OAuth2, OpenID Connect et la mobilité

Philippe Depouilly

4 Octobre 2016

mathrice

OAuth2

- Avec le déploiement des applications mobiles et des services en ligne est apparu l'utilisation massive des APIs REST dans la perspective d'inter-opérer plus simplement entre producteurs de services
- Les opérateurs de ses services se sont entendus afin de définir un protocole d'échange d'identités (que l'on appellera fédération) et de délégation de droits (consentement) appelé OAuth
- Avec un aspect vertueux de ne pas multiplier ses identités et moins vertueux de tracer au maximum les usages

mathrice

OAuth2

- OAuth, un cadre fixant un protocole d'autorisation, est apparu en 2006
- Faible, inconsistant dans sa version 1, il a évolué en 2012 dans sa version 2
- OAuth2 définit un mécanisme de délégation d'autorisation de services en ligne, ce n'est pas un protocole d'authentification en soit
- Une authentification permet de garantir à une application quel utilisateur tente d'y accéder et si cet utilisateur est toujours présent quand il y accède
- OAuth2 ne définit rien à ce sujet (dans la RFC, cela est précisé comme étant "beyond the scope of this specification")

mathrice

Vocabulaire OAuth2

- Resource Owner (RO) : l'utilisateur, le propriétaire des ressources
- Resource Server (RS) : API traitant et fournissant des ressources
- Authorization Server (AS) : fournisseur d'accréditation et de jetons après une authentification réussie
- Client : application qui va authentifier le RO sur le AS puis accéder et exploiter les ressources venant du RS

mathrice

Schéma de principe : RFC 6749

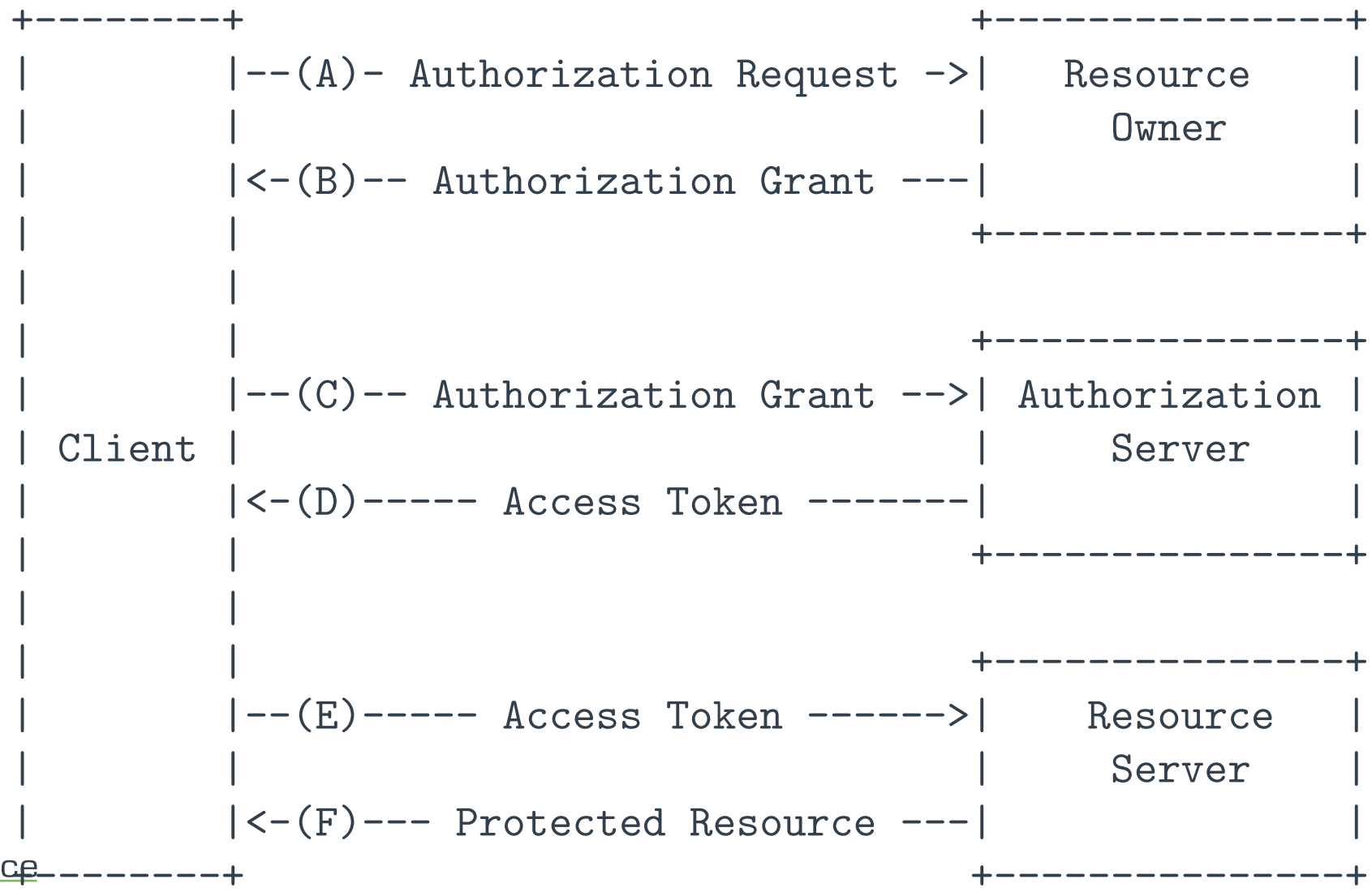


Schéma de principe : fonctionnement vu du Client

- L'utilisateur (Resource Owner) souhaite accéder à l'application cliente (le Client).
- Le Client ne le connaissant pas va solliciter le RO afin d'obtenir une autorisation
- Une fois cette autorisation obtenue, il l'a soumet au service d'autorisation (Authorization Server) en fournissant aussi la preuve (plus ou moins vérifiable, nous verrons plus loin) du Client
- Ces informations validées, le AS retourne un jetons d'accès (access token)
- En présentant ce jetons, le Client pourra accéder à des ressources provenant du Resource Server

Généralement, le Client redirige le Resource Owner vers l'Authorization Server afin d'obtenir une autorisation. Il existe d'autres moyens de valider le RO (nous verrons aussi plus loin).

mathrice

OAuth2 vs authentification

- OAuth2 décrit les échanges entre les différents protagonistes d'un service en ligne (appelé Client) afin que le propriétaire d'une ressource (RO : l'utilisateur) consente (Consent screen) depuis un service d'autorisation (AS) que des données le concernant soient traitées par une API (RS)
- Ces échanges s'effectuent par des échanges de jetons (token) dans les en-têtes ou corps de requêtes HTTP
- Le "Consent screen" est un aspect important dans le processus d'autorisation : lorsque le RO s'est authentifié, il lui est clairement affiché quelles données l'AS va diffuser vers le Client et si oui ou non il accepte la diffusion de ces données (RFC 6819)

mathrice

OAuth2 vs authentification

- OAuth2 propose quatre façons de valider la présence de l'utilisateur
  - authorization code : la plus courante, dédié à une application hébergée
  - implicit : plus légère (le client n'est pas vraiment validé : application embarquée, avec des données utilisateur peu sensibles ou bien une application intégrée à l'OS)
  - resource owner password credentials : plus robuste que implicit (mais le Client voit passer au moins une fois l'identifiant et mot de passe, donc le client est de confiance)
  - client credentials (l'utilisateur ne présente aucune clé, les données sont de réputation publique)

mathrice

OAuth2 vs authentification

- Il est indispensable d'ajouter des fonctionalités à OAuth2 afin de réaliser une solution fiable et robuste d'authentification et d'autorisation.
- Par exemple OAuth2 ne précise pas comment :
  - éviter le re-jeu de jetons via d'autres clients et ainsi l'usurpation d'identité.
  - garantir que ce jeton provient bien de l'Authorization Server (signature)
  - etc.

mathrice

OAuth2 vs authentification

- C'est pour cela que chaque fournisseur de service basé sur OAuth2 a sa propre implantation, afin de déployer un service d'authentification et d'autorisation basé sur OAuth2 et de fiabiliser le service
- Partant des spécifications de OAtuh2, les acteurs du marché ont donc développé des stratégies propres ajoutant soit des champs supplémentaires dans le corps de retour des APIs, soit des mécanismes de signature de type JWT (Json Web Token), soit d'introspection de jeton (RFC 7662), voire plusieurs mécanismes simultanés

mathrice

Exemple de stratégie OAuth2 dans Mathrice

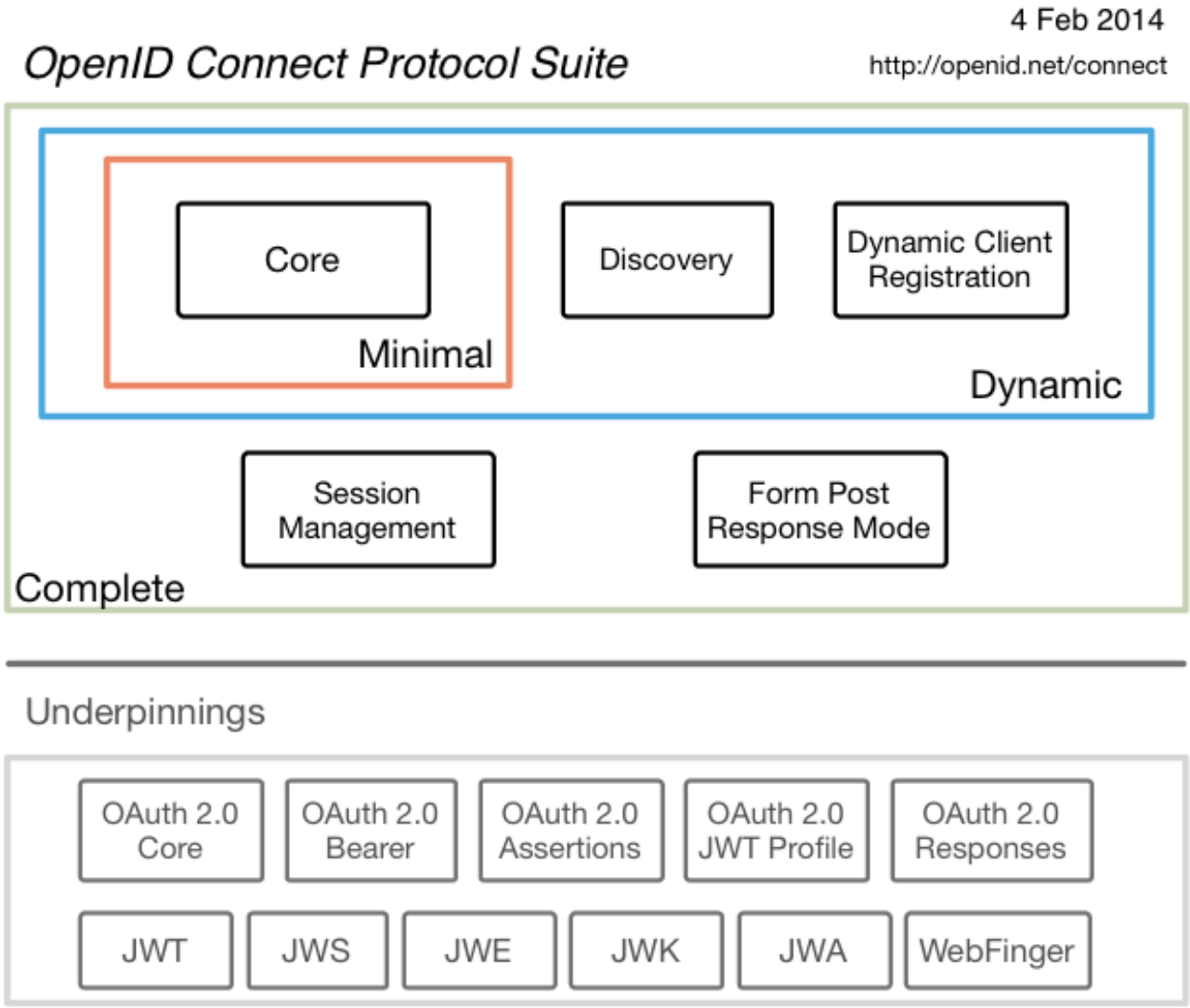
```ruby
module OmniAuth
  module Strategies
    class Plm < OmniAuth::Strategies::OAuth2
      option :name, 'plm'
      option :client_options, {
        site:          'https://plm.math.cnrs.fr/sp',
        authorize_url: 'https://plm.math.cnrs.fr/sp/oauth/authorize',
        token_url:     'https://plm.math.cnrs.fr/sp/oauth/token'
      }

      uid do
        raw_info['uid']
      end

      info do
        {
          email: raw_info['email'],
        }
      end

      extra do
        { raw_info: raw_info }
      end

      def raw_info
        @raw_info ||= access_token.get('/sp/me').parsed
      end
    end
  end
end
```

Apports de OpenID Connect

- OpenID Connect est une proposition d'authentification et autorisation basé sur OAuth2 qui offre en plus des mécanismes robustes de signature et d'encryption afin de garantir et fiabiliser les échanges

mathrice

Apports de OpenID Connect



mathrice

Apports de OpenID Connect

- Signature et chiffrement des réponses avec JWT
- IDToken : jeton de profil utilisateur associé à un jetons de session
- Nonce : code de non rejeu à valider lors de la récupération d'un IDToken (envoyé par le client lors de l'autorisation)
- Mécanisme de découverte des URLs et certificats

mathrice

Exemple d'un mécanisme de découverte


```
{
  "issuer": "https://plm.math.cnrs.fr/sp",
  "authorization_endpoint": "https://plm.math.cnrs.fr/sp/oauth/authorize",
  "token_endpoint": "https://plm.math.cnrs.fr/sp/oauth/token",
  "userinfo_endpoint": "https://plm.math.cnrs.fr/sp/oauth/userinfo",
  "jwks_uri": "https://plm.math.cnrs.fr/certs",
  "response_types_supported":
  [ "code", "token", "id_token", "code token", "code id_token", "token id_token", "code token id_token", "none" ],
  "subject_types_supported": [ "public" ],
  "id_token_signing_alg_values_supported": [ "RS256" ],
  "scopes_supported": [ "openid", "profile" ],
  "token_endpoint_auth_methods_supported":
  [ "client_secret_post", "client_secret_basic" ],
  "claims_supported":
  [ "aud", "nonceme", "exp", "displayName", "iat", "iss", "sub" ]
}
```

Exemple de stratégie OpenID Connect dans Mathrice

```
Rails.application.config.middleware.use OmniAuth::Builder do
  provider :openid_connect,
  name: :plm,
  issuer: 'https://plm.math.cnrs.fr/sp',
  discovery: true,
  #
  # soit on ne met pas de scope (dans ce cas le scope openid est envoye),
  # soit on met openid et legacyplm afin d'obtenir les infos de convergence
  #
  scope: 'openid_legacyplm',
  client_options: {
   port: 443,
   scheme: "https",
   host: 'plm.math.cnrs.fr',
   identifier: 'CLIENT_ID',
   secret: 'SECRET_ID',
   redirect_uri: 'https://plm.math.cnrs.fr/oauth-client/auth/plm/callback'
  }
end
```

OAuth2 dans Mathrice

- Doorkeeper 4
- Application Rails
- Toute une panoplie de mécanismes/greffons pour gérer, déléguer l'authentification
- Facilement extensible

OAuth2 dans Mathrice

- Extension OpenID Connect au dessus de DookKeeper
- Permet d'inter-opérer avec des clients tel que mod_OIDC pour Apache
- Une authentification robuste, chiffrée et signée

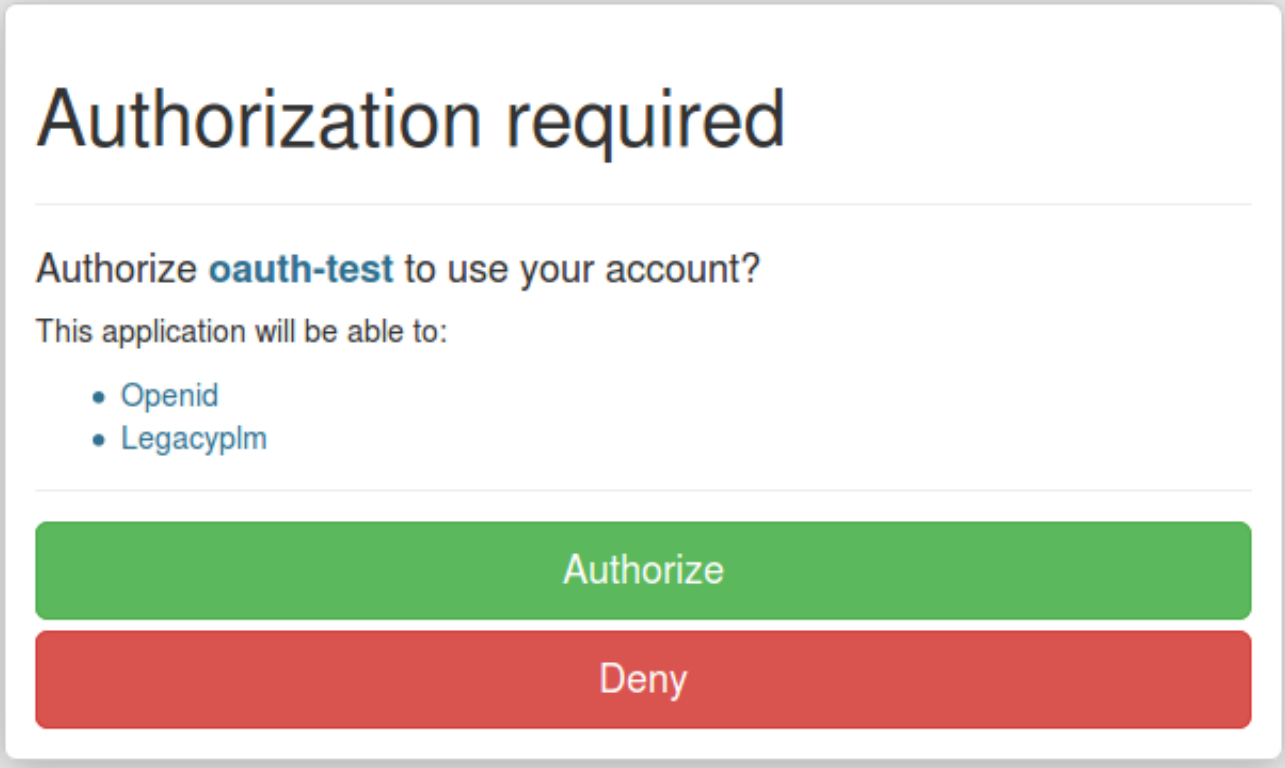
mathrice

OAuth2 dans Mathrice

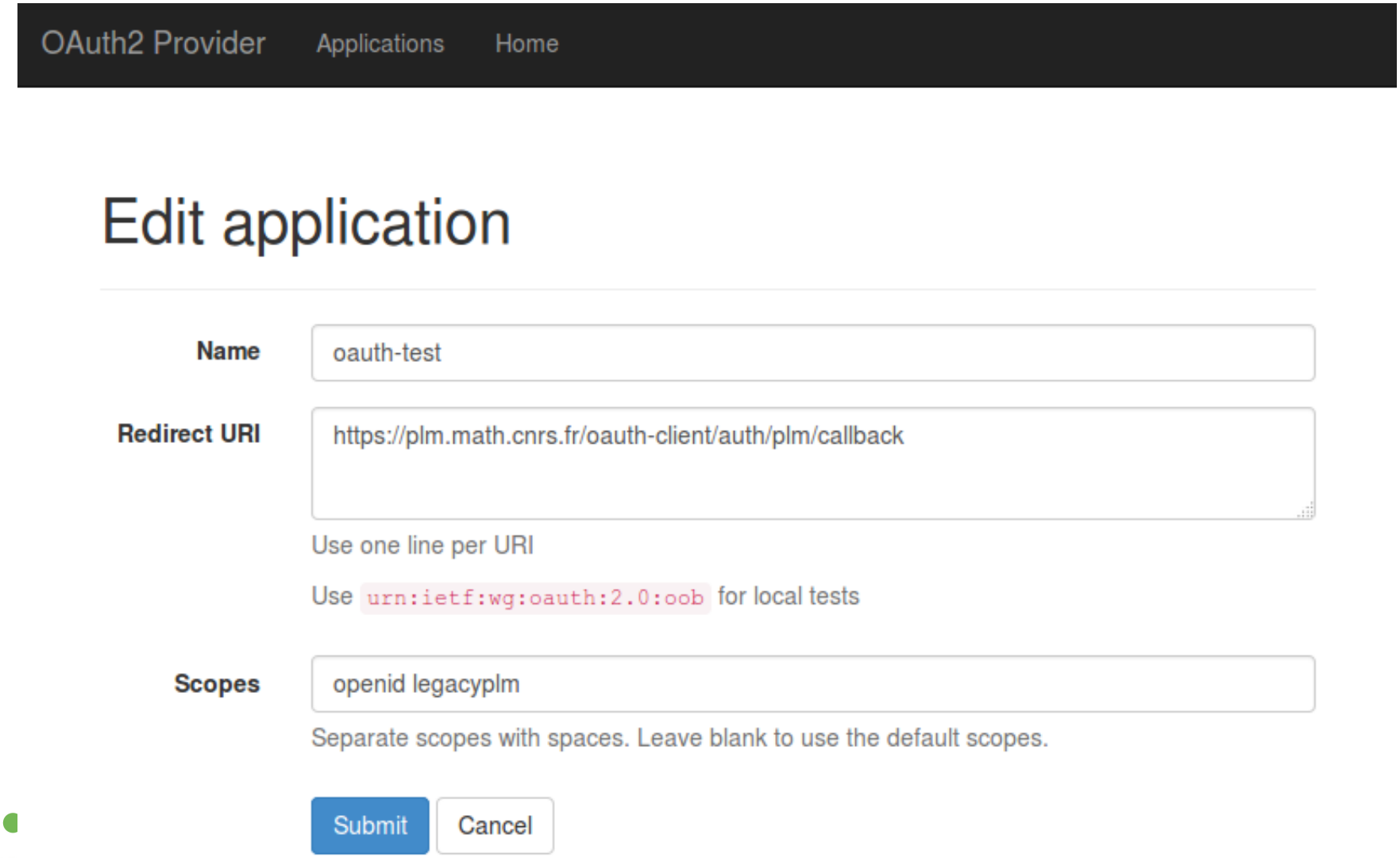
- L'Authorization Server est protégé par Shibboleth
- C'est un Service Provider dans la Fédération RENATER
- Permet de déployer une panoplie de services sous une authentification robuste pour une large communauté d'utilisateurs

mathrice

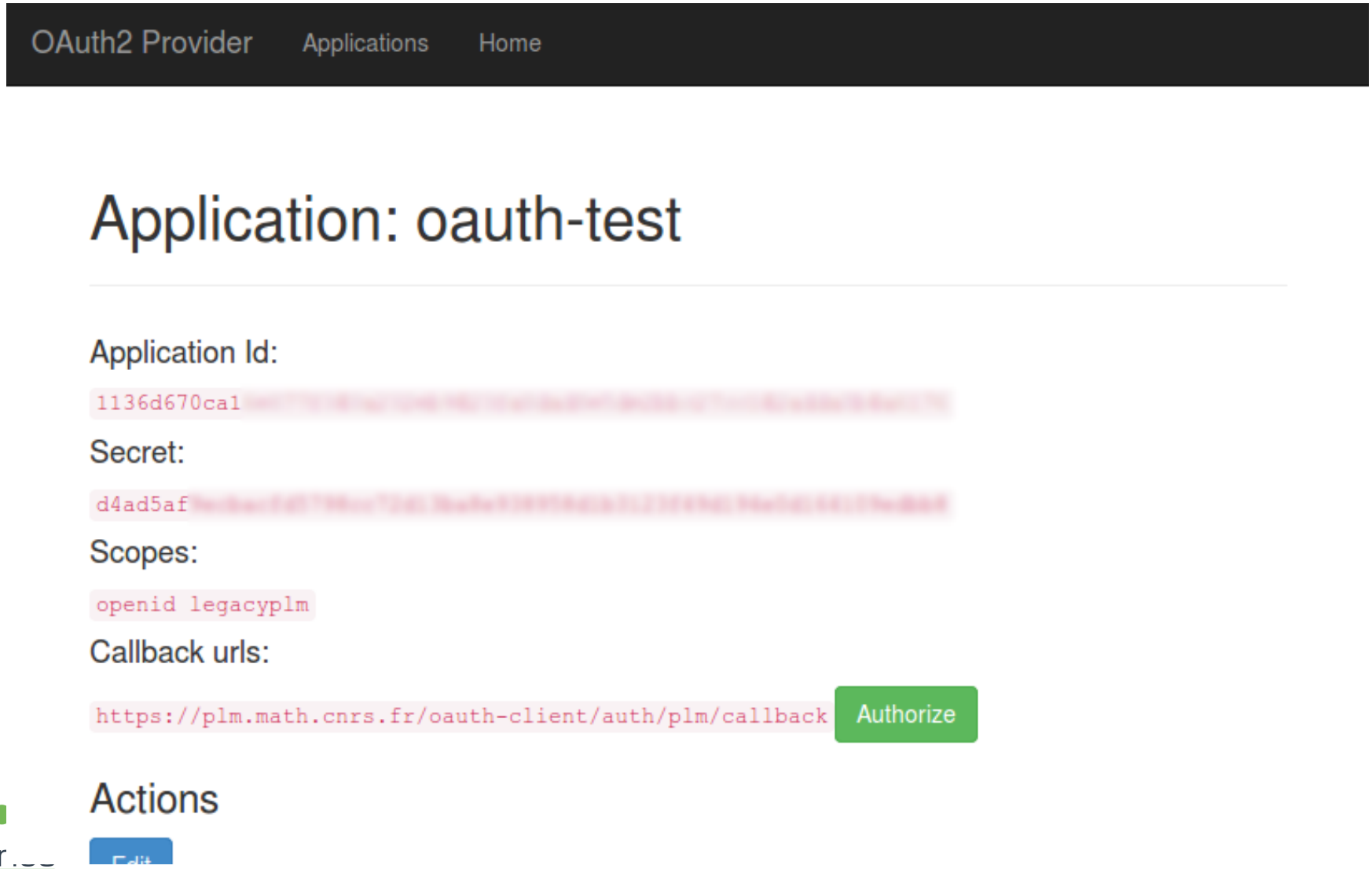
OAuth2 dans Mathrice



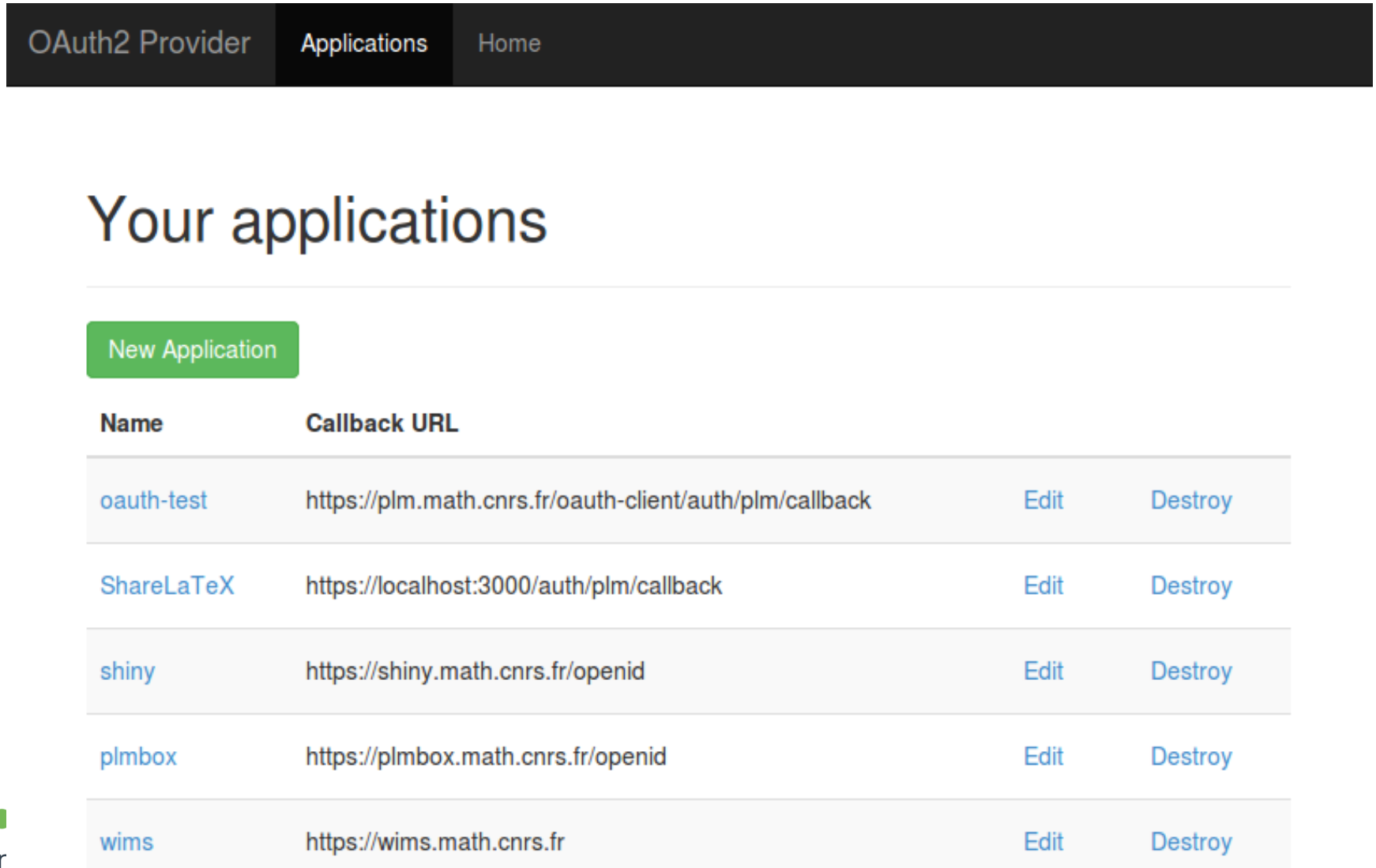
OAuth2 dans Mathrice



OAuth2 dans Mathrice



OAuth2 dans Mathrice



Exemple d'authentification OAuth2

Premier accès au Client en mode non authentifié

```
GET "/sp/oauth/authorize?client_id=[CLIENT_ID]&redirect_uri=https%3A%2F%2Fexample.math.cnrs.fr%2Fauth%2Fplm%2Fcallback&response
Processing by Doorkeeper::AuthorizationsController#new as HTML
Parameters: {
  "client_id"=>[CLIENT_ID],
  "redirect_uri"=>"https://example.math.cnrs.fr/auth/plm/callback",
  "response_type"=>"code",
  "state"=>"5cf5a4cfb39e259b390f613bba2844dfa08fe771804e9f1f"
  }
Redirected to https://plm.math.cnrs.fr/PLMlogin?https://plm.math.cnrs.fr/sp/users/auth/shibboleth
Completed 302

=> Ecran d'authentification Shibboleth (Service Provider)
GET "/sp/users/auth/shibboleth"

=> Authentifcation réussie, appel d'une URL protégée par Shibboleth (donc le résultat de l'authentification est propagée dans l
GET "/sp/users/auth/shibboleth/callback"

=> Mon utilisateur existe dans ma base de données de l'Autorisation Server :
Processing by Users::OmniauthCallbacksController#shibboleth as HTML
SELECT  "users".* FROM "users" WHERE "users"."email" = ? LIMIT 1  [["email", "philippe.depouilly@math.u-bordeaux.fr"]]

=> retourne un UUID local : 640
```

Exemple d'authentification OAuth2

Authentification réussie, on revient sur l'autorisation

```
Redirected to https://plm.math.cnrs.fr/sp/oauth/authorize?client_id=[CLIENT_ID]&redirect_uri=https%3A%2F%2Fexample.math.cnrs.fr
GET "/sp/oauth/authorize?client_id=[CLIENT_ID]&redirect_uri=https%3A%2F%2Fexample.math.cnrs.fr%2Fauth%2Fplm%2Fcallback&response
Processing by Doorkeeper::AuthorizationsController#new as HTML
Parameters: {
  "client_id"=>[CLIENT_ID],
  "redirect_uri"=>"https://example.math.cnrs.fr/auth/plm/callback",
  "response_type"=>"code",
  "state"=>"5cf5a4cfb39e259b390f613bba2844dfa08fe771804e9f1f"
  }

SELECT  "users".* FROM "users" WHERE "users"."id" = ?  ORDER BY "users"."id" ASC LIMIT 1  [["id", 640]]
SELECT  "oauth_applications".* FROM "oauth_applications" WHERE "oauth_applications"."uid" = ? LIMIT 1  [["uid", [CLIENT_ID]]]
SELECT  1 AS one FROM "oauth_access_grants" WHERE "oauth_access_grants"."token" = '1e106492b56a4b1ee25a8a8c33aefdb596a050a8cb29
INSERT INTO "oauth_access_grants" ("application_id", "resource_owner_id", "expires_in", "redirect_uri", "scopes", "token",
"created_at") VALUES (?, ?, ?, ?, ?, ?, ?) [["application_id", 17], ["resource_owner_id", 640], ["expires_in", 600],
["redirect_uri", "https://example.math.cnrs.fr/auth/plm/callback"],  ["scopes", "public profile"],
["token", "1e106492b56a4b1ee25a8a8c33aefdb596a050a8cb297fc6b8ba4c476930cb34"], ["created_at", "2016-09-27 14:33:25.333596"]]

Redirected to https://example.math.cnrs.fr/auth/plm/callback?code=1e106492b56a4b1ee25a8a8c33aefdb596a050a8cb297fc6b8ba4c476930c
```

Nous avons un jetons d'accès

Exemple d'authentification OAuth2

Obtention d'un jetons d'accès aux APIs

```
POST "/sp/oauth/token"
Processing by Doorkeeper::TokensController#create as */*
Parameters: {
  "client_id"=>[CLIENT_ID],
  "client_secret"=>"[CLIENT_SECRET]",
  "code"=>"1e106492b56a4b1ee25a8a8c33aefdb596a050a8cb297fc6b8ba4c476930cb34",
  "grant_type"=>"authorization_code",
  "redirect_uri"=>"https://example.math.cnrs.fr/auth/plm/callback"
  }

SELECT  "oauth_access_grants".* FROM "oauth_access_grants" WHERE "oauth_access_grants"."token" = ? LIMIT 1
[["token", "1e106492b56a4b1ee25a8a8c33aefdb596a050a8cb297fc6b8ba4c476930cb34"]]
SELECT  "oauth_applications".* FROM "oauth_applications" WHERE "oauth_applications"."uid" = ? AND "oauth_applications"."secret"
SELECT  "oauth_access_grants".* FROM "oauth_access_grants" WHERE "oauth_access_grants"."id" = ? LIMIT 1   [["id", 13800]]
UPDATE "oauth_access_grants" SET "revoked_at" = ? WHERE "oauth_access_grants"."id" = ?  [["revoked_at", "2016-09-27 14:33:25.41
SELECT  "oauth_applications".* FROM "oauth_applications" WHERE "oauth_applications"."id" = ? LIMIT 1  [["id", 17]]
SELECT  1 AS one FROM "oauth_access_tokens" WHERE "oauth_access_tokens"."token" = 'f3b1662719ef8f004777c459991637cbea24336c1a91
INSERT INTO "oauth_access_tokens" ("application_id", "resource_owner_id", "scopes", "expires_in", "created_at", "token") VALUES
[["application_id", 17], ["resource_owner_id", 640], ["scopes", "public profile"], ["expires_in", 7200],
["created_at", "2016-09-27 14:33:25.418563"], ["token", "f3b1662719ef8f004777c459991637cbea24336c1a91412456abab8df2242226"]]
SELECT  "users".* FROM "users" WHERE "users"."id" = ? LIMIT 1  [["id", 640]]
```

Exemple d'authentification OAuth2

Accès à l'API (Resource Server) permettant d'obtenir des informations utilisateur

```
GET "/sp/me"
SELECT  "oauth_access_tokens".* FROM "oauth_access_tokens" WHERE "oauth_access_tokens"."token" = ? LIMIT 1
[["token", "f3b1662719ef8f004777c459991637cbea24336c1a91412456abab8df2242226"]]
SELECT  "oauth_access_tokens".* FROM "oauth_access_tokens" WHERE "oauth_access_tokens"."refresh_token" = ? LIMIT 1  [["refresh_
SELECT  "users".* FROM "users" WHERE "users"."id" = ? LIMIT 1  [["id", 640]]
```

mathrice

Exemple d'authentification OpenID Connect

Apparition du Nonce

```
GET "/sp/oauth/authorize?client_id=[CLIENT_ID]
&nonce=60ccd8736f77b4a54346c2014b41d27f
&redirect_uri=https%3A%2F%2Fplm.math.cnrs.fr%2Foauth-client%2Fauth%2Fplm%2Fcallback
&response_type=code&scope=openid+legacyplm
&state=9dd7fe1ac8e0d38c67bb2930cbba4f24"
Parameters:
{
  "client_id"=>[CLIENT_ID],
  "nonce"=>"60ccd8736f77b4a54346c2014b41d27f",
  "redirect_uri"=>"https://plm.math.cnrs.fr/oauth-client/auth/plm/callback",
  "response_type"=>"code", "scope"=>"openid legacyplm",
  "state"=>"9dd7fe1ac8e0d38c67bb2930cbba4f24"
}

INSERT INTO "oauth_access_grants" ("application_id", "resource_owner_id", "expires_in",
"redirect_uri", "scopes", "nonce", "token", "created_at") VALUES (?, ?, ?, ?, ?, ?, ?, ?)
[["application_id", 2], ["resource_owner_id", 640], ["expires_in", 600],
["redirect_uri", "https://plm.math.cnrs.fr/oauth-client/auth/plm/callback"],
["scopes", "openid legacyplm"], ["nonce", "60ccd8736f77b4a54346c2014b41d27f"],
["token", "b69c467796ec2085209853f24e4af0b21f25f2cd9dd3113beeeaecbdee12142b"],
["created_at", "2016-09-30 15:39:05.955547"]]

POST "/sp/oauth/token"
Parameters:
{
  "grant_type"=>"authorization_code",
  "code"=>"[FILTERED]",
  "redirect_uri"=>"https://plm.math.cnrs.fr/oauth-client/auth/plm/callback",
  "scope"=>"openid legacyplm"
}
```

Exemple d'authentification OpenID Connect

Récupération des informations relatives au IDToken

```
GET "/sp/oauth/userinfo"
SELECT  "oauth_access_tokens".* FROM "oauth_access_tokens" WHERE "oauth_access_tokens"."token" = ? LIMIT 1
[["token", "b5fad6658306f4172806bf47aa48f5ec3602c7e853eeddcfa53199d485046ff9"]]
SELECT  "oauth_access_tokens".* FROM "oauth_access_tokens" WHERE
"oauth_access_tokens"."refresh_token" = ? LIMIT 1  [["refresh_token", ""]]
SELECT  "users".* FROM "users" WHERE "users"."id" = ? LIMIT 1  [["id", 640]]

=> Retournera vers le Client :

"id_token":"eyJ0eXAiOiJKV1QiLCJraWQiOiJwbG0iLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJodHRwczovL3BsbS5tYXRoLmNucnMuZnIvc3AiLCJzdWIiOiJwa

Qui peut être déchiffré grâce à la clé publique en :

{
  "iss": "https://plm.math.cnrs.fr/sp",
  "sub": "philippe.depouilly@math.u-bordeaux.fr",
  "aud": [CLIENT_ID],
  "exp": 1475427207,
  "iat": 1475427087,
  "mailplm": "philippe.depouilly@math.u-bordeaux.fr",
  "statusplm": "plm",
  "nonce": "60ccd8736f77b4a54346c2014b41d27f"
}
```

Références

- OAuth2 https://oauth.net/2/
- JWT https://jwt.io/introduction/
- OpenID connect http://openid.net/connect/
- DoorKeeper https://github.com/doorkeeper-gem/doorkeeper
- OpenID Connect pour DoorKeeper https://github.com/playon/doorkeeper-openid_connect
- Déploiement Oauth2 et OpenID Connect sur la PLM : https://plm.wiki.math.cnrs.fr/servicesnumeriques/identites

mathrice