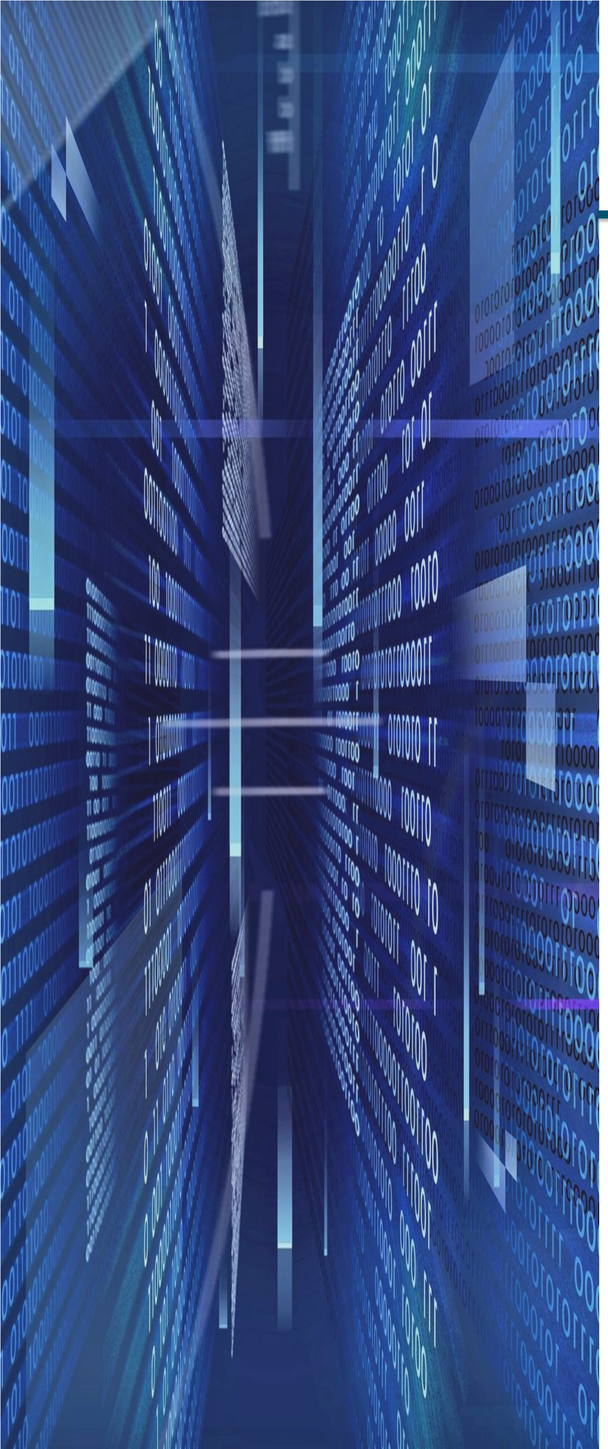


Access Control

Prof. Federica Paci

Lecture Outline

- Access Control Overview
  - What is Access Control
  - Access Control Elements
  - Discretionary Access Control
  - Role-based Access Control
  - Attribute-based Access Control
  - OAuth authorization flows

elementi dell'access controll

Learning outcomes

- At the end of this session, you should be able to:
  - Provide a definition of discretionary access control models
  - Provide a definition of role-based access control models
  - Provide a definition of attribute-based access control models
  - Discuss the limitations of discretionary access control models
  - Discuss the limitations of role-based of access control models
  - Discuss the limitations of attribute-based of access control models
  - Provide examples of threats to access control systems

What is Access Control

- Central element of cyber security
- The prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner
- Involves users and groups
  - authenticate to system
  - assigned access rights to certain resources on system

protegge le risorse di un'organizzazione
- Da l'accesso non consentito
- Da abuso di permessi

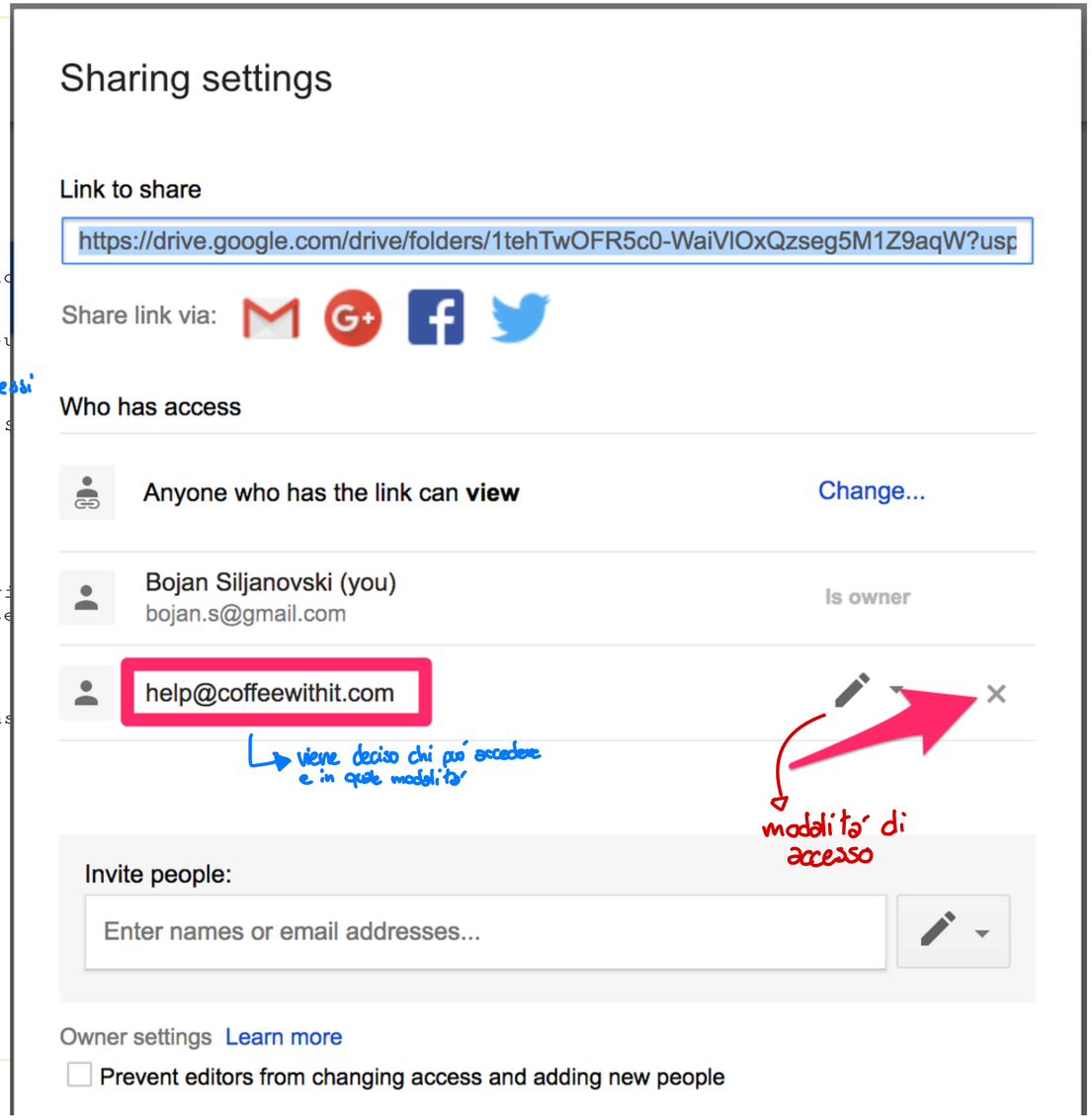
Access Control in Use

sono implementazioni dell'access controll

accesso fisico

```php
<?php
Class Acl {

   private $db;
private $user_empty = false;

   //initialize the database object here
   function __construct() {
     $this->db = new db;
   }

   function check($permission,$userid,$group_id

     //we check the user permissions first
     If(!$this->user_permissions($permission,$u
        return false;
     }

     if(!$this->group_permissions($permission,$
        return false;
     }

     return true;

   }

   function user_permissions($permission,$useri
$this->db->q("SELECT COUNT(*) AS count FROM use

$f = $this->db->f();

     If($f['count']>0) {
   $this->db->q("SELECT * FROM user_permissions

   $f = $this->db->f();

   If($f['permission_type']==0) {
     return false;
   }

   return true;

     }
     $this->setUserEmpty('true');

     return true;

   }
```

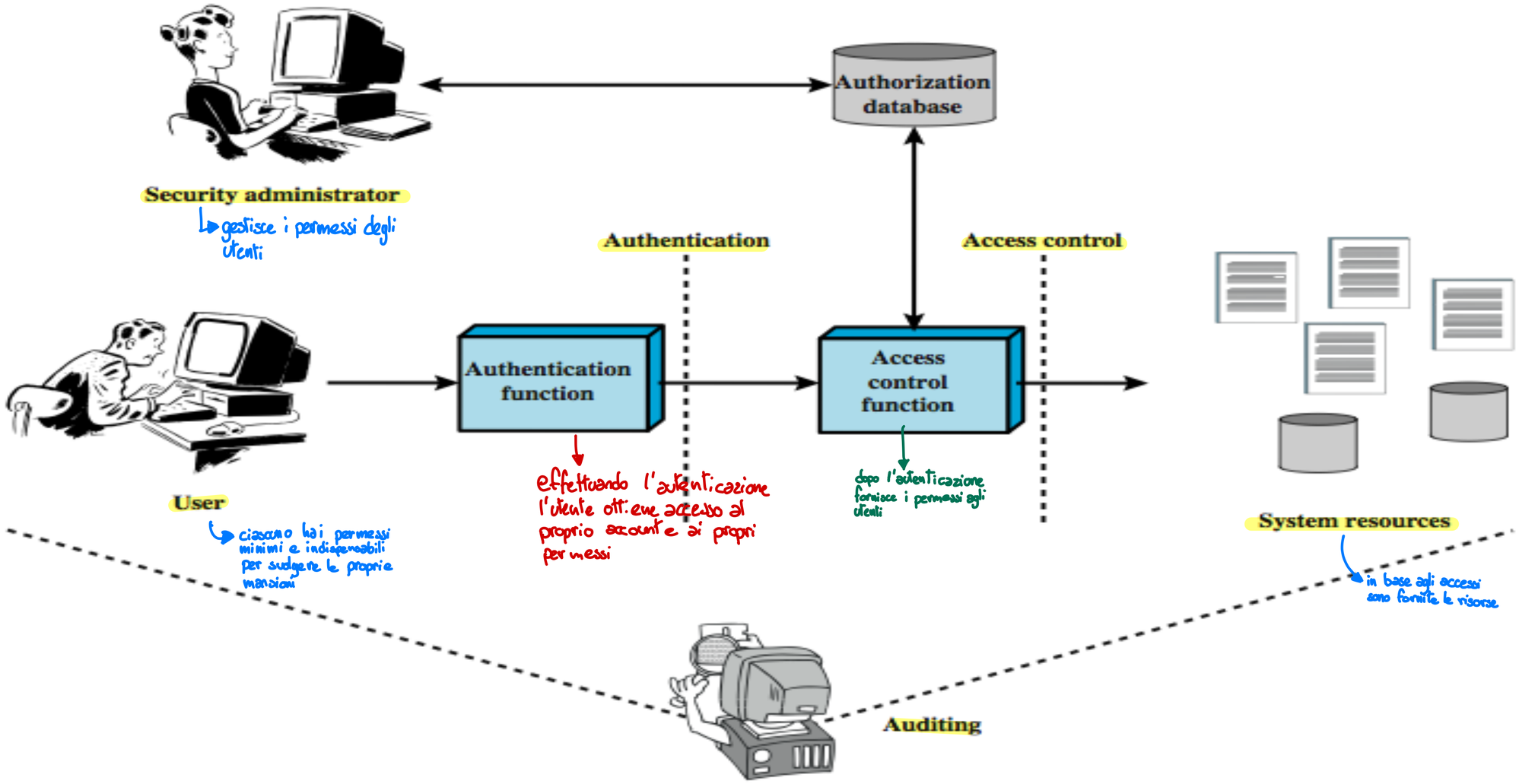
presenti nel codice dove viene controllato se l'utente ha i permessi per eseguire un azione



viene deciso chi può accedere e in quale modalità

modalità di accesso

Access Control Overview

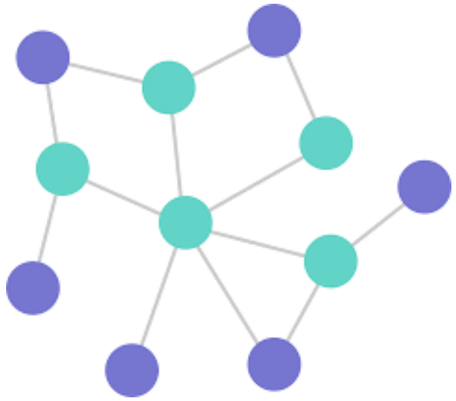


Access Control Principles

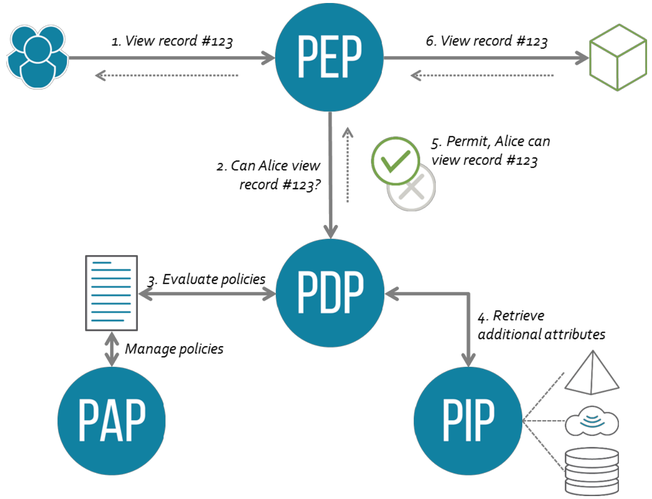
- Authentication
  - The verification an identity claimed by or for a system entity
- Authorization
  - The granting of a right or permission to a system entity to access a system resource
- Accountability → fondamentale, nessuno deve poter negare di aver svolto un azione in realtà fatta da lui
  - The monitoring and processing of user accesses to system resources

Access Control System

Access Control Policies



Access Control Model



Access Control Mechanism

Access Control Policy Elements

Se non consideriamo il modello le access control policy specificano 3 elementi:

- **subject** - entity that can access objects
  - a process representing user/application
- **object** - access controlled resource
  - e.g. files, directories, records, programs etc
- **access right** (or **permission**) - way in which subject accesses an object
  - e.g. read, write, execute, delete, create, search

Types of Access Control Models

Ho 4 modelli principalmente

- Discretionary Access Control (DAC)
  - Access based on the identity of the subjects
- Mandatory Access Control (MAC)
  - Access based on objects security labels and subjects security clearances
- Role based Access Control (RBAC)
  - Access based on the role played by a subject
- Attribute-based Access Control (ABAC)
  - Access is based on attributes of the subject, the object, and the context

Discretionary Access Control

- Access to data objects (files, directories, etc.) is permitted based on the identity of users.
- Explicit access rules that establish who can, or cannot, execute which actions on which resources.
- Discretionary: users can be given the ability of passing on their privileges to other users, where granting and revocation of privileges is regulated by an administrative policy
- often provided using an **access matrix**

Discrezionalità o delega dei diritti

Access Control Structures

Access Control Matrix

utenti sono le righe
risorse sono le colonne

Il problema è che non è scalabile

objects

subjects	news.doc	photo.png	fun.com
alice	read	view edit	view
bob	read write	view edit	view modify
charlie			
dave		view	

Access Control Structures

Access Control List

Stored with the object

per ogni risorsa
abbiamo utente e permesso associato
miglioriamo prestazioni perche salvo solo i permessi presenti
complico un po' perche se devo rimuovere un utente devo scansionare tutte le liste
Diventa difficile gestire i permessi di un utente

news.doc

bob	read write
alice	read

photo.png

alice	view, edit
bob	view, edit
dave	view

fun.com

alice	view
bob	view modify

Access Control Structures

- Capability List
  - Stored with the subject

rende più facile capire i permessi di un utente ed eventualmente gestirli
rende più semplice anche la delega

Alice's capability

news.doc	photo.png	fun.com
read	view, edit	view

Bob's capability

news.doc	photo.png	fun.com
read write	view, edit	view edit

Charlie's capability

Dave's capability

photo.png
view

Exercise 1

- Imagine you are John, a user of social network Famebook. Famebook adopts DAC as access control model. John has 253 friends on Famebook. Use access control list to specify the following access control policies:
  - John wants his friend Alice to have view access on his photo album ”Spanish Holidays”
  - John wants to share only with his wife his status ”Happy like the first day! Happy Anniversary my love!”
  - John wants to share with all his friends on Famebook a funny video of his cat

Time: 5 min

DAC Limitations

- Managing a policy is a complex task in a large system
  - Set of subjects or objects is large
  - Set of subjects or objects change frequently
- Capability Lists
  - Difficult to get an overview of permissions granted on a given object
- Access Control Lists
  - Difficult to get an overview of permissions granted to a given user

Role based access control (RBAC)

privilegi assegnati ai ruoli non agli utenti

- Widely adopted access control model
- Based on the role played by a user within an organization
- Roles are assigned access rights to resources
- Users are assigned to roles
  - Inherit access rights of the role they play

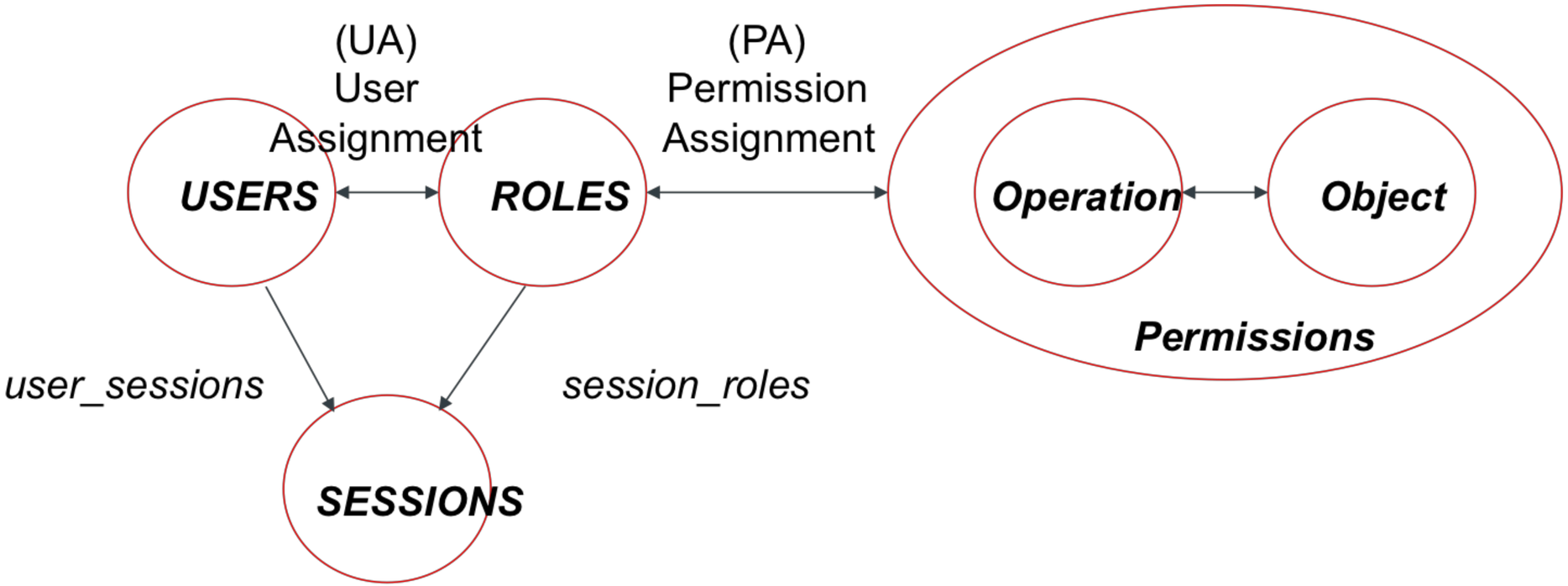
RBAC Family

- Defined by Ravi Sandhu in 1996
- ANSI standard since 2004
- $RBAC_0$
  - Core model: users, roles, permissions, sessions
- $RBAC_1$
  - $RBAC_0$ + Role Hierarchies
- $RBAC_2$
  - $RBAC_1$ + Constraints

3 modelli

RBAC Family

$RBAC_0$: Core RBAC



Un utente può avere uno o più ruoli

Un utente può essere assegnato a più ruoli e di conseguenza ho il concetto di sessione
ogni utente deve attivare il proprio ruolo e quindi avrà permessi diversi a seconda della sessione

Another simple example

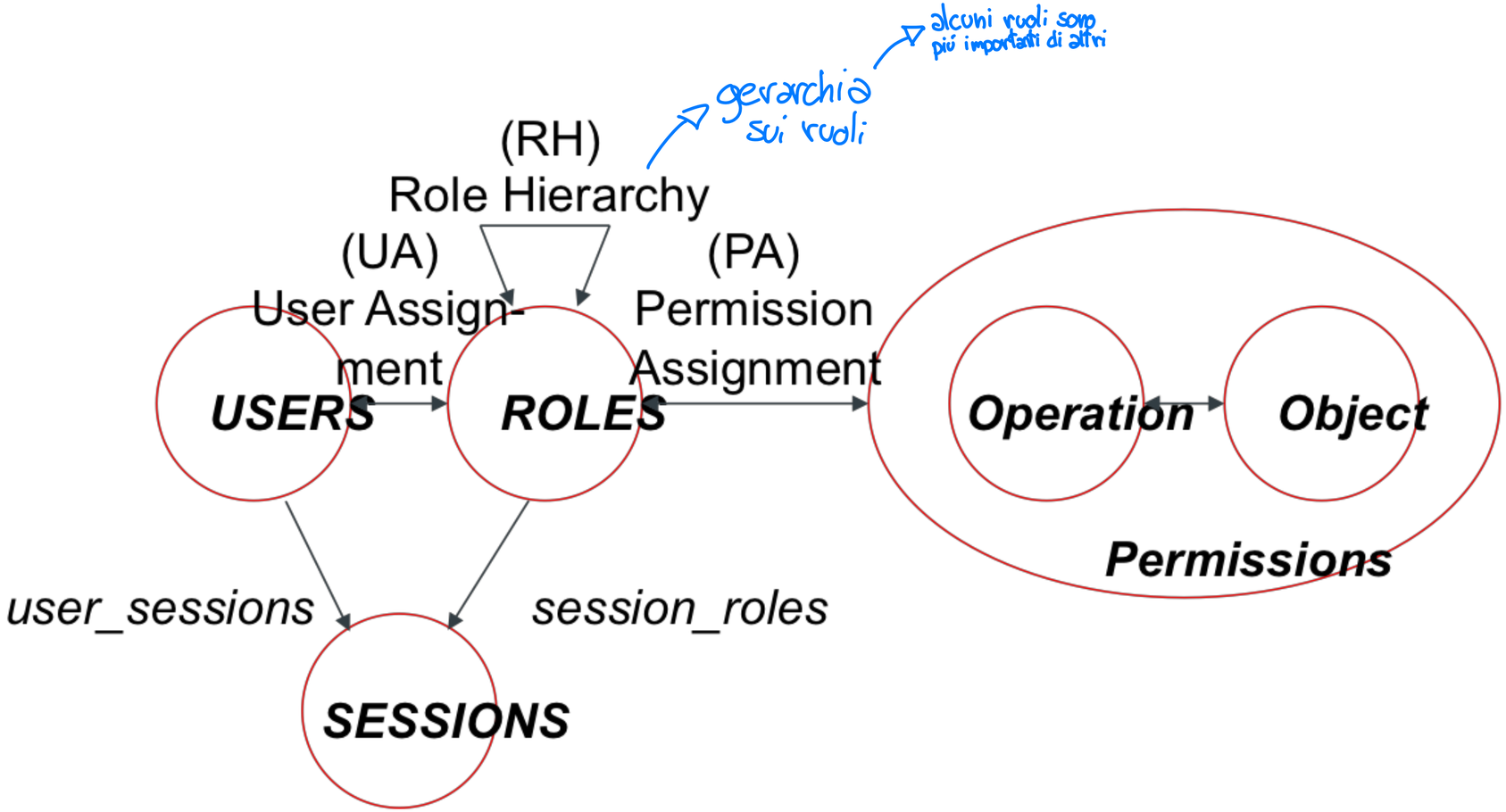
- Roles
  - Lecturer
  - PhD student
  - Associate Professor
  - Full Professor
  - Head of Department
- Objects
  - Exam paper
  - Coursework
  - Module Marks

Another simple example

- Lecturer
  - create exam paper, create coursework, mark exam paper, mark coursework, view and upload the module marks
- PhD student
  - view coursework description, mark a coursework
- Associate Professor
- Full Professor
- Head of Department

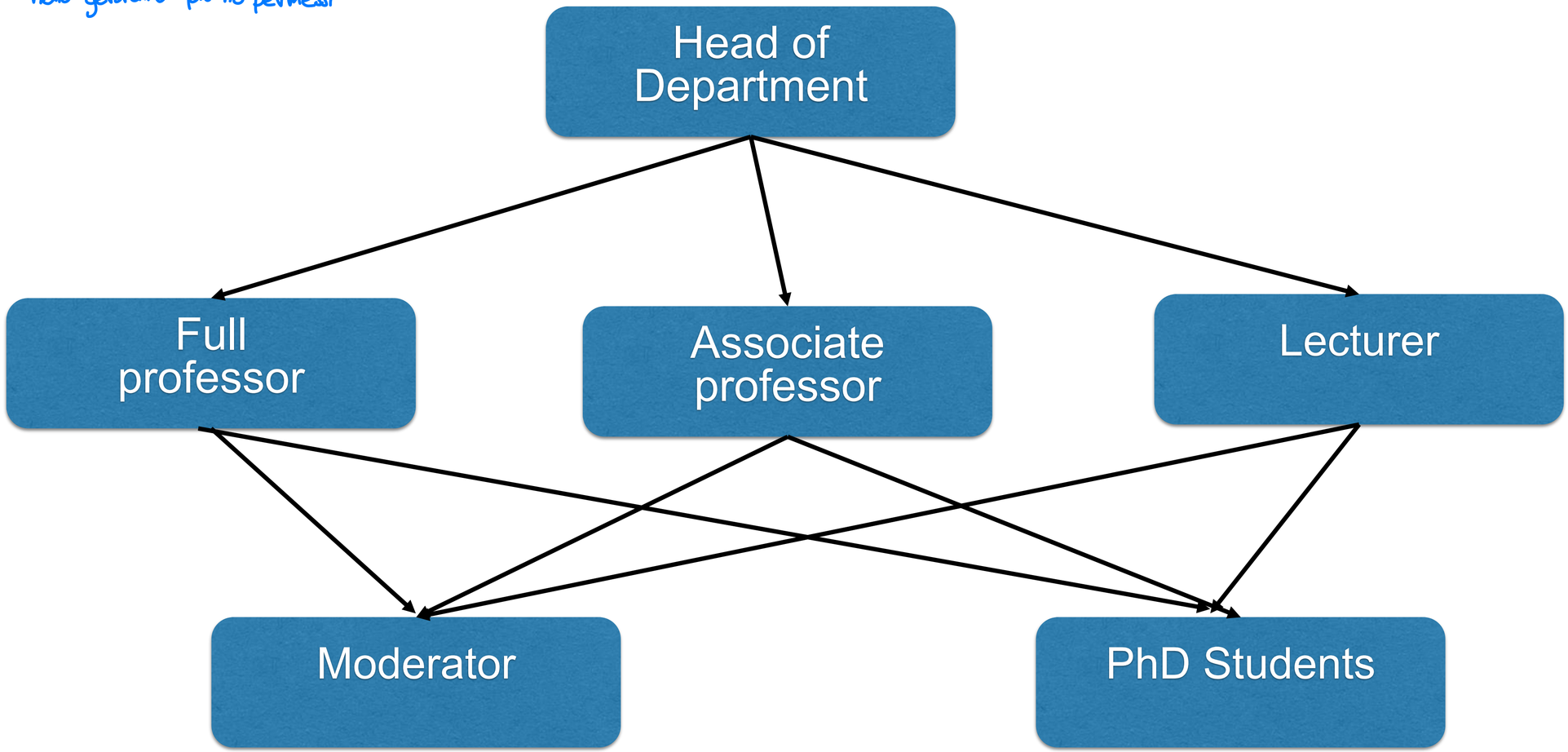
RBAC Family

$RBAC_1$: Core RBAC + Role Hierarchies



$RBAC_1$: Role Hierarchy

Tipicamente più si è in alto nella gerarchia più ho permessi

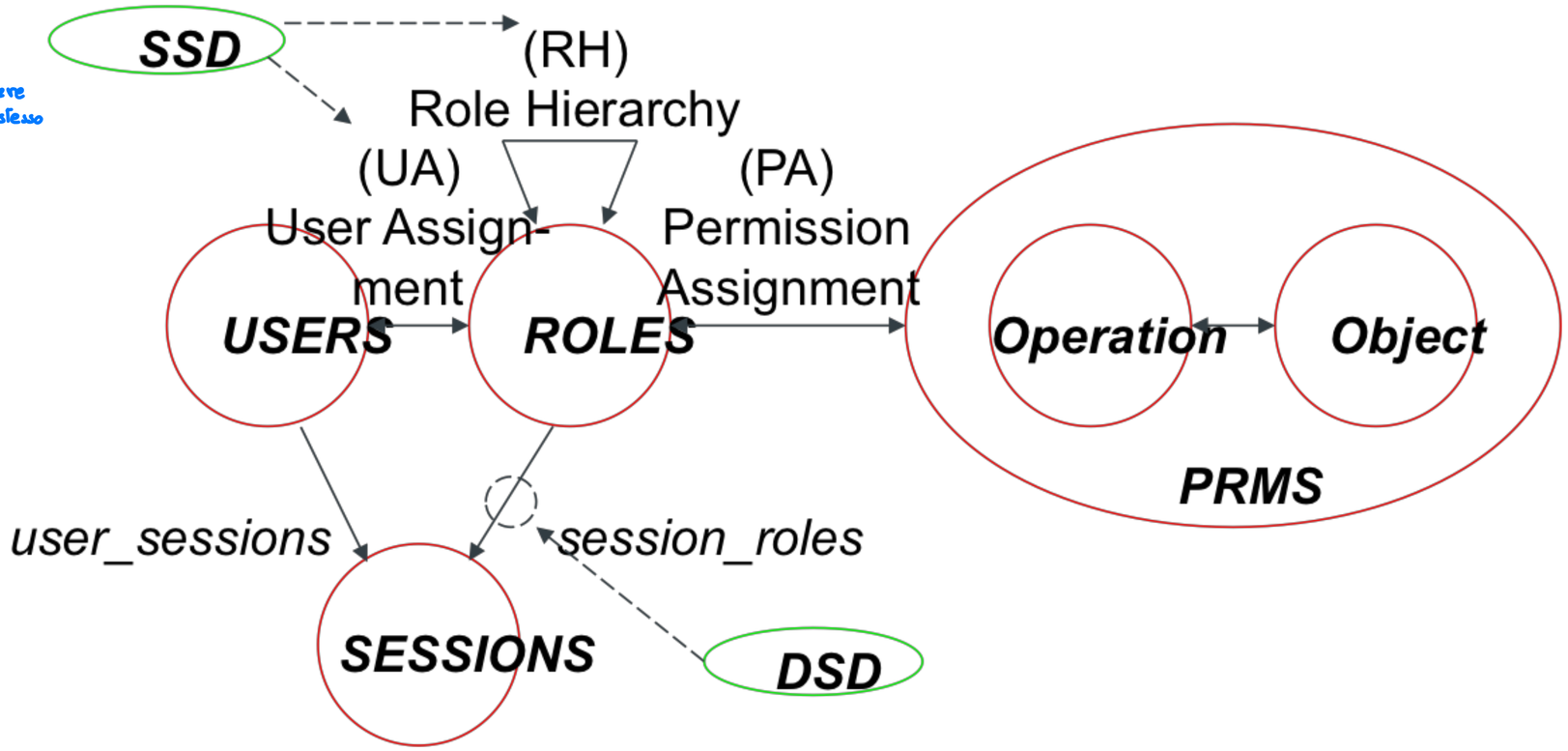


RBAC Family

$RBAC_2$:RBAC1 + Separation of Duty Constraints

Aggiunge Constraint ovvero dei vincoli

Ad esempio non posso essere docente e studente dello stesso corso



Separation of Duty Constraints

Static Separation of Duty

pair (role set, n)

a user cannot be assigned to more than n roles in the role set

e.g (student, professor) a user cannot be assigned to both the role of student and professor

Dynamic Separation of Duty

pair (role set, n)

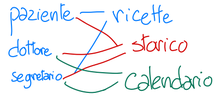
a user cannot activate more than n roles in the role set within the same session

e.g the user Federica cannot activate at the same time the role of professor and student for Foundations of Security and Privacy

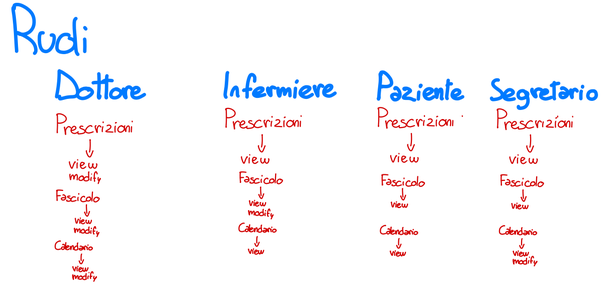
Exercise 2

Let’s imagine that John and Alice are two family doctors working for the clinic HappyHealth. Jane and Bob are two nurses working in the same clinic. Mary and Claire are two patients being treated at the same clinic. Jennifer and Marco work as secretary at the clinic.

The protected objects are the prescriptions, the health records of the patients, and the calendar with doctors' appointments.



Specify the role-based access control model that assigns to the users the permissions on the protected objects prescriptions, health records, and doctor’s calendar.



Time: 5 min

RBAC Advantages

- Efficient administering and monitoring of permissions
  - No need to manually assign users to permissions
  - Automatically done by assigning users to roles
- Reduce Employee Downtime

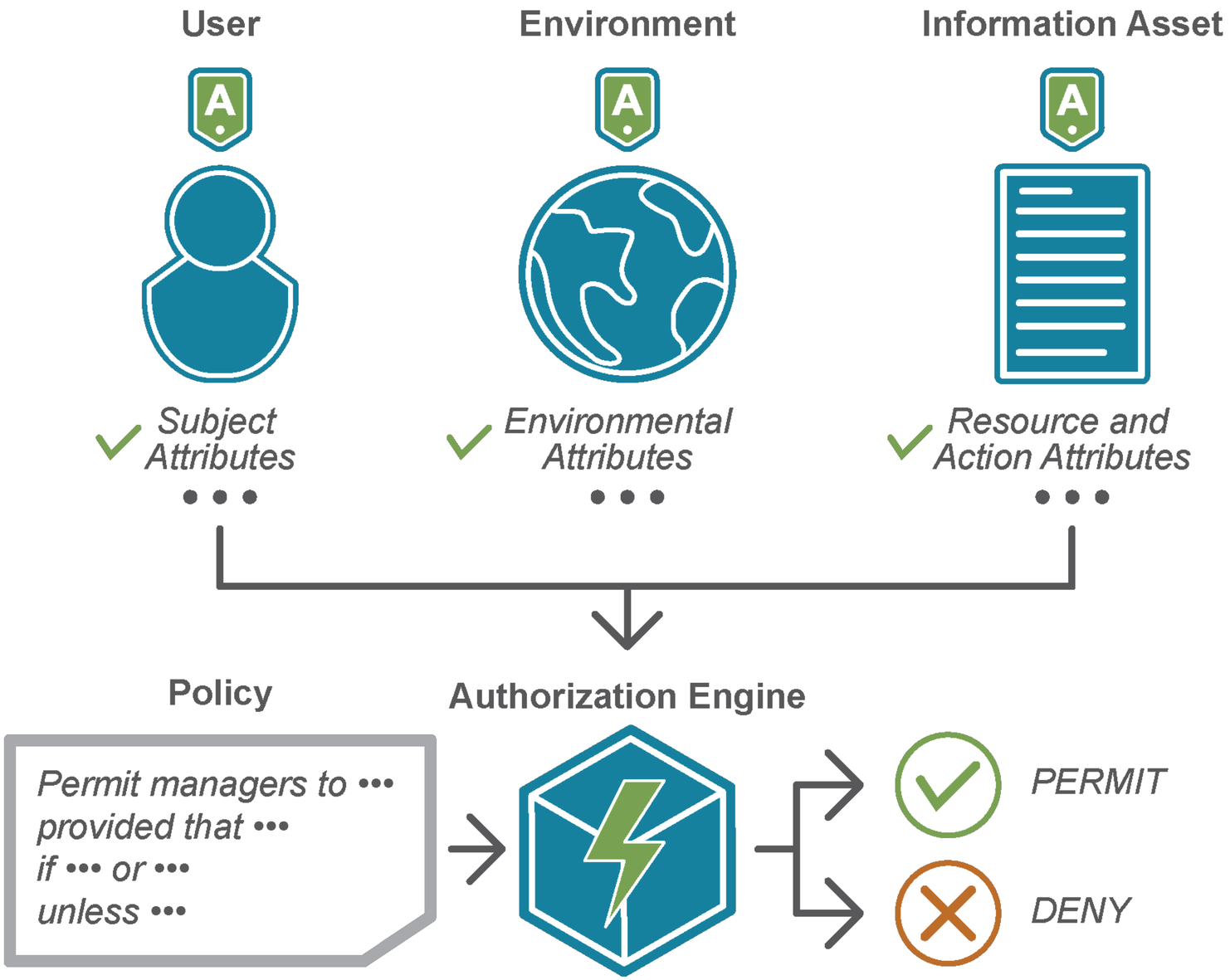
Threats to RBAC

- Access control is very hard to implement correctly
  - A European bank with over 50,000 employees, 1400 branches and more than 6 million customers
    - 1300 roles → Spesso viene creato un ruolo ad-hoc senza cercare un ruolo dove inserire l'utente
    - 1000 role changes in 3 months → diventa di difficile gestione
- Organizations over-entitle employees
  - 50 to 90% of employees are over-entitled in large organizations (Eric Johnson)

RBAC in Use

- Healthcare
  - NHS
- Banking and Finance
  - Banks, Insurance companies
- Commercial products
  - DBMS
- Cloud platforms
  - Microsoft Azure

Attribute Based Access Control



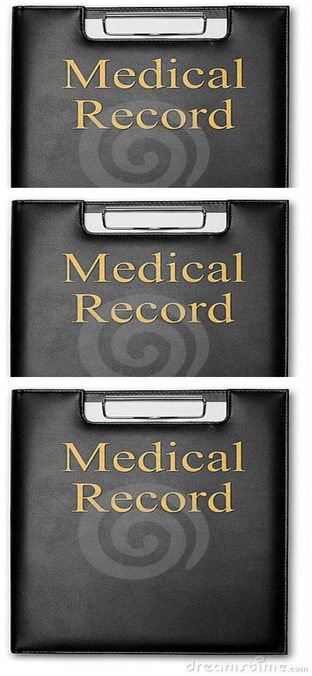
An Access Control Scenario

BART SIMPSON

BART's
PHISYCIAN

REGIONE
VENETO

HEALTHCARE
RECORDS
MANAGEMENT
SYSTEM



A Policy Example

Practitioners in the **Cardiology Department** can **View** the **Records** of **Heart Patients** using **Hospital Computer** within **Hospital Ward**

Resource Attributes

Patient Disease

Subject Attributes

Cardiology Department

Practitioner

Action Attributes

View

Environment Attributes

Hospital Ward

Hospital Computer

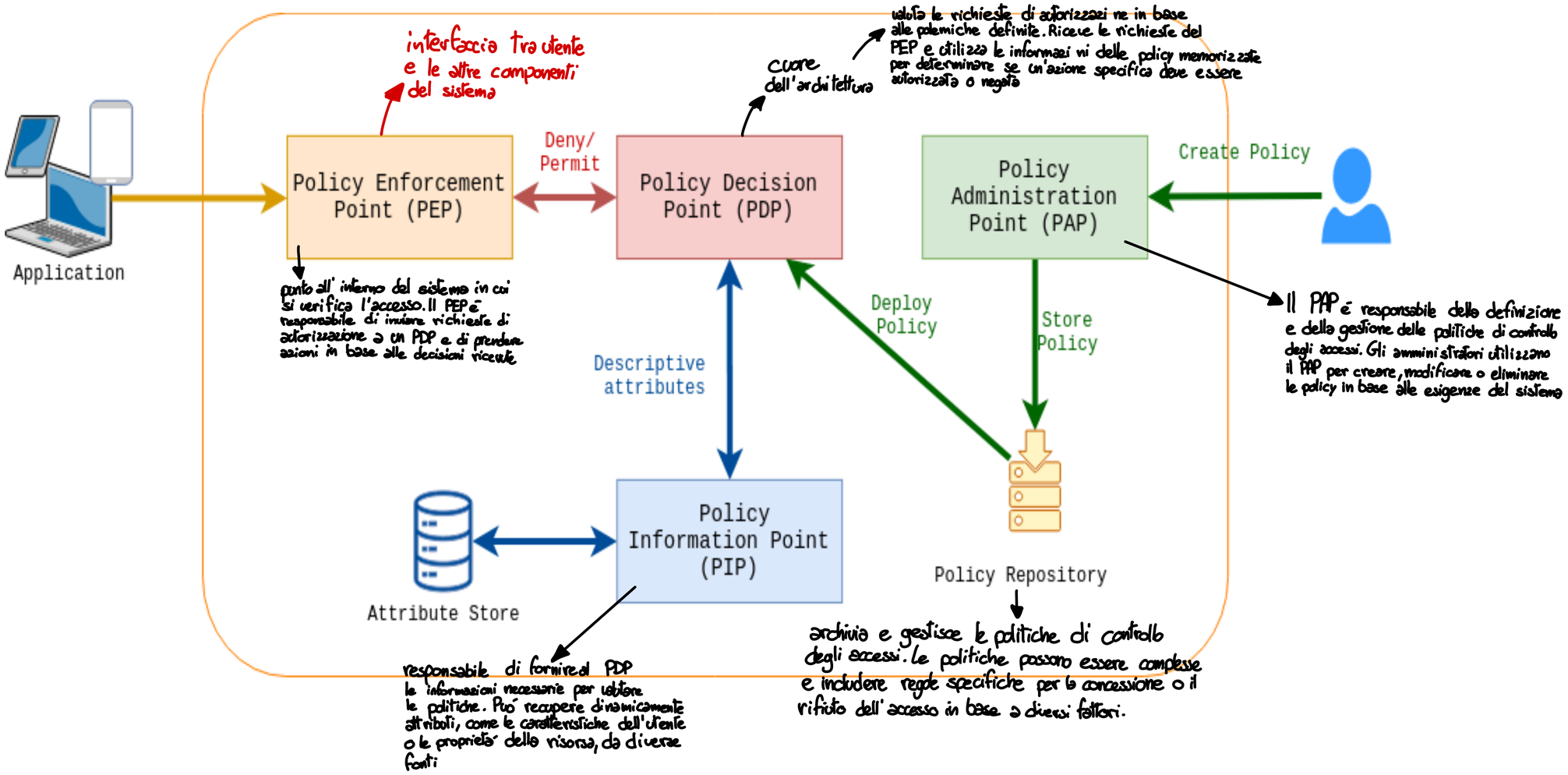
XACML —▷ protocollo standard basato su XML

- The eXtensible Access Control Markup Language is an OASIS Standard
- The XACML standard provides
  - XML-based Policy Language
  - XML-based Request and Response Language
  - Standard data-types, functions, combining algorithms
  - An architecture defining the major components in an implementation
  - Privacy profile, RBAC profile

• Protocollo standard utilizzato per la definizione di politiche di controllo degli accessi in ambienti informatici.
  └▶ utilizzato per implementare controlli di accesso basati su attributi

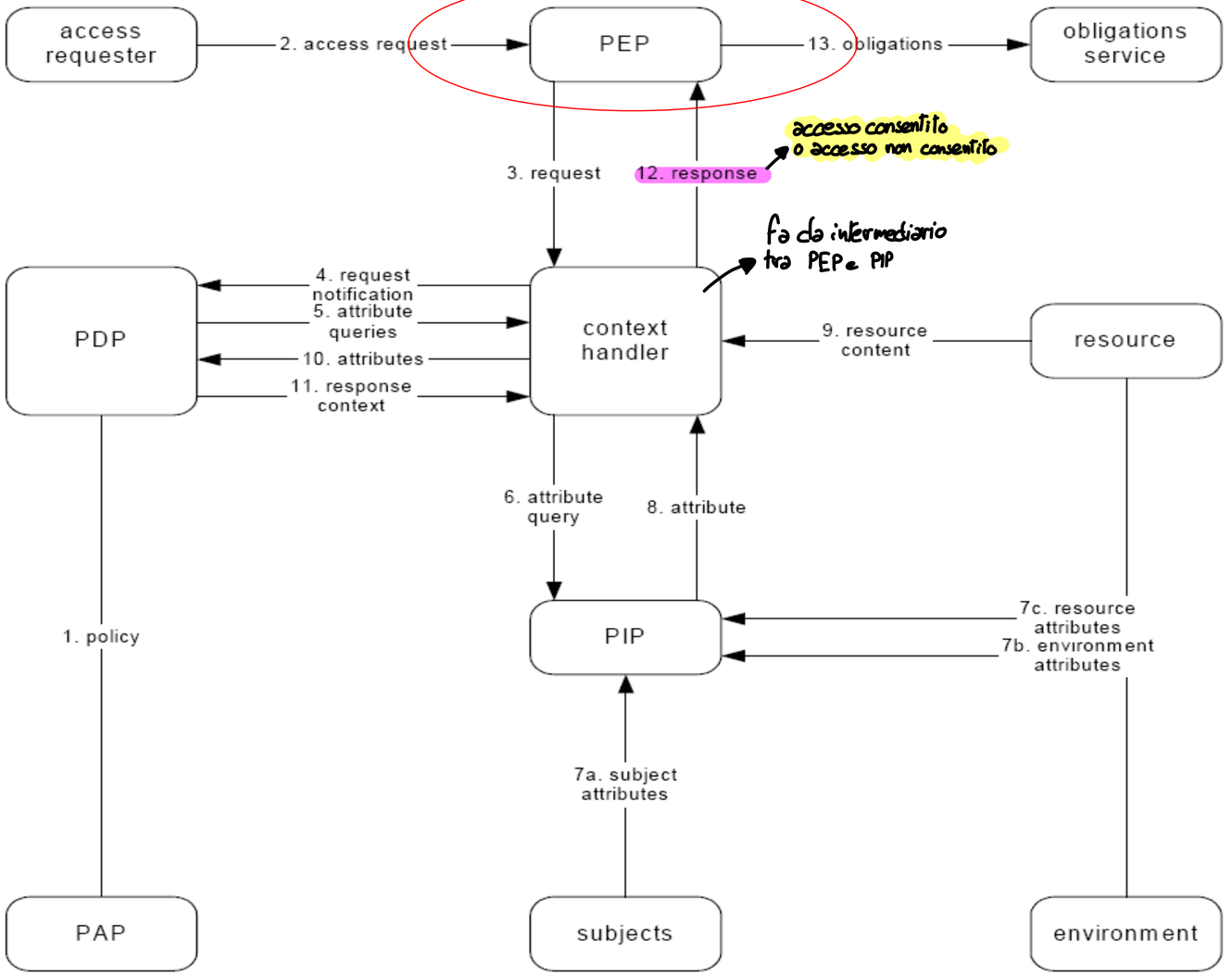
XACML Architecture

collaborano per la gestione delle politiche di controllo degli accessi



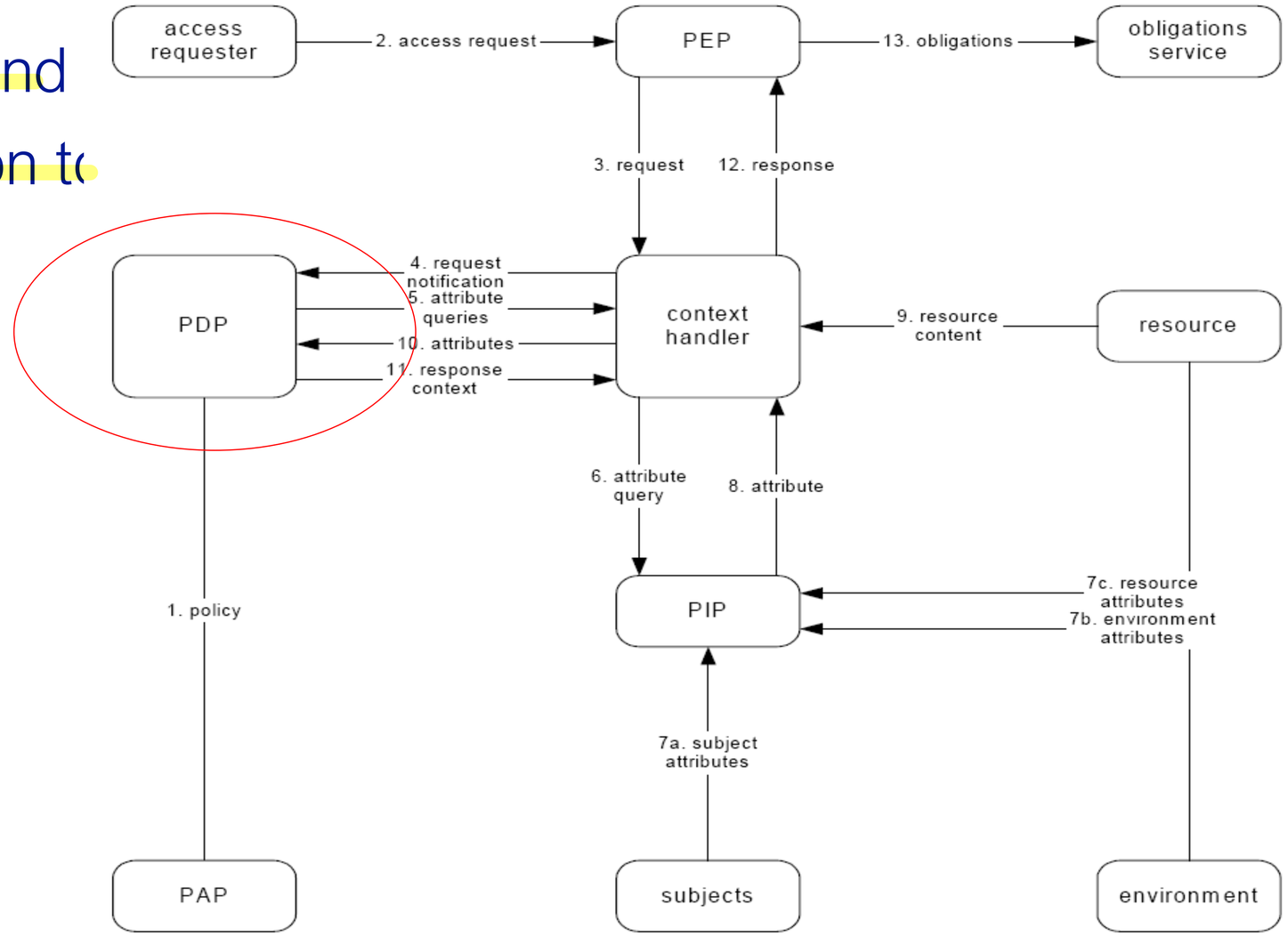
Policy Enforcement Point

- Entity protecting the resource (e.g. file system)
- Performs access control by making decision requests and enforcing authorization decisions and executing obligations



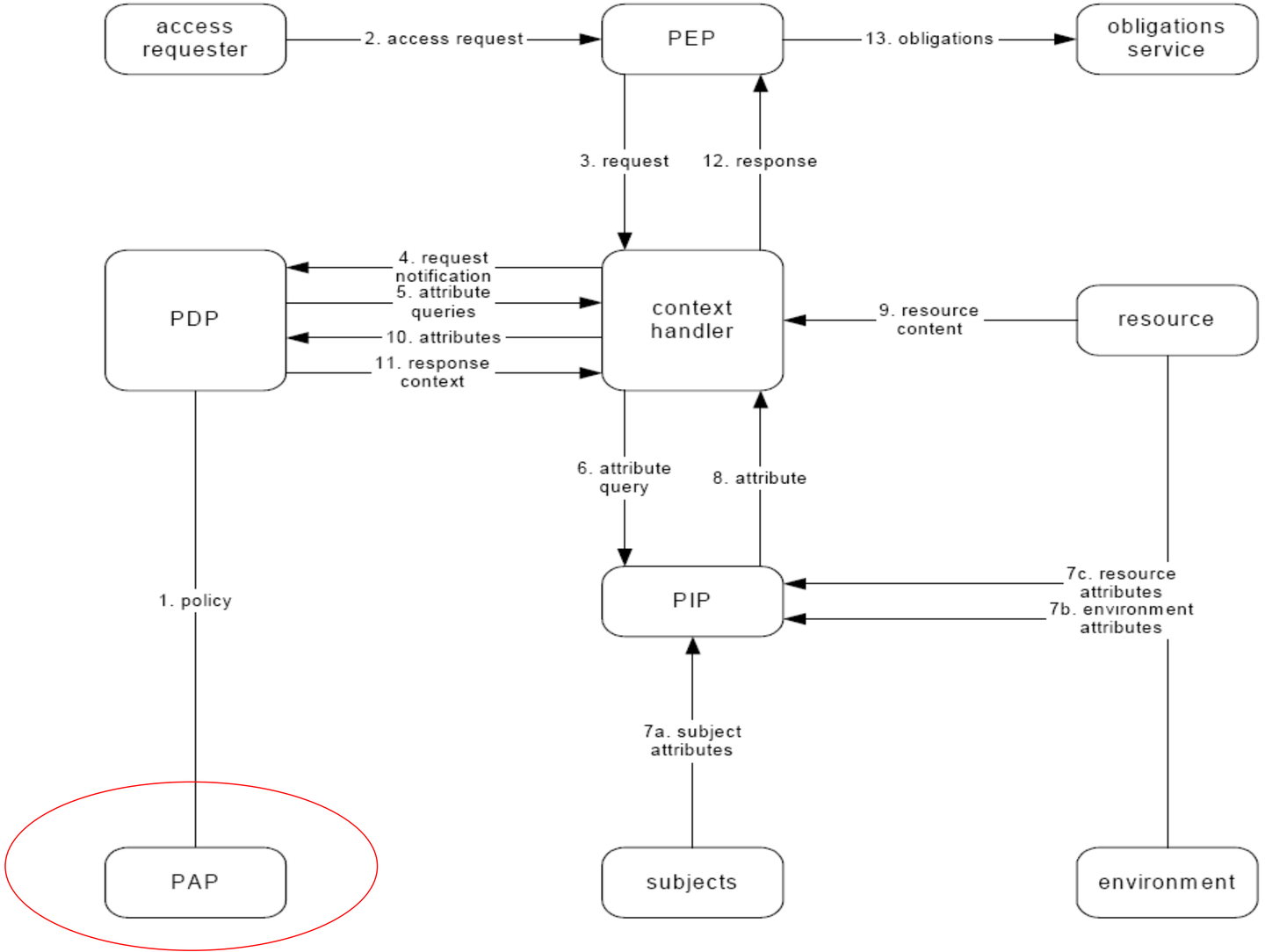
Policy Decision Point

- Receives and examines the request
- Retrieves applicable policies
- Evaluates the applicable policy and
- Returns the authorization decision t



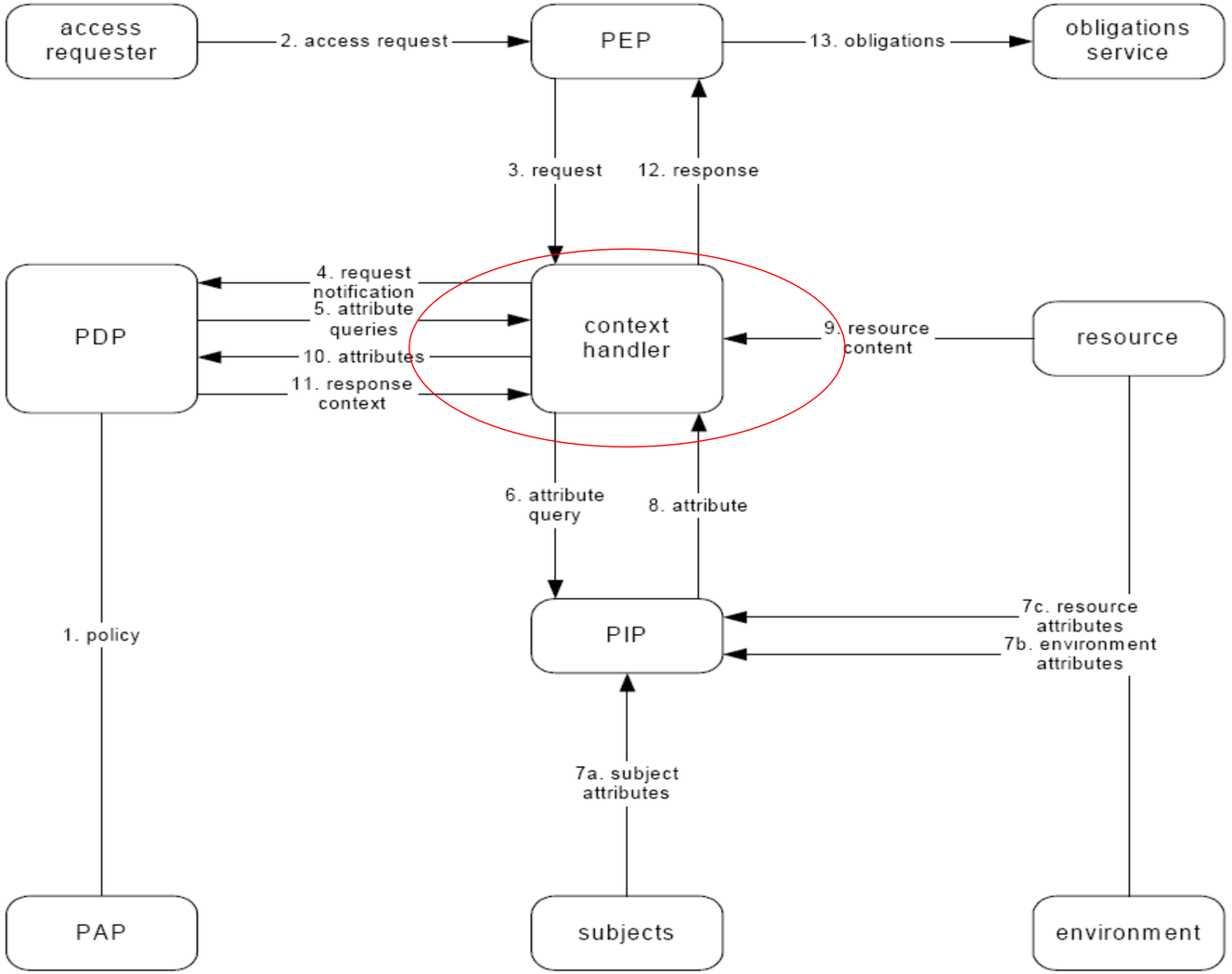
Policy Administration Point

- Creates security policies and stores these policies in the repository



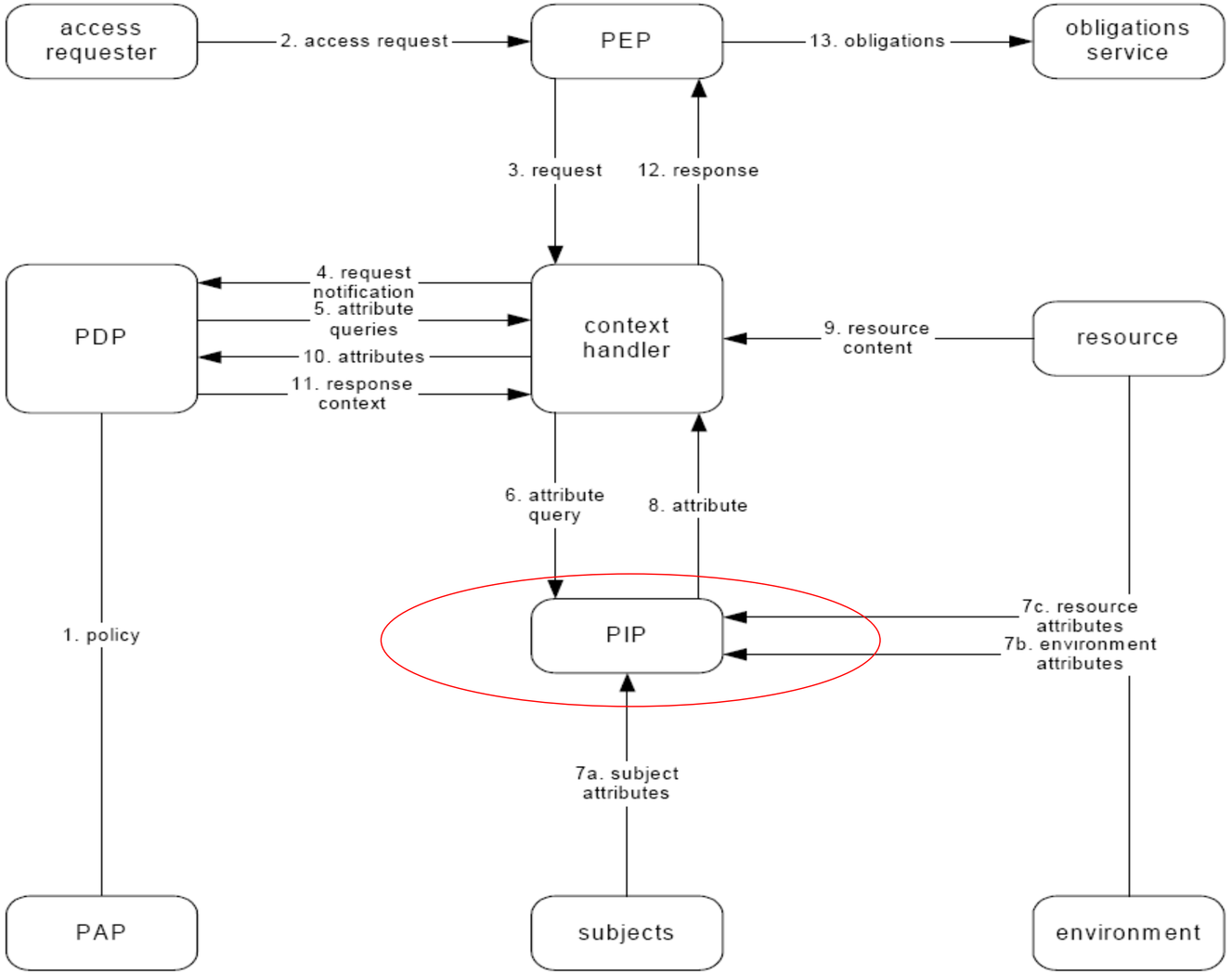
Context Handler

- It converts the requests in its native format to the XACML canonical form and to convert the authorization decisions in the XACML canonical form to the native format

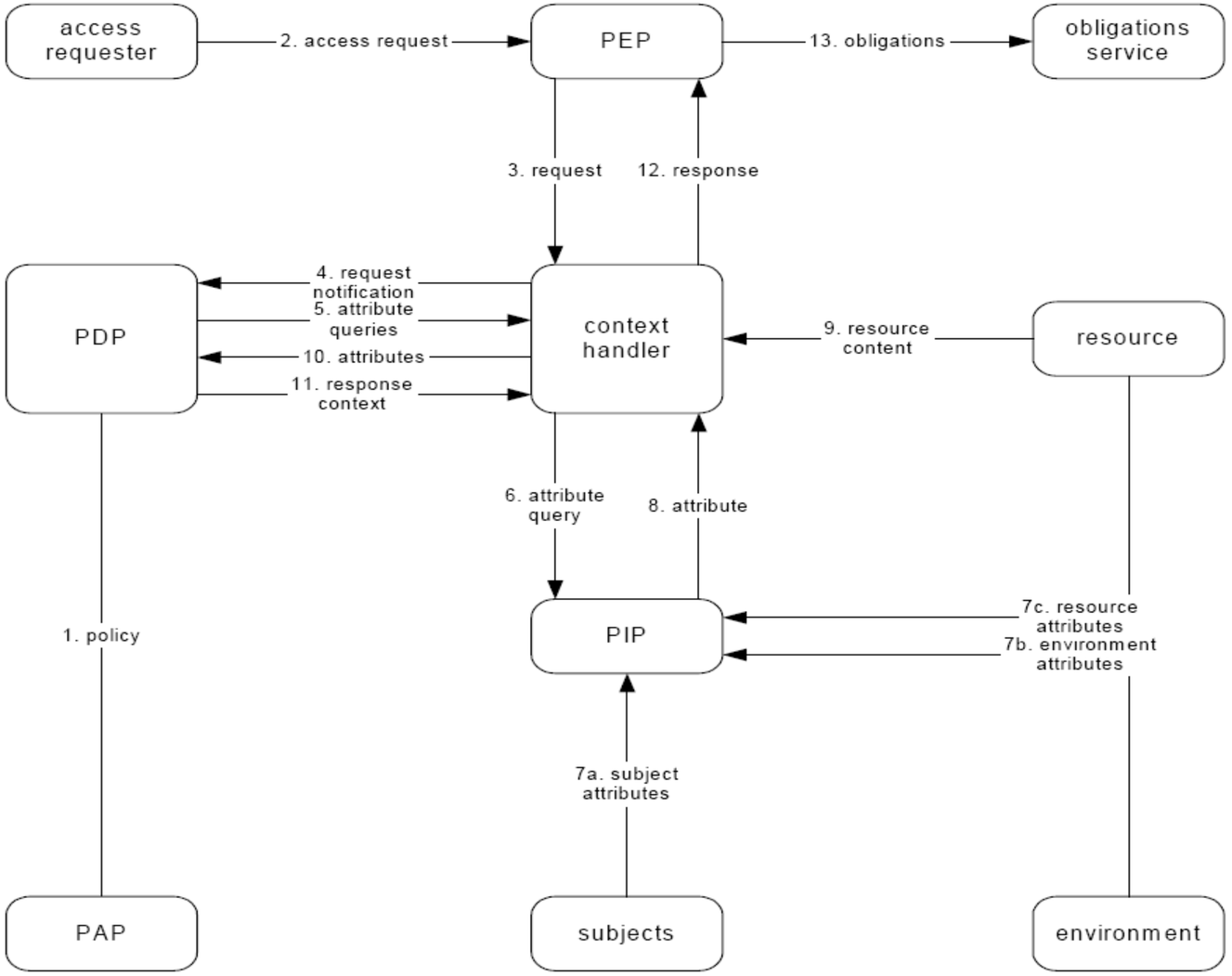


Policy Information Point

- serves as the source of attribute values, or the data required for policy evaluation



Authorization Flow



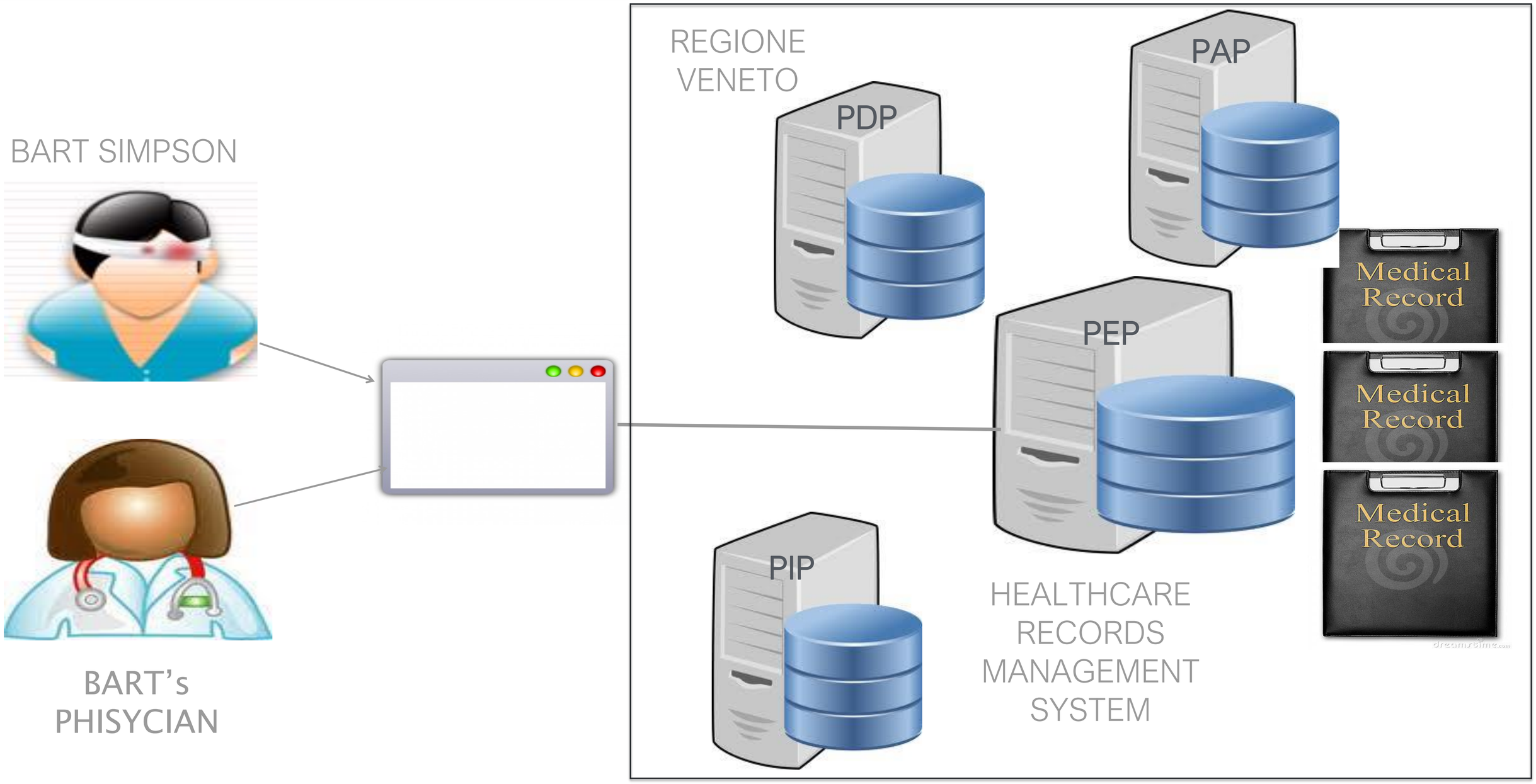
Data Flow Model

1. PAPs write **policies** and **policy sets** and make them available to the PDP.
2. The access requester sends a request for access to the PEP
3. The PEP sends the request for **access** to the context handler in its native request format, optionally including **attributes** of the **subjects**, **resource**, **action** and **environment**
4. The **context handler** constructs an XACML request **context** and send it to the PDP
5. The PDP requests any additional **subject**, **resource**, **action**, and **environment attributes** from the context handler
6. The context handler requests the attributes from a PIP
7. The PIP obtains the requested **attributes**
8. The PIP returns the requested **attributes** to the context handler

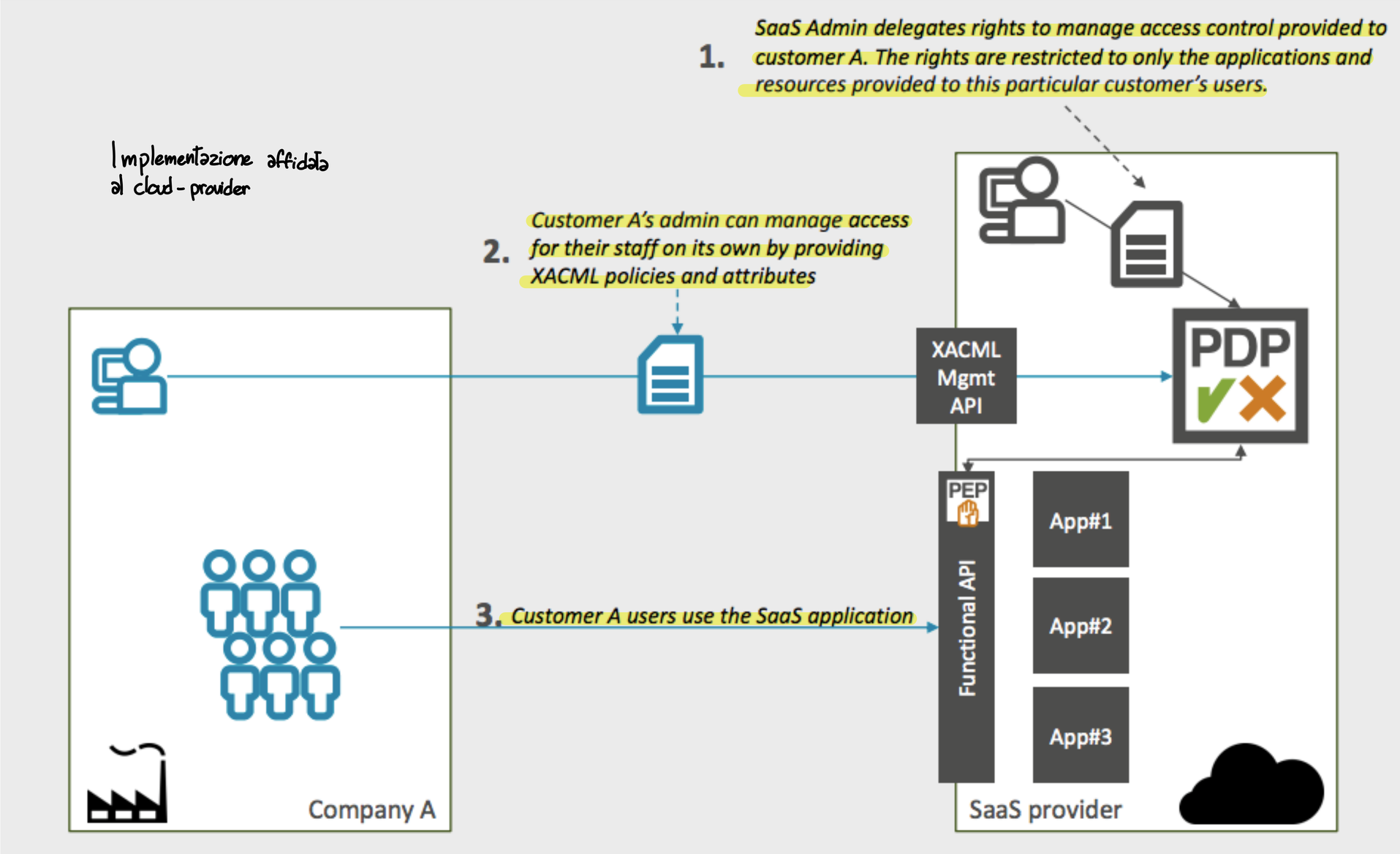
Data Flow Model

9. Optionally, the context handler includes the **resource** in the context
10. The context handler sends the requested **attributes** and (optionally) the resource to the PDP. The PDP evaluates the **policy**
11. The PDP returns the response **context** (including the **authorization decision**) to the context handler
12. The context handler translates the response context to the native response format of the PEP. The context handler returns the response to the PEP
13. The PEP fulfills the **obligations**
14. (Not shown) If **access** is permitted, then the PEP permits **access** to the **resource**; otherwise, it denies **access**

A Centralized Scenario



A Cloud Scenario



XACML language key components

```
<Policy>
   <Target>
      <Resources>
      <Subjects>
      <Actions>
   </Target>
   <RuleSet ruleCombiningAlgId = “DenyOverrides”>
      <Rule ruleId=“R1”>
      <Rule ruleId=“R2”>
        …
      <Obligations>
   <RuleSet>
</Policy>
```

Le regole devono essere sempre valutate al policy decision point

```
<Rule RuleId=“R2”
        Effect=“Deny”>
   <Target>
      <Resources>
      <Subjects>
      <Actions>
   <Condition>
</Rule>
```

```
<Rule RuleId=“R1”
        Effect=“Permit”>
   <Target>
      <Resources>
      <Subjects>
      <Actions>
   <Condition>
</Rule>
```

XACML Language Key Components

- <PolicySet > is the key top-level element which aggregates other <PolicySet> elements or <Policy> elements
- The <Policy> element is composed principally of <Target>, <Rule> and <Obligation> elements and is evaluated at the PDP to yield and access decision.
- The <Rule> element provides the conditions which test the relevant attributes within a Policy.
- The <Target> element is used to associate a requested resource with an applicable Policy.
- Combining Algorithms are used to reconcile multiple outcomes of policies' evaluations into a single decision

Target Element

- Designed to find the policies that apply to a request
- It specifies conditions against subjects, resources, actions in an access request
- <Target> is the top element
  - <AnyOf> element express disjunction of <AllOf> elements
  - <AllOf> element express conjunction of <Match> element
  - <Match>specifies a condition against subjects, resources, and actions

Rule Element

- Basic element of a
- Main elements
  - Target
    - Conditions to determine if a rule is applicable to a decision request
  - Effect
    - Consequence of the evaluation to True of a rule
  - Condition
    - Boolean expression that refines the applicability of a policy
  - Obligation (Optional)
    - Operations/actions to be performed with an authorization decision

Policies and Policy Sets

- Policy
  - Smallest element PDP can evaluate
  - Contains: Description, Target, Rules, Obligations, Rule Combining Algorithm
- Policy Set
  - Allows Policies and Policy Sets to be combined
  - Use not required
  - Contains: Description, Target, Policies, Policy Sets, Policy References, Policy Set References, Obligations, Policy Combining Algorithm

Policy Element

- Smallest entity that can be presented to the PDP for evaluation
- <Policy> is the top element
  - **<Target>** define the applicability of the policy to a decision request
  - **<Rule>** elements express different access control rules
  - **Rule Combining Algorithms** specify how rules can be combined to make an authorization decision
  - **<ObligationExpression>** element express obligations to be fulfilled by the PEP e.g *send an email to the patient when his record is accessed*

Combining Algorithms

- Deny-Overrides
  - Deny decision has priority over Permit Decision
  - If all decision is Indeterminate the result is Indeterminate
  - If all decision is Permit, the result is Permit
  - Otherwise the result is NotApplicable
- Permit-Overrides
  - Permit decision has priority over Deny Decision
  - If all decision is Indeterminate the result is Indeterminate
  - If all decision is Deny, the result is Deny
  - Otherwise the result is NotApplicable

Combining Algorithms

- First-applicable
  - The result is the effect of the first rule/policy whose Target evaluates to True
- Only-one-applicable
  - Apply only to PolicySet
  - If no Policy is applicable then the result is NotApplicable
  - If more than one Policy is applicable then the result is Indeterminate
  - If only one policy is applicable, the result is the result of evaluating the policy

Request Context

- It encapsulates a decision request to be submitted to PDP
- **<Request>** is the top element
  - **<Attributes>** elements to specify the attributes of subject, resource, action and environment
  - "Category" attribute specify if the attributes are related to subject, resource, action or environment
    - Each Attribute is encapsulated by a <Attribute> element
    - The attribute "AttributeId" identifies the attribute
    - The <AttributeValue> sub-element specifies the attribute value
  - <Content> element can encapsulate the resource itself

Response Context

- It encapsulates the authorization decision produced by the PDP
- <Response> is the top element
  - <Decision> element specifies contains the result of evaluating the decision request against the applicable policy
  - Possible Authorization Decisions: Permit, Deny, Indeterminate, Not Applicable
  - <Obligations> element specifies the obligation to be fulfilled by the PEP

Summary

- Access control specifies who or what may have access to a system resource and the type of access that is permitted
- Four types of access control policies
  - Discretionary
  - Mandatory
  - Role-based
  - Attribute-based
- Access control is very hard to implement correctly
  - Employees are often over-entitled

Summary

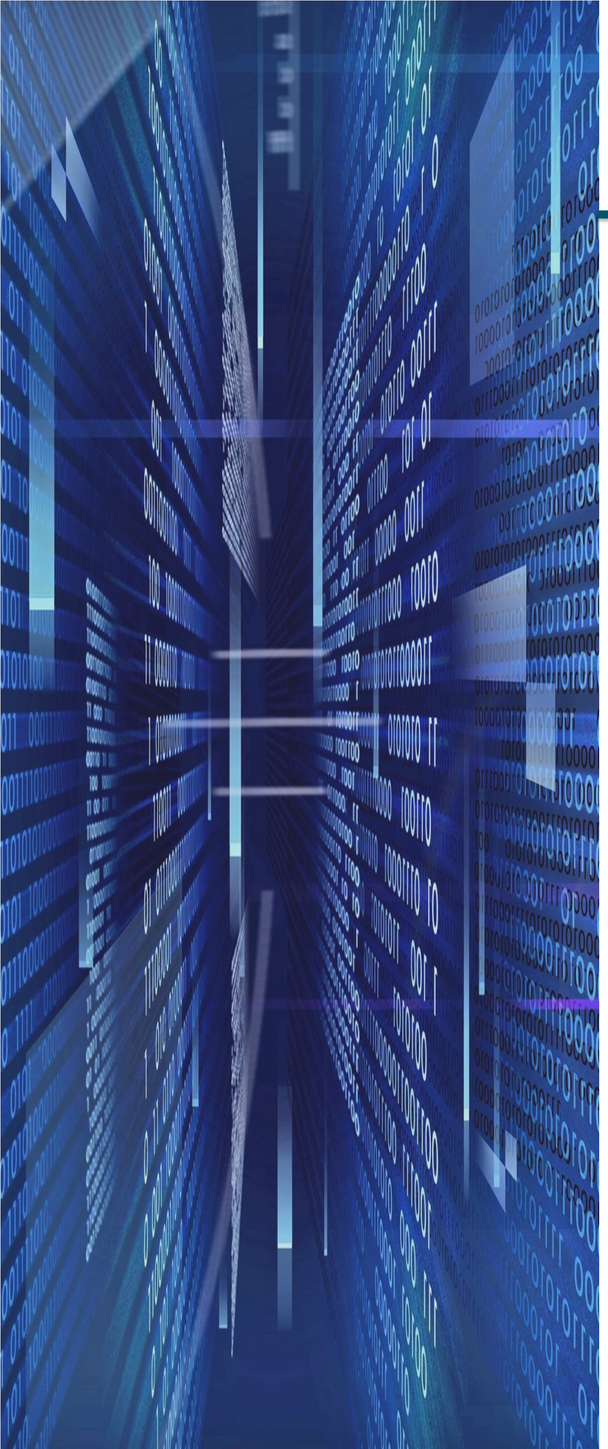
- Traditional access control policies do not allow to express real-world access control policies
- Attribute based access control allows to specify access control policies as conditions on subject, object, action and environment attributes
- XACML is the standard to specify ABAC policies
- XACML provides
  - A policy language, an architecture, a request/response protocol to make an authorization decisions

Resources

- Chapter 5- D. Gollmann. Computer Security
- Chapter 4 –W. Stallings and L. Brown. Computer Security – Principles and Practices
- INCITS 359-2012 – Information Technology Role Based Access Control
  - http://www.techstreet.com/products/1837530

Recommended Readings

- Guide to Attribute Based Access Control (ABAC) Definition and Considerations NIST Special Publication 800-162.
  - http://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.sp.800-162.pdf
- eXtensible Access Control Markup Language (XACML) XACML Version 3.0.
  - http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf
- A beginner's guide to XACML
  - https://medium.com/identity-beyond-borders/a-beginners-guide-to-xacml-6dc75b547d55



OAuth Authorization Flow

Prof. Federica Paci

What is OAuth?

- It is a standard **authorization protocol** that allows a third-party application to access protected resource hosted on a HTTP server
- It requests an **access token** from the Authorization Server
- Then the third-party application uses the access token to request access to the protected resources

- Viene usato il token d'accesso e non è necessario condividere le credenziali di accesso

Actors

- **Resource Owner:** entity capable of granting access to a protected resource
  - utente che possiede le risorse protette e decide di condividerle
- **Resource Server**: the server that stores that resource owner resources
  - server che ospita le risorse protette a cui il client cerca di accedere
    Verifica la validità del token di accesso emesso dall'Authorization Server
- **Authorization Server:** the server issuing access token to the client after authenticating the resource owner and obtaining its authorization
  - server responsabile di autenticare l'utente e concedere l'autorizzazione al client per accedere alle risorse
- **Client:** a third-party application that requests access to protected resources on-behalf of resource owner and with its approval
  - applicazione che richiede l'accesso alle risorse del Resource Owner
    Deve essere registrata e autenticata presso il servizio di autenticazione

Authorization Flows

- Authorization Code Grant Flow
- Authorization Code Grant Flow with PCKE
- Resource Owner Password
- Client Credential
- Device Flow

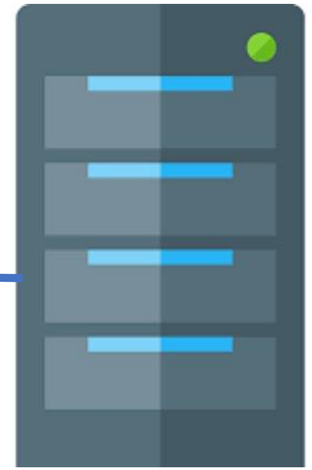
Lo standard accetta diversi scenari di autorizzazione

OAuth -The Authorization Code Flow

Alice – Resource Owner

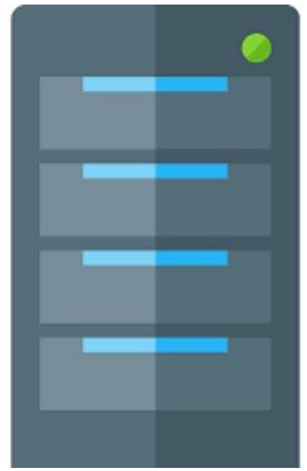
registrazione dell'applicazione sull'authorization server

Credit Union Bank - Authorization Server



https://example.com/oauth2/register?
application_name=Mybank&
redirect_uri=https://example.com/callback&
grant_type=code

client_id & client_secret



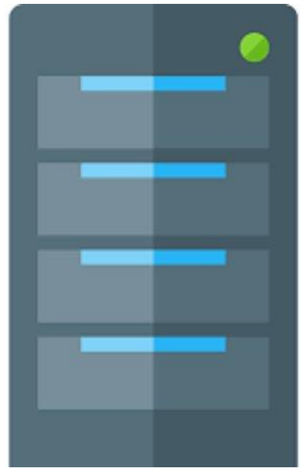
My Bank App – Client Application

Credit Union Bank - Resource Server

OAuth - The Authorization Code Flow

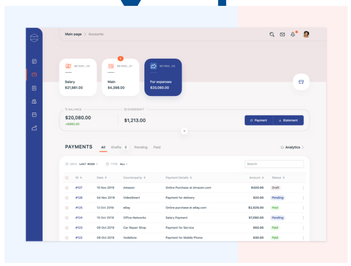
Alice – Resource Owner

Credit Union Bank - Authorization Server



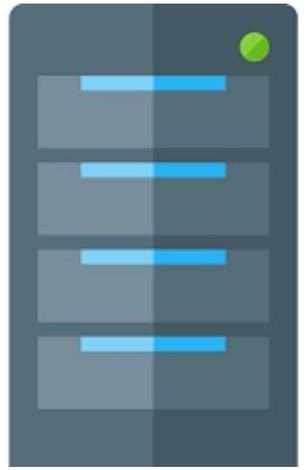
Authorization Request

Alice Authenticates



My Bank App – Client Application

```
https://example.com/oauth2/authorize?
response_type=code&
client_id=<client id>&
redirect_uri=https://example.com/callback&
scope=bankaccount&
state=123xyz
```

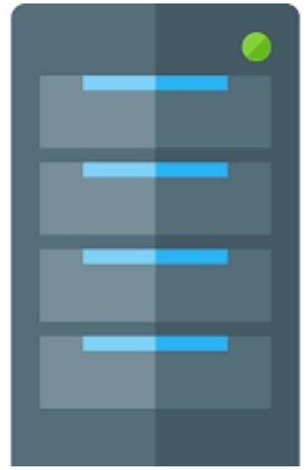


Credit Union Bank - Resource Server

OAuth -The Authorization Code Flow

Alice – Resource Owner

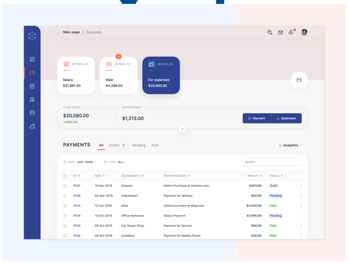
Credit Union Bank - Authorization Server



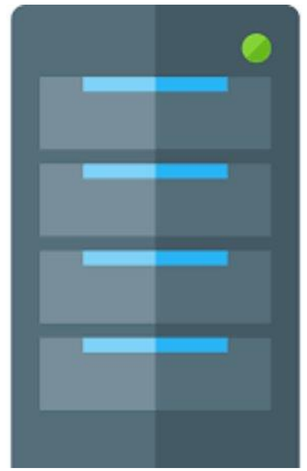
Authorization Code

https://example.com/oauth2/authorize?
code=8042d4cad58c66d0a5edfad89521192&
state=123xyz

stesso valore



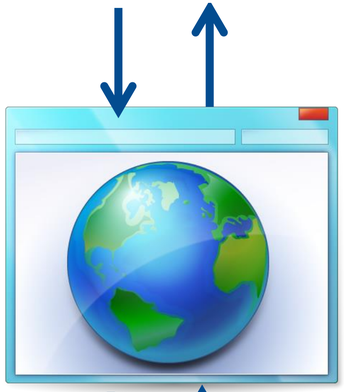
My Bank App – Client Application

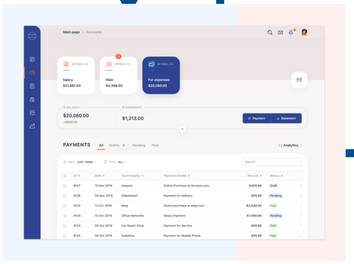


Credit Union Bank - Resource Server

OAuth - The Authorization Code Flow

Alice – Resource Owner



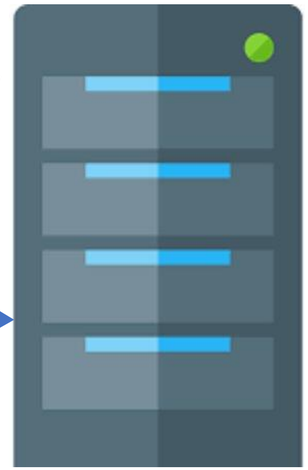


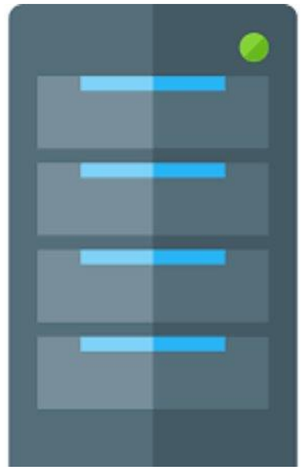
My Bank App – Client Application

POST https://example.com/oauth2/token
grant_type=authorization_code&
code=8042d4cad58c66d0a5edfad89521192&
redirect_uri=https://example.com/callback&
client_id=<client id>&
client_secret=<client secret>

Authorization Code & Redirection URI

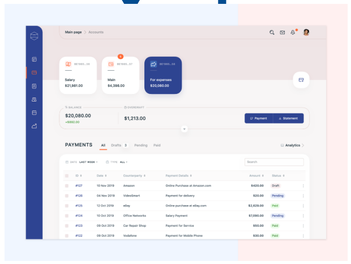
Credit Union Bank - Authorization Server





Credit Union Bank - Resource Server

OAuth -The Authorization Code Grant Flow



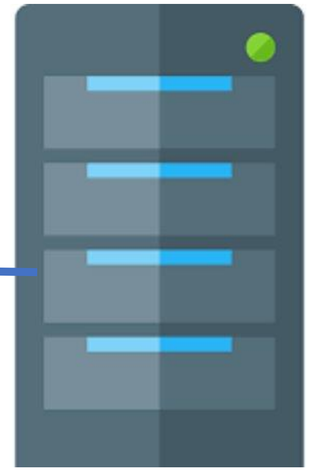
My Bank App –
Client Application

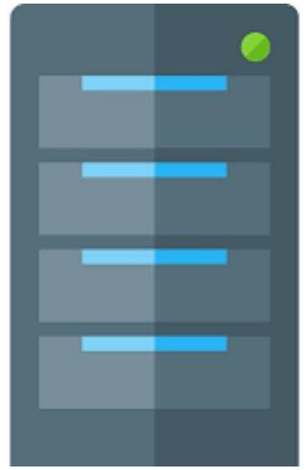
```
HTTP/1.1 200 OK
 Content-Type: application/json;charset=UTF-8
    {
      "access_token":"2YotnFZFEjr1zCsicMWpAA",
      "token_type":”bearer",
      "expires_in":3600,
      "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA”
    }
```

Se i dati inseriti rispettano quelli dell'authorization server viene rilasciato il token

Access token & Refresh token

Credit Union Bank -
Authorization Server

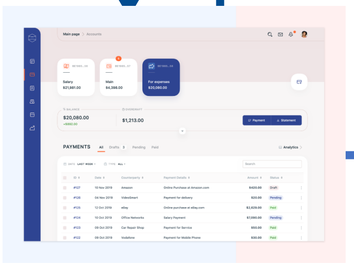




Credit Union Bank -
Rsource Server

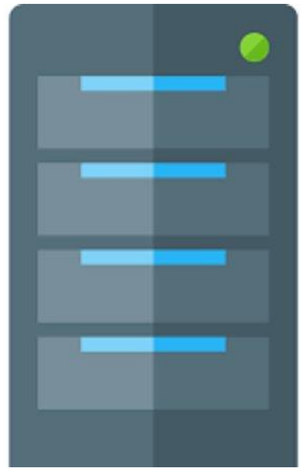
OAuth - The Authorization Code Flow

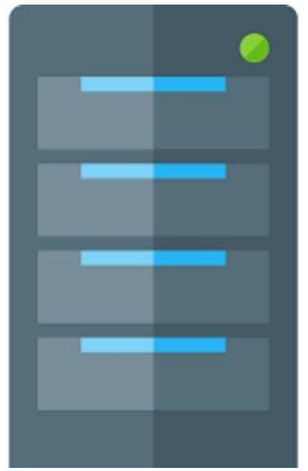
Alice – Resource Owner



Access token

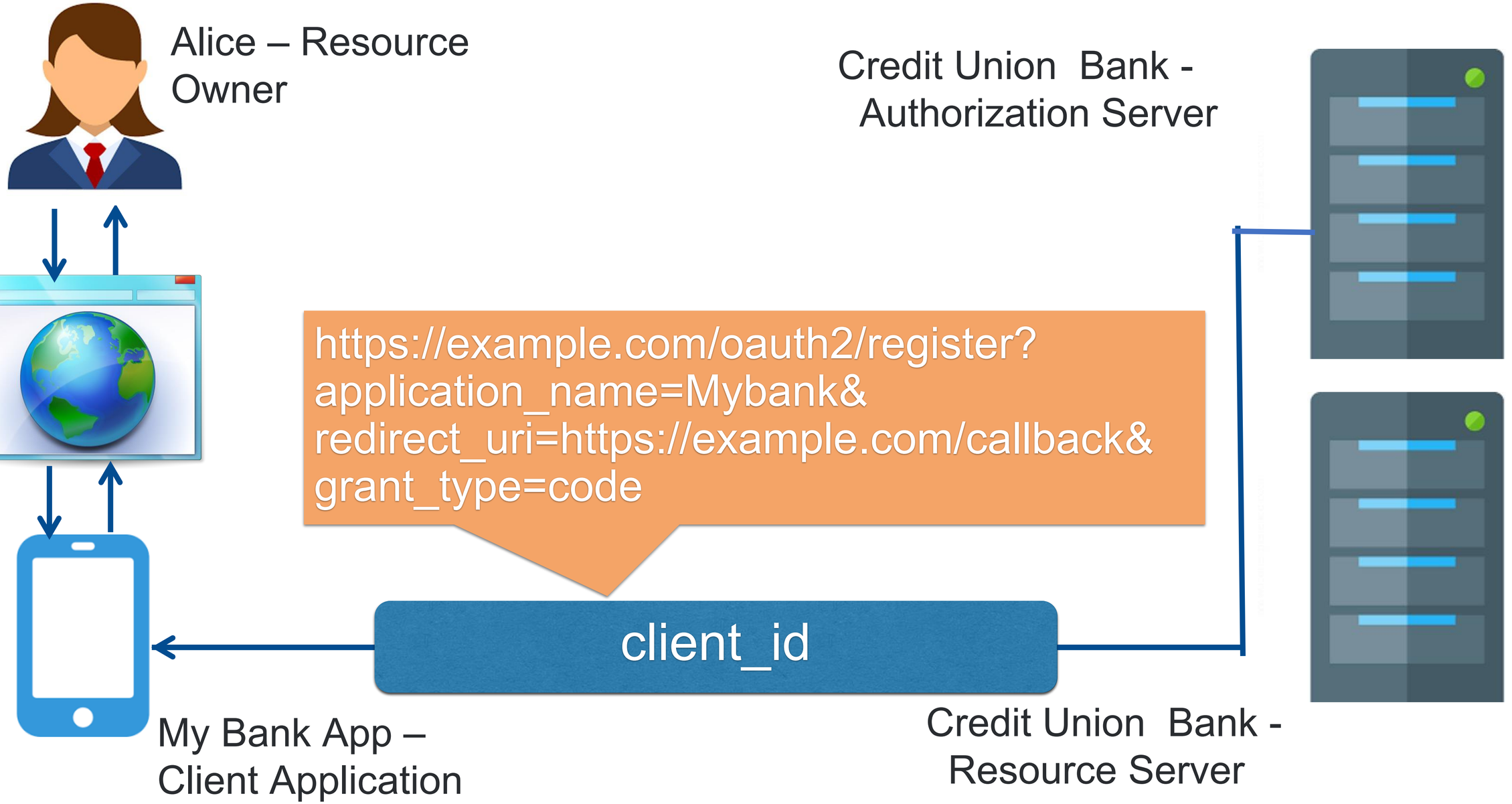
Credit Union Bank - Authorization Server





Credit Union Bank - Resource Server

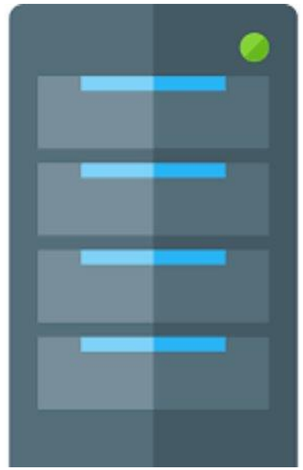
OAuth -The Authorization Code Flow with PCKE

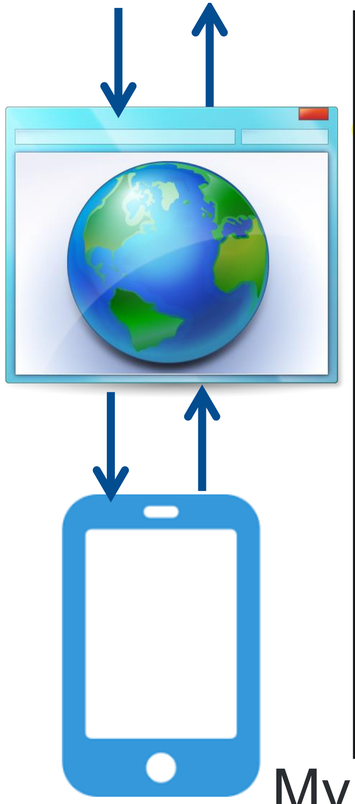


OAuth -The Authorization Code Flow with PCKE

Alice – Resource Owner

Credit Union Bank - Authorization Server





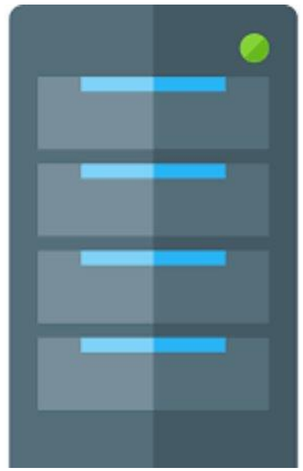
The Client generates a PCKE code verifier, a random string 43-128 characters

5d2309e5bb73b864f989753887fe52f79ce5270395e25862da6940d5

- The Client generates the PCKE Code Challenge

base64url(sha256(code verifier)

viene fatto l'hash



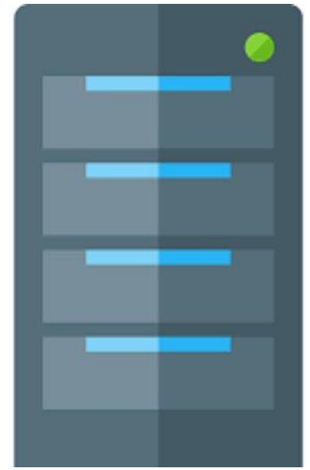
My Bank App – Client Application

Credit Union Bank - Resource Server

OAuth - The Authorization Code Flow

Alice – Resource Owner

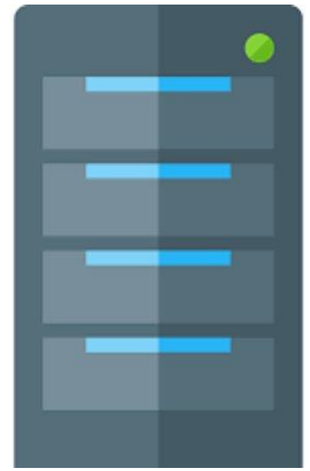
Credit Union Bank - Authorization Server



Alice Authenticates

https://example.com/oauth2/authorize?
response_type=code&
client_id=<client id>&
redirect_uri=https://example.com/callback&
scope=bankaccount&
state=123xyz
code_challenge=CODE_CHALLENGE
code_challenge_method=S256

My Bank App – Client Application

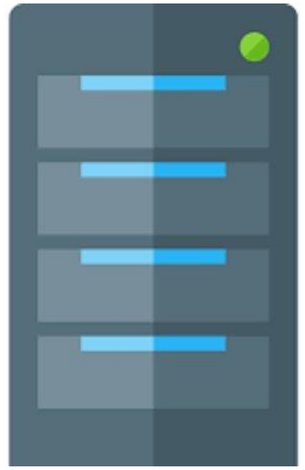


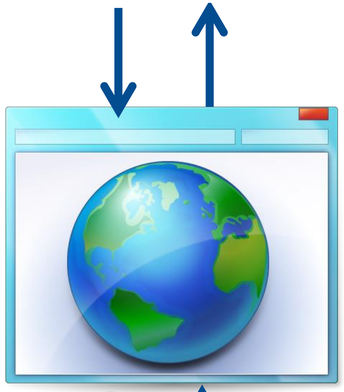
Credit Union Bank - Resource Server

OAuth -The Authorization Code Flow

Alice – Resource Owner

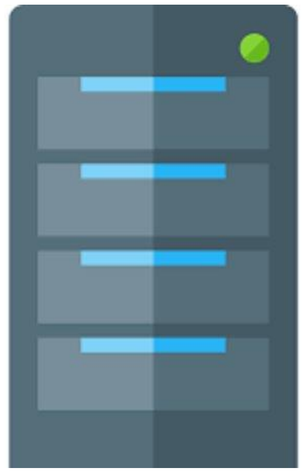
Credit Union Bank - Authorization Server





Authorization Code

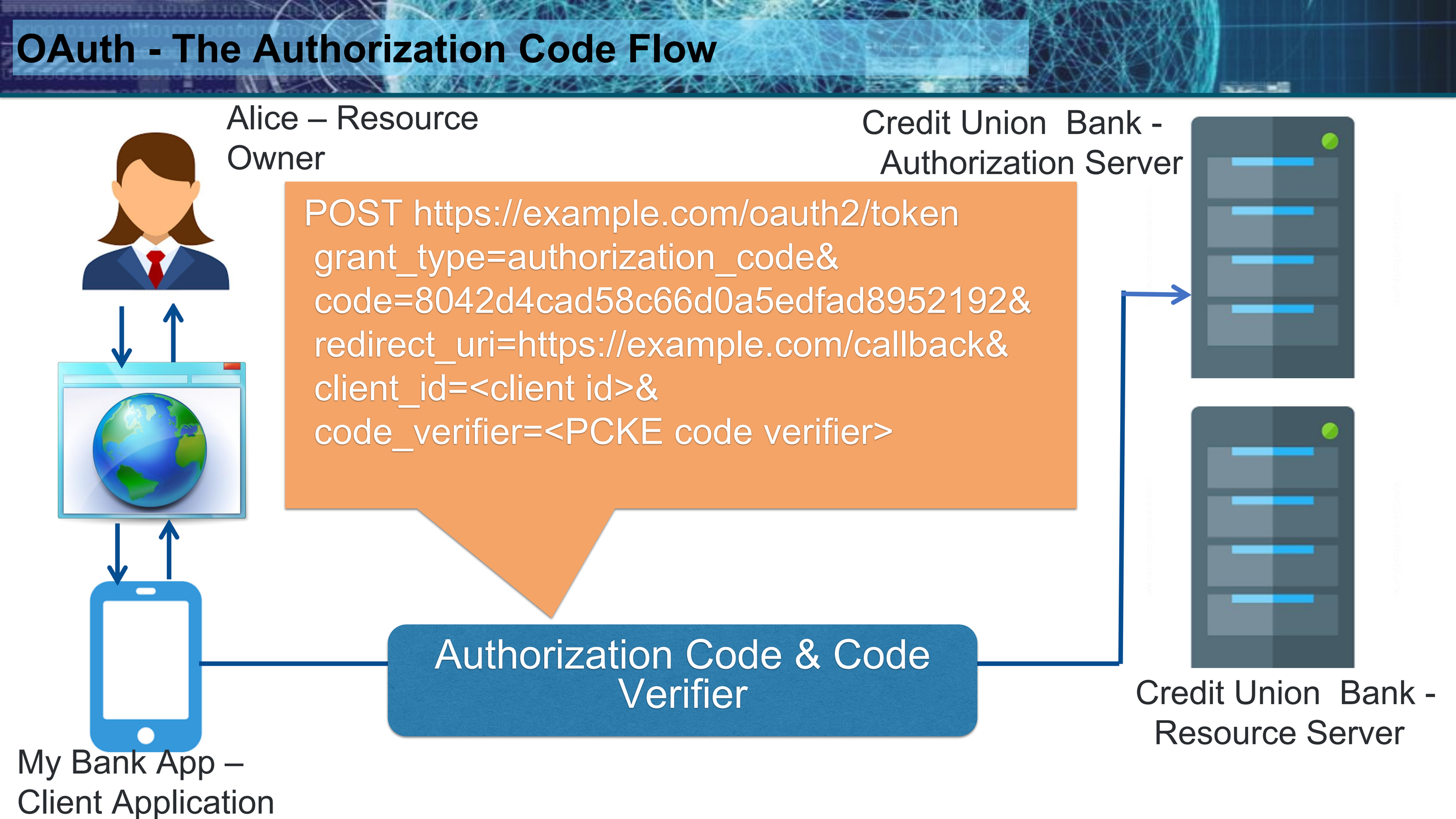
https://example.com/oauth2/authorize?
code=8042d4cad58c66d0a5edfad89521192&
state=123xyz



My Bank App – Client Application

Credit Union Bank - Resource Server

OAuth - The Authorization Code Flow



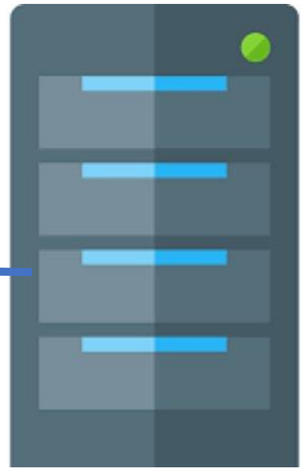
OAuth -The Authorization Code Grant Flow

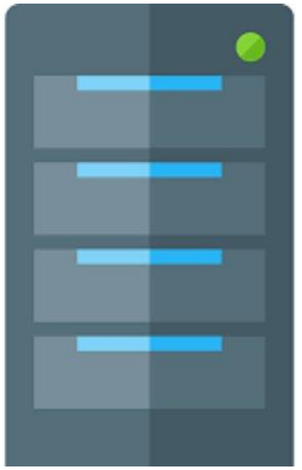
My Bank App –
Client Application

```
HTTP/1.1 200 OK
 Content-Type: application/json;charset=UTF-8
     {
       "access_token":"2YotnFZFEjr1zCsicMWpAA",
       "token_type":”bearer",
       "expires_in":3600,
       "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA”
      }
```

Access token & Refresh token

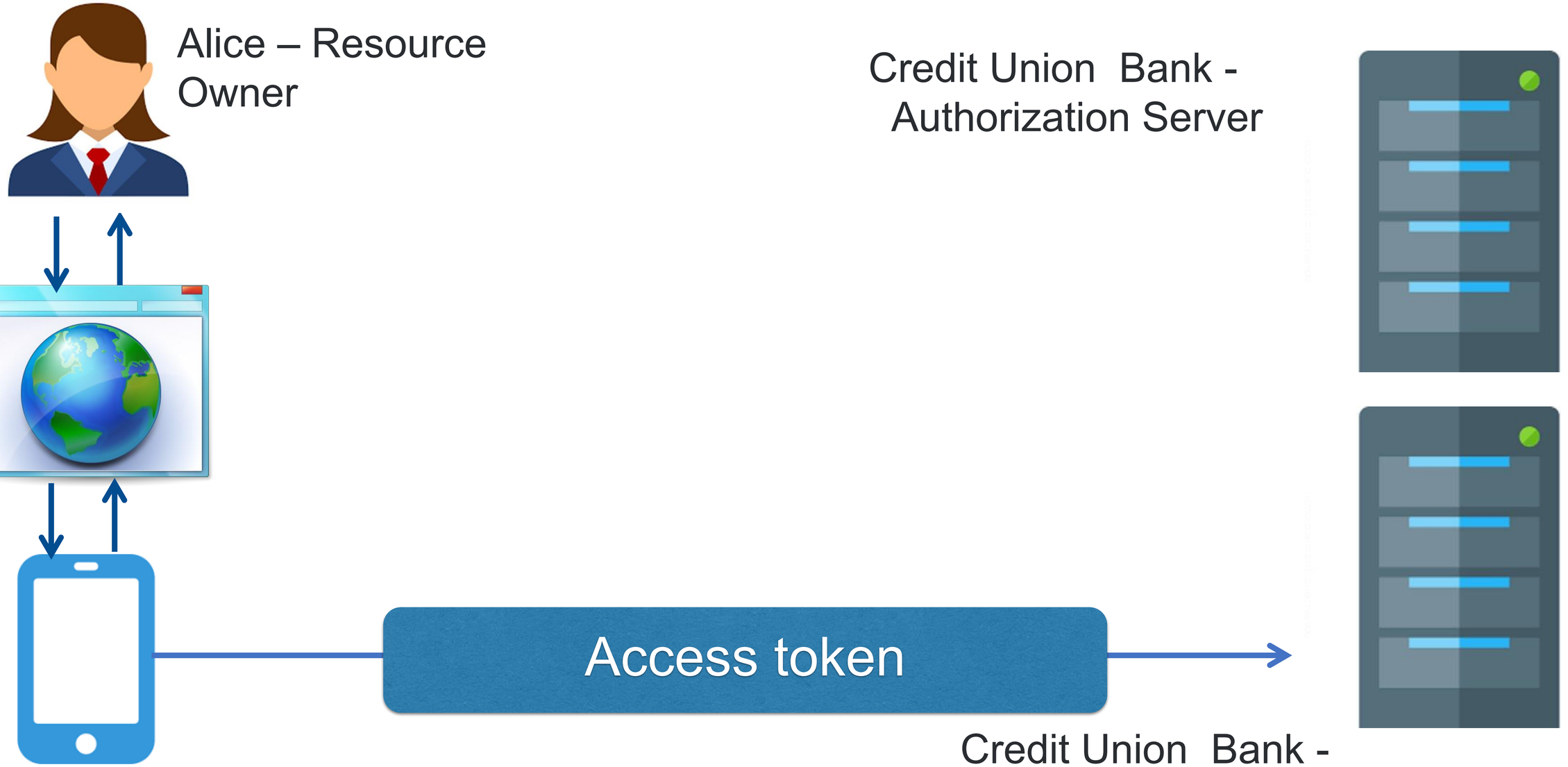
Credit Union Bank -
Authorization Server





Credit Union Bank -
Rsource Server

OAuth - The Authorization Code Flow



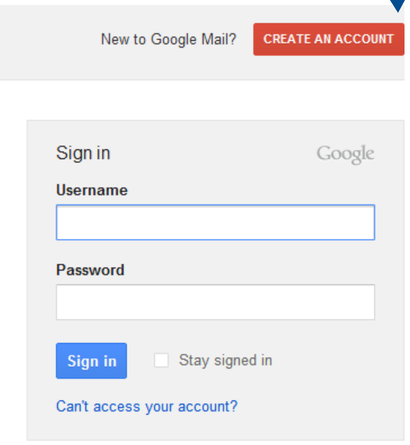
Resource Owner Password Grant Flow

Google login a

POST https://google.com/oauth2/token
grant_type=password&
client_id=<client id>&
username=johhdoe&
password=A4dm1kl
scope=email

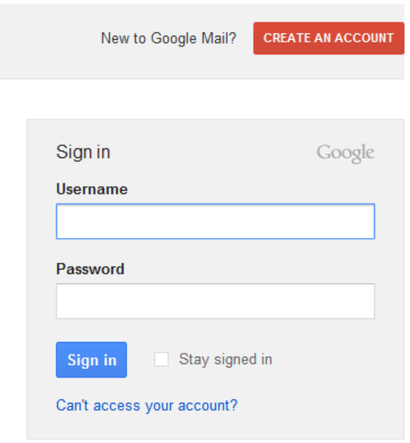
flusso diverso è un flusso password

Google login and password



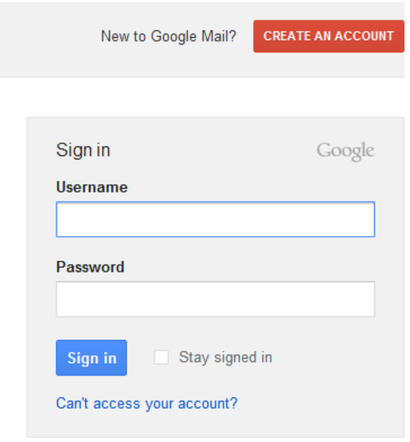
Resource Owner Password Grant Flow

```
HTTP/1.1 200 OK
 Content-Type: application/json;charset=UTF-8
     {
       "access_token":"2YotnFZFEjr1zCsicMWpAA",
       "token_type":”bearer",
       "expires_in":3600,
       "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA”
      }
```



Access Token

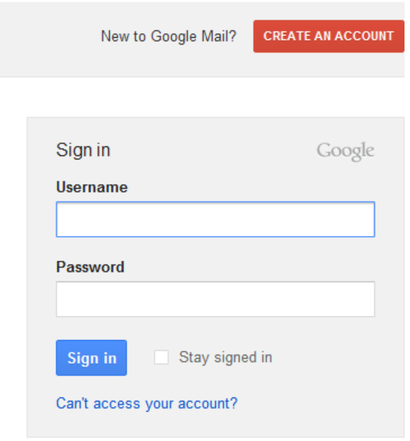
Resource Owner Password Grant Flow



curl –H “Authorization: bearer 2YotnFZFEjr1zCsicMWpAA” \
https://google.com/gmail

Access Email Request &
Access Token

Resource Owner Password Grant Flow



Google

Client Credential Grant Flow

Access Token Request

Google

POST https://google.com/oauth2/token
grant_type=client_credential&
client_id=<client id>&
client_secret=<client secret>

Google Cloud Storage

Client Credential Grant Flow

Accesso viene richiesto per conto di un' applicazione e non di un'utente

Access Token

```
HTTP/1.1 200 OK
 Content-Type: application/json;charset=UTF-8
     {
       "access_token":"2YotnFZFEjr1zCsicMWpAA",
       "token_type":”bearer",
       "expires_in":3600,
     }
```

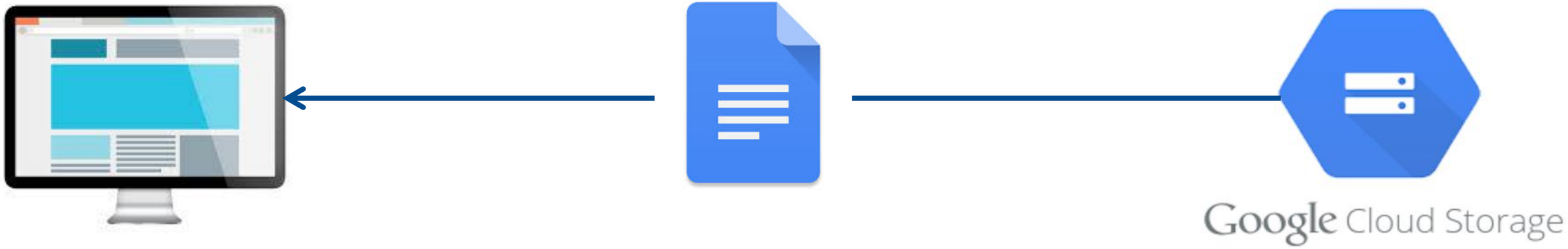
ogle Cloud Storage

Client Credential Grant Flow

Access Files Request &
Access Token

Google Cloud Storage

Client Credential Grant Flow



Device Flow

Accesso con dispositivi con limitata capacità di input

Google

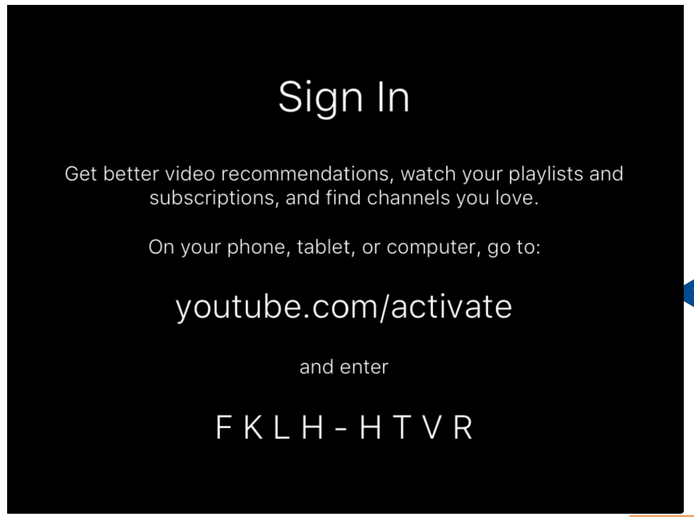
YouTube TV

tv

Access Token Request

POST https://google.com/oauth2/token
response_type=device_code&
client_id=<client id>

Device Flow

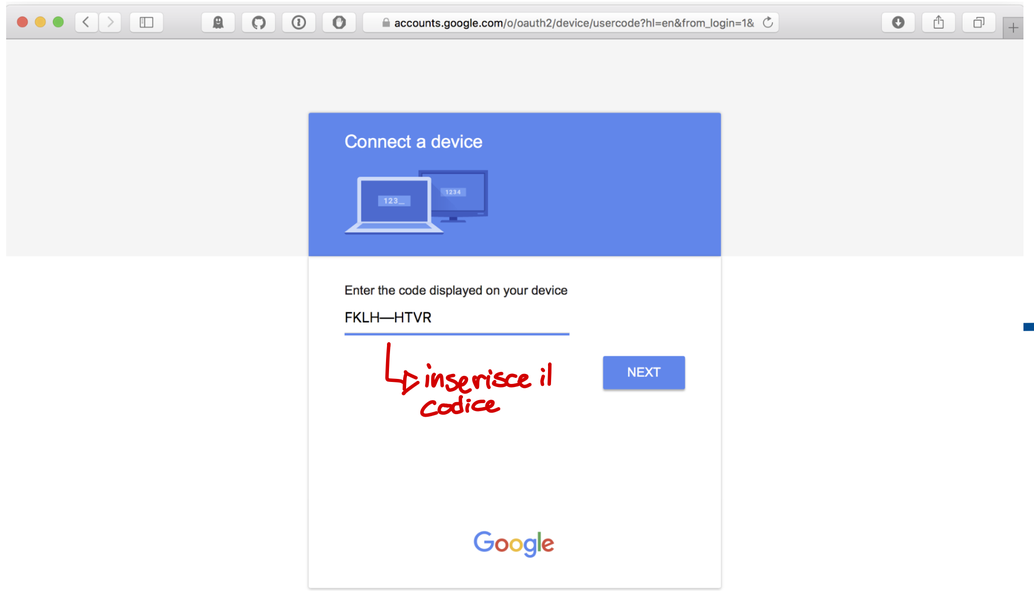


Google

Login Info


```
{
"verification_uri": "youtube.com/activate",
    "user_code": "FKLH-HTVR",
    "device_code": "74tq5miHKB",
    "interval": 5
}
```

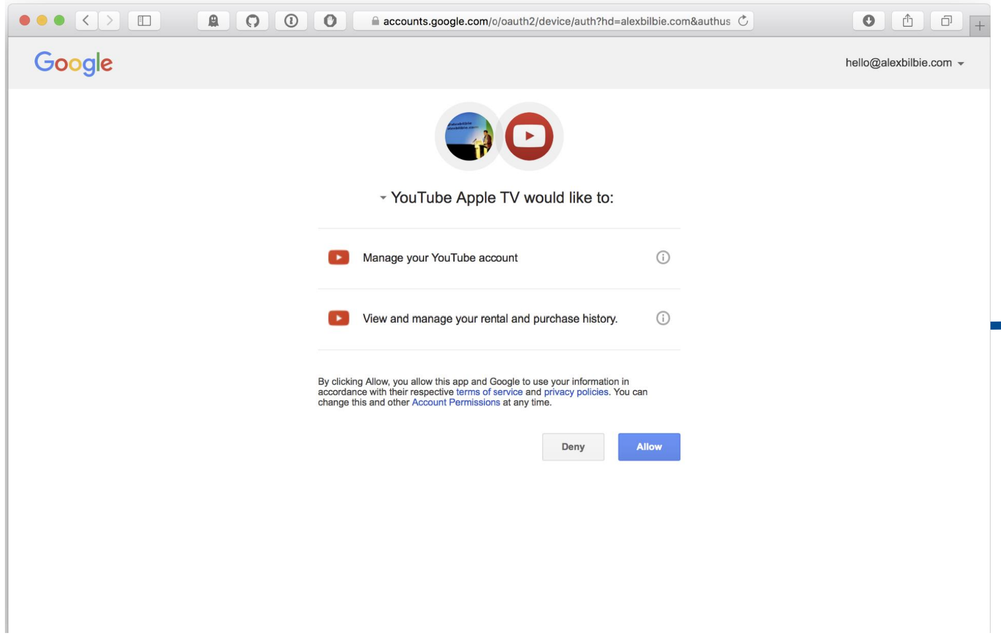
Device Flow



Google

User Code

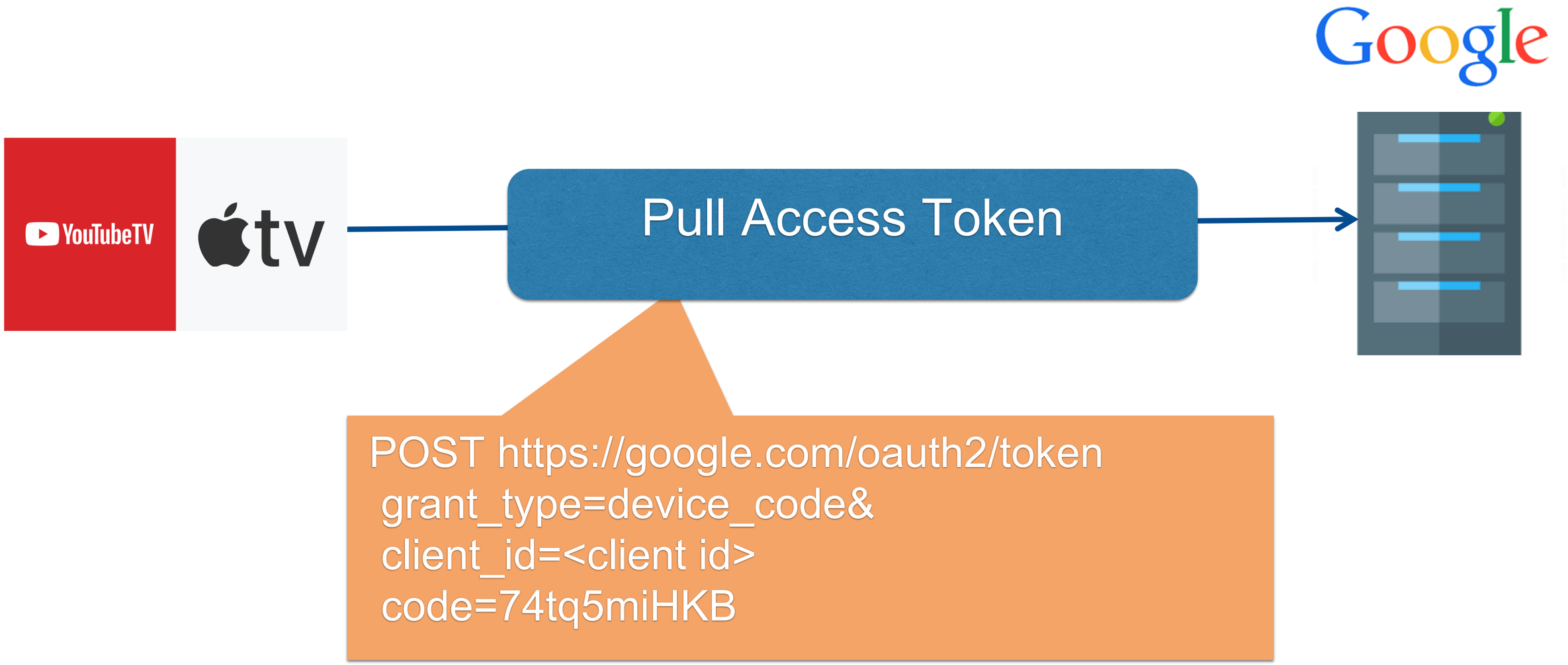
Device Flow



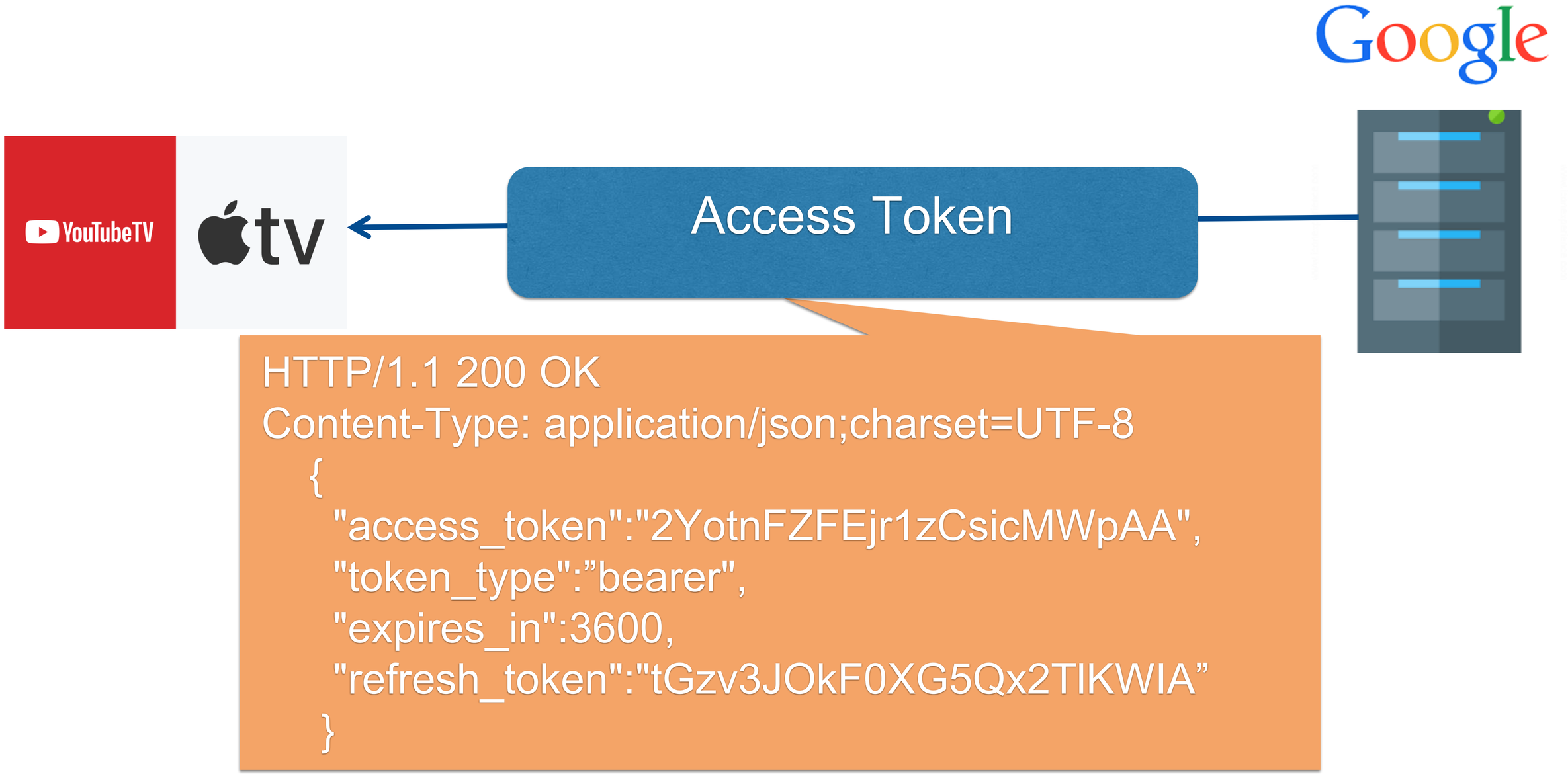
Google

Authorization

Device Flow


Google
YouTubeTV
tv
Pull Access Token
POST https://google.com/oauth2/token
grant_type=device_code&
client_id=<client id>
code=74tq5miHKB

Device Flow


Google
YouTubeTV
tv
Access Token
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
"access_token":"2YotnFZFEjr1zCsicMWpAA",
"token_type":”bearer",
"expires_in":3600,
"refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA”
}

Exercise 2 – Smart Home

- Alice lives in a smart home with a smart lighting system
- Alice can control her smart light system with MyBulb mobile app
- The smart light systems exposes APIs to switch/on and off the smart bulbs that are part of the system
- Think about:
  - Who are the actors?
  - What are the protected resources?
  - What grant flow would you use?
- Time: 5 min

Summary

- OAuth is an authorization protocol that allows an end-user (the resource owner) to grant access to a client application to protected resources with the end-user consent
- OAuth supports three main grant flows
  - Authorization code grant flow
    a. the client application is a web app which has a backend server
    b. access is requested on behalf of the end user
  - Authorization code grant flow
    a. the client application is a browser–based app or mobile app
  - Resource owner password grant flow
    a. the client application is a a first party application
    b. access is requested on behalf of the end user
  - Client credential grant flow
    a. access is requested on behalf of the client application itself
  - Device grant flow
    a. access is requested devices with no browser or limited input capability

Resources

- OAuth Web site
  - https://oauth.net/2/
- OAuth specification
  - https://tools.ietf.org/html/rfc6749
- OAuth simplified explanation
  - https://aaronparecki.com/oauth-2-simplified/
- OAuth security
  - https://tools.ietf.org/html/rfc6819