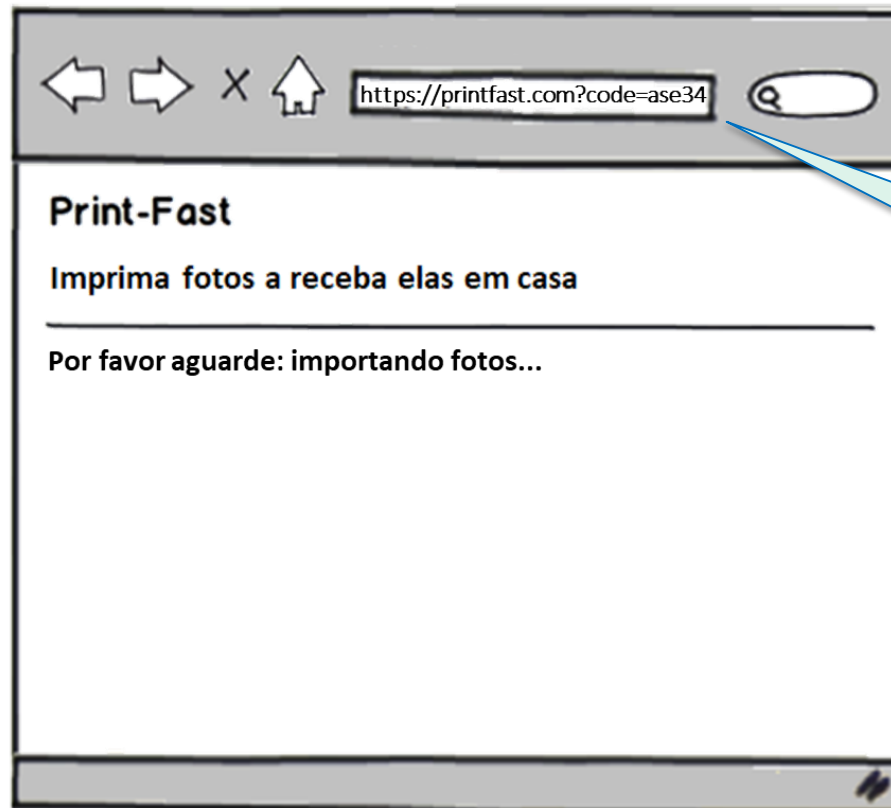
**scope:** tipos de recursos que Cliente deseja acessar (nomes não são padronizados)

➔ Dono de Recursos pode **limitar acesso** a um subconjunto do escopo requisitado

# Protocolo OAuth: funcionamento

- Passo 1: Obter código de autorização

*Resultado da concessão de autorização*



**Código de  
Autorização:  
ase34**

# Protocolo OAuth: funcionamento

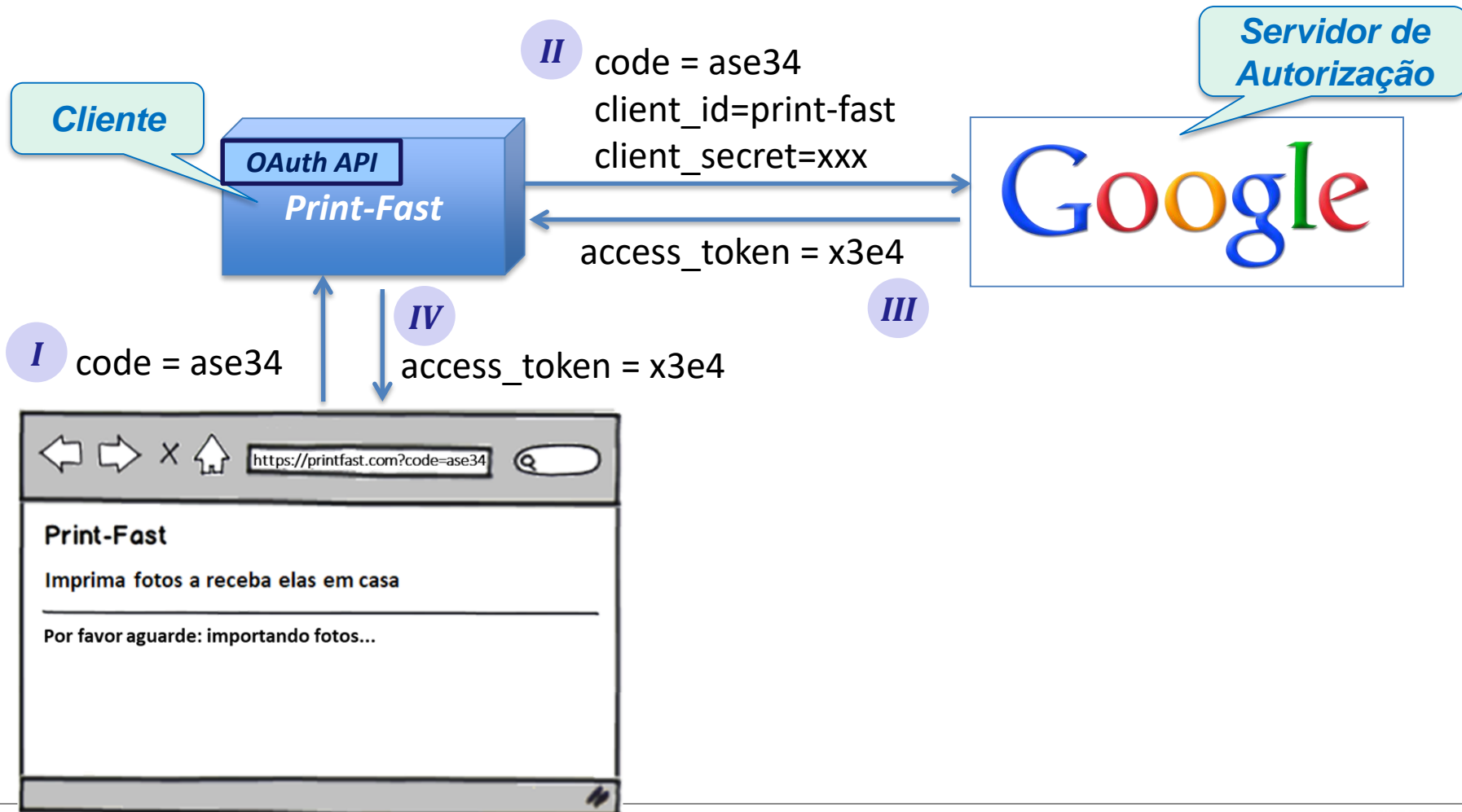
## *Requisição de autorização*

client\_id=print-fast  
redirect\_url=http://printfast.com  
scope=profile,email,photos

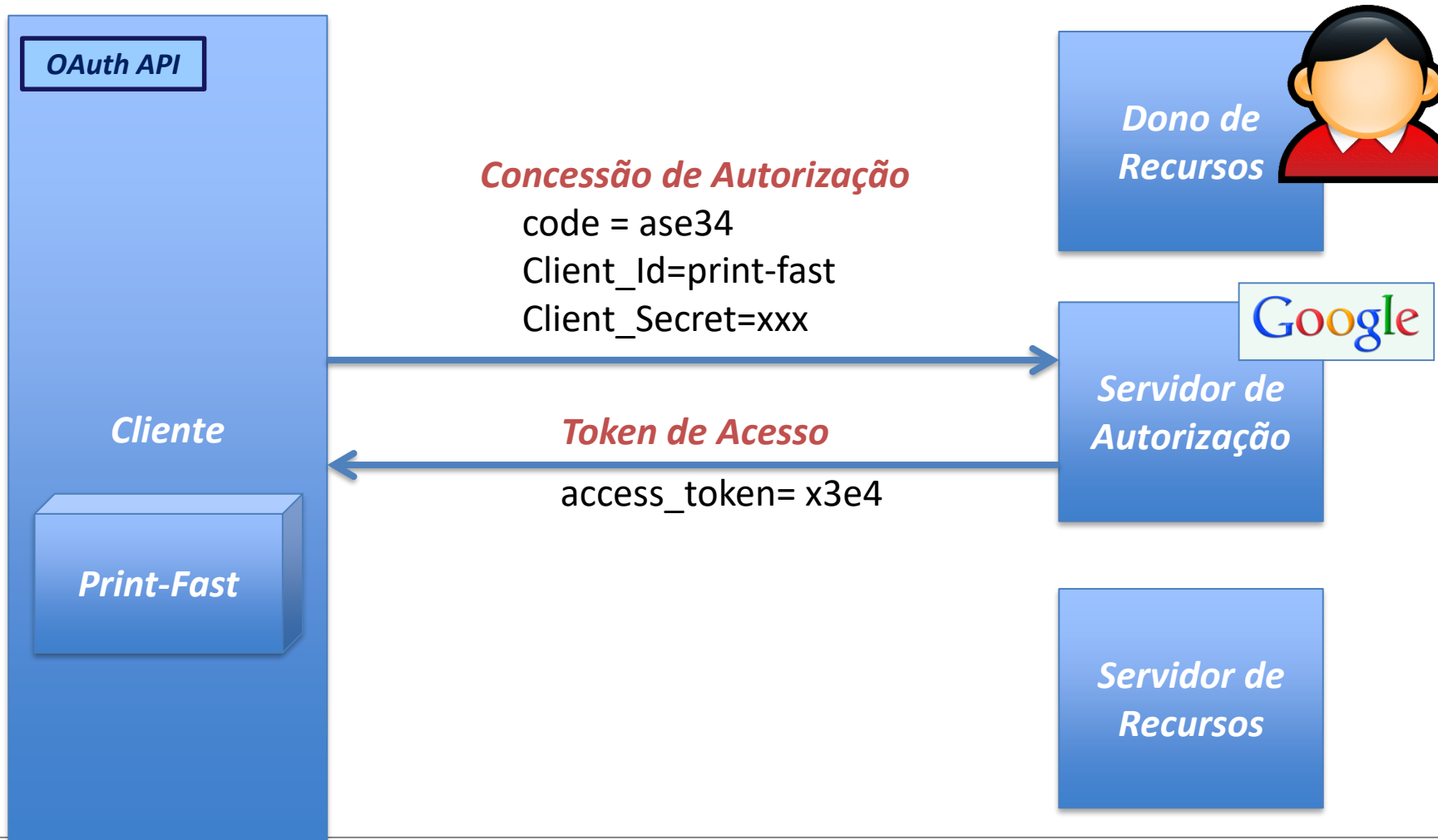


# Protocolo OAuth: funcionamento

- Passo 2: Trocar código de autorização por token de acesso

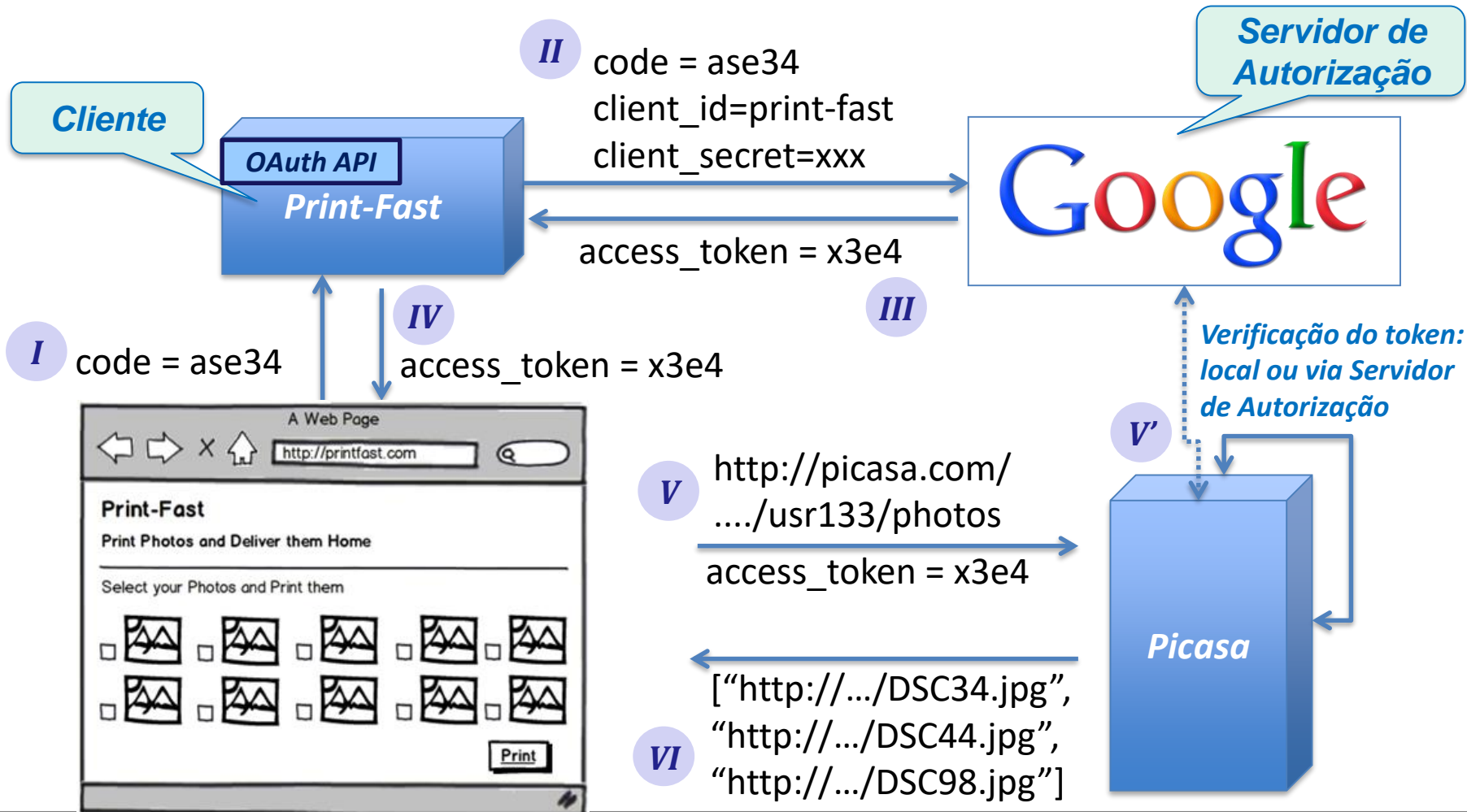


# Protocolo OAuth: funcionamento

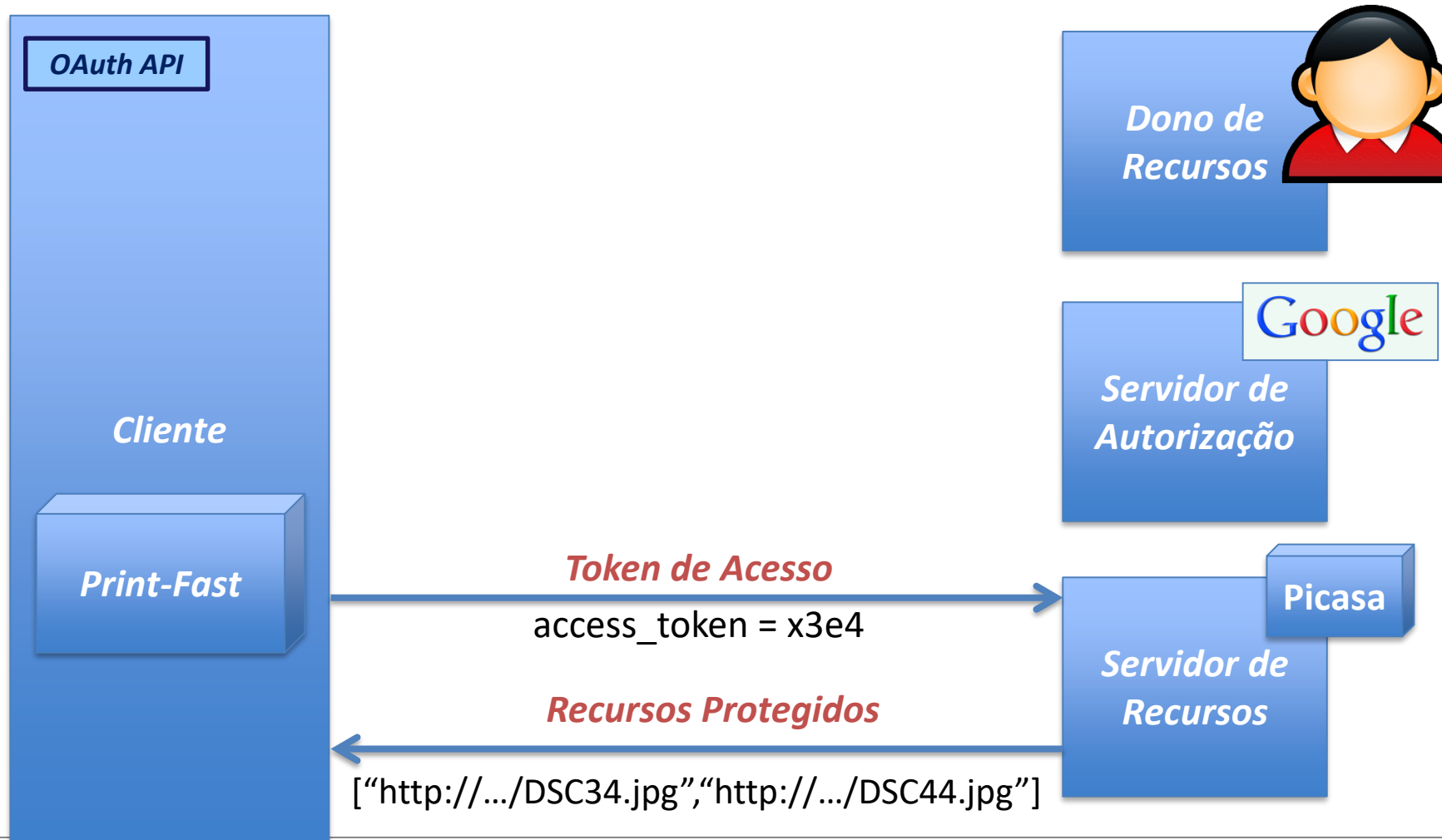


# Protocolo OAuth: funcionamento

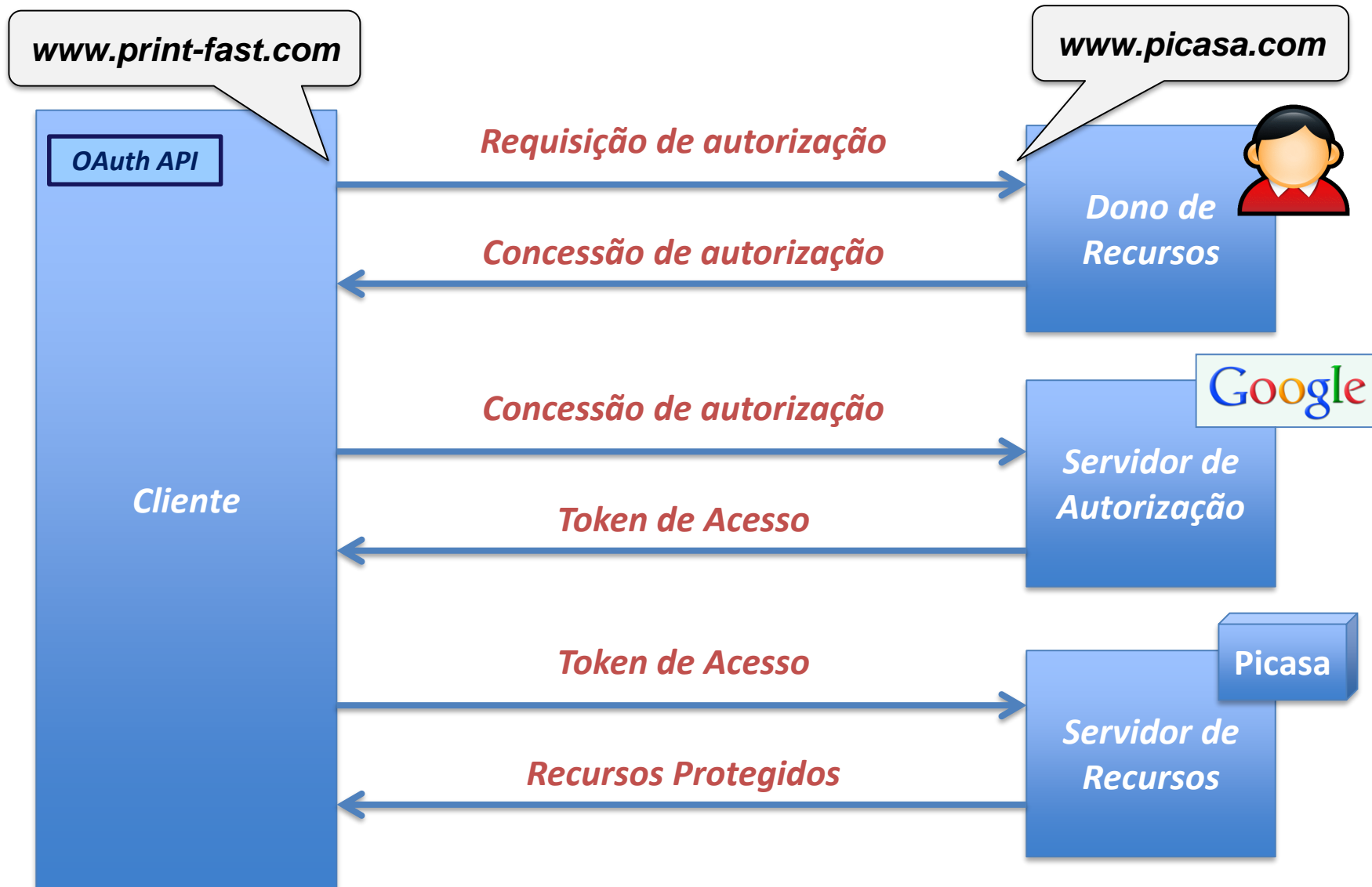
- Passo 3: Acessar recursos protegidos



# Protocolo OAuth: funcionamento



# Protocolo OAuth: funcionamento





# Protocolo OAuth: funcionamento

- Outros fluxos possíveis

- Token de acesso pode expirar ou ser revogado com o tempo

- Erro ao acessar recurso com token inválido (expirado/revogado)

- **Refresh token** pode ser obtido junto com token de acesso

- Depois, Passo 2 pode ser executado sem passar por Passo 1

- **Concessão com credenciais**

- Cliente pode usar credenciais do usuário diretamente

- Ex.: **Cliente** é uma **aplicação no desktop do Dono do Recurso**, o qual fornece sua senha para acessar os recursos

- Ex.: **Cliente é o próprio Dono dos Recursos** (comunicação entre máquinas -- *machine-to-machine*, ou M2M)

- Nesse caso, Servidor de Autorização pode ser contatado diretamente para obtenção do token de acesso (passo 2)

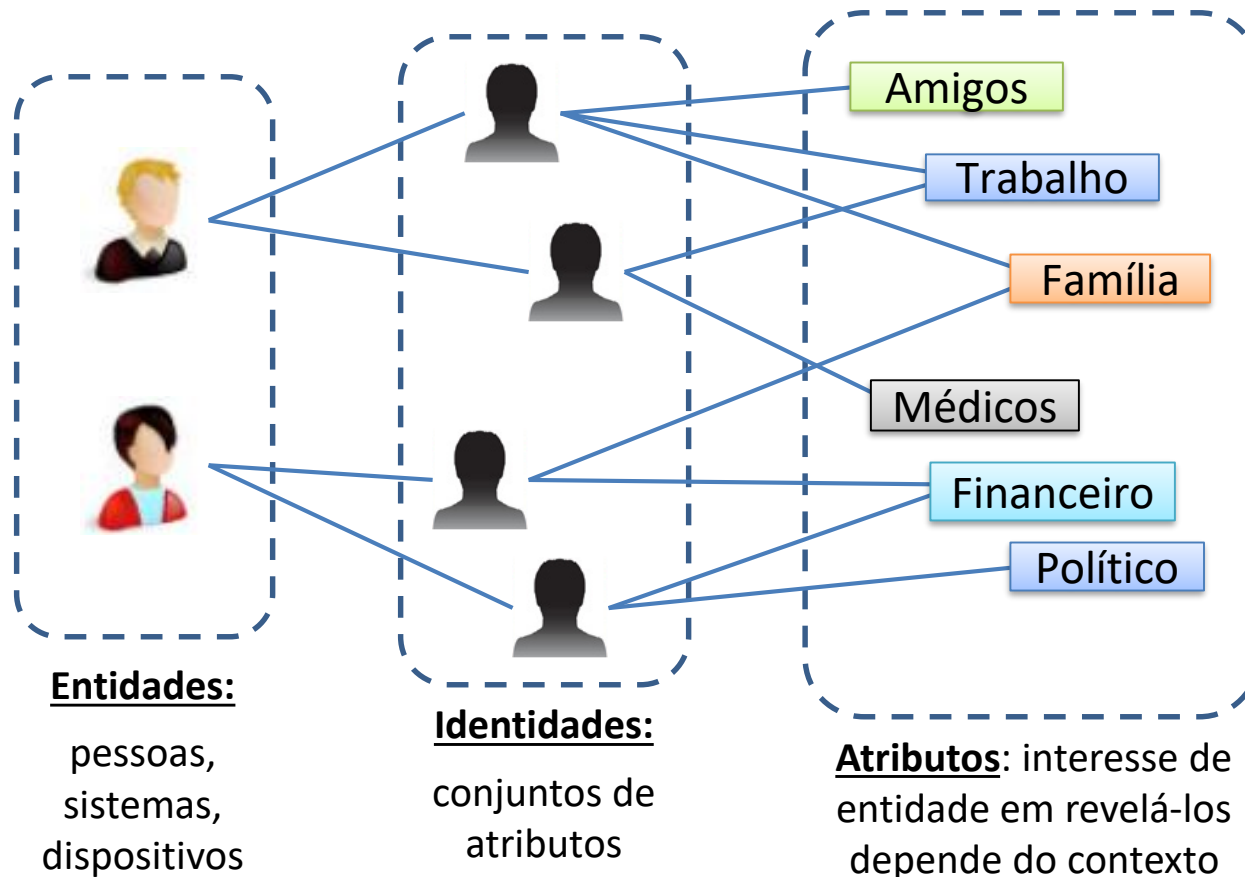
# Oauth + Identidades: OIDC

- **OpenID Connect (OIDC):** <https://openid.net/connect/>
  - Identidades Digitais Verificáveis
    - Conceito relacionado com “Identidade Federada”, “Identidade Descentralizada”, “Identidade Auto-soberana”, “Identidade Centrada no usuário”, ...



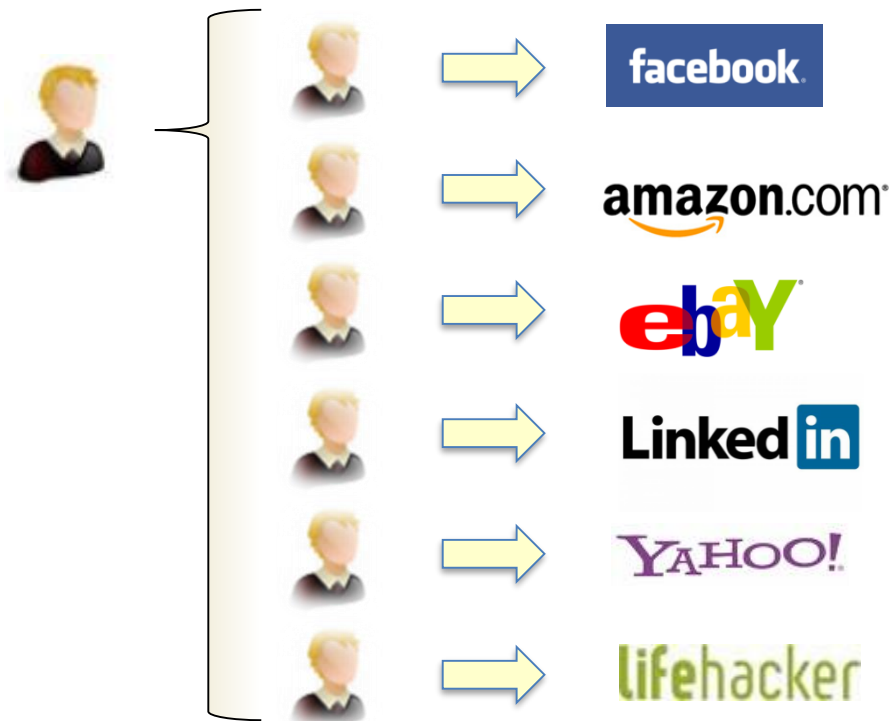
# Identities na Internet

- Gerenciamento de identidades: como identificar as várias entidades na web?



# Identidades na Internet

- Modelo básico de gestão de identidades na web:
  - Contas (identidades) independentes
  - Criação de uma conta diferente para cada serviço



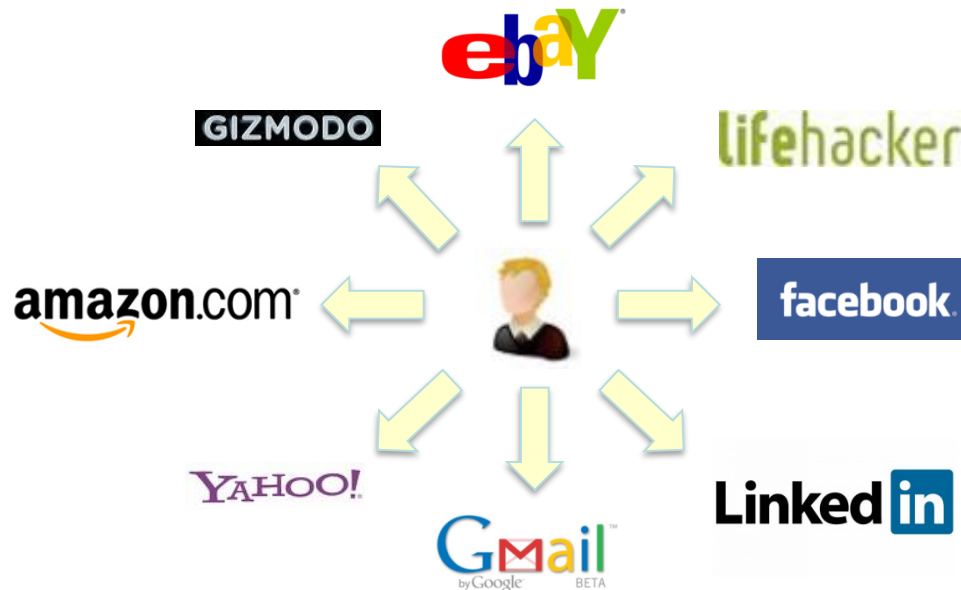
# Identidades na Internet

- Modelo básico de gestão de identidades na web:
  - **Centrado no Provedor de Serviço:** ele acaba sendo o “dono” das informações dos usuários...



# Identidades na Internet

- Gerenciamento de Identidades do **OpenID (Connect)**
  - **Modelo centrado no usuário:** ele quem **escolhe o(s) identificador(es)** usados em diferentes serviços
    - E **pode escolher quais atributos** de suas identidades fornecer aos serviços: **não é preciso criar novas contas...**
    - Permite *single sign on (SSO)*: um **único login** na navegação



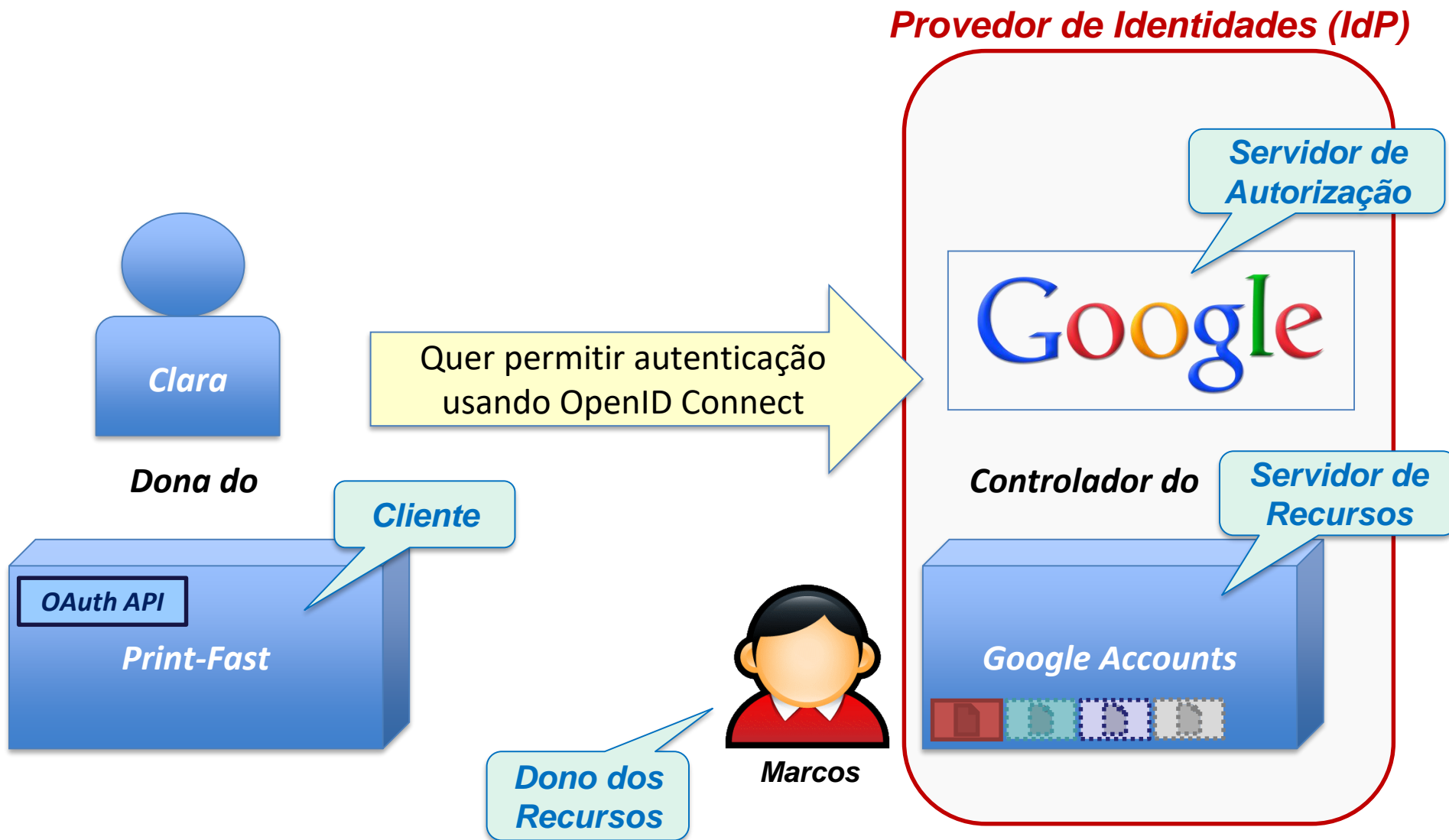
# OpenID Connect



- Projeto da OpenID Foundation (<http://openid.net/>)
  - Mas bem diferente do OpenID original...
- Em suma: **camada de identidade sobre OAuth 2.0**
  - Requisição de Autorização (passo 1) inclui como **escopo** “**openid**” (mandatório)
    - Outros escopos do padrão OpenID Connect (opcionais): “profile”, “email”, “address”, “phone”
  - Client recebe **ID token**
    - Além de poder receber tokens de acesso na mesma requisição
  - Mecanismo por trás do “Entrar com Google”
  - Bastante simples e leve!

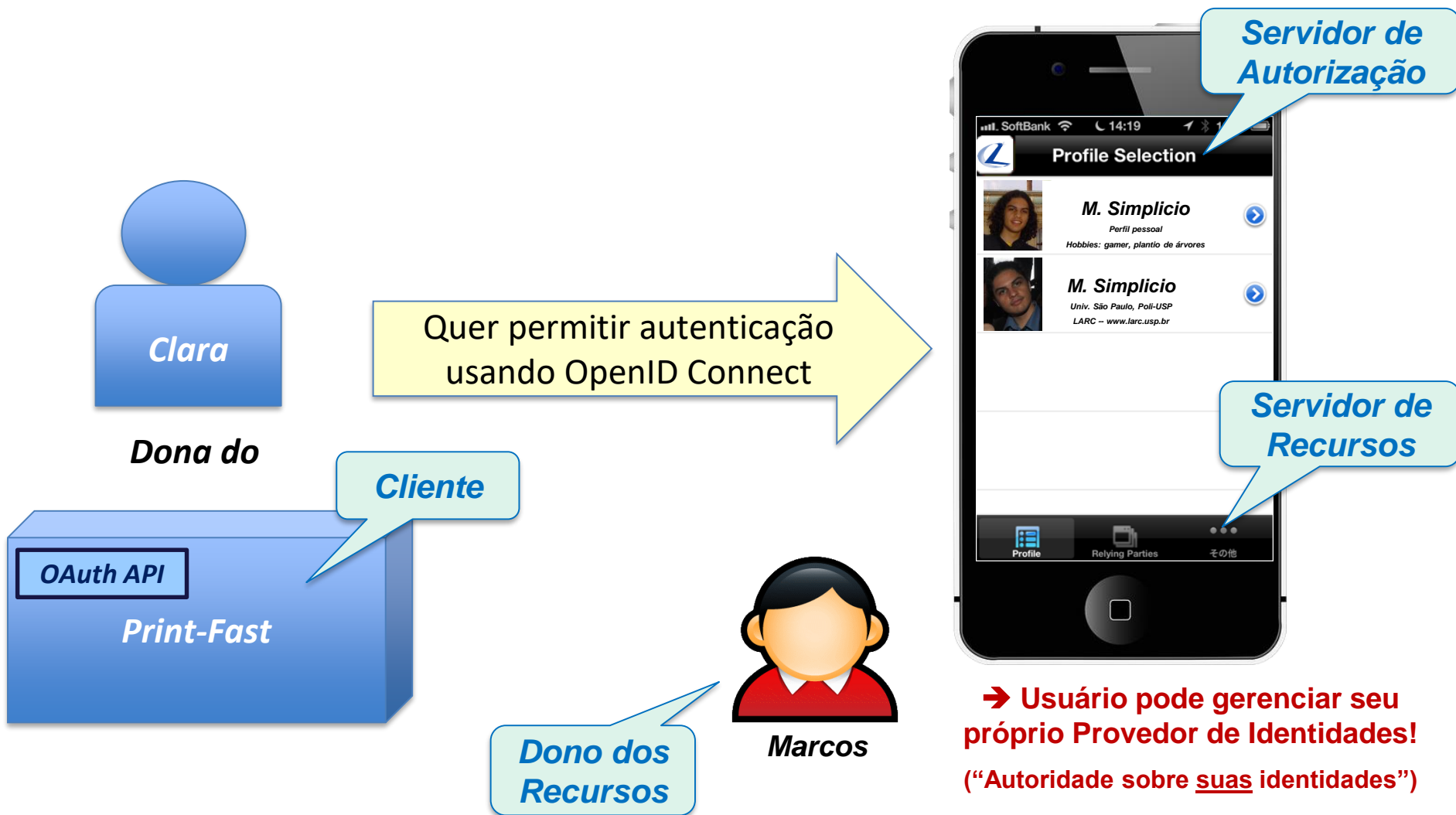


# OpenID Connect: Cenário





# OpenID Connect: Cenário



# OpenID Connect: funcionamento

- Passo 1: Obter código de autorização

## *Requisição de autorização*



## *Concessão de autorização*



**URL de redirecionamento:** `https://msimplicio-idp.com.br/?client_id=saintpeterhospital&scope=openid,email,profile &redirect_uri=https://saintpeter.com.br&response_type=code`

O restante do fluxo é igual ao do OAuth, mas recebe-se no final um **ID token**

# OpenID Connect: ID token

- Objeto no formato JWT (JSON Web Token): representação compacta
  - **iss:** autoridade que emitiu identificador → "https://msimplicio-idp.com.br"
  - **sub:** identificador do usuário → "marcos.simplicio"
  - **aud:** p/ quem token foi gerado → "https://saintpeter.com.br"
  - **iat** e **exp:** data de emissão e expiração → 1311280969, 1311281969
  - **contém assinatura e pode ser cifrado** se desejado

Identidade do usuário naquele  
provedor: descentralizado!



# OpenID Connect: OpenID4VC

- **Credenciais verificáveis:** privacidade e portabilidade de identidades e atributos associados



- Facilita relações de confiança entre diferentes domínios
- Caveat: **verificabilidade** da identidade/atributos depende da **confiança** que verificador atribui a **emissor das credenciais**!

- **Usuários no controle:**

- Decidem **a quem/quando apresentar suas credenciais**
  - Não é possível prevenir repasse de dados, mas previne-se personificação
- **Escolhem os emissores** dos quais obter suas credenciais
  - Podem ser auto-emitidas!
- Podem apresentar suas credenciais **sem interação com emissor**; emissor e verificador interagem sem necessidade de relação de confiança prévia





# Blockchain, Criptomoedas & Tecnologias Descentralizadas

## Tecnologias descentralizadas: Delegação de acesso e Identidades Federadas

Prof. Dr. Marcos A. Simplicio Jr. – [mjunior@larc.usp.br](mailto:mjunior@larc.usp.br)  
Escola Politécnica, Universidade de São Paulo

# Referências

- OAuth 2.0 (online). URL: <http://oauth.net/2>
- Welcome to OpenID Connect (online). URL: <https://openid.net/connect/>
- K. Yasuda, T. Lodderstedt, D. Chadwick, K. Nakamura, J. Vercammen (2022). "OpenID for Verifiable Credentials - A Shift in the Trust Model Brought by Verifiable Credentials". OpenID URL: <https://openid.net/2022/05/12/openid-for-verifiable-credentials-whitepaper/>
- D. Hardt (2012) "The OAuth 2.0 Authorization Framework". Internet Engineering Task Force (IETF), RFC 6749, ISSN 2070-1721. URL: <https://www.rfc-editor.org/rfc/rfc6749.html>
- M. Jones, J. Bradley, N. Sakimura (2015) "JSON Web Token (JWT)". Internet Engineering Task Force (IETF), RFC 7519, ISSN 2070-1721. URL: <https://www.rfc-editor.org/rfc/rfc7519.html>
- T. Spencer (2013) "OAuth and OpenID Connect Deep Dive". Youtube. URL: <https://www.youtube.com/watch?v=XGmUlyggXVo>
- R. Ghatol (2015) "OAuth 2.0 in Depth". SlideShare. URL: <https://slideplayer.com/slide/6119467/>

# OAuth: teste você mesmo @

- Cliente e servidor simples para testes:
  - Cliente: <http://term.ie/oauth/example/client.php>
  - Servidor: <http://term.ie/oauth/example/index.php>
- OAuth playground do Google
  - <https://developers.google.com/oauthplayground/>
    - Teste, por exemplo: Google Calendar API com a sua conta; execute operações como “List CalendarList” e “List Events”