API Platform

Table des matières

Installation du projet

Version utilisée de PHP : 8.3

```
composer create-project symfony/skeleton forum 6.3.*
```

```
composer require symfony/orm-pack
```

```
composer require --dev symfony/maker-bundle
```

```
cp .env .env.local
```

Base de données

```
DATABASE_URL="mysql://login????:XXXXXXXXXXXXXXX@127.0.0.1:3306/dbforum?
serverVersion=10.3.39-MariaDB"
```

```
php bin/console doctrine:database:create
```

```
composer require symfony/security-bundle
```

```
php bin/console make:user
```

```
php bin/console make:entity User
```

Ajouter
nom : string de 30 caractères non nul
prenom : string de 30 caractères non nul
dateInscription : datetime non nul

```
php bin/console make:entity Message
```

titre : string 50 caractères non nul
datePoste : datetime non nul
contenu : text non nul

champ user relation ManyToOne vers User
champ parent ManyToOne vers Message

```
php bin/console make:migration
php bin/console doctrine:migrations:migrate
```

Construire un jeu d’essai

```
composer require orm-fixtures --dev
```

```
composer require fakerphp/faker
```

Dans DataFixtures :

UserFixtures.php

```
<?php

namespace App\DataFixtures;

use Doctrine\Bundle\FixturesBundle\Fixture;
use Doctrine\Persistence\ObjectManager;
use Faker\Factory;
use App\Entity\User;
use Symfony\Component\PasswordHasher\Hasher\UserPasswordHasherInterface;
class UserFixtures extends Fixture
{
    private $faker;
    private $passwordHasher;
    public function __construct(UserPasswordHasherInterface $passwordHasher){
        $this->faker=Factory::create("fr_FR");
        $this->passwordHasher= $passwordHasher;
  }

    public function load(ObjectManager $manager): void
    {
        for($i=0;$i<10;$i++){
            $user = new User();
            $user->setNom($this->faker->lastName())
            ->setPrenom($this->faker->firstName())
            ->setEmail(strtolower($user->getPrenom()).'.'.strtolower($user->getNom()).'@'.$this-
>faker->freeEmailDomain())
            ->setPassword($this->passwordHasher->hashPassword($user, strtolower($user-
>getPrenom())))
            ->setDateInscription($this->faker->dateTimeThisYear());
            $this->addReference('user'.$i, $user);
            $manager->persist($user);
        }
        $manager->flush();
    }
}
```

Vous pouvez également supprimer les accents dans le prénom puisqu’il est utilisé pour le mot de passe. Cela se fait en utilisant le slug.

```
composer require symfony/string
```

```
<?php

namespace App\DataFixtures;

use Doctrine\Bundle\FixturesBundle\Fixture;
use Doctrine\Persistence\ObjectManager;
use Faker\Factory;
use App\Entity\User;
use Symfony\Component\PasswordHasher\Hasher\UserPasswordHasherInterface;
use Symfony\Component\String\Slugger\AsciiSlugger;

class UserFixtures extends Fixture
{
    private $faker;
    private $passwordHasher;

    public function __construct(UserPasswordHasherInterface $passwordHasher){
        $this->faker=Factory::create("fr_FR");
        $this->passwordHasher= $passwordHasher;
```

```
 }

    public function load(ObjectManager $manager): void
    {
        $slugger = new AsciiSlugger();

        for($i=0;$i<10;$i++){
            $user = new User();
            $user->setNom($this->faker->lastName())
            ->setPrenom($this->faker->firstName())
            ->setEmail(strtolower($user->getPrenom()).'.'.strtolower($user->getNom()).'@'.$this-
>faker->freeEmailDomain())
            ->setPassword($this->passwordHasher->hashPassword($user, $slugger-
>slug(strtolower($user->getPrenom()))))
            ->setDateInscription($this->faker->dateTimeThisYear());
            $this->addReference('user'.$i, $user);
            $manager->persist($user);
        }
        $manager->flush();
    }
}
```

MessageFixtures.php

```
<?php

namespace App\DataFixtures;

use Doctrine\Bundle\FixturesBundle\Fixture;
use Doctrine\Persistence\ObjectManager;
use Faker\Factory;
use App\Entity\User;
use App\Entity\Message;
use Doctrine\Common\DataFixtures\DependentFixtureInterface;

class MessageFixtures extends Fixture implements DependentFixtureInterface
{
    private $faker;

    public function __construct(){
        $this->faker=Factory::create("fr_FR");

    }

    public function load(ObjectManager $manager): void
    {
        for($i=0;$i<10;$i++){
            $message = new Message();
            $message->setTitre($this->faker->sentence(3));
            $message->setDatePoste($this->faker->dateTimeThisYear());
            $message->setContenu($this->faker->paragraph());
            $message->setUser($this->getReference('user'.mt_rand(0,9)));
            $manager->persist($message);
            $this->addReference('message'.$i, $message);
        }
        for($i=0;$i<20;$i++){
            $message = new Message();
            $message->setTitre($this->faker->sentence(3));
            $message->setDatePoste($this->faker->dateTimeThisYear());
            $message->setContenu($this->faker->paragraph());
            $message->setUser($this->getReference('user'.mt_rand(0,9)));
            $message->setParent($this->getReference('message'.mt_rand(0,9)));
            $manager->persist($message);
        }

        $manager->flush();
    }

    public function getDependencies()
    {
        return [
            UserFixtures::class,
        ];
    }
}
```

```
php bin/console doctrine:fixtures:load
```

Installation et configuration d’API Platform

```
composer require api
```

```
composer require symfony/apache-pack
```

En root, dans */etc/apache2/conf-available/forum.conf*

```
Alias /forum /var/www/html/forum/public
<Directory /var/www/html/forum/public>
AllowOverride All
Order Allow,Deny
Allow from All
</Directory>
```

```
a2enconf forum.conf
```

```
systemctl reload apache2
```

User.php

```
<?php

namespace App\Entity;

use App\Repository\UserRepository;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\DBAL\Types\Types;
use Doctrine\ORM\Mapping as ORM;
use Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface;
use Symfony\Component\Security\Core\User\UserInterface;
use ApiPlatform\Metadata\ApiResource;

#[ApiResource()]
#[ORM\Entity(repositoryClass: UserRepository::class)]
class User implements UserInterface, PasswordAuthenticatedUserInterface
```

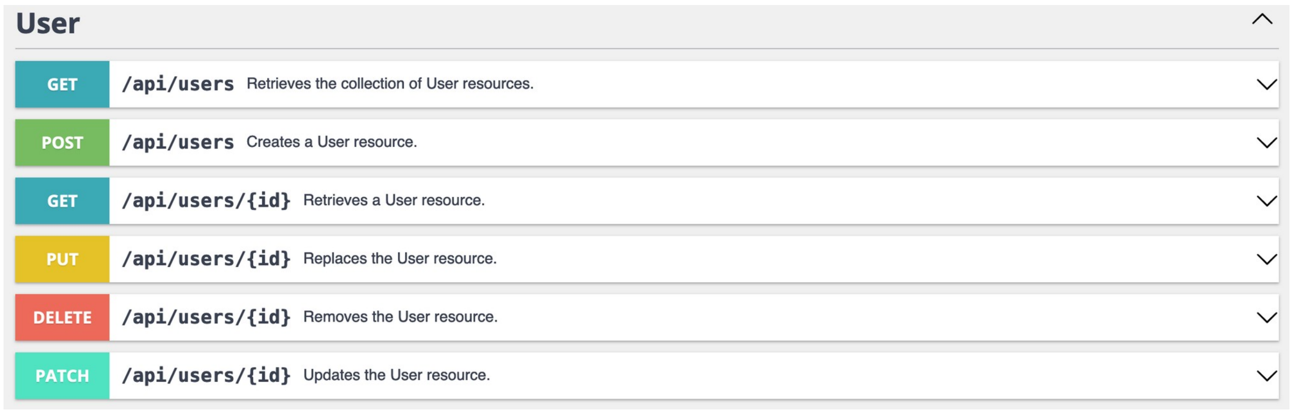
Message.php

```
<?php

namespace App\Entity;

use App\Repository\MessageRepository;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\DBAL\Types\Types;
use Doctrine\ORM\Mapping as ORM;
use ApiPlatform\Metadata\ApiResource;

#[ApiResource()]
#[ORM\Entity(repositoryClass: MessageRepository::class)]
class Message
```

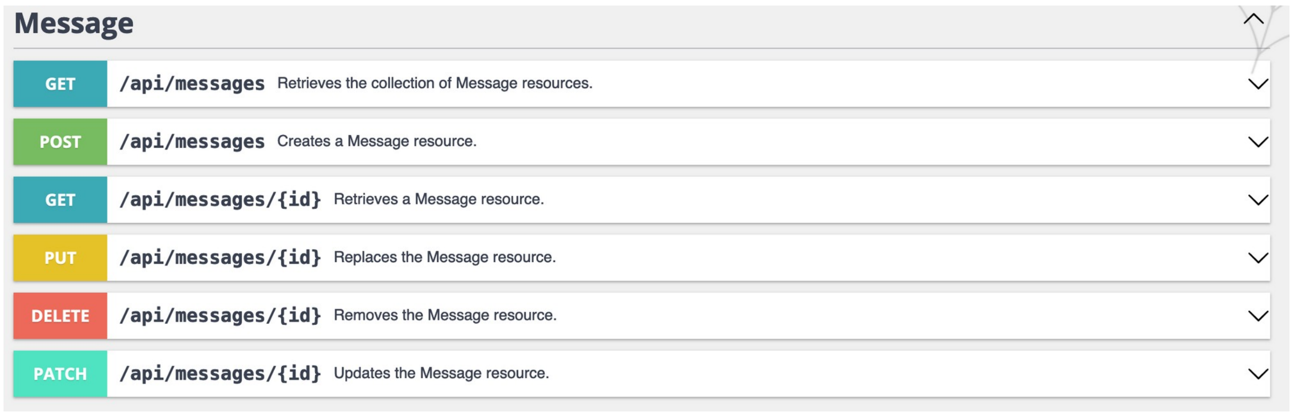
Changer la route de la documentation (catalogue des API)
Dans config/routes.yaml

```
api_doc:
   path: /doc
   controller: api_platform.swagger_ui.action
```

https://s3-4987.nuage-peda.fr/forum/doc


User

GET /api/users Retrieves the collection of User resources.
POST /api/users Creates a User resource.
GET /api/users/{id} Retrieves a User resource.
PUT /api/users/{id} Replaces the User resource.
DELETE /api/users/{id} Removes the User resource.
PATCH /api/users/{id} Updates the User resource.


Message

GET /api/messages Retrieves the collection of Message resources.
POST /api/messages Creates a Message resource.
GET /api/messages/{id} Retrieves a Message resource.
PUT /api/messages/{id} Replaces the Message resource.
DELETE /api/messages/{id} Removes the Message resource.
PATCH /api/messages/{id} Updates the Message resource.

Si vous ne voyez pas de changement dans le catalogue des API, pensez à faire la commande suivante :

```
php bin/console cache:clear
```

collectionOperation
itemOperation

Récupérer l'ensemble des utilisateurs

```
curl -X 'GET' \
  'https://s3-4987.nuage-peda.fr/forum/api/users?page=1' \
  -H 'accept: application/ld+json'
```

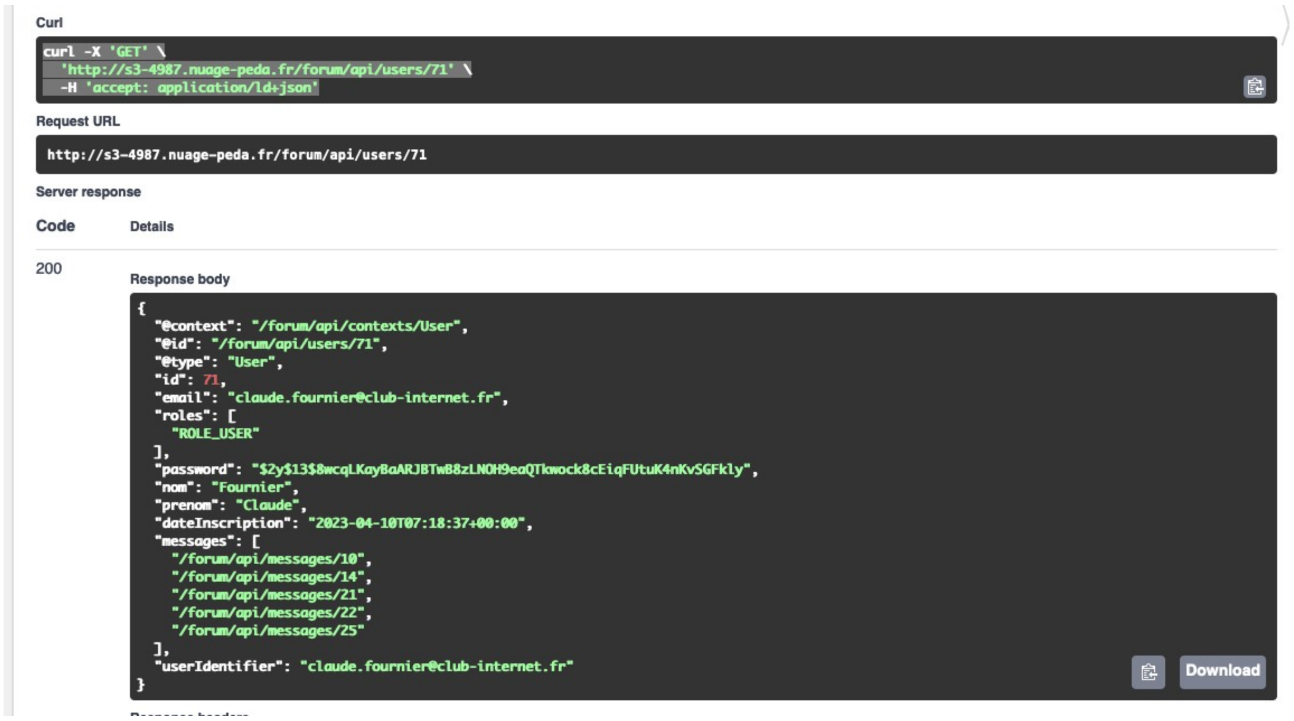
Le résultat est en JSON.

```
{
  "@context": "/forum/api/contexts/User",
  "@id": "/forum/api/users",
  "@type": "hydra:Collection",
  "hydra:totalItems": 10,
  "hydra:member": [
    {
      "@id": "/forum/api/users/61",
      "@type": "User",
      "id": 61,
      "email": "xavier.becker@noos.fr",
      "roles": [
        "ROLE_USER"
      ],
      "password":
"$2y$13$QRpBm3F4RtNGm0Qxl2WxK.CfCn9hb1v7GJ6vpuC0zr3HY4dCSP9Xi",
      "nom": "Becker",
      "prenom": "Xavier",
      "dateInscription": "2023-04-07T14:32:16+00:00",
      "messages": [
        "/forum/api/messages/5",
        "/forum/api/messages/7",
        "/forum/api/messages/13"
      ],
      "userIdentifier": "xavier.becker@noos.fr"
    },
    {
      "@id": "/forum/api/users/62",
      "@type": "User",
      "id": 62,
      "email": "thibaut.rodriguez@laposte.net",
      "roles": [
        "ROLE_USER"
      ],
      "password": "$2y$13$nIUxV9ikoV.V/yE.PKcsdOnSnTpmox1/9iBqCzV53RSYGtrm/Eif2",
      "nom": "Rodriguez",
      "prenom": "Thibaut",
      "dateInscription": "2023-03-02T07:37:10+00:00",
      "messages": [
        "/forum/api/messages/10",
        "/forum/api/messages/23"
      ],
      "userIdentifier": "thibaut.rodriguez@laposte.net"
    },
    {
      "@id": "/forum/api/users/63",
      "@type": "User",
      "id": 63,
      "email": "astrid.rousseau@orange.fr",
      "roles": [
        "ROLE_USER"
      ],
      "password":
"$2y$13$7wBFiKlLDr0Y5WmR5nnbz.z5Qh3JpV4ax3swY7jbyOMMoy1Rykeyu",
      "nom": "Rousseau",
      "prenom": "Astrid",
      "dateInscription": "2023-06-19T21:46:00+00:00",
      "messages": [
      "/forum/api/messages/15",
      "/forum/api/messages/16",
      "/forum/api/messages/19",
      "/forum/api/messages/24",
      "/forum/api/messages/27",
      "/forum/api/messages/28"
    ],
    "userIdentifier": "astrid.rousseau@orange.fr"
  },
  {
    "@id": "/forum/api/users/64",
    "@type": "User",
    "id": 64,
    "email": "amélie.guillou@noos.fr",
    "roles": [
      "ROLE_USER"
    ],
    "password": "$2y$13$84tFsHGh7prxbJ/r37hLw.Xfphb4WKsPFRT78nr9ndBrdNswQI0Le",
    "nom": "Guillou",
    "prenom": "Amélie",
    "dateInscription": "2023-07-26T07:32:19+00:00",
    "messages": [
      "/forum/api/messages/4",
      "/forum/api/messages/22"
    ],
    "userIdentifier": "amélie.guillou@noos.fr"
  },
  {
    "@id": "/forum/api/users/65",
    "@type": "User",
    "id": 65,
    "email": "léon.marechal@wanadoo.fr",
    "roles": [
      "ROLE_USER"
    ],
    "password":
"$2y$13$ek0T9sK5gPFuS6VcFX3vhuNddiDu1.nvBxQJiLW1wzEpgV9OSVYIy",
    "nom": "Marechal",
    "prenom": "Léon",
    "dateInscription": "2023-01-26T10:07:39+00:00",
    "messages": [
      "/forum/api/messages/1",
      "/forum/api/messages/3",
      "/forum/api/messages/6",
      "/forum/api/messages/30"
    ],
    "userIdentifier": "léon.marechal@wanadoo.fr"
  },
  {
    "@id": "/forum/api/users/66",
    "@type": "User",
    "id": 66,
    "email": "marc.boucher@free.fr",
    "roles": [
      "ROLE_USER"
    ],
    "password":
"$2y$13$kLcRkXBJeYoEcfNm7MdX6.n41NdCG3bhbDTI8MFJr1ztLeQUAPrmK",
    "nom": "Boucher",
    "prenom": "Marc",
    "dateInscription": "2023-01-04T14:55:57+00:00",
    "messages": [
      "/forum/api/messages/2",
      "/forum/api/messages/8",
      "/forum/api/messages/18",
      "/forum/api/messages/29"
    ],
    "userIdentifier": "marc.boucher@free.fr"
  },
  {
    "@id": "/forum/api/users/67",
    "@type": "User",
    "id": 67,
    "email": "alice.lenoir@free.fr",
    "roles": [
      "ROLE_USER"
    ],
    "password":
"$2y$13$Ow1NwgD5C1gUrPvrLX0gh.JwKnCaWFl6giBnJAAucTCgYU/Snv6J.",
    "nom": "Lenoir",
    "prenom": "Alice",
    "dateInscription": "2023-06-24T17:11:18+00:00",
    "messages": [
      "/forum/api/messages/9",
      "/forum/api/messages/12",
      "/forum/api/messages/17",
      "/forum/api/messages/21"
    ],
    "userIdentifier": "alice.lenoir@free.fr"
  },
  {
    "@id": "/forum/api/users/68",
    "@type": "User",
    "id": 68,
    "email": "constance.renard@free.fr",
    "roles": [
      "ROLE_USER"
    ],
    "password":
"$2y$13$Tjyi8WUgT8hJjp4.uUqTX.NND9c5XUmR.9ZF5HHGFJYvOt7Z/Sr92",
      "nom": "Renard",
      "prenom": "Constance",
      "dateInscription": "2023-05-23T00:54:13+00:00",
      "messages": [
        "/forum/api/messages/11",
        "/forum/api/messages/14",
        "/forum/api/messages/26"
      ],
      "userIdentifier": "constance.renard@free.fr"
    },
    {
      "@id": "/forum/api/users/69",
      "@type": "User",
      "id": 69,
      "email": "jean.guyon@noos.fr",
      "roles": [
        "ROLE_USER"
      ],
      "password": "$2y$13$6.N84K308s8mnir1WvuImuFm1roxKcIRtlFYoKc5nSxfnyYuTpf/a",
      "nom": "Guyon",
      "prenom": "Jean",
      "dateInscription": "2023-01-16T22:49:44+00:00",
      "messages": [
        "/forum/api/messages/20"
      ],
      "userIdentifier": "jean.guyon@noos.fr"
    },
    {
      "@id": "/forum/api/users/70",
      "@type": "User",
      "id": 70,
      "email": "dominique.julien@free.fr",
      "roles": [
        "ROLE_USER"
      ],
      "password": "$2y$13$b2.jgB/Xz/upE.9E3olbKOdvH3Mt6DbqJeMjZbDyNi7go9yuRVEe.",
      "nom": "Julien",
      "prenom": "Dominique",
      "dateInscription": "2023-01-28T16:09:12+00:00",
      "messages": [
        "/forum/api/messages/25"
      ],
      "userIdentifier": "dominique.julien@free.fr"
    }
  ]
}
```

Commentaires :
"@id": "/forum/api/users/70"
la clé @id représente une URI (Uniform Resource Identifier) unique qui identifie la ressource.

Récupérer qu’un seul utilisateur :

```
curl -X 'GET' \
  'http://s3-4987.nuage-peda.fr/forum/api/users/71' \
  -H 'accept: application/ld+json'
```


Curl

```
curl -X 'GET' \
  'http://s3-4987.nuage-peda.fr/forum/api/users/71' \
  -H 'accept: application/ld+json'
```

Request URL

```
http://s3-4987.nuage-peda.fr/forum/api/users/71
```

Server response

Code Details

200

Response body

```
{
  "@context": "/forum/api/contexts/User",
  "@id": "/forum/api/users/71",
  "@type": "User",
  "id": 71,
  "email": "claude.fournier@club-internet.fr",
  "roles": [
    "ROLE_USER"
  ],
  "password": "$2y$13$8wcqLKayBaARJBTwB8zLNOH9eaQTkwock8cEiqFUtuK4nKvSGFkly",
  "nom": "Fournier",
  "prenom": "Claude",
  "dateInscription": "2023-04-10T07:18:37+00:00",
  "messages": [
    "/forum/api/messages/10",
    "/forum/api/messages/14",
    "/forum/api/messages/21",
    "/forum/api/messages/22",
    "/forum/api/messages/25"
  ],
  "userIdentifier": "claude.fournier@club-internet.fr"
}
```

Pagination

Modifier la pagination pour l’ensemble des API

Par défaut, il affiche les 30 premiers éléments de la table. Vous pouvez changer cette valeur par pour toutes les API dans le fichier api_platform.yaml

```
api_platform:
    title: Hello API Platform
    version: 1.0.0
    # Good defaults for REST APIs
    defaults:
        stateless: true
        cache_headers:
            vary: ['Content-Type', 'Authorization', 'Origin']
        extra_properties:
            standard_put: true
        pagination_items_per_page: 5
```

```
curl -X 'GET' \
  'https://s3-4987.nuage-peda.fr/forum/api/users?page=1' \
  -H 'accept: application/ld+json'
```

Voici le résultat si je limite à 5 le nombre de résultats par page :

```
{
  "@context": "/forum/api/contexts/User",
  "@id": "/forum/api/users",
  "@type": "hydra:Collection",
  "hydra:totalItems": 10,
  "hydra:member": [
    {
      "@id": "/forum/api/users/61",
      "@type": "User",
      "id": 61,
      "email": "xavier.becker@noos.fr",
      "roles": [
        "ROLE_USER"
      ],
      "password":
"$2y$13$QRpBm3F4RtNGm0Qxl2WxK.CfCn9hb1v7GJ6vpuC0zr3HY4dCSP9Xi",
      "nom": "Becker",
      "prenom": "Xavier",
      "dateInscription": "2023-04-07T14:32:16+00:00",
      "messages": [
        "/forum/api/messages/5",
        "/forum/api/messages/7",
        "/forum/api/messages/13"
      ],
      "userIdentifier": "xavier.becker@noos.fr"
    },
    {
      "@id": "/forum/api/users/62",
      "@type": "User",
      "id": 62,
      "email": "thibaut.rodriguez@laposte.net",
      "roles": [
        "ROLE_USER"
      ],
      "password": "$2y$13$nIUxV9ikoV.V/yE.PKcsdOnSnTpmox1/9iBqCzV53RSYGtrm/Eif2",
      "nom": "Rodriguez",
      "prenom": "Thibaut",
      "dateInscription": "2023-03-02T07:37:10+00:00",
      "messages": [
        "/forum/api/messages/10",
        "/forum/api/messages/23"
      ],
      "userIdentifier": "thibaut.rodriguez@laposte.net"
    },
    {
      "@id": "/forum/api/users/63",
      "@type": "User",
      "id": 63,
      "email": "astrid.rousseau@orange.fr",
      "roles": [
        "ROLE_USER"
      ],
      "password":
"$2y$13$7wBFiKlLDr0Y5WmR5nnbz.z5Qh3JpV4ax3swY7jbyOMMoy1Rykeyu",
      "nom": "Rousseau",
      "prenom": "Astrid",
      "dateInscription": "2023-06-19T21:46:00+00:00",
      "messages": [
        "/forum/api/messages/15",
        "/forum/api/messages/16",
        "/forum/api/messages/19",
        "/forum/api/messages/24",
        "/forum/api/messages/27",
        "/forum/api/messages/28"
      ],
      "userIdentifier": "astrid.rousseau@orange.fr"
    },
    {
      "@id": "/forum/api/users/64",
      "@type": "User",
      "id": 64,
      "email": "amélie.guillou@noos.fr",
      "roles": [
        "ROLE_USER"
      ],
      "password": "$2y$13$84tFsHGh7prxbJ/r37hLw.Xfphb4WKsPFRT78nr9ndBrdNswQI0Le",
      "nom": "Guillou",
      "prenom": "Amélie",
      "dateInscription": "2023-07-26T07:32:19+00:00",
      "messages": [
        "/forum/api/messages/4",
        "/forum/api/messages/22"
      ],
      "userIdentifier": "amélie.guillou@noos.fr"
    },
    {
      "@id": "/forum/api/users/65",
      "@type": "User",
      "id": 65,
      "email": "léon.marechal@wanadoo.fr",
      "roles": [
        "ROLE_USER"
      ],
      "password":
"$2y$13$ek0T9sK5gPFuS6VcFX3vhuNddiDu1.nvBxQJiLW1wzEpgV9OSVYIy",
      "nom": "Marechal",
      "prenom": "Léon",
      "dateInscription": "2023-01-26T10:07:39+00:00",
      "messages": [
        "/forum/api/messages/1",
        "/forum/api/messages/3",
        "/forum/api/messages/6",
        "/forum/api/messages/30"
      ],
      "userIdentifier": "léon.marechal@wanadoo.fr"
    }
  ],
  "hydra:view": {
    "@id": "/forum/api/users?page=1",
    "@type": "hydra:PartialCollectionView",
    "hydra:first": "/forum/api/users?page=1",
    "hydra:last": "/forum/api/users?page=2",
    "hydra:next": "/forum/api/users?page=2"
  }
}
```

Commentaires :

Le résultat me renvoie le nombre total de valeurs dans la table et me donne l’adresse de la première page de résultat, la dernière et la prochaine.

Pagination personnalisée sur chaque entité

```php
<?php

namespace App\Entity;

use App\Repository\MessageRepository;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\DBAL\Types\Types;
use Doctrine\ORM\Mapping as ORM;
use ApiPlatform\Metadata\ApiResource;

#[ApiResource(paginationItemsPerPage: 10)]
#[ORM\Entity(repositoryClass: MessageRepository::class)]
class Message
```

Commentaire :
Nous aurons au maximum 10 messages par page.

Désactiver la pagination

Nous pouvons désactiver la pagination et tout renvoyer. Ce qui n’est pas forcement intéressant suivant le volume de données.

```
api_platform:
    defaults:
        pagination_enabled: false
```

Les Groups

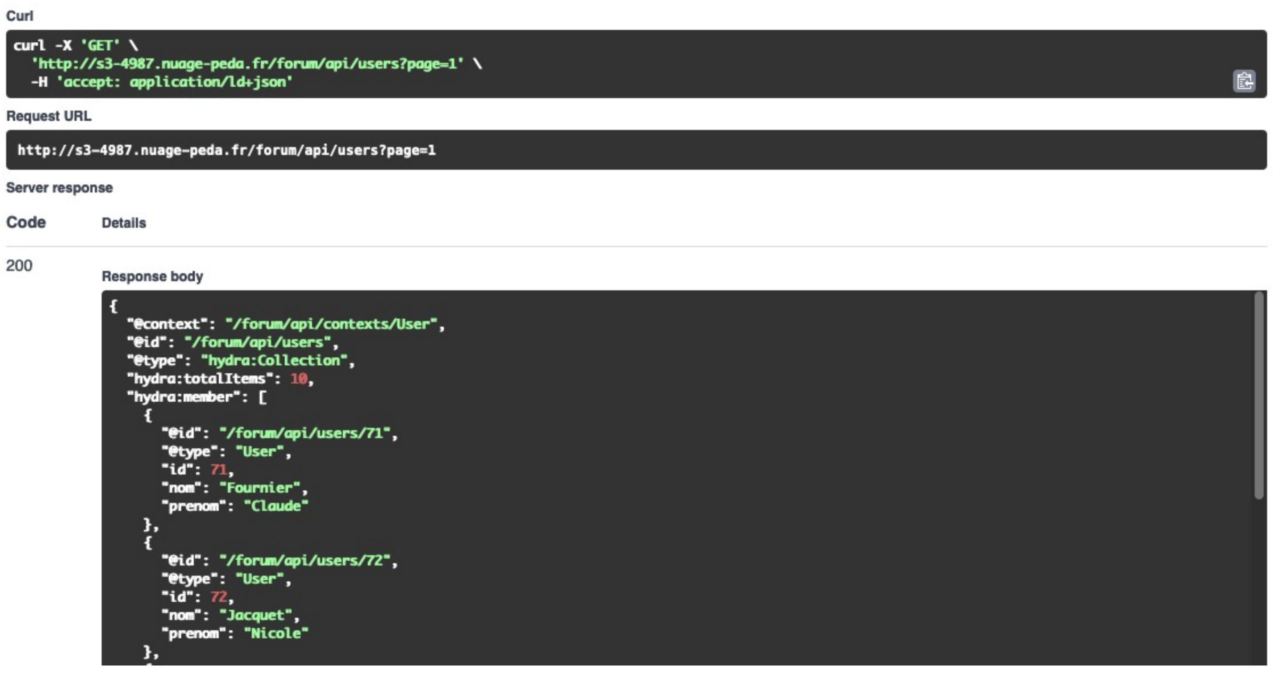
```
<?php

namespace App\Entity;

use App\Repository\UserRepository;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\DBAL\Types\Types;
use Doctrine\ORM\Mapping as ORM;
use Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface;
use Symfony\Component\Security\Core\User\UserInterface;
use ApiPlatform\Metadata\ApiResource;
use Symfony\Component\Serializer\Annotation\Groups;
use ApiPlatform\Metadata\Get;
use ApiPlatform\Metadata\GetCollection;
use ApiPlatform\Metadata\Patch;
use ApiPlatform\Metadata\Delete;
use ApiPlatform\Metadata\Post;
use ApiPlatform\Metadata\Put;

#[ApiResource(operations: [
            new GetCollection(normalizationContext: ['groups' => 'user:list']),
            new Post(),
            new Get(normalizationContext: ['groups' => 'user:item']),
            new Put(),
            new Patch(),
            new Delete(),

        ],)]
#[ORM\Entity(repositoryClass: UserRepository::class)]
class User implements UserInterface, PasswordAuthenticatedUserInterface
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    #[Groups(['user:list', 'user:item'])]
    private ?int $id = null;
```

```
#[ORM\Column(length: 180, unique: true)]
#[Groups(['user:item'])]
private ?string $email = null;

#[ORM\Column]
private array $roles = [];

/**
 * @var string The hashed password
 */
#[ORM\Column]
private ?string $password = null;

#[ORM\Column(length: 30)]
#[Groups(['user:list', 'user:item'])]
private ?string $nom = null;

#[ORM\Column(length: 30)]
#[Groups(['user:list', 'user:item'])]
private ?string $prenom = null;

#[ORM\Column(type: Types::DATETIME_MUTABLE)]
private ?\DateTimeInterface $dateInscription = null;

#[ORM\OneToMany(mappedBy: 'user', targetEntity: Message::class, orphanRemoval: true)]
private Collection $messages;

public function __construct()
{
    $this->messages = new ArrayCollection();
}

public function getId(): ?int
{
    return $this->id;
}

public function getEmail(): ?string
{
    return $this->email;
}

public function setEmail(string $email): static
{
    $this->email = $email;

    return $this;
}
```

```
/**
 * A visual identifier that represents this user.
 *
 * @see UserInterface
 */
public function getUserIdentifier(): string
{
    return (string) $this->email;
}

/**
 * @see UserInterface
 */
public function getRoles(): array
{
    $roles = $this->roles;
    // guarantee every user at least has ROLE_USER
    $roles[] = 'ROLE_USER';

    return array_unique($roles);
}

public function setRoles(array $roles): static
{
    $this->roles = $roles;

    return $this;
}

/**
 * @see PasswordAuthenticatedUserInterface
 */
public function getPassword(): string
{
    return $this->password;
}

public function setPassword(string $password): static
{
    $this->password = $password;

    return $this;
}

/**
 * @see UserInterface
 */
public function eraseCredentials(): void
{
    // If you store any temporary, sensitive data on the user, clear it here
```

```
        // $this->plainPassword = null;
    }

    public function getNom(): ?string
    {
        return $this->nom;
    }

    public function setNom(string $nom): static
    {
        $this->nom = $nom;

        return $this;
    }

    public function getPrenom(): ?string
    {
        return $this->prenom;
    }

    public function setPrenom(string $prenom): static
    {
        $this->prenom = $prenom;

        return $this;
    }

    public function getDateInscription(): ?\DateTimeInterface
    {
        return $this->dateInscription;
    }

    public function setDateInscription(\DateTimeInterface $dateInscription): static
    {
        $this->dateInscription = $dateInscription;

        return $this;
    }

    /**
     * @return Collection<int, Message>
     */
    public function getMessages(): Collection
    {
        return $this->messages;
    }

    public function addMessage(Message $message): static
    {
        if (!$this->messages->contains($message)) {
            $this->messages->add($message);
            $message->setUser($this);
        }

        return $this;
    }

    public function removeMessage(Message $message): static
    {
        if ($this->messages->removeElement($message)) {
            // set the owning side to null (unless already changed)
            if ($message->getUser() === $this) {
                $message->setUser(null);
            }
        }

        return $this;
    }
}
```

Lorsque j’appelle l’api GET /users je ne vois pas les emails dans le résultat


Curl

```
curl -X 'GET' \
  'http://s3-4987.nuage-peda.fr/forum/api/users?page=1' \
  -H 'accept: application/ld+json'
```

Request URL

```
http://s3-4987.nuage-peda.fr/forum/api/users?page=1
```

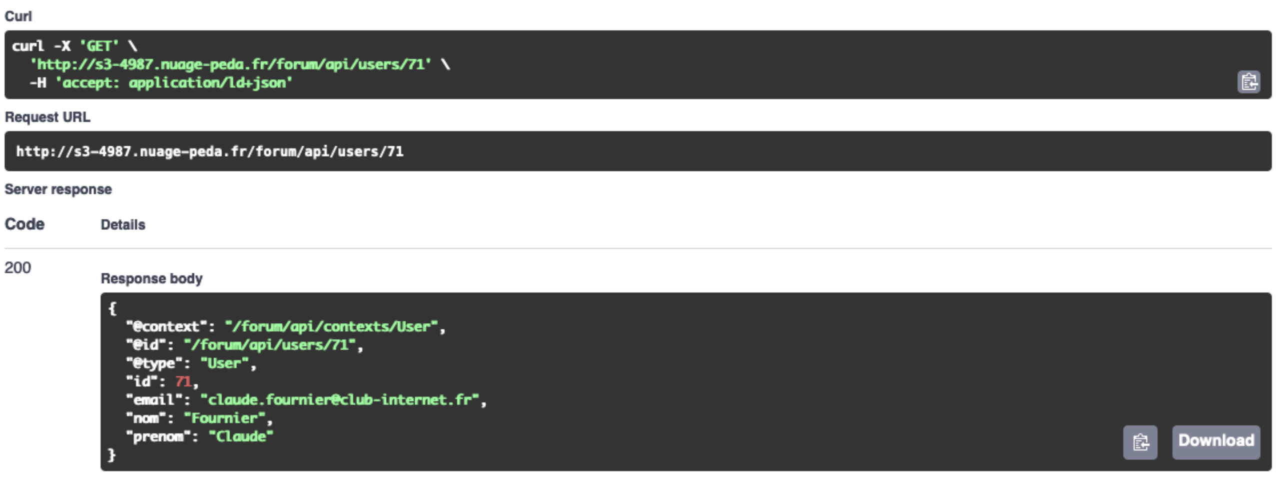
Server response

Code Details

200 Response body

```
{
  "@context": "/forum/api/contexts/User",
  "@id": "/forum/api/users",
  "@type": "hydra:Collection",
  "hydra:totalItems": 10,
  "hydra:member": [
    {
      "@id": "/forum/api/users/71",
      "@type": "User",
      "id": 71,
      "nom": "Fournier",
      "prenom": "Claude"
    },
    {
      "@id": "/forum/api/users/72",
      "@type": "User",
      "id": 72,
      "nom": "Jacquet",
      "prenom": "Nicole"
    },
```

Lorsque j’appelle l’api GET /users/71 l’email apparaît bien dans le résultat


Curl

```
curl -X 'GET' \
  'http://s3-4987.nuage-peda.fr/forum/api/users/71' \
  -H 'accept: application/ld+json'
```

Request URL

```
http://s3-4987.nuage-peda.fr/forum/api/users/71
```

Server response

Code Details

200

Response body

```
{
  "@context": "/forum/api/contexts/User",
  "@id": "/forum/api/users/71",
  "@type": "User",
  "id": 71,
  "email": "claude.fournier@club-internet.fr",
  "nom": "Fournier",
  "prenom": "Claude"
}
```

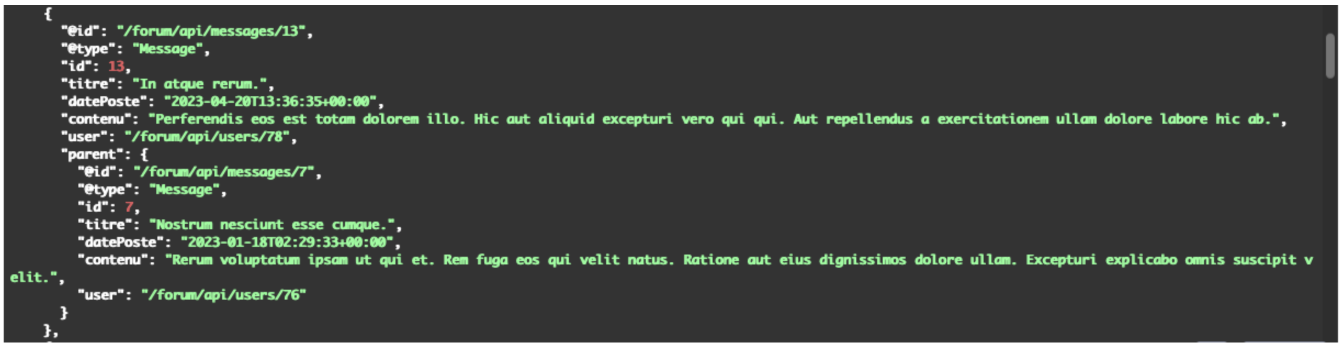
```
<?php

namespace App\Entity;

use App\Repository\MessageRepository;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\DBAL\Types\Types;
use Doctrine\ORM\Mapping as ORM;
use ApiPlatform\Metadata\ApiResource;
use Symfony\Component\Serializer\Annotation\Groups;
use ApiPlatform\Metadata\Get;
use ApiPlatform\Metadata\GetCollection;
use ApiPlatform\Metadata\Patch;
use ApiPlatform\Metadata\Delete;
use ApiPlatform\Metadata\Post;
use ApiPlatform\Metadata\Put;

#[ApiResource(paginationItemsPerPage: 10,operations: [
    new GetCollection(normalizationContext: ['groups' => 'message:list']),
    new Post(),
    new Get(normalizationContext: ['groups' => 'message:item']),
    new Put(),
    new Patch(),
    new Delete(),

],)]

#[ORM\Entity(repositoryClass: MessageRepository::class)]
class Message
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
```

```php
#[Groups(['message:list', 'message:item'])]
private ?int $id = null;

#[ORM\Column(length: 50)]
#[Groups(['message:list', 'message:item'])]
private ?string $titre = null;

#[ORM\Column(type: Types::DATETIME_MUTABLE)]
#[Groups(['message:list', 'message:item'])]
private ?\DateTimeInterface $datePoste = null;

#[ORM\Column(type: Types::TEXT)]
#[Groups(['message:list', 'message:item'])]
private ?string $contenu = null;

#[ORM\ManyToOne(inversedBy: 'messages')]
#[ORM\JoinColumn(nullable: false)]
#[Groups(['message:list', 'message:item'])]
private ?User $user = null;

#[ORM\ManyToOne(targetEntity: self::class, inversedBy: 'messages')]
#[ORM\JoinColumn(nullable: true)]
#[Groups(['message:list', 'message:item'])]
private ?self $parent = null;

#[ORM\OneToMany(mappedBy: 'parent', targetEntity: self::class, orphanRemoval: true)]
private Collection $messages;

public function __construct()
{
    $this->messages = new ArrayCollection();
}

public function getId(): ?int
{
    return $this->id;
}

public function getTitre(): ?string
{
    return $this->titre;
}

public function setTitre(string $titre): static
{
    $this->titre = $titre;

    return $this;
}
```

```
public function getDatePoste(): ?\DateTimeInterface
{
   return $this->datePoste;
}

public function setDatePoste(\DateTimeInterface $datePoste): static
{
   $this->datePoste = $datePoste;

   return $this;
}

public function getContenu(): ?string
{
   return $this->contenu;
}

public function setContenu(string $contenu): static
{
   $this->contenu = $contenu;

   return $this;
}

public function getUser(): ?User
{
   return $this->user;
}

public function setUser(?User $user): static
{
   $this->user = $user;

   return $this;
}

public function getParent(): ?self
{
   return $this->parent;
}

public function setParent(?self $parent): static
{
   $this->parent = $parent;

   return $this;
}

/**
 * @return Collection<int, self>
```

```
     */
    public function getMessages(): Collection
    {
        return $this->messages;
    }

    public function addMessage(self $message): static
    {
        if (!$this->messages->contains($message)) {
            $this->messages->add($message);
            $message->setParent($this);
        }

        return $this;
    }

    public function removeMessage(self $message): static
    {
        if ($this->messages->removeElement($message)) {
            // set the owning side to null (unless already changed)
            if ($message->getParent() === $this) {
                $message->setParent(null);
            }
        }

        return $this;
    }
}
```

```
http://s3-4987.nuage-peda.fr/forum/api/messages?page=1
```



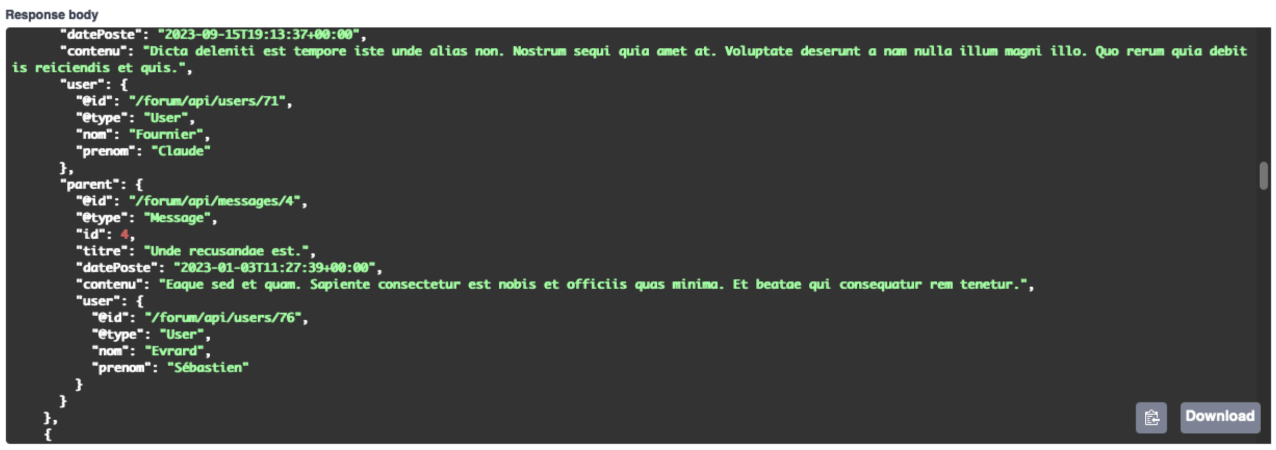
```
    {
      "@id": "/forum/api/messages/13",
      "@type": "Message",
      "id": 13,
      "titre": "In atque rerum.",
      "datePoste": "2023-04-20T13:36:35+00:00",
      "contenu": "Perferendis eos est totam dolorem illo. Hic aut aliquid excepturi vero qui qui. Aut repellendus a exercitationem ullam dolore labore hic ab.",
      "user": "/forum/api/users/78",
      "parent": {
        "@id": "/forum/api/messages/7",
        "@type": "Message",
        "id": 7,
        "titre": "Nostrum nesciunt esse cumque.",
        "datePoste": "2023-01-18T02:29:33+00:00",
        "contenu": "Rerum voluptatum ipsam ut qui et. Rem fuga eos qui velit natus. Ratione aut eius dignissimos dolore ullam. Excepturi explicabo omnis suscipit v
elit.",
        "user": "/forum/api/users/76"
      }
    },
```

Commentaires :

Nous voyons le détail du message parent. En revanche, nous ne voyons pas le détail de l'utilisateur.

```
<?php

namespace App\Entity;
```

```
use App\Repository\UserRepository;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\DBAL\Types\Types;
use Doctrine\ORM\Mapping as ORM;
use Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface;
use Symfony\Component\Security\Core\User\UserInterface;
use ApiPlatform\Metadata\ApiResource;
use Symfony\Component\Serializer\Annotation\Groups;
use ApiPlatform\Metadata\Get;
use ApiPlatform\Metadata\GetCollection;
use ApiPlatform\Metadata\Patch;
use ApiPlatform\Metadata\Delete;
use ApiPlatform\Metadata\Post;
use ApiPlatform\Metadata\Put;

#[ApiResource(operations: [
            new GetCollection(normalizationContext: ['groups' => 'user:list']),
            new Post(),
            new Get(normalizationContext: ['groups' => 'user:item']),
            new Put(),
            new Patch(),
            new Delete(),

        ],)]
#[ORM\Entity(repositoryClass: UserRepository::class)]
class User implements UserInterface, PasswordAuthenticatedUserInterface
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    #[Groups(['user:list', 'user:item'])]
    private ?int $id = null;

    #[ORM\Column(length: 180, unique: true)]
    #[Groups(['user:item'])]
    private ?string $email = null;

    #[ORM\Column]
    private array $roles = [];

    /**
     * @var string The hashed password
     */
    #[ORM\Column]
    private ?string $password = null;

    #[ORM\Column(length: 30)]
    #[Groups(['user:list', 'user:item','message:list', 'message:item'])]
    private ?string $nom = null;

    #[ORM\Column(length: 30)]
    #[Groups(['user:list', 'user:item','message:list', 'message:item'])]
    private ?string $prenom = null;
```



```
      "datePoste": "2023-09-15T19:13:37+00:00",
      "contenu": "Dicta deleniti est tempore iste unde alias non. Nostrum sequi quia amet at. Voluptate deserunt a nam nulla illum magni illo. Quo rerum quia debit
is reiciendis et quis.",
      "user": {
        "@id": "/forum/api/users/71",
        "@type": "User",
        "nom": "Fournier",
        "prenom": "Claude"
      },
      "parent": {
        "@id": "/forum/api/messages/4",
        "@type": "Message",
        "id": 4,
        "titre": "Unde recusandae est.",
        "datePoste": "2023-01-03T11:27:39+00:00",
        "contenu": "Eaque sed et quam. Sapiente consectetur est nobis et officiis quas minima. Et beatae qui consequatur rem tenetur.",
        "user": {
          "@id": "/forum/api/users/76",
          "@type": "User",
          "nom": "Evrard",
          "prenom": "Sébastien"
        }
      }
    },
    {
```

Commentaire :
Nous avons maintenant le détail de l'utilisateur.

Les filtres

Recherches

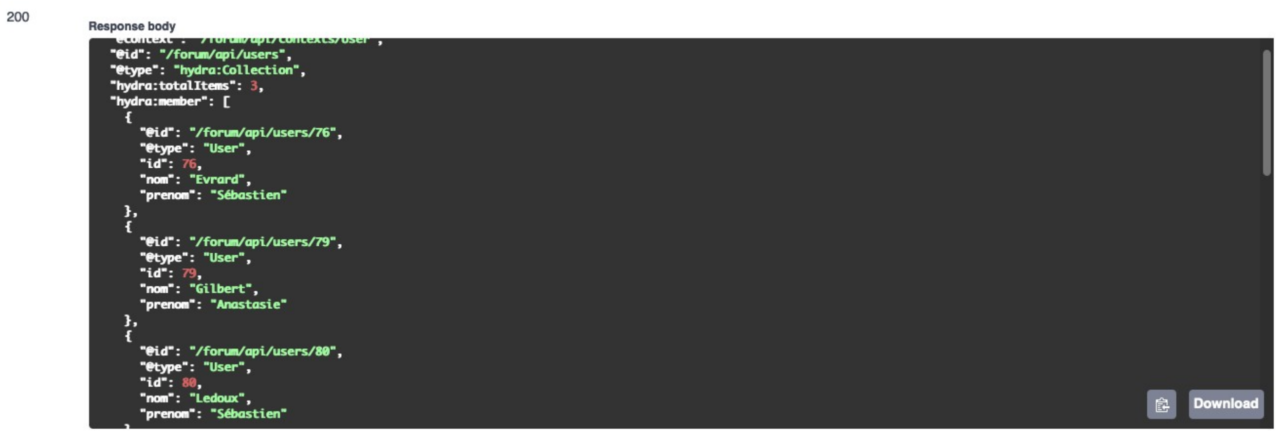
```
<?php

namespace App\Entity;

use App\Repository\UserRepository;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\DBAL\Types\Types;
use Doctrine\ORM\Mapping as ORM;
use Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface;
use Symfony\Component\Security\Core\User\UserInterface;
use ApiPlatform\Metadata\ApiResource;
use Symfony\Component\Serializer\Annotation\Groups;
use ApiPlatform\Metadata\Get;
use ApiPlatform\Metadata\GetCollection;
use ApiPlatform\Metadata\Patch;
use ApiPlatform\Metadata\Delete;
use ApiPlatform\Metadata\Post;
use ApiPlatform\Metadata\Put;
use ApiPlatform\Metadata\ApiFilter;
use ApiPlatform\Doctrine\Orm\Filter\SearchFilter;
```

```
#[ApiResource(operations: [
        new GetCollection(normalizationContext: ['groups' => 'user:list']),
        new Post(),
        new Get(normalizationContext: ['groups' => 'user:item']),
        new Put(),
        new Patch(),
        new Delete(),

    ],)]
#[ApiFilter(SearchFilter::class, properties: ['id' => 'exact', 'nom' => 'exact', 'prenom' => 'partial'])]
#[ORM\Entity(repositoryClass: UserRepository::class)]
class User implements UserInterface, PasswordAuthenticatedUserInterface
```

Commentaires :
Lorsque nous recherchons un utilisateur avec la propriété « exact », il est nécessaire de mettre sa valeur complète alors que « partial » signifie qu’il est inutile de connaître la valeur complète.

Exemple : Liste des tous les utilisateurs qui possèdent un « s » dans leur prénom

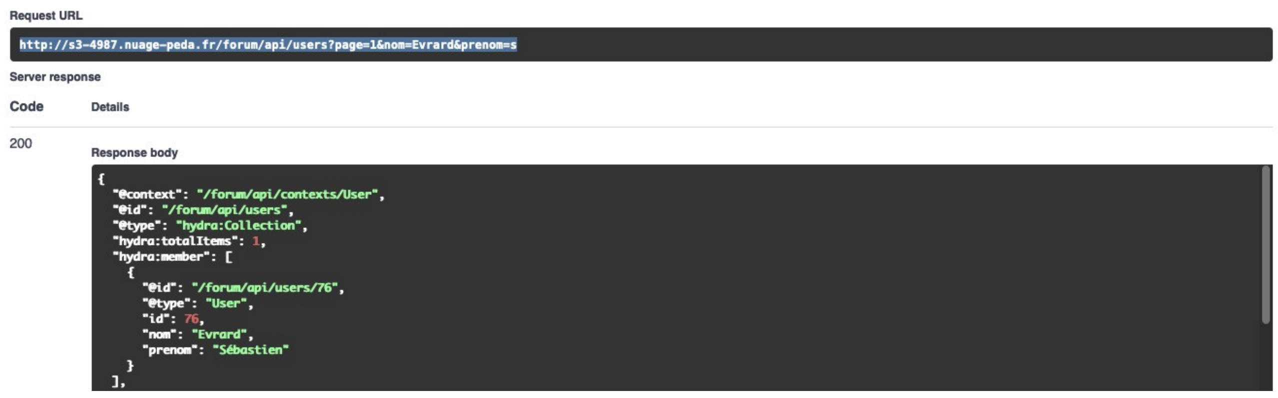
```
curl -X 'GET' \
  'http://s3-4987.nuage-peda.fr/forum/api/users?page=1&prenom=s' \
  -H 'accept: application/ld+json'
```



```
  "@id": "/forum/api/users",
  "@type": "hydra:Collection",
  "hydra:totalItems": 3,
  "hydra:member": [
    {
      "@id": "/forum/api/users/76",
      "@type": "User",
      "id": 76,
      "nom": "Evrard",
      "prenom": "Sébastien"
    },
    {
      "@id": "/forum/api/users/79",
      "@type": "User",
      "id": 79,
      "nom": "Gilbert",
      "prenom": "Anastasie"
    },
    {
      "@id": "/forum/api/users/80",
      "@type": "User",
      "id": 80,
      "nom": "Ledoux",
      "prenom": "Sébastien"
```

Nous pouvons combiner les filtres.
Exemple : cherchons un utilisateur qui a un « s » dans on prénom et dont le nom est « Evrard ».

```
http://s3-4987.nuage-peda.fr/forum/api/users?page=1&nom=Evrard&prenom=s
```


Request URL

```
http://s3-4987.nuage-peda.fr/forum/api/users?page=1&nom=Evrard&prenom=s
```

Server response

Code Details

200

Response body

```
{
  "@context": "/forum/api/contexts/User",
  "@id": "/forum/api/users",
  "@type": "hydra:Collection",
  "hydra:totalItems": 1,
  "hydra:member": [
    {
      "@id": "/forum/api/users/76",
      "@type": "User",
      "id": 76,
      "nom": "Evrard",
      "prenom": "Sébastien"
    }
  ],
```

Chercher tous les messages d’un utilisateur donné en paramètre

```
<?php

namespace App\Entity;

use App\Repository\MessageRepository;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\DBAL\Types\Types;
use Doctrine\ORM\Mapping as ORM;
use ApiPlatform\Metadata\ApiResource;
use Symfony\Component\Serializer\Annotation\Groups;
use ApiPlatform\Metadata\Get;
use ApiPlatform\Metadata\GetCollection;
use ApiPlatform\Metadata\Patch;
use ApiPlatform\Metadata\Delete;
use ApiPlatform\Metadata\Post;
use ApiPlatform\Metadata\Put;
use ApiPlatform\Metadata\ApiFilter;
use ApiPlatform\Doctrine\Orm\Filter\SearchFilter;

#[ApiResource(paginationItemsPerPage: 10,operations: [
    new GetCollection(normalizationContext: ['groups' => 'message:list']),
    new Post(),
    new Get(normalizationContext: ['groups' => 'message:item']),
    new Put(),
    new Patch(),
    new Delete(),

],)]
#[ApiFilter(SearchFilter::class, properties: ['user' => 'exact'])]
```

```
curl -X 'GET' \ 'http://s3-4987.nuage-peda.fr/forum/api/messages?page=1&user=71' \ -H 'accept:
application/ld+json'
```

Tri

```
<?php

namespace App\Entity;

use App\Repository\MessageRepository;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\DBAL\Types\Types;
use Doctrine\ORM\Mapping as ORM;
use ApiPlatform\Metadata\ApiResource;
use Symfony\Component\Serializer\Annotation\Groups;
use ApiPlatform\Metadata\Get;
use ApiPlatform\Metadata\GetCollection;
use ApiPlatform\Metadata\Patch;
use ApiPlatform\Metadata\Delete;
use ApiPlatform\Metadata\Post;
use ApiPlatform\Metadata\Put;
use ApiPlatform\Metadata\ApiFilter;
use ApiPlatform\Doctrine\Orm\Filter\SearchFilter;
use ApiPlatform\Doctrine\Orm\Filter\OrderFilter;

#[ApiResource(paginationItemsPerPage: 10,operations: [
    new GetCollection(normalizationContext: ['groups' => 'message:list']),
    new Post(),
    new Get(normalizationContext: ['groups' => 'message:item']),
    new Put(),
    new Patch(),
    new Delete(),

],)]
#[ApiFilter(SearchFilter::class, properties: ['user' => 'exact'])]
#[ApiFilter(OrderFilter::class, properties: ['id' => 'ASC', 'titre' => 'DESC'])]
```

```
curl -X 'GET' \
  'http://s3-4987.nuage-peda.fr/forum/api/messages?page=1&order%5Bid%5D=asc&order
%5Btitre%5D=desc' \
  -H 'accept: application/ld+json'
```

Il existe énormément de filtres. Vous pouvez accéder à la documentation officielle ici https://api-platform.com/docs/core/filters/

Gestion des droits d’accès

```
composer req lexik/jwt-authentication-bundle
```

```
php bin/console lexik:jwt:generate-keypair
```

Copier le bloc qui ressemble à celui-ci du .env au .env.local

```
###> lexik/jwt-authentication-bundle ###
JWT_SECRET_KEY=%kernel.project_dir%/config/jwt/private.pem
JWT_PUBLIC_KEY=%kernel.project_dir%/config/jwt/public.pem
JWT_PASSPHRASE=53803e4d703e90b0c93346542f5dd5556e94df3b3b1256fe56148c1b0de2a45
5
###< lexik/jwt-authentication-bundle ###
```

Dans config/packages/lexik_jwt_authentication.yaml

```
lexik_jwt_authentication:
    secret_key: '%env(resolve:JWT_SECRET_KEY)%'
    public_key: '%env(resolve:JWT_PUBLIC_KEY)%'
    pass_phrase: '%env(JWT_PASSPHRASE)%'
    token_ttl: 3600
```

Dans config/routes.yaml

```
authentication_token:
    path: /api/authentication_token
    methods: ['POST']
```

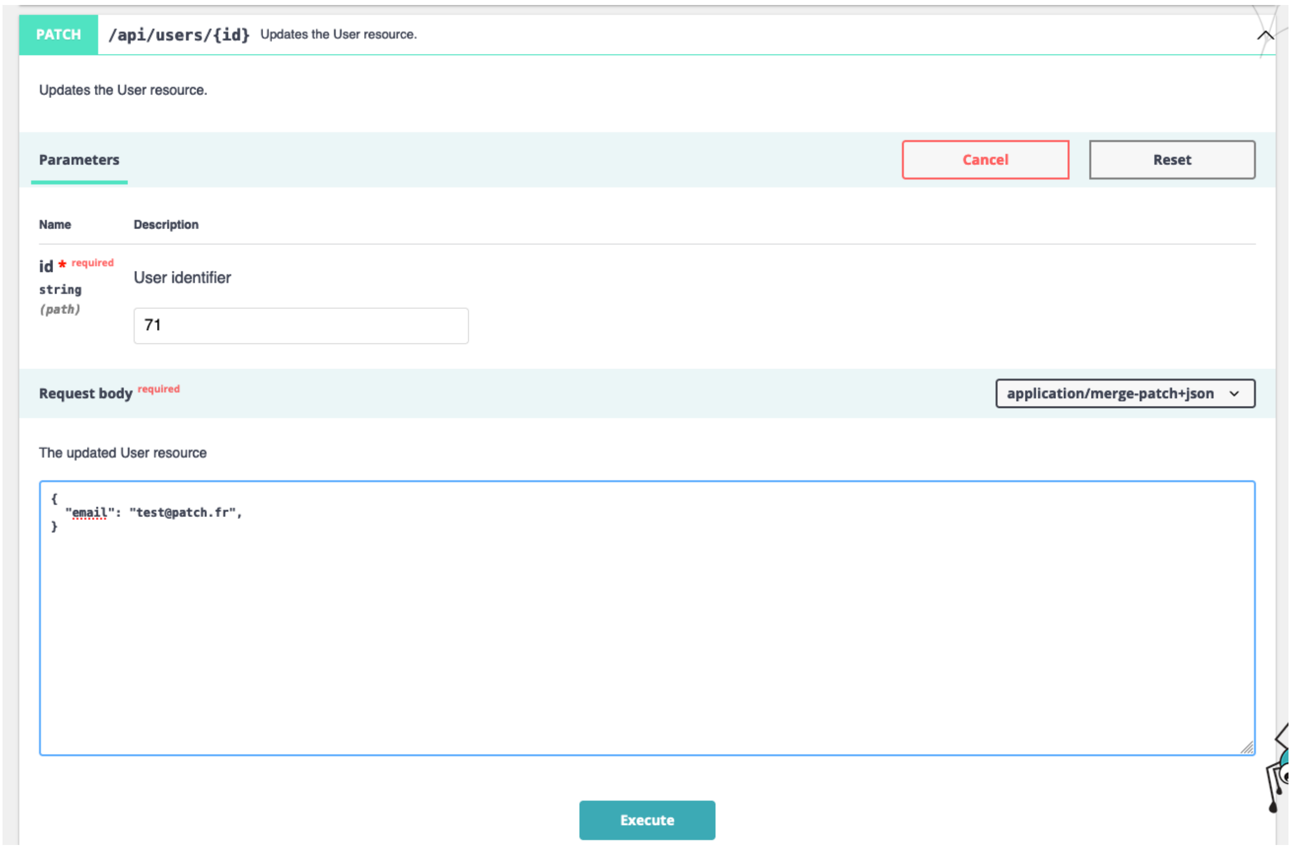
```
php bin/console debug:router
```

config/packages/security.yaml

```
 main:
            provider: app_user_provider
            pattern: ^/api
            stateless: true
            json_login:
                username_path: email
                check_path: /api/authentication_token
                success_handler: lexik_jwt_authentication.handler.authentication_success
                failure_handler: lexik_jwt_authentication.handler.authentication_failure
            jwt: ~
```

```
use ApiPlatform\Metadata\Put;
use ApiPlatform\Metadata\ApiFilter;
use ApiPlatform\Doctrine\Orm\Filter\SearchFilter;
```

```
#[ApiResource(operations: [
        new GetCollection(normalizationContext: ['groups' => 'user:list']),
        new Post(),
        new Get(normalizationContext: ['groups' => 'user:item']),
        new Put(),
        new Patch(security: "is_granted('ROLE_ADMIN') or object == user"),
        new Delete(),

    ],)]
```

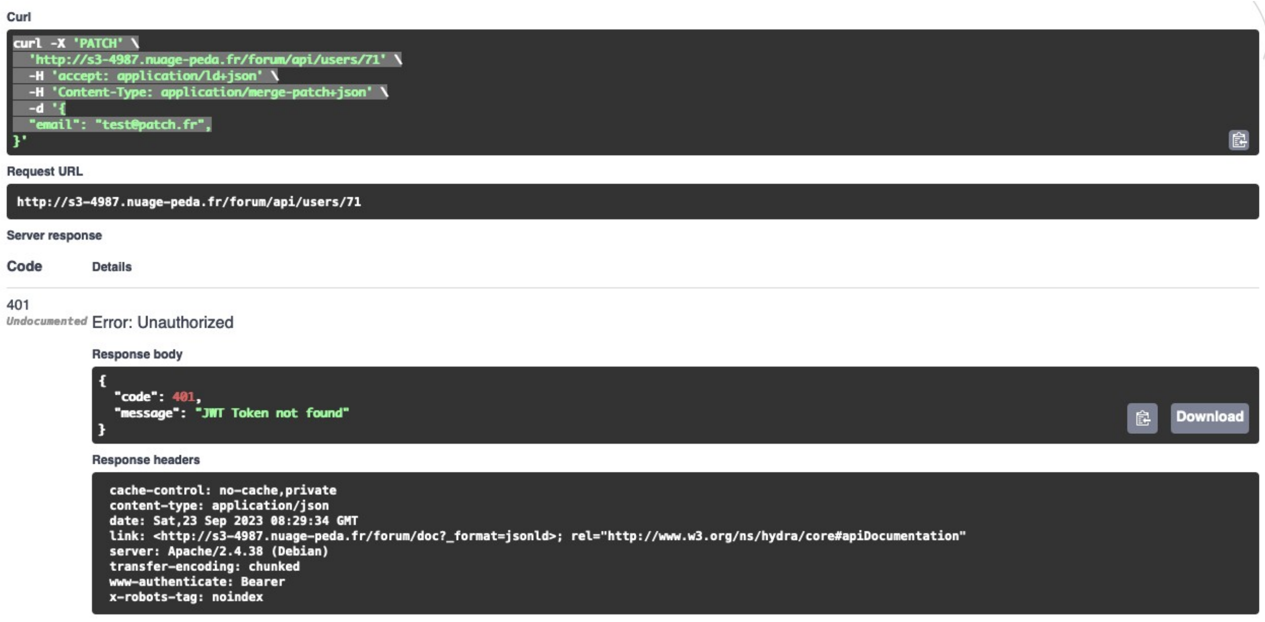

PATCH
/api/users/{id} Updates the User resource.
Updates the User resource.
Parameters
Cancel
Reset
Name
Description
id * required
string
(path)
User identifier
71
Request body required
application/merge-patch+json
The updated User resource
Execute

```
{
  "email": "test@patch.fr",
}
```

Je tente de modifier l’email de l’utilisateur 71. Adaptez par rapport à votre base de données.

```
curl -X 'PATCH' \
  'http://s3-4987.nuage-peda.fr/forum/api/users/71' \
  -H 'accept: application/ld+json' \
  -H 'Content-Type: application/merge-patch+json' \
  -d '{
  "email": "test@patch.fr",
}'
```

Voici le résultat :


Curl

```
curl -X 'PATCH' \
  'http://s3-4987.nuage-peda.fr/forum/api/users/71' \
  -H 'accept: application/ld+json' \
  -H 'Content-Type: application/merge-patch+json' \
  -d '{
  "email": "test@patch.fr",
}'
```

Request URL

```
http://s3-4987.nuage-peda.fr/forum/api/users/71
```

Server response

Code Details

401
Undocumented Error: Unauthorized

Response body

```
{
  "code": 401,
  "message": "JWT Token not found"
}
```

Response headers

```
cache-control: no-cache,private
content-type: application/json
date: Sat,23 Sep 2023 08:29:34 GMT
link: <http://s3-4987.nuage-peda.fr/forum/doc?_format=jsonld>; rel="http://www.w3.org/ns/hydra/core#apiDocumentation"
server: Apache/2.4.38 (Debian)
transfer-encoding: chunked
www-authenticate: Bearer
x-robots-tag: noindex
```

Commentaire :

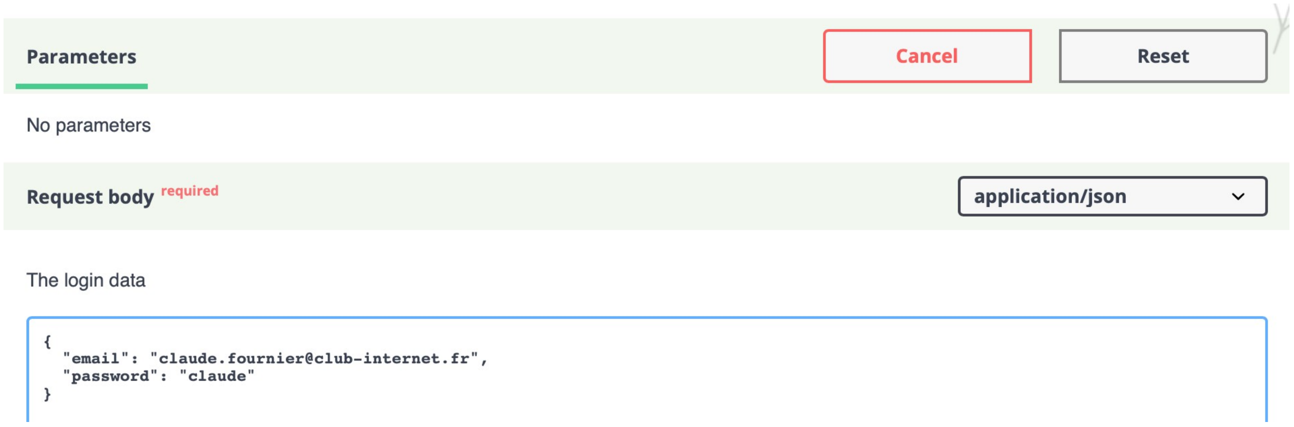
L’utilisation de l’API étant protégé, il envoie une erreur 401 car le token n’a pas été trouvé.

Dans /*src*/OpenApi/JwtDecorator.php

```
<?php
// api/src/OpenApi/JwtDecorator.php
declare(strict_types=1);
namespace App\OpenApi;
use ApiPlatform\OpenApi\Factory\OpenApiFactoryInterface;
use ApiPlatform\OpenApi\OpenApi;
use ApiPlatform\OpenApi\Model;
final class JwtDecorator implements OpenApiFactoryInterface
{
    public function __construct(
        private OpenApiFactoryInterface $decorated
    ) {}
    public function __invoke(array $context = []): OpenApi
    {
        $openApi = ($this->decorated)($context);
        $schemas = $openApi->getComponents()->getSchemas();
        $schemas['Token'] = new \ArrayObject([
            'type' => 'object',
            'properties' => [
                'token' => [
                    'type' => 'string',
                    'readOnly' => true,
                ],
            ],
        ]);
        $schemas['Credentials'] = new \ArrayObject([
            'type' => 'object',
            'properties' => [
                'email' => [
                    'type' => 'string',
                    'example' => 'johndoe@example.com',
                ],
                'password' => [
                    'type' => 'string',
                    'example' => 'apassword',
                ],
            ],
        ]);
        $schemas = $openApi->getComponents()->getSecuritySchemes() ?? [];
        $schemas['JWT'] = new \ArrayObject([
            'type' => 'http',
            'scheme' => 'bearer',
            'bearerFormat' => 'JWT',
        ]);

        $pathItem = new Model\PathItem(
            ref: 'JWT Token',
            post: new Model\Operation(
                operationId: 'postCredentialsItem',
                tags: ['Token'],
                responses: [
                    '200' => [
                        'description' => 'Get JWT token',
                        'content' => [
                            'application/json' => [
                                'schema' => [
                                    '$ref' => '#/components/schemas/Token',
                                ],
                            ],
                        ],
                    ],
                ],
                summary: 'Get JWT token to login.',
                requestBody: new Model\RequestBody(
                    description: 'Generate new JWT Token',
                    content: new \ArrayObject([
                        'application/json' => [
                            'schema' => [
                                '$ref' => '#/components/schemas/Credentials',
                            ],
                        ],
                    ]),
                ),
                security: [],
            ),
        );
        $openApi->getPaths()->addPath('/authentication_token', $pathItem);
        return $openApi;
    }
}
```

Dans config/services.yaml :

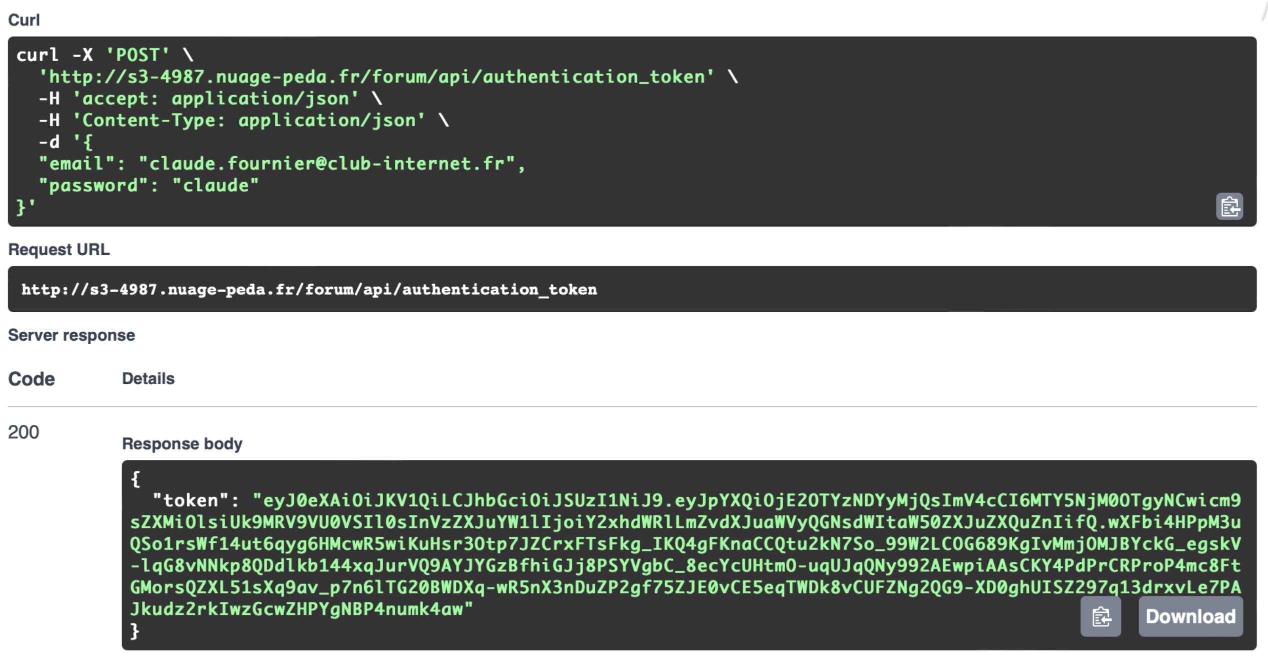
```
# This file is the entry point to configure your own services.
# Files in the packages/ subdirectory configure your dependencies.

# Put parameters here that don't need to change on each machine where the app is deployed
# https://symfony.com/doc/current/best_practices.html#use-parameters-for-application-
configuration
parameters:


services:
    # default configuration for services in *this* file
    _defaults:
        autowire: true      # Automatically injects dependencies in your services.
        autoconfigure: true # Automatically registers your services as commands, event subscribers,
etc.

    # makes classes in src/ available to be used as services
    # this creates a service per class whose id is the fully-qualified class name
    App\:
        resource: '../src/'
        exclude:
            - '../src/DependencyInjection/'
            - '../src/Entity/'
            - '../src/Kernel.php'
    App\OpenApi\JwtDecorator:
        decorates: 'api_platform.openapi.factory'
        arguments: ['@.inner']
    # add more service definitions when explicit configuration is needed
    # please note that last definitions always *replace* previous ones
```



No parameters

The login data

```
{
  "email": "claude.fournier@club-internet.fr",
  "password": "claude"
}
```


Curl

```
curl -X 'POST' \
  'http://s3-4987.nuage-peda.fr/forum/api/authentication_token' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "email": "claude.fournier@club-internet.fr",
  "password": "claude"
}'
```

Request URL

```
http://s3-4987.nuage-peda.fr/forum/api/authentication_token
```

Server response

Code Details

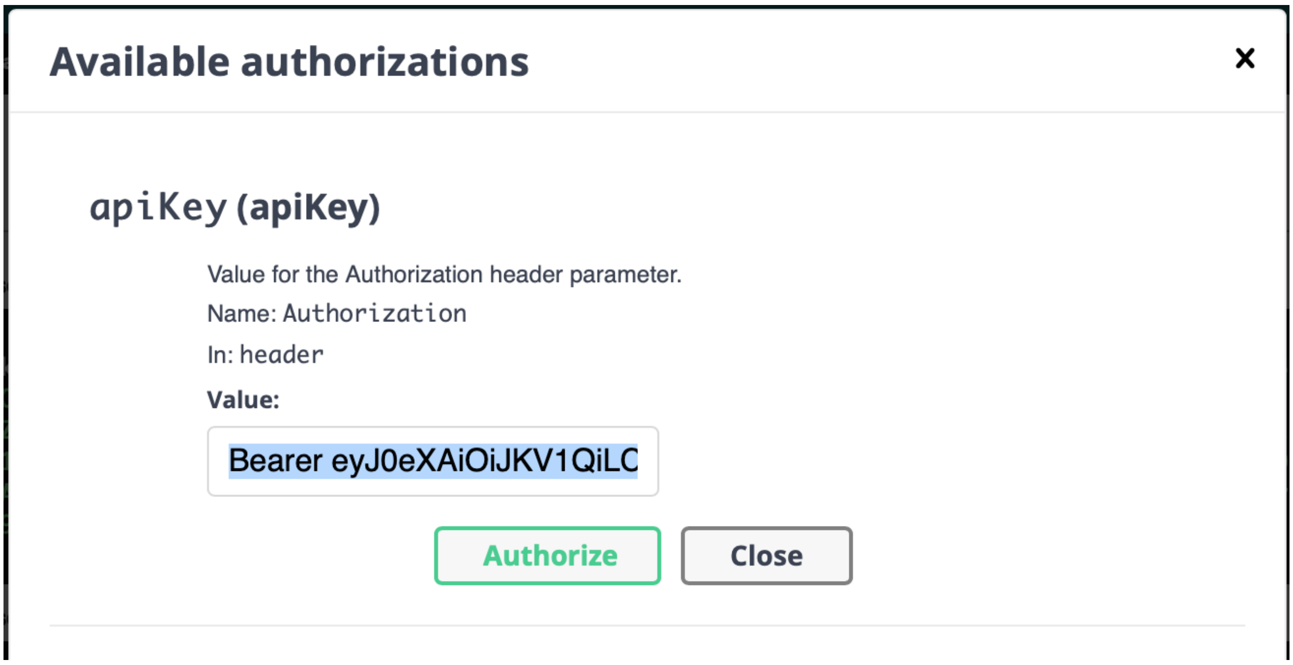
200

Response body

```
{
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpYXQiOjE2OTYzNDYyMjQsImV4cCI6MTY5NjM0OTgyNCwicm9sZXMiOlsiUk9MRV9VU0VSIl0sInVzZXJuYW1lIjoiY2xhdWRlLmZvdXJuaWVyQGNsdWItaW50ZXJuZXQuZnIifQ.wXFbi4HPpM3uQSo1rsWf14ut6qyg6HMcwR5wiKuHsr3Otp7JZCrxFTsFkg_IKQ4gFKnaCCQtu2kN7So_99W2LCOG689KgIvMmjOMJBYckG_egskV-lqG8vNNkp8QDdlkb144xqJurVQ9AYJYGzBfhiGJj8PSYVgbC_8ecYcUHtmO-uqUJqQNy992AEwpiAAsCKY4PdPrCRProP4mc8FtGMorsQZXL51sXq9av_p7n6lTG20BWDXq-wR5nX3nDuZP2gf75ZJE0vCE5eqTWDk8vCUFZNg2QG9-XD0ghUISZ297q13drxvLe7PAJkudz2rkIwzGcwZHPYgNBP4numk4aw"
}
```

config/packages/api_platform.yaml

```
api_platform:
    title: Hello API Platform
    version: 1.0.0
    # Good defaults for REST APIs
    defaults:
        stateless: true
        cache_headers:
            vary: ['Content-Type', 'Authorization', 'Origin']
        extra_properties:
            standard_put: true
        normalization_context:
            skip_null_values: true
    swagger:
        api_keys:
            apiKey:
                name: Authorization
                type: header
```



Available authorizations

apiKey (apiKey)

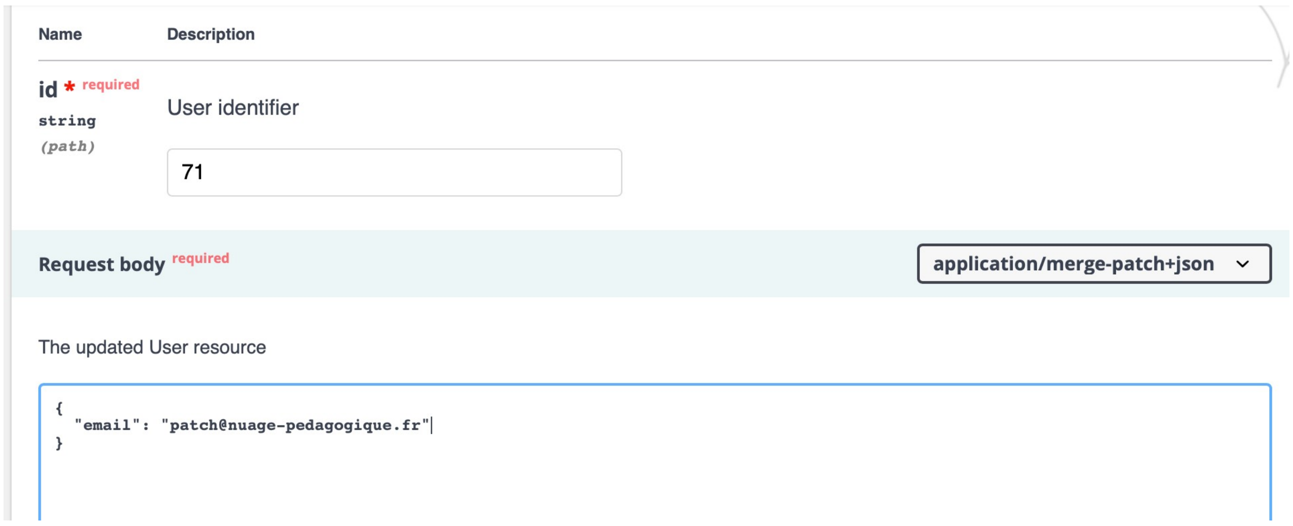
Value for the Authorization header parameter.

Name: Authorization

In: header

Value:

Bearer eyJ0eXAiOiJKV1QiLC

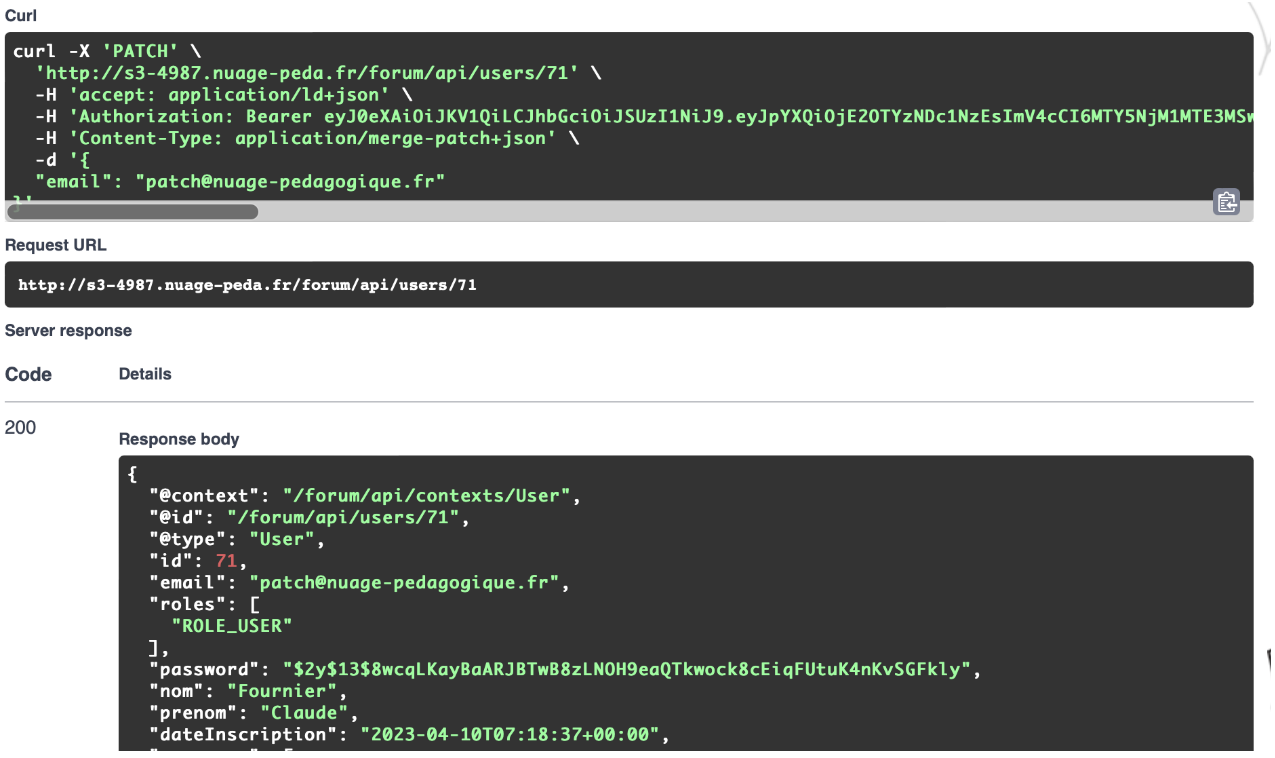


Name	Description
id * required string (path)	User identifier 71

Request body required

The updated User resource

```
{
  "email": "patch@nuage-pedagogique.fr"
}
```



Curl

```
curl -X 'PATCH' \
  'http://s3-4987.nuage-peda.fr/forum/api/users/71' \
  -H 'accept: application/ld+json' \
  -H 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpYXQiOjE2OTYzNDc1NzEsImV4cCI6MTY5NjM1MTE3MSw
  -H 'Content-Type: application/merge-patch+json' \
  -d '{
  "email": "patch@nuage-pedagogique.fr"
```

Request URL

```
http://s3-4987.nuage-peda.fr/forum/api/users/71
```

Server response

Code Details

200

Response body

```
{
  "@context": "/forum/api/contexts/User",
  "@id": "/forum/api/users/71",
  "@type": "User",
  "id": 71,
  "email": "patch@nuage-pedagogique.fr",
  "roles": [
    "ROLE_USER"
  ],
  "password": "$2y$13$8wcqLKayBaARJBTwB8zLNOH9eaQTkwock8cEiqFUtuK4nKvSGFkly",
  "nom": "Fournier",
  "prenom": "Claude",
  "dateInscription": "2023-04-10T07:18:37+00:00",
```

Chiffre le mot de passe lors d’une inscription :

php bin/console make:state-processor et taper UserStateProcessor

src/State/UserStateProcessor.php

```
<?php namespace App\State;
// https://api-platform.com/docs/core/state-processors/
use ApiPlatform\State\ProcessorInterface;
use ApiPlatform\Metadata\Operation;
use App\Entity\User;
use Symfony\Component\PasswordHasher\Hasher\UserPasswordHasherInterface;

class UserStateProcessor implements ProcessorInterface {
    private $passwordHasher;
    private $decorated;

    public function __construct(private ProcessorInterface $persistProcessor,
UserPasswordHasherInterface $passwordHasher) {
        $this->passwordHasher = $passwordHasher;
    }

    public function process($data, Operation $operation, array $uriVariables = [], array $context =
[])
    {
        // call your persistence layer to save $data
        if ($data->getPassword()) {
            $data->setPassword(
                $this->passwordHasher->hashPassword($data, $data->getPassword())
            );
        }
        $result = $this->persistProcessor->process($data, $operation, $uriVariables, $context);
        return $result;
    }
}
```

config/services.yaml

```
   App\State\UserStateProcessor:
        bind:
            $persistProcessor: '@api_platform.doctrine.orm.state.persist_processor'
```

src/Entity/User

```
<?php

namespace App\Entity;

use App\Repository\UserRepository;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\DBAL\Types\Types;
use Doctrine\ORM\Mapping as ORM;
use Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface;
use Symfony\Component\Security\Core\User\UserInterface;
use ApiPlatform\Metadata\ApiResource;
use Symfony\Component\Serializer\Annotation\Groups;
use ApiPlatform\Metadata\Get;
use ApiPlatform\Metadata\GetCollection;
use ApiPlatform\Metadata\Patch;
use ApiPlatform\Metadata\Delete;
use ApiPlatform\Metadata\Post;
use ApiPlatform\Metadata\Put;
use ApiPlatform\Metadata\ApiFilter;
use ApiPlatform\Doctrine\Orm\Filter\SearchFilter;
use App\State\UserStateProcessor;


#[ApiResource(
            operations: [
            new GetCollection(normalizationContext: ['groups' => 'user:list']),
            new Post(processor: UserStateProcessor::class),
            new Get(normalizationContext: ['groups' => 'user:item']),
            new Put(),
            new Patch(security: "is_granted('ROLE_ADMIN') or object == user"),
            new Delete(),

        ],)]
```

Recevoir des données avec l'authentification

Introduction

Dans le chapitre précédent, nous avons réussi à nous authentifier et à utiliser des API protégées. La communication se réalise avec un token. Nous allons voir dans ce chapitre, la méthode permettant de récupérer des données supplémentaires concernant l'utilisateur qui s'est authentifié.

A. Mise en place d'un écouteur d'événement

Symfony permet de gérer et de créer des événements, c'est-à-dire des situations qui ont lieu sur le site (une connexion, une inscription, etc.).
Les créateurs du package « LexikJWTAuthenticationBundle » fournissent un certain nombre d'événements. Par exemple, le moment où l'authentification a échoué ou au contraire a réussi. Vous pouvez jeter un œil dans la documentation suivante pour plus de détail : LexikJWTAuthenticationBundle

Afin de déterminer qu'un événement se produit, nous devons créer un écouteur(« Listener »). C'est celui qui nous prévient. Dans notre cas, après une authentification réussie, c'est le moment d'y ajouter des données supplémentaires. Ainsi, nous allons écrire le code qui s'occupe de compléter l'envoi du token en écrivant une nouvelle classe et une méthode qui fera ce travail.

Dans votre répertoire « src », créez le répertoire « EventListener », qui nous permettra d'ajouter une classe par événement que nous souhaitons écouter.
À l'intérieur de ce répertoire, créez le fichier « AuthenticationSuccessListener.php ».
src/EventListener/AuthenticationSuccessListener.php

```
<?php

namespace App\EventListener;
use Symfony\Component\Security\Core\User\UserInterface;
use Lexik\Bundle\JWTAuthenticationBundle\Event\AuthenticationSuccessEvent;



class AuthenticationSuccessListener{

  /**
  * @param AuthenticationSuccessEvent $event
 */
 public function onAuthenticationSuccessResponse(AuthenticationSuccessEvent $event)
 {
  $data = $event->getData();
  $user = $event->getUser();

  if (!$user instanceof UserInterface) {
      return;
  }

  $data['data'] = array(
      'roles' => $user->getRoles(),
      'id' => $user->getId(),
      'nom' => $user->getNom(),
      'prenom' => $user->getPrenom(),
  );

  $event->setData($data);
 }
}
```

Commentaires :
$data = $event->getData(); // permet de récupérer un tableau de données lié à l’événement qui a eu lieu. Nous le stockons dans la variable $data
$user = $event->getUser(); // Nous récupérons les données concernant l’utilisateur connecté.
if (!$user instanceof UserInterface) { // Nous vérifions que l’utilisateur récupéré est bien une instance de UserInterface. Dans le cas contraire, nous quittons cette méthode.
$data['data'] = array(// permet de créer une clé « data » dans le tableau « $data ».
Cette clé permet la création d’un tableau dans lequel nous stockons les rôles de l’utilisateur, son identifiant, son nom et son prénom.
$event->setData($data); // nous retournons les données complétées à l’événement.

B. Configuration du service

Maintenant que nous avons écrit notre événement, nous devons lui dire de se lancer dans le cas où une authentification s’exécute avec succès.
Dans le fichier « config/services.yaml » :

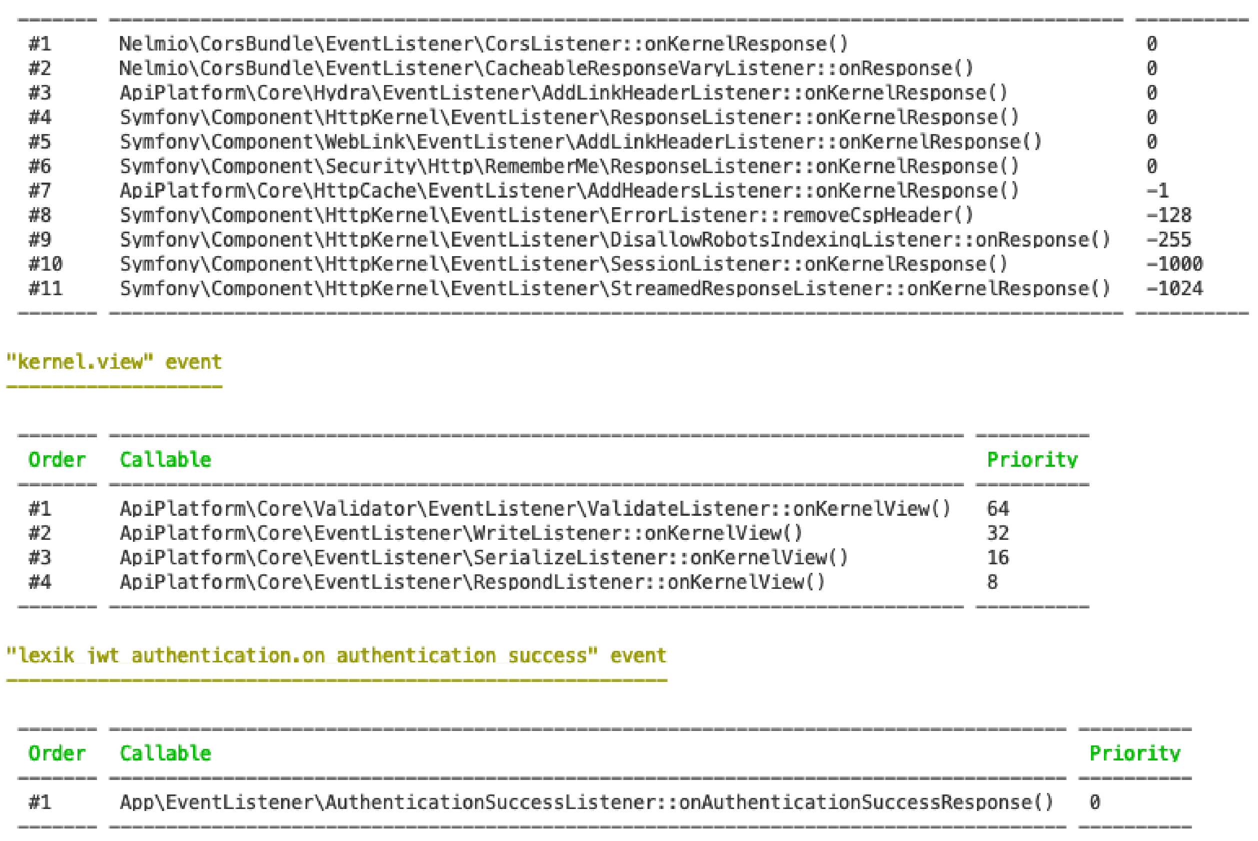
```
services:
    acme_api.event.authentication_success_listener:
        class: App\EventListener\AuthenticationSuccessListener
        tags:
            - { name: kernel.event_listener, event: lexik_jwt_authentication.on_authentication_success, method: onAuthenticationSuccessResponse }
```

Commentaires :
acme_api.event.authentication_success_listener: // nom du service à exécuter, c’est un identifiant qui doit être unique
class: App\EventListener\AuthenticationSuccessListener // e chemin vers votre nouvelle classe

name: kernel.event_listener // nom du gestionnaire d'événements
"event »: lexik_jwt_authentication.on_authentication_success // événement du package lorsque l'authentification s'est bien déroulée.
method: onAuthenticationSuccessResponse // nom de la méthode qu'il faut exécuter

Vous pouvez vérifier que votre écouteur est bien enregistré en exécutant la ligne suivante :

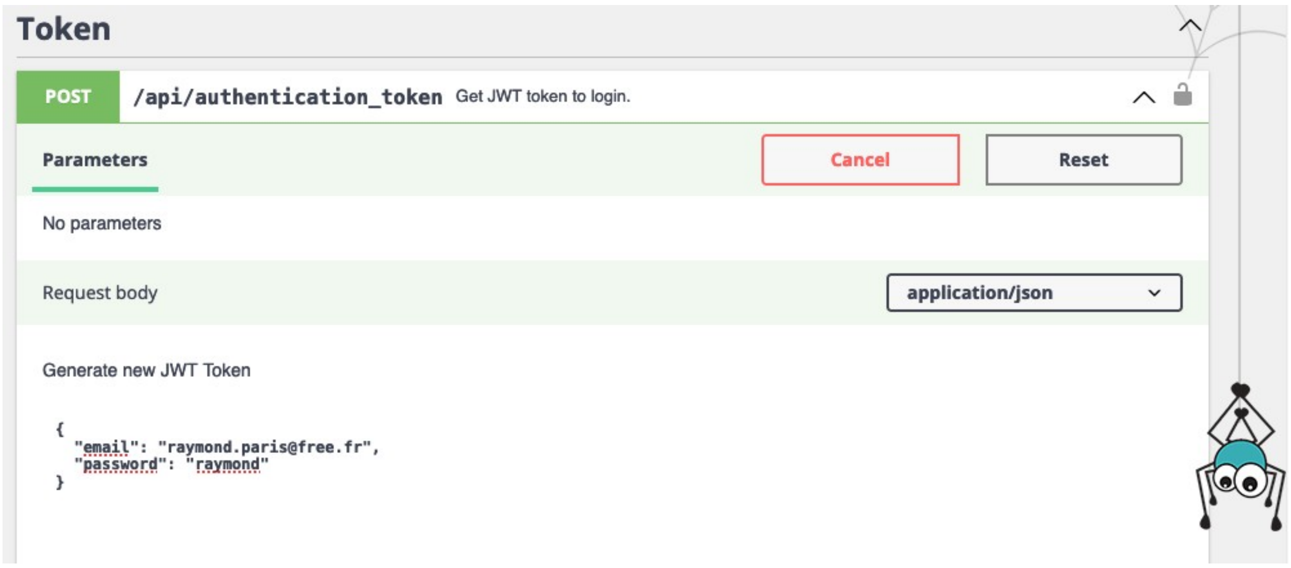
```
php bin/console debug:event-dispatcher
```



```
 ------- ------------------------------------------------------------------------------------- ----------
  #1      Nelmio\CorsBundle\EventListener\CorsListener::onKernelResponse()                       0
  #2      Nelmio\CorsBundle\EventListener\CacheableResponseVaryListener::onResponse()            0
  #3      ApiPlatform\Core\Hydra\EventListener\AddLinkHeaderListener::onKernelResponse()         0
  #4      Symfony\Component\HttpKernel\EventListener\ResponseListener::onKernelResponse()        0
  #5      Symfony\Component\WebLink\EventListener\AddLinkHeaderListener::onKernelResponse()      0
  #6      Symfony\Component\Security\Http\RememberMe\ResponseListener::onKernelResponse()        0
  #7      ApiPlatform\Core\HttpCache\EventListener\AddHeadersListener::onKernelResponse()        -1
  #8      Symfony\Component\HttpKernel\EventListener\ErrorListener::removeCspHeader()            -128
  #9      Symfony\Component\HttpKernel\EventListener\DisallowRobotsIndexingListener::onResponse()   -255
  #10     Symfony\Component\HttpKernel\EventListener\SessionListener::onKernelResponse()         -1000
  #11     Symfony\Component\HttpKernel\EventListener\StreamedResponseListener::onKernelResponse()   -1024
 ------- ------------------------------------------------------------------------------------- ----------

"kernel.view" event
-------------------

 ------- ------------------------------------------------------------------------ ----------
  Order   Callable                                                                 Priority
 ------- ------------------------------------------------------------------------ ----------
  #1      ApiPlatform\Core\Validator\EventListener\ValidateListener::onKernelView()   64
  #2      ApiPlatform\Core\EventListener\WriteListener::onKernelView()               32
  #3      ApiPlatform\Core\EventListener\SerializeListener::onKernelView()           16
  #4      ApiPlatform\Core\EventListener\RespondListener::onKernelView()             8
 ------- ------------------------------------------------------------------------ ----------

"lexik_jwt_authentication.on_authentication_success" event
----------------------------------------------------------

 ------- ---------------------------------------------------------------------------------- ----------
  Order   Callable                                                                           Priority
 ------- ---------------------------------------------------------------------------------- ----------
  #1      App\EventListener\AuthenticationSuccessListener::onAuthenticationSuccessResponse()   0
 ------- ---------------------------------------------------------------------------------- ----------
```

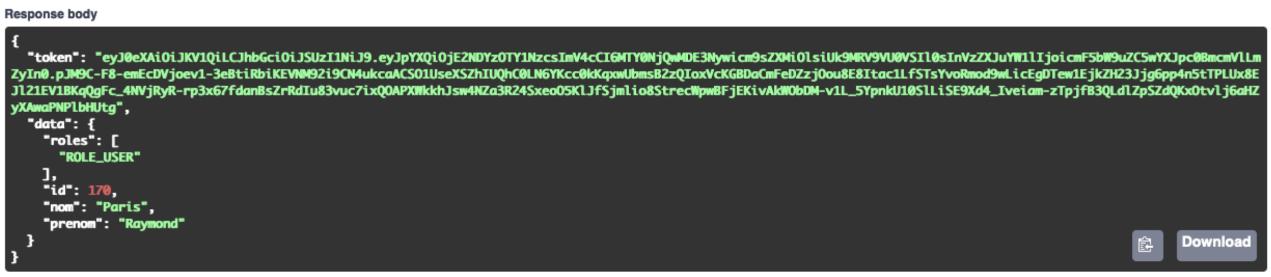
C. Vérification

Vous pouvez maintenant vérifier que l'événement fonctionne en vous rendant sur le catalogue des API. Connectez-vous en indiquant un email et un mot de passe.



```
{
  "email": "raymond.paris@free.fr",
  "password": "raymond"
}
```

Voici un exemple de réponse :



```
{
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpYXQiOjE2NDYzOTY1NzcsImV4cCI6MTY0NjQwMDE3Nywicm9sZXMiOlsiUk9MRV9VU0VSIl0sInVzZXJuYW1lIjoicmF5bW9uZC5wYXJpc0BmcmVlLm
ZyIn0.pJM9C-F8-emEcDVjoev1-3eBtiRbiKEVNM92i9CN4ukcaACSO1UseXSZhIUQhC0LN6YKcc0kKqxwUbmsB2zQIoxVcKGBDaCmFeDZzjOou8E8Itac1LfSTsYvoRmod9wLicEgDTew1EjkZH23Jjg6pp4n5tTPLUx8E
Jl21EV1BKqQgFc_4NVjRyR-rp3x67fdanBsZrRdIu83vuc7ixQOAPXWkkhJsw4NZa3R24SxeoO5KlJfSjmlio8StrecWpwBFjEKivAkWObDM-v1L_5YpnkU10SlLiSE9Xd4_Iveiam-zTpjfB3QLdlZpSZdQKxOtvlj6aHZ
yXAwaPNPlbHUtg",
  "data": {
    "roles": [
      "ROLE_USER"
    ],
    "id": 170,
    "nom": "Paris",
    "prenom": "Raymond"
  }
}
```

Commentaires :
Vous voyez la clé « token » comme dans le chapitre 2 et la clé « data ». Dans celle-ci, vous retrouvez les données supplémentaires souhaitées.