API Platform sans Doctrine ?

Jérôme Tamarelle — GromNaN

SymfonyLive
PARIS 2025
MARCH 27 - 28, 2025

Jérôme Tamarelle

@GromNaN

MongoDB®

Symfony

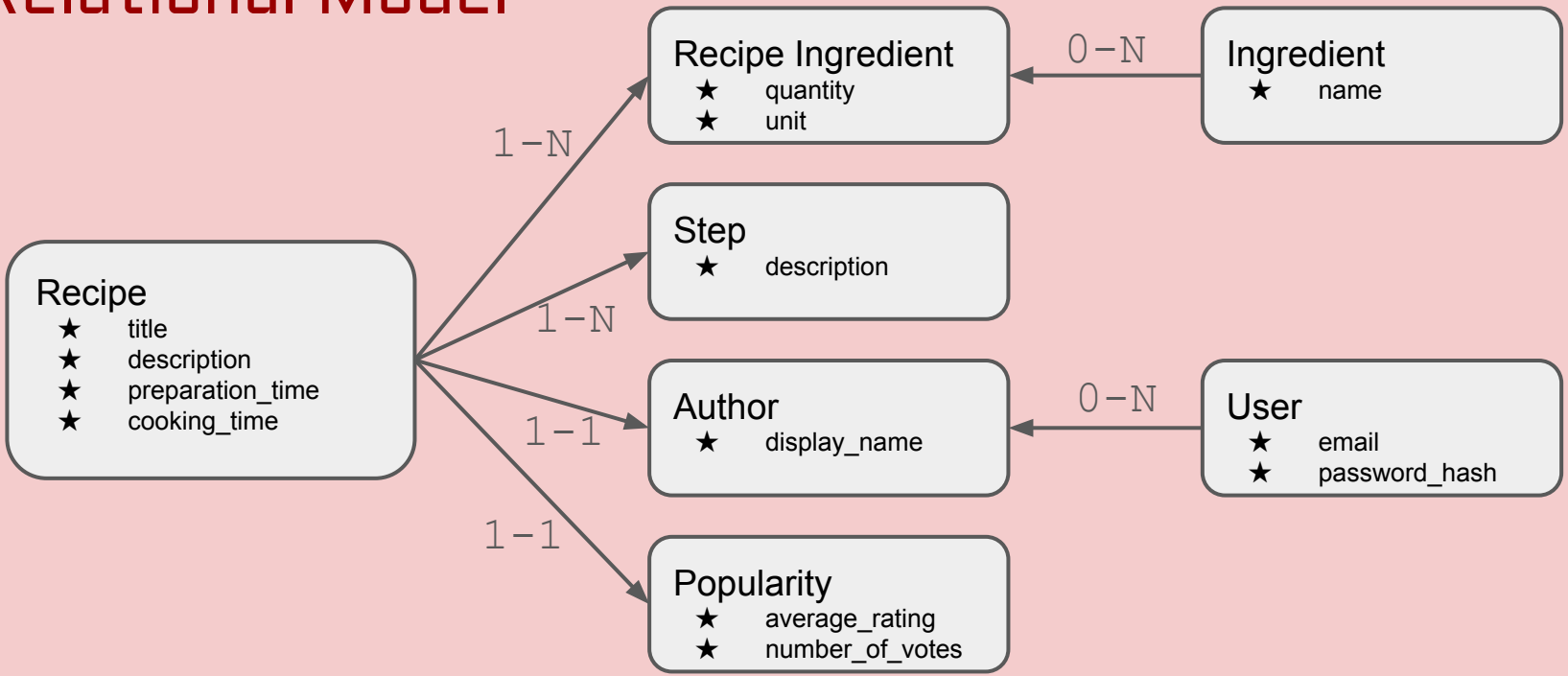
SymfonyLive
BERLIN 2025

APRIL 3 - 4, 2025

SymfonyLive
PARIS 2025
MARCH 27 - 28, 2025


```
{
  "id": "1a9b58d0-e7c1-4e94-be89-30dfcbf2131e",
  "title": "Lasagna Bolognese",
  "description": "A traditional Italian dish, tasty and comforting.",
  "preparation_time": 30,
  "cooking_time": 45,
  "ingredients": [
    { "name": "Lasagna noodles", "quantity": 250, "unit": "g" },
    { "name": "Ground beef", "quantity": 500, "unit": "g" },
    { "name": "Crushed tomatoes", "quantity": 400, "unit": "g" },
    { "name": "Onion", "quantity": 1, "unit": "unit" },
    { "name": "Béchamel sauce", "quantity": 300, "unit": "ml" },
    { "name": "Grated cheese", "quantity": 100, "unit": "g" }
  ],
  "steps": [
    "Sauté the chopped onion in a pan with a little oil.",
    "Add the ground beef and cook for 5 minutes.",
    "...",
  ],
  "author": {
    "name": "Chef Mario",
    "user_id": "83ec0f93-e3b5-49ba-8466-84ceb38df788"
  },
  "popularity": { "average_rating": 4.7, "number_of_votes": 128 }
}
```

Relational Model



```php
#[ORM\Entity]
class Recipe
{
    #[ORM\Id, ORM\GeneratedValue, ORM\Column]
    public ?int $id = null;

    #[ORM\Column(length: 255)]
    public string $title;

    #[ORM\Column(type: 'text', nullable: true)]
    public ?string $description = null;

    #[ORM\Column(type: 'integer')]
    public int $preparationTime;

    #[ORM\OneToMany(mappedBy: 'recipe', targetEntity: RecipeIngredient::class]
    public Collection $ingredients;

    #[ORM\OneToMany(mappedBy: 'recipe', targetEntity: Step::class]
    #[ORM\OrderBy(['stepNumber' => 'ASC'])]
    public Collection $steps;

    // ...
}
```

```php
#[ORM\Entity]
class Recipe
{
    #[ORM\Id, ORM\GeneratedValue, ORM\Column]
    public ?int $id = null;

    #[ORM\Column(length: 255)]
    public string $title;

    #[ORM\Column(type: 'text', nullable: true)]
    public ?string $description = null;

    #[ORM\Column(type: 'integer')]
    public int $preparationTime;

    #[ORM\OneToMany(mappedBy: 'recipe', targetEntity: RecipeIngredient::class]
    public Collection $ingredients;

    #[ORM\OneToMany(mappedBy: 'recipe', targetEntity: Step::class]
    #[ORM\OrderBy(['stepNumber' => 'ASC'])]
    public Collection $steps;

    // ...
}
```

```php
#[ORM\Entity]
class Recipe
{
    #[ORM\Id, ORM\GeneratedValue, ORM\Column]
    public ?int $id = null;

    #[ORM\Column(length: 255)]
    public string $title;

    #[ORM\Column(type: 'text', nullable: true)]
    public ?string $description = null;

    #[ORM\Column(type: 'integer')]
    public int $preparationTime;

    #[ORM\OneToMany(mappedBy: 'recipe', targetEntity: RecipeIngredient::class]
    public Collection $ingredients;

    #[ORM\OneToMany(mappedBy: 'recipe', targetEntity: Step::class]
    #[ORM\OrderBy(['stepNumber' => 'ASC'])]
    public Collection $steps;

    // ...
}
```

Transform an Entity into an API with a single attribute

```php
#[ApiResource]
#[ORM\Entity]
class Recipe
{
    #[ORM\Id, ORM\GeneratedValue, ORM\Column]
    public ?int $id = null;

    #[ORM\Column(length: 255)]
    public string $title;

    #[ORM\Column(type: 'text', nullable: true)]
    public ?string $description = null;

    #[ORM\Column(type: 'integer')]
    public int $preparationTime;

    #[ORM\OneToMany(mappedBy: 'recipe', targetEntity: RecipeIngredient::class]
    public Collection $ingredients;

    #[ORM\OneToMany(mappedBy: 'recipe', targetEntity: Step::class]
    #[ORM\OrderBy(['stepNumber' => 'ASC'])]
    public Collection $steps;

    // ...
}
```

GET /api/recipes?<filters>

GET /api/recipes/<id>

POST /api/recipes

PATCH /api/recipes/<id>

DELETE /api/recipes/<id>

```
### Collection JSON LD + Hydra
GET https://localhost:8000/api/recipe

### Single JSON LD + Hydra
GET https://localhost:8000/api/recipe/1
Accept: application/ld+json

### Single HAL+JSON
GET https://localhost:8000/api/recipe/1
Accept: application/hal+json

### Single JSON
GET https://localhost:8000/api/recipe/1
Accept: application/json

### Single XML
GET https://localhost:8000/api/recipe/1
Accept: application/xml

### Single YAML
GET https://localhost:8000/api/recipe/1
Accept: application/x-yaml

### Single CSV
GET https://localhost:8000/api/recipe/1
Accept: text/csv
```

Support many formats


```
{
 "@context": "/api/contexts/Recipe",
 "@id": "/api/recipes/1",
 "@type": "Recipe",
 "id": 1,
 "title": "Lasagna Bolognese",
 "description": "A traditional Italian dish, tasty and comforting.",
 "preparationTime": 30,
 "cookingTime": 45,
 "ingredients": [
        {
                "@type": "RecipeIngredient",
                "@id": "/api/.well-known/genid/0403e35d2fac4966f006",
                "id": 1,
                "recipe": "/api/recipes/1",
                "ingredient": {
                        "@type": "Ingredient",
                        "@id": "/api/ingredients/1",
                        "id": 1,
                        "name": "Lasagna Sheets"
                },
                "quantity": 12,
                "unit": "sheets"
        },
```


```
{
 "@context": "/api/contexts/Recipe",
 "@id": "/api/recipes/1",
 "@type": "Recipe",
 "id": 1,
 "title": "Lasagna Bolognese",
 "description": "A traditional Italian dish, tasty and comforting.",
 "preparationTime": 30,
 "cookingTime": 45,
 "ingredients": [
       {
             "@type": "RecipeIngredient",
             "@id": "/api/.well-known/genid/0403e35d2fac4966f006",
             "id": 1,
             "recipe": "/api/recipes/1",
             "ingredient": {
                   "@type": "Ingredient",
                   "@id": "/api/ingredients/1",
                   "id": 1,
                   "name": "Lasagna Sheets"
             },
             "quantity": 12,
             "unit": "sheets"
       },
```


```
{
 "@context": "/api/contexts/Recipe",
 "@id": "/api/recipes/1",
 "@type": "Recipe",
 "id": 1,

 "steps": [
        {
                "@type": "Step",
                "@id": "/api/.well-known/genid/5eed77035d333fd33600",
                "id": 1,
                "recipe": "/api/recipes/1",
                "stepNumber": 1,
                "description": "Chop onions, garlic, carrots, and celery finely."
        },
        {
                "@type": "Step",
                "@id": "/api/.well-known/genid/aff9800da0188530f625",
                "id": 2,
                "recipe": "/api/recipes/1",
                "stepNumber": 2,
                "description": "Heat olive oil in a pan, sauté vegetables."
        },
```


```
{
 "@context": "/api/contexts/Recipe",
 "@id": "/api/recipes/1",
 "@type": "Recipe",
 "id": 1,

 "author_name": "Chef Mario",
 "author_user": {
        "@type": "User",
        "@id": "/api/.well-known/genid/ac364f39a787a780cde9",
        "id": 5,
        "email": "chef@cuisine.dev",
        "userIdentifier": "chef@cuisine.dev",
        "roles": [
                "ROLE_USER"
        ],
        "password": "Délice"
},
```


Should not be exposed

```php
#[ORM\ManyToOne(targetEntity: User::class, inversedBy: 'recipes')]
#[Ignore]
public ?User $author_user;
```

Serializer attributes

```
SELECT count(*) AS sclr_0 FROM recipe r0_

SELECT ... FROM recipe r0_ ORDER BY r0_.id ASC LIMIT 30

SELECT ... FROM popularity t0 WHERE t0.recipe_id = ?                    x30

SELECT ... FROM recipe_ingredient t0 WHERE t0.recipe_id = ?             x30

SELECT ... FROM ingredient t0 WHERE t0.id = ?                           x30 x10

SELECT ... FROM step t0 WHERE t0.recipe_id = ? ORDER BY t0.step_number ASC   x30
```

= 300 SQL queries !!!

```
select
      id,
      title,
      description,
      preparation time,
      cooking time,
      author_name,
```

How to get relations in a single query?

```
from recipe
where ...;
```

```php
#[ORM\OneToMany(mappedBy: 'recipe', targetEntity: Step::class, fetch: 'EAGER')]
#[ORM\OrderBy(['stepNumber' => 'ASC'])]
#[Groups(['recipe:read', 'recipe:write'])]
public Collection $steps;
```

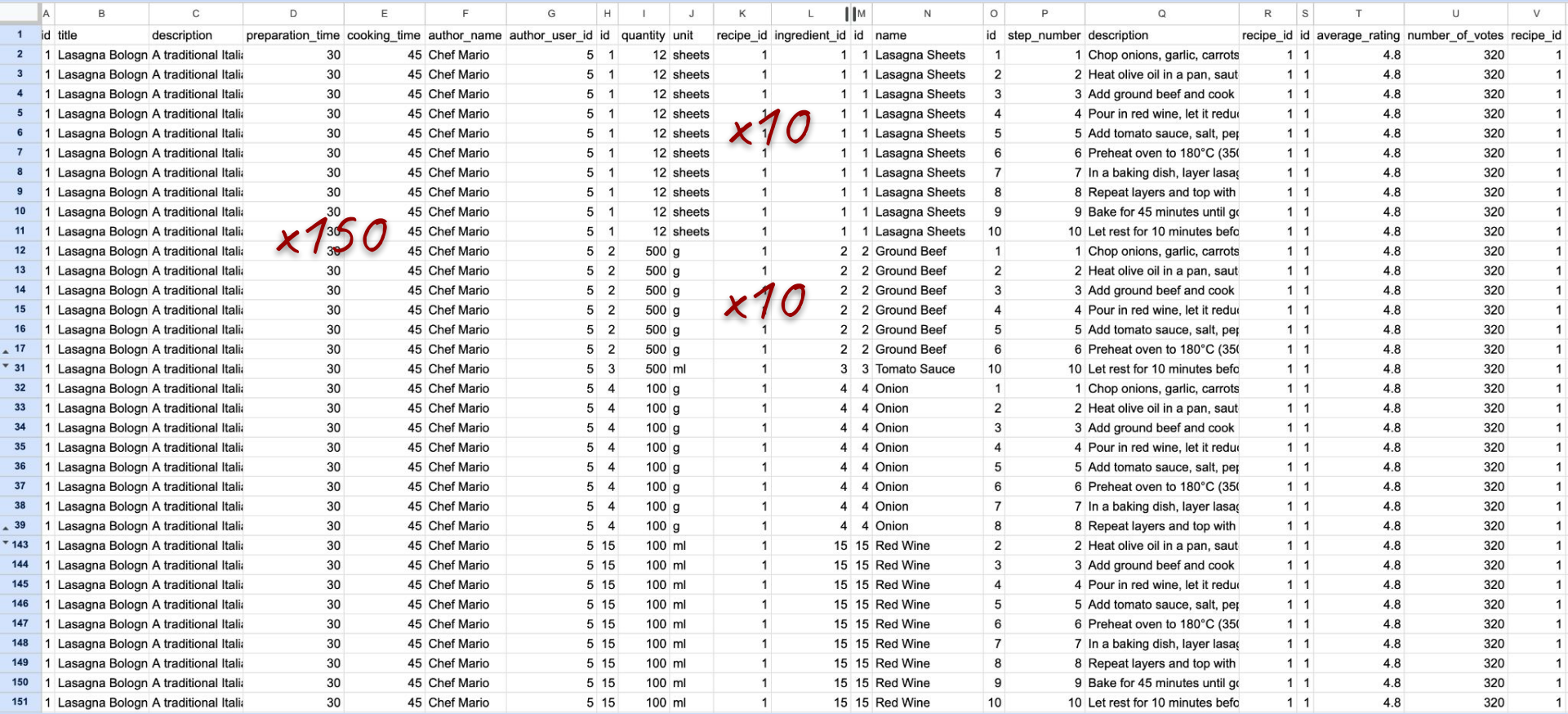
Eager Loading optimize queries

Eager Loading optimize queries

```
select ... from recipe r0_
    left join recipe_ingredient r1_ on r0_.id = r1_.recipe_id
    left join step s2_ on r0_.id = s2_.recipe_id
    left join popularity p3_ on r0_.id = p3_.recipe_id
where r0_.id = ? order by s2_.step_number asc

select ... from user t0 where t0.id in (?)

select ... from ingredient t0 where t0.id in (?)
```

```
select

  ...

from recipe
     left join recipe_ingredient
          on recipe.id = recipe_ingredient.recipe_id
     left join ingredient
          on recipe_ingredient.ingredient_id = ingredient.id
     left join step
          on recipe.id = step.recipe_id
     left join popularity
          on recipe.id = popularity.recipe_id

where ...;
```

Cartesian JOIN all the tables



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	id	title	description	preparation_time	cooking_time	author_name	author_user_id	id	quantity	unit	recipe_id	ingredient_id	id	name	id	step_number	description	recipe_id	id	average_rating	number_of_votes	recipe_id
2	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	1	12	sheets	1	1	1	Lasagna Sheets	1	1	Chop onions, garlic, carrots	1	1	4.8	320	1
3	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	1	12	sheets	1	1	1	Lasagna Sheets	2	2	Heat olive oil in a pan, saut	1	1	4.8	320	1
4	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	1	12	sheets	1	1	1	Lasagna Sheets	3	3	Add ground beef and cook	1	1	4.8	320	1
5	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	1	12	sheets	1	1	1	Lasagna Sheets	4	4	Pour in red wine, let it redu	1	1	4.8	320	1
6	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	1	12	sheets	1	1	1	Lasagna Sheets	5	5	Add tomato sauce, salt, pep	1	1	4.8	320	1
7	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	1	12	sheets	1	1	1	Lasagna Sheets	6	6	Preheat oven to 180°C (350	1	1	4.8	320	1
8	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	1	12	sheets	1	1	1	Lasagna Sheets	7	7	In a baking dish, layer lasag	1	1	4.8	320	1
9	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	1	12	sheets	1	1	1	Lasagna Sheets	8	8	Repeat layers and top with	1	1	4.8	320	1
10	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	1	12	sheets	1	1	1	Lasagna Sheets	9	9	Bake for 45 minutes until go	1	1	4.8	320	1
11	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	1	12	sheets	1	1	1	Lasagna Sheets	10	10	Let rest for 10 minutes befo	1	1	4.8	320	1
12	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	2	500	g	1	2	2	Ground Beef	1	1	Chop onions, garlic, carrots	1	1	4.8	320	1
13	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	2	500	g	1	2	2	Ground Beef	2	2	Heat olive oil in a pan, saut	1	1	4.8	320	1
14	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	2	500	g	1	2	2	Ground Beef	3	3	Add ground beef and cook	1	1	4.8	320	1
15	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	2	500	g	1	2	2	Ground Beef	4	4	Pour in red wine, let it redu	1	1	4.8	320	1
16	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	2	500	g	1	2	2	Ground Beef	5	5	Add tomato sauce, salt, pep	1	1	4.8	320	1
17	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	2	500	g	1	2	2	Ground Beef	6	6	Preheat oven to 180°C (350	1	1	4.8	320	1
31	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	3	500	ml	1	3	3	Tomato Sauce	10	10	Let rest for 10 minutes befo	1	1	4.8	320	1
32	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	4	100	g	1	4	4	Onion	1	1	Chop onions, garlic, carrots	1	1	4.8	320	1
33	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	4	100	g	1	4	4	Onion	2	2	Heat olive oil in a pan, saut	1	1	4.8	320	1
34	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	4	100	g	1	4	4	Onion	3	3	Add ground beef and cook	1	1	4.8	320	1
35	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	4	100	g	1	4	4	Onion	4	4	Pour in red wine, let it redu	1	1	4.8	320	1
36	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	4	100	g	1	4	4	Onion	5	5	Add tomato sauce, salt, pep	1	1	4.8	320	1
37	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	4	100	g	1	4	4	Onion	6	6	Preheat oven to 180°C (350	1	1	4.8	320	1
38	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	4	100	g	1	4	4	Onion	7	7	In a baking dish, layer lasag	1	1	4.8	320	1
39	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	4	100	g	1	4	4	Onion	8	8	Repeat layers and top with	1	1	4.8	320	1
143	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	15	100	ml	1	15	15	Red Wine	2	2	Heat olive oil in a pan, saut	1	1	4.8	320	1
144	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	15	100	ml	1	15	15	Red Wine	3	3	Add ground beef and cook	1	1	4.8	320	1
145	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	15	100	ml	1	15	15	Red Wine	4	4	Pour in red wine, let it redu	1	1	4.8	320	1
146	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	15	100	ml	1	15	15	Red Wine	5	5	Add tomato sauce, salt, pep	1	1	4.8	320	1
147	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	15	100	ml	1	15	15	Red Wine	6	6	Preheat oven to 180°C (350	1	1	4.8	320	1
148	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	15	100	ml	1	15	15	Red Wine	7	7	In a baking dish, layer lasag	1	1	4.8	320	1
149	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	15	100	ml	1	15	15	Red Wine	8	8	Repeat layers and top with	1	1	4.8	320	1
150	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	15	100	ml	1	15	15	Red Wine	9	9	Bake for 45 minutes until go	1	1	4.8	320	1
151	1	Lasagna Bologn	A traditional Itali	30	45	Chef Mario	5	15	100	ml	1	15	15	Red Wine	10	10	Let rest for 10 minutes befo	1	1	4.8	320	1

More data • More network transfer • More memory & CPU usage

```
select
    id,
    title,
    description,
    preparation_time,
    cooking_time,
    author_name,

    (
        select json_object('average_rating', popularity.average_rating,
                           'number_of_votes', popularity.number_of_votes)
        from popularity
        where popularity.recipe_id = recipe.id
    )
    as popularity

from recipe
where ...;
```

```
{"average_rating":4.8,"number_of_votes":320}
```

```sql
select
    id,
    title,
    description,
    preparation_time,
    cooking_time,
    author_name,

    (
        select json_group_array(step.description)
        from step
        where step.recipe_id = recipe.id
        order by step.step_number
    )
    as steps

from recipe
where ...;
```

```
[
        "Chop onions, garlic, carrots, and celery finely.,"
        "Heat olive oil in a pan, sauté vegetables until soft.,"
        "Add ground beef and cook until browned.,"
        "Pour in red wine, let it reduce for 5 minutes.,"
        "Add tomato sauce, salt, pepper, and oregano. Simmer for 20
        minutes.","Preheat oven to 180°C (350°F).,"
        "Let rest for 10 minutes before serving."
]
```

```sql
select
    id,
    title,
    description,
    preparation_time,
    cooking_time,
    author_name,
      (
          select json_group_array(
                  json_object('quantity', recipe_ingredient.quantity,
                              'unit',     recipe_ingredient.unit,
                              'name',     ingredient.name             ))
          from recipe_ingredient
          inner join ingredient on recipe_ingredient.ingredient_id = ingredient.id
          where recipe_ingredient.recipe_id = recipe.id
          order by ingredient.name
      )
      as ingredients,
from recipe
where ...;
```

```
[
        {"quantity":12.0,"unit":"sheets","name":"Lasagna Sheets"},
        {"quantity":500.0,"unit":"g","name":"Ground Beef"},
        {"quantity":500.0,"unit":"ml","name":"Tomato Sauce"},
        {"quantity":100.0,"unit":"g","name":"Onion"},
        {"quantity":2.0,"unit":"cloves","name":"Garlic"},
        {"quantity":100.0,"unit":"g","name":"Carrot"}
]
```

```sql
select
   id,
   title,
   description,
   preparation_time,
   cooking_time,
   author_name,
   (
       select json_group_array(step.description)
       from step
       where step.recipe_id = recipe.id
       order by step.step_number
   ) as steps,
   (
       select json_object('average_rating', popularity.average_rating, 'number_of_votes',
popularity.number_of_votes)
       from popularity
       where popularity.recipe_id = recipe.id
   ) as popularity,
   (
       select json_group_array(json_object('quantity', recipe_ingredient.quantity, 'unit',
recipe_ingredient.unit, 'name', ingredient.name))
       from recipe_ingredient
       inner join ingredient on recipe_ingredient.ingredient_id = ingredient.id
       where recipe_ingredient.recipe_id = recipe.id
       order by ingredient.name
   ) as ingredients
from recipe;
```

id	1
title	Lasagna Bolognese
description	A traditional Italian dish, tasty and comforting.
preparation_time	30
cooking_time	45
author_name	Chef Mario
popularity	{"average_rating":4.8,"number_of_votes":320}
steps	["Chop onions, garlic, carrots, and celery finely.","Heat olive oil in a pan, sauté vegetables until soft.","Add ground beef and cook until browned.","Pour in red wine, let it reduce for 5 minutes.","Add tomato sauce, salt, pepper, and oregano. Simmer for 20 minutes.","Preheat oven to 180°C (350°F).","In a baking dish, layer lasagna sheets, meat sauce, and béchamel sauce.","Repeat layers and top with mozzarella and parmesan.","Bake for 45 minutes until golden brown.","Let rest for 10 minutes before serving."]
ingredients	[{"quantity":12.0,"unit":"sheets","name":"Lasagna Sheets"},{"quantity":500.0,"unit":"g","name":"Ground Beef"},{"quantity":500.0,"unit":"ml","name":"Tomato Sauce"},{"quantity":100.0,"unit":"g","name":"Onion"},{"quantity":2.0,"unit":"cloves","name":"Garlic"},{"quantity":100.0,"unit":"g","name":"Carrot"},{"quantity":50.0,"unit":"g","name":"Celery"},{"quantity":2.0,"unit":"tbsp","name":"Olive Oil"},{"quantity":400.0,"unit":"ml","name":"Bechamel Sauce"},{"quantity":100.0,"unit":"g","name":"Parmesan Cheese"},{"quantity":150.0,"unit":"g","name":"Mozzarella"},{"quantity":1.0,"unit":"tsp","name":"Salt"},{"quantity":1.0,"unit":"tsp","name":"Black Pepper"},{"quantity":1.0,"unit":"tsp","name":"Oregano"},{"quantity":100.0,"unit":"ml","name":"Red Wine"}]

```php
class Recipe
{
    public string $id;

    public string $title;

    public ?string $description;

    public int $preparationTime;

    public int $cookingTime;

    public array $ingredients;

    public array $steps;

    public string $authorName;

    public array $popularity;
}
```

```php
#[ApiResource(provider: SqlRecipeState::class )]

class Recipe
{
    public string $id;

    public string $title;

    public ?string $description;

    public int $preparationTime;

    public int $cookingTime;

    public array $ingredients;

    public array $steps;

    public string $authorName;

    public array $popularity;
}
```

Custom State Provider

```php
use ApiPlatform\State\ProviderInterface;

/** @implements ProviderInterface<Recipe> */
readonly class SqlRecipeState implements ProviderInterface
{
    public function __construct(private \Doctrine\DBAL\Connection $connection) {}

    public function provide(Operation $operation, array $uriVariables, array $context): Recipe
    {
        $id = $uriVariables['id'];

        $stmt = $this->connection->prepare('select ... where id = :id');
        $stmt->bindValue('id', $id);
        $results = $stmt->executeQuery()->fetchAllAssociative();



        $results = array_map($this->hydrateRecipe(...), $results);



        return $results[0] ?? null;
    }
}
```

Service autoconfigured

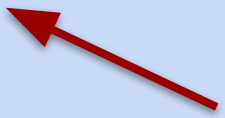
```php
use ApiPlatform\State\ProviderInterface;

/** @implements ProviderInterface<Recipe> */
readonly class SqlRecipeState implements ProviderInterface
{
    public function __construct(private \Doctrine\DBAL\Connection $connection) {}

    public function provide(Operation $operation, array $uriVariables, array $context): Recipe
    {
        $id = $uriVariables['id'];

        $stmt = $this->connection->prepare('select ... where id = :id');
        $stmt->bindValue('id', $id);
        $results = $stmt->executeQuery()->fetchAllAssociative();



        $results = array_map($this->hydrateRecipe(...), $results);



        return $results[0] ?? null;
    }
}
```

Context from the Request

```php
use ApiPlatform\State\ProviderInterface;

/** @implements ProviderInterface<Recipe> */
readonly class SqlRecipeState implements ProviderInterface
{
    public function __construct(private \Doctrine\DBAL\Connection $connection) {}

    public function provide(Operation $operation, array $uriVariables, array $context): Recipe
    {
        $id = $uriVariables['id'];

        $stmt = $this->connection->prepare('select ... where id = :id');
        $stmt->bindValue('id', $id);
        $results = $stmt->executeQuery()->fetchAllAssociative();



        $results = array_map($this->hydrateRecipe(...), $results);



        return $results[0] ?? null;
    }
}
```

Custom SQL query

```php
use ApiPlatform\State\ProviderInterface;

/** @implements ProviderInterface<Recipe> */
readonly class SqlRecipeState implements ProviderInterface
{
    public function __construct(private \Doctrine\DBAL\Connection $connection) {}

    public function provide(Operation $operation, array $uriVariables, array $context): Recipe
    {
        $id = $uriVariables['id'];

        $stmt = $this->connection->prepare('select ... where id = :id');
        $stmt->bindValue('id', $id);
        $results = $stmt->executeQuery()->fetchAllAssociative();



        $results = array_map($this->hydrateRecipe(...), $results);



        return $results[0] ?? null;
    }
}
```



You are responsible of mapping the results to DTO objects

```php
public function hydrateRecipe(array $data): Recipe
{

    $recipe = new Recipe();
    $recipe->id = $data['id'];
    $recipe->title = $data['title'];
    $recipe->description = $data['description'];
    $recipe->preparationTime = $data['preparation_time'];
    $recipe->cookingTime = $data['cooking_time'];
    $recipe->authorName = $data['author_name'];
    $recipe->ingredients = json_decode($data['ingredients'], true);
    $recipe->steps = json_decode($data['steps'], true);
    $recipe->popularity = json_decode($data['popularity'], true);

    return $recipe;
}
```

Parse JSON documents returned by the database

```php
use ApiPlatform\State\ProviderInterface;
use AutoMapper\AutoMapper;

readonly class SqlRecipeState implements ProviderInterface
{
    public function __construct(Connection $connection, private AutoMapper $autoMapper) {}

    public function provide(Operation $operation, array $uriVariables, array $context): Recipe
    {
        $id = $uriVariables['id'];

        $stmt = $this->connection->prepare('select ... where id = :id');
        $stmt->bindValue('id', $id);
        $results = $stmt->executeQuery()->fetchAllAssociative();


        $resourceClass = $operation->getClass();
        $results = $this->autoMapper->mapCollection($results, $resourceClass);


        return $results[0] ?? null;
    }
}
```

Package jolicode/automapper

```php
use AutoMapper\Attribute\MapFrom;
#[ApiResource(provider: SqlRecipeState::class )]
class Recipe
{
    public string $id;

    public string $title;

    public ?string $description;

    public int $preparationTime;

    public int $cookingTime;
    #[MapFrom(source: 'array', transformer: 'json_decode')]
    public array $ingredients;
    #[MapFrom(source: 'array', transformer: 'json_decode')]
    public array $steps;

    public string $authorName;
    #[MapFrom(source: 'array', transformer: 'json_decode')]
    public \stdClass $popularity;
}
```

Parse JSON documents returned by the database


```
{
  "id": 1,
  "title": "Lasagna Bolognese",
  "description": "A traditional Italian dish, tasty and comforting.",
  "preparation_time": 30,
  "cooking_time": 45,
  "ingredients": [
    { "name": "Lasagna noodles", "quantity": 250, "unit": "g" },
    { "name": "Ground beef", "quantity": 500, "unit": "g" },
    { "name": "Crushed tomatoes", "quantity": 400, "unit": "g" },
    { "name": "Onion", "quantity": 1, "unit": "unit" },
    { "name": "Béchamel sauce", "quantity": 300, "unit": "ml" },
    { "name": "Grated cheese", "quantity": 100, "unit": "g" }
  ],
  "steps": [
    "Sauté the chopped onion in a pan with a little oil.",
    "Add the ground beef and cook for 5 minutes.",
    "...",
  ],
  "popularity": { "average_rating": 4.7, "number_of_votes": 128 }
}
```

32 SQL queries to insert 1 recipe

```
insert into recipe (title, description, preparation_time, cooking_time, author_name,
author_user_id)
     values (?, ?, ?, ?, ?, ?)
insert into ingredient (name)
     values (?)
insert into recipe_ingredient (quantity, unit, recipe_id, ingredient_id)
     values (?, ?, ?, ?)
insert into step (step_number, description, recipe_id)
     values (?, ?, ?)
insert into popularity (average_rating, number_of_votes, recipe_id)
     values (?, ?, ?)
```

x10

x10

x10

10 SQL queries to remove 1 step from a recipe

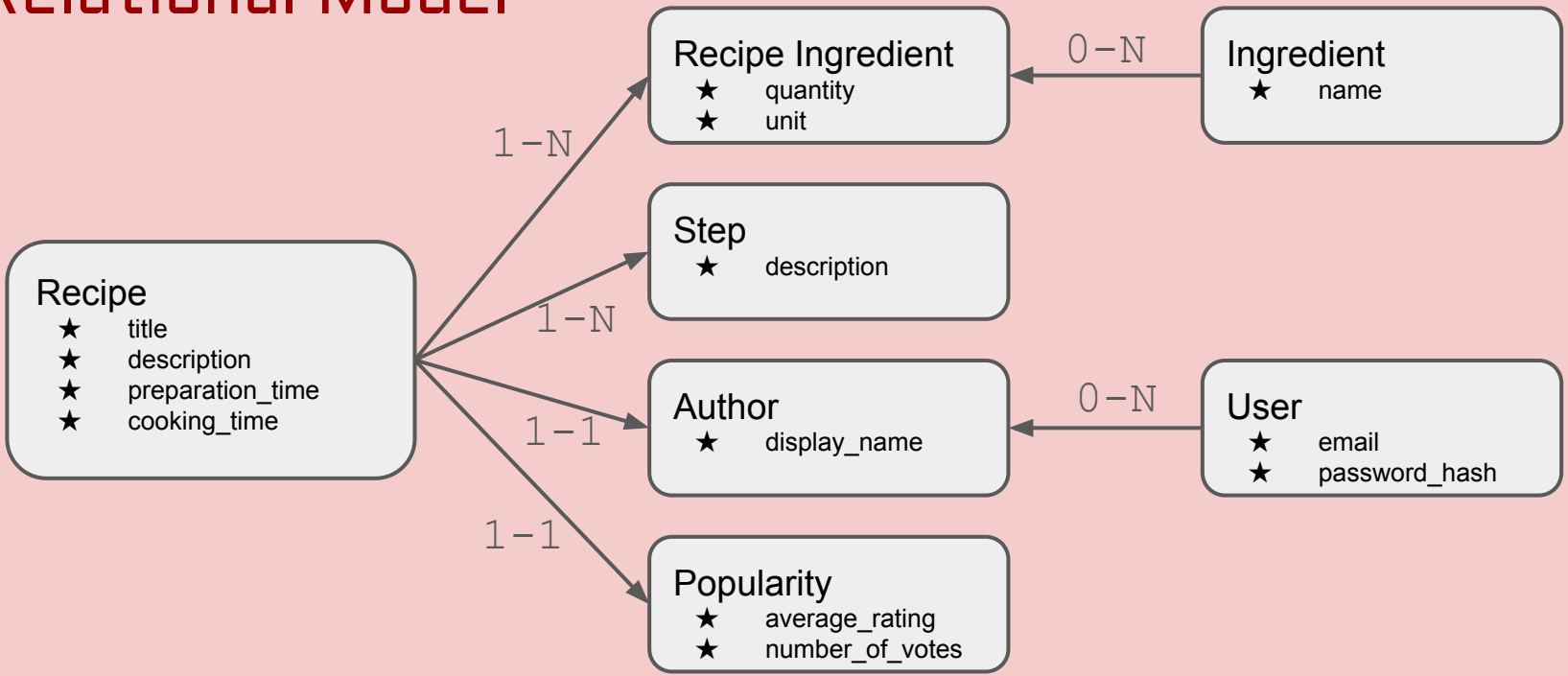
```
update step set step_number = ? where id = ?
delete from step where id = ?
```

x10

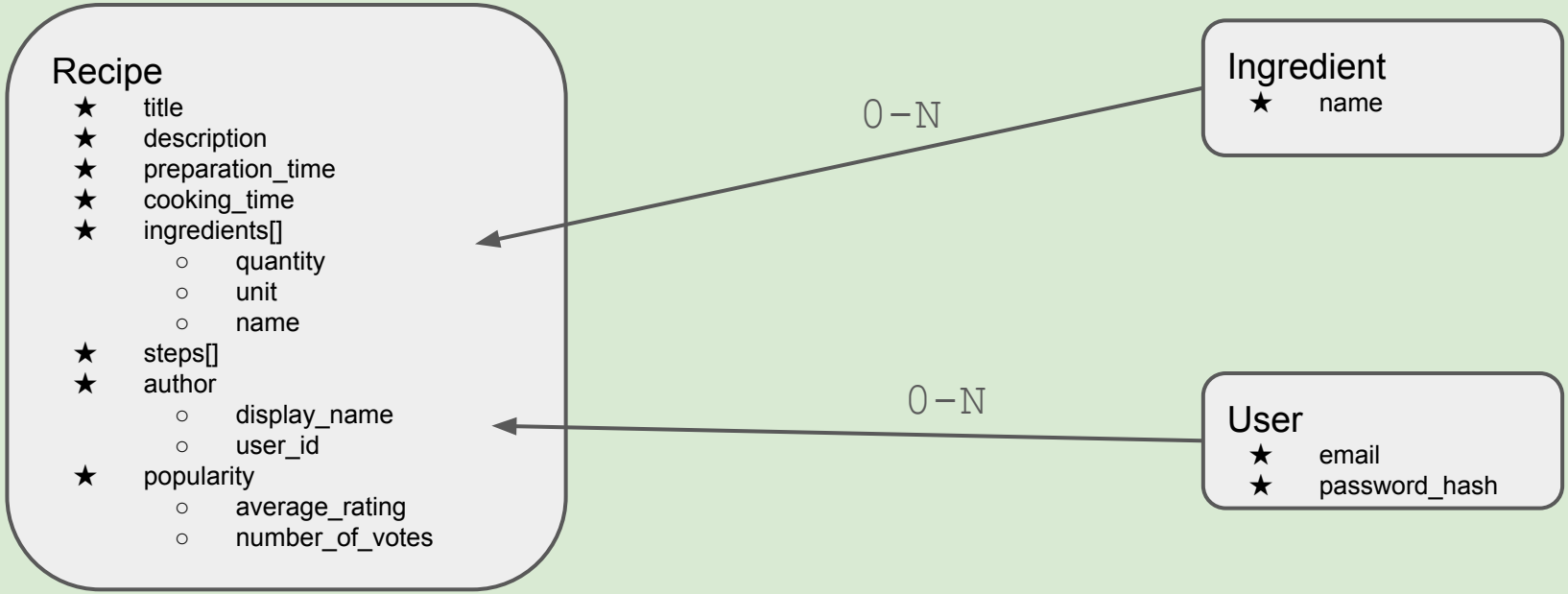
I gave up for write operations

POST PATCH

Relational Model



Document Model



Flavor

MongoDB®

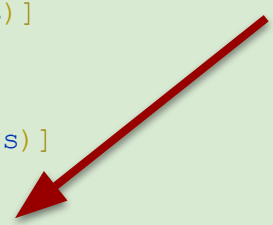
```php
#[ODM\Document(collection: 'recipes')]
class Recipe
{
    #[ODM\Id] public string $id;
    #[ODM\Field] public string $title;
    #[ODM\Field] public ?string $description = null;
    #[ODM\Field] public int $preparationTime;

    /** @var Collection<Ingredient> */
    #[ODM\EmbedMany(targetDocument: Ingredient::class)]
    public Collection $ingredients;

    /** @var Collection<Step> */
    #[ODM\EmbedMany(targetDocument: Step::class)]
    public Collection $steps;

    #[ODM\EmbedOne(targetDocument: Author::class)]
    public Author $author;

    #[ODM\EmbedOne(targetDocument: Popularity::class)]
    public ?Popularity $popularity = null;
}
```

```php
#[ODM\Document(collection: 'recipes')]
class Recipe
{
    #[ODM\Id] public string $id;
    #[ODM\Field] public string $title;
    #[ODM\Field] public ?string $description = null;
    #[ODM\Field] public int $preparationTime;

    /** @var Collection<Ingredient> */
    #[ODM\EmbedMany(targetDocument: Ingredient::class)]
    public Collection $ingredients;

    /** @var Collection<Step> */
    #[ODM\EmbedMany(targetDocument: Step::class)]
    public Collection $steps;

    #[ODM\EmbedOne(targetDocument: Author::class)]
    public Author $author;

    #[ODM\EmbedOne(targetDocument: Popularity::class)]
    public ?Popularity $popularity = null;
}
```

Columns are Fields

```php
#[ODM\Document(collection: 'recipes')]
class Recipe
{
    #[ODM\Id] public string $id;
    #[ODM\Field] public string $title;
    #[ODM\Field] public ?string $description = null;
    #[ODM\Field] public int $preparationTime;

    /** @var Collection<Ingredient> */
    #[ODM\EmbedMany(targetDocument: Ingredient::class)]
    public Collection $ingredients;

    /** @var Collection<Step> */
    #[ODM\EmbedMany(targetDocument: Step::class)]
    public Collection $steps;

    #[ODM\EmbedOne(targetDocument: Author::class)]
    public Author $author;

    #[ODM\EmbedOne(targetDocument: Popularity::class)]
    public ?Popularity $popularity = null;
}
```

Embeds a sub-document



```php
#[ODM\Document(collection: 'recipes')]
class Recipe
{
    #[ODM\Id] public string $id;
    #[ODM\Field] public string $title;
    #[ODM\Field] public ?string $description = null;
    #[ODM\Field] public int $preparationTime;

    /** @var Collection<Ingredient> */
    #[ODM\EmbedMany(targetDocument: Ingredient::class)]
    public Collection $ingredients;

    /** @var Collection<Step> */
    #[ODM\EmbedMany(targetDocument: Step::class)]
    public Collection $steps;

    #[ODM\EmbedOne(targetDocument: Author::class)]
    public Author $author;

    #[ODM\EmbedOne(targetDocument: Popularity::class)]
    public ?Popularity $popularity = null;
}
```

Embeds a list of sub-documents

```php
#[ApiResource(shortName: 'recipes')]
#[ODM\Document(collection: 'recipes')]
class Recipe
{
    #[ODM\Id] public string $id;
    #[ODM\Field] public string $title;
    #[ODM\Field] public ?string $description = null;
    #[ODM\Field] public int $preparationTime;

    /** @var Collection<Ingredient> */
    #[ODM\EmbedMany(targetDocument: Ingredient::class)]
    public Collection $ingredients;

    /** @var Collection<Step> */
    #[ODM\EmbedMany(targetDocument: Step::class)]
    public Collection $steps;

    #[ODM\EmbedOne(targetDocument: Author::class)]
    public Author $author;

    #[ODM\EmbedOne(targetDocument: Popularity::class)]
    public ?Popularity $popularity = null;
}
```


```
{
 "id": "67da8b6c606149379d0a7ba7",
 "title": "Lasagna Bolognese",
 "description": "A traditional Italian dish, tasty and comforting.",
 "preparationTime": 30,
 "cookingTime": 45,
 "ingredients": [
   {"id": "67e4191a2f9b557eb10f6537","name": "Lasagna
Sheets","quantity": 12.0,"unit": "sheets"},
   {"id": "67e4191a2f9b557eb10f6538","name": "Ground Beef","quantity":
500.0,"unit": "g"},
 ],
 "steps": [
   {"id": "67e4191a2f9b557eb10f6546","stepNumber": 1,"description":
"Chop onions, garlic, carrots, and celery finely."},
   {"id": "67e4191a2f9b557eb10f6547","stepNumber": 2,"description":
"Heat olive oil in a pan, sauté vegetables until soft."},
 ]
 "popularity": {"id": "67e4191a2f9b557eb10f6550","averageRating":
4.8,"numberOfVotes": 320}
}
```


```
{
 "id": "67da8b6c606149379d0a7ba7",
 "title": "Lasagna Bolognese",
 "description": "A traditional Italian dish, tasty and comforting.",
 "preparationTime": 30,
 "cookingTime": 45,
 "ingredients": [
   {"id": "67e4191a2f9b557eb10f6537","name": "Lasagna
Sheets","quantity": 12.0,"unit": "sheets"},
   {"id": "67e4191a2f9b557eb10f6538","name": "Ground Beef","quantity":
500.0,"unit": "g"},
 ],
 "steps": [
   {"id": "67e4191a2f9b557eb10f6546","stepNumber": 1,"description":
"Chop onions, garlic, carrots, and celery finely."},
   {"id": "67e4191a2f9b557eb10f6547","stepNumber": 2,"description":
"Heat olive oil in a pan, sauté vegetables until soft."},
 ]
 "popularity": {"id": "67e4191a2f9b557eb10f6550","averageRating":
4.8,"numberOfVotes": 320}
}
```

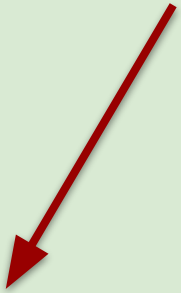

Not useful

```php
#[ODM\EmbeddedDocument]
class Step
{
    #[ODM\Id]
    #[\Symfony\Component\Serializer\Attribute\Ignore]
    public string $id;

    #[ODM\Field]
    public int $stepNumber;

    #[ODM\Field]
    public string $description;
}
```

Never serialized

```php
#[ODM\EmbeddedDocument]
class Step
{
    #[ODM\Id]
    public string $id;

    #[ODM\Field]
    public int $stepNumber;

    #[Groups(['recipe:read', 'recipe:write'])]
    #[ODM\Field]
    public string $description;
}
```

Only this field is serialized in this context



```php
#[ApiResource(
    normalizationContext: ['groups' => ['recipe:read']],
    denormalizationContext: ['groups' => ['recipe:write']],
)]
#[ODM\Document(collection: 'recipes')]
#[Groups(['recipe:read', 'recipe:write'])]
class Recipe
{
    #[ODM\Id]
    public string $id;

    // ...

}
```

Configuration for the Symfony Serializer


```
{
 "id": "67da8b6c606149379d0a7ba7",
 "title": "Lasagna Bolognese",
 "description": "A traditional Italian dish, tasty and comforting.",
 "preparationTime": 30,
 "cookingTime": 45,
 "ingredients": [
   {"name": "Lasagna Sheets","quantity": 12.0,"unit": "sheets"},
   {"name": "Ground Beef","quantity": 500.0,"unit": "g"},
 ],
 "steps": [
   {"description": "Chop onions, garlic, carrots, and celery finely."},
   {"description": "Heat olive oil in a pan, sauté vegetables until
soft."},
 ]
 "popularity": {"averageRating": 4.8,"numberOfVotes": 320}
}
```

```php
#[ApiResource(
    provider: State::class,
    processor: State::class,
)]
class Recipe
{
    public string $id;
    public string $title;
    public ?string $description;
    public int $preparationTime;
    public int $cookingTime;
    public array $ingredients;
    public array $steps;
    public array $popularity;
}
```

API Resource

Model Class

Data Transfert Object

Central Object

```
GET /api/recipes?<filters>

GET /api/recipes/<id>

POST /api/recipes

PATCH /api/recipes/<id>

DELETE /api/recipes/<id>
```

```php
readonly class State implements ProviderInterface
{
    public function __construct(
        private \MongoDB\Database $database,
        private \AutoMapper\AutoMapper $autoMapper,
    ) {}

    public function provide(Operation $operation, array $uriVariables, array $context): object
    {
        $resourceClass = $operation->getClass();

        if ($operation instanceof \ApiPlatform\Metadata\Get ||
            $operation instanceof \ApiPlatform\Metadata\Patch ||
            $operation instanceof \ApiPlatform\Metadata\Delete
        ) {
            $objectId = new \MongoDB\BSON\ObjectId($uriVariables['id']);
            $results = $this->getCollection($operation)->find(['_id' => $objectId])->toArray();
            $results = $this->autoMapper->mapCollection($results, $resourceClass);

            return $results[0] ?? null;
        }
    }
}
```

API Platform Provider

```php
readonly class State implements ProviderInterface
{
    public function __construct(
        private \MongoDB\Database $database,
        private \AutoMapper\AutoMapper $autoMapper,
    ) {}

    public function provide(Operation $operation, array $uriVariables, array $context): object
    {
        $resourceClass = $operation->getClass();

        if ($operation instanceof \ApiPlatform\Metadata\Get ||
            $operation instanceof \ApiPlatform\Metadata\Patch ||
            $operation instanceof \ApiPlatform\Metadata\Delete
        ) {
            $objectId = new \MongoDB\BSON\ObjectId($uriVariables['id']);
            $results = $this->getCollection($operation)->find(['_id' => $objectId])->toArray();
            $results = $this->autoMapper->mapCollection($results, $resourceClass);

            return $results[0] ?? null;
        }
    }
}
```

Operations requiring a single document to be read

```php
readonly class State implements ProviderInterface
{
    public function __construct(
        private \MongoDB\Database $database,
        private \AutoMapper\AutoMapper $autoMapper,
    ) {}

    public function provide(Operation $operation, array $uriVariables, array $context): object
    {
        $resourceClass = $operation->getClass();

        if ($operation instanceof \ApiPlatform\Metadata\Get ||
            $operation instanceof \ApiPlatform\Metadata\Patch ||
            $operation instanceof \ApiPlatform\Metadata\Delete
        ) {
            $objectId = new \MongoDB\BSON\ObjectId($uriVariables['id']);
            $results = $this->getCollection($operation)->find(['_id' => $objectId])->toArray();
            $results = $this->autoMapper->mapCollection($results, $resourceClass);

            return $results[0] ?? null;
        }
    }
}
```

MongoDB query syntax uses BSON

```php
readonly class State implements ProviderInterface
{
    public function __construct(
        private \MongoDB\Database $database,
        private \AutoMapper\AutoMapper $autoMapper,
    ) {}

    public function provide(Operation $operation, array $uriVariables, array $context): object
    {
        $resourceClass = $operation->getClass();

        if ($operation instanceof \ApiPlatform\Metadata\Get ||
            $operation instanceof \ApiPlatform\Metadata\Patch ||
            $operation instanceof \ApiPlatform\Metadata\Delete
        ) {
            $objectId = new \MongoDB\BSON\ObjectId($uriVariables['id']);
            $results = $this->getCollection($operation)->find(['_id' => $objectId])->toArray();
            $results = $this->autoMapper->mapCollection($results, $resourceClass);

            return $results[0] ?? null;
        }
    }
}
```

Generic State using configuration attached to the resource class

```php
readonly class State implements ProviderInterface
{
    public function __construct(
        private \MongoDB\Database $database,
        private \AutoMapper\AutoMapper $autoMapper,
    ) {}

    public function provide(Operation $operation, array $uriVariables, array $context): object
    {
        $resourceClass = $operation->getClass();

        if ($operation instanceof \ApiPlatform\Metadata\Get ||
            $operation instanceof \ApiPlatform\Metadata\Patch ||
            $operation instanceof \ApiPlatform\Metadata\Delete
        ) {
            $objectId = new \MongoDB\BSON\ObjectId($uriVariables['id']);
            $results = $this->getCollection($operation)->find(['_id' => $objectId])->toArray();
            $results = $this->autoMapper->mapCollection($results, $resourceClass);

            return $results[0] ?? null;
        }
    }
}
```

Map into the Resource Class

```php
readonly class State implements ProviderInterface
{
    public function __construct(
        private \MongoDB\Database $database,
        private \AutoMapper\AutoMapper $autoMapper,
    ) {}

    public function provide(Operation $operation, array $uriVariables, array $context): object
    {
        $resourceClass = $operation->getClass();

        if ($operation instanceof \ApiPlatform\Metadata\Get ||
            $operation instanceof \ApiPlatform\Metadata\Patch ||
            $operation instanceof \ApiPlatform\Metadata\Delete
        ) {
            $objectId = new \MongoDB\BSON\ObjectId($uriVariables['id']);
            $results = $this->getCollection($operation)->find(['_id' => $objectId])->toArray();
            $results = $this->autoMapper->mapCollection($results, $resourceClass);

            return $results[0] ?? null;
        }
    }
}
```

GET /api/recipes?<filters>

GET /api/recipes/<id>

POST /api/recipes

PATCH /api/recipes/<id>

DELETE /api/recipes/<id>

```php
readonly class State implements ProcessorInterface
{
    public function __construct(
        private \MongoDB\Database $database,
        private \AutoMapper\AutoMapper $autoMapper,
    ) {}

    public function process(mixed $data, Operation $operation, array $uriVariables, array $context)
    {
        if ($operation instanceof Post) {
            $data->id ??= (string) new ObjectId();
            $document = $this->autoMapper->map($data, 'array');
            $this->getCollection($operation)->insertOne($document);
        }

        if ($operation instanceof Patch) {
            $document = $this->autoMapper->map($data, 'array');
            $this->getCollection($operation)->replaceOne(
                ['_id' => new ObjectId($data->id)],
                $document);
        }

        if ($operation instanceof Delete) {
            $this->getCollection($operation)->deleteOne(['_id' => new ObjectId($data->id)]);
        }

        return $data;
    }
}
```

```php
readonly class State implements ProcessorInterface
{
    public function __construct(
        private \MongoDB\Database $database,
        private \AutoMapper\AutoMapper $autoMapper,
    ) {}

    public function process(mixed $data, Operation $operation, array $uriVariables, array $context)
    {
        if ($operation instanceof Post) {
            $data->id ??= (string) new ObjectId();
            $document = $this->autoMapper->map($data, 'array');
            $this->getCollection($operation)->insertOne($document);
        }

        if ($operation instanceof Patch) {
            $document = $this->autoMapper->map($data, 'array');
            $this->getCollection($operation)->replaceOne(
                ['_id' => new ObjectId($data->id)],
                $document);
        }

        if ($operation instanceof Delete) {
            $this->getCollection($operation)->deleteOne(['_id' => new ObjectId($data->id)]);
        }

        return $data;
    }
}
```

For write operations

```php
readonly class State implements ProcessorInterface
{
    public function __construct(
        private \MongoDB\Database $database,
        private \AutoMapper\AutoMapper $autoMapper,
    ) {}

    public function process(mixed $data, Operation $operation, array $uriVariables, array $context)
    {
        if ($operation instanceof Post) {
            $data->id ??= (string) new ObjectId();
            $document = $this->autoMapper->map($data, 'array');
            $this->getCollection($operation)->insertOne($document);
        }

        if ($operation instanceof Patch) {
            $document = $this->autoMapper->map($data, 'array');
            $this->getCollection($operation)->replaceOne(
                ['_id' => new ObjectId($data->id)],
                $document);
        }

        if ($operation instanceof Delete) {
            $this->getCollection($operation)->deleteOne(['_id' => new ObjectId($data->id)]);
        }

        return $data;
    }
}
```

Create the document

```php
readonly class State implements ProcessorInterface
{
    public function __construct(
        private \MongoDB\Database $database,
        private \AutoMapper\AutoMapper $autoMapper,
    ) {}

    public function process(mixed $data, Operation $operation, array $uriVariables, array $context)
    {
        if ($operation instanceof Post) {
            $data->id ??= (string) new ObjectId();
            $document = $this->autoMapper->map($data, 'array');
            $this->getCollection($operation)->insertOne($document);
        }

        if ($operation instanceof Patch) {
            $document = $this->autoMapper->map($data, 'array');
            $this->getCollection($operation)->replaceOne(
                ['_id' => new ObjectId($data->id)],
                $document);
        }

        if ($operation instanceof Delete) {
            $this->getCollection($operation)->deleteOne(['_id' => new ObjectId($data->id)]);
        }

        return $data;
    }
}
```

```php
readonly class State implements ProcessorInterface
{
    public function __construct(
        private \MongoDB\Database $database,
        private \AutoMapper\AutoMapper $autoMapper,
    ) {}

    public function process(mixed $data, Operation $operation, array $uriVariables, array $context)
    {
        if ($operation instanceof Post) {
            $data->id ??= (string) new ObjectId();
            $document = $this->autoMapper->map($data, 'array');
            $this->getCollection($operation)->insertOne($document);
        }

        if ($operation instanceof Patch) {
            $document = $this->autoMapper->map($data, 'array');
            $this->getCollection($operation)->replaceOne(
                ['_id' => new ObjectId($data->id)],
                $document);
        }

        if ($operation instanceof Delete) {
            $this->getCollection($operation)->deleteOne(['_id' => new ObjectId($data->id)]);
        }

        return $data;
    }
}
```

Remove the document

GET /api/recipes?<filters>

GET /api/recipes/<id>

POST /api/recipes

PATCH /api/recipes/<id>

DELETE /api/recipes/<id>

```php
#[ApiResource(
    provider: State::class,
    processor: State::class,
)]
class Recipe
{
    public string $id;

    #[ApiFilter(SearchFilter::class, strategy: 'partial')]
    public string $title;

    public ?string $description;
    public int $preparationTime;
    public int $cookingTime;
    public array $ingredients;
    public array $steps;
    public array $popularity;
}
```

Enable filter on this field

```php
#[\Symfony\Component\DependencyInjection\AttributeExclude]
class SearchFilter implements \ApiPlatform\Metadata\FilterInterface
{
    public function __construct(private array $properties) {}

    public function apply(array $query, array $context): array
    {
        foreach ($this->properties as $property => $strategy) {
            if (!array_key_exists($property, $context['filters'])) {
                continue;
            }

            $value = $context['filters'][$property];
            $query[$property] = match ($strategy) {
                'exact' => ['$eq' => $value],
                'partial' => ['$regex' => new Regex(preg_quote($value), 'i')],
            };
        }

        return $query;
    }
```

```php
#[\Symfony\Component\DependencyInjection\Attribute\Exclude]
class SearchFilter implements \ApiPlatform\Metadata\FilterInterface
{
    public function __construct(private array $properties) {}

    public function apply(array $query, array $context): array
    {
        foreach ($this->properties as $property => $strategy) {
            if (!array_key_exists($property, $context['filters'])) {
                continue;
            }

            $value = $context['filters'][$property];
            $query[$property] = match ($strategy) {
                'exact' => ['$eq' => $value],
                'partial' => ['$regex' => new Regex(preg_quote($value), 'i')],
            };
        }

        return $query;
    }
```

services defined by a compiler pass for each API resource

```php
#[\Symfony\Component\DependencyInjection\AttributeExclude]
class SearchFilter implements \ApiPlatform\Metadata\FilterInterface
{
    public function __construct(private array $properties) {}

    public function apply(array $query, array $context): array
    {
        foreach ($this->properties as $property => $strategy) {
            if (!array_key_exists($property, $context['filters'])) {
                continue;
            }

            $value = $context['filters'][$property];
            $query[$property] = match ($strategy) {
                'exact' => ['$eq' => $value],
                'partial' => ['$regex' => new Regex(preg_quote($value), 'i')],
            };
        }

        return $query;
    }
```

From the HTTP request query string

```php
#[\Symfony\Component\DependencyInjection\AttributeExclude]
class SearchFilter implements \ApiPlatform\Metadata\FilterInterface
{
    public function __construct(private array $properties) {}

    public function apply(array $query, array $context): array
    {
        foreach ($this->properties as $property => $strategy) {
            if (!array_key_exists($property, $context['filters'])) {
                continue;
            }

            $value = $context['filters'][$property];
            $query[$property] = match ($strategy) {
                'exact' => ['$eq' => $value],
                'partial' => ['$regex' => new Regex(preg_quote($value), 'i')],
            };
        }

        return $query;
    }
```

Appends the query condition using property configuration

```php
readonly class State implements ProviderInterface
{
    public function __construct(
        private Database $database,
        #[Autowire(service: 'api_platform.filter_locator')] private ServiceProviderInterface $filters,
        private AutoMapper $autoMapper,
    ) {}

    public function provide(Operation $operation, array $uriVariables, array $context)
    {
        if ($operation instanceof GetCollection) {
            $query = [];
            foreach ($operation->getFilters() as $filterId) {
                $filter = $this->filters->get($filterId);
                $query = $filter->apply($query, $context);
            }

            $results = $this->getCollection($operation)->find($query)->toArray();

            $results = $this->autoMapper->mapCollection($results, $operation->getClass());

            return $results;
        }
    }
}
```

Provider for listing operations

```php
readonly class State implements ProviderInterface
{
    public function __construct(
        private Database $database,
        #[Autowire(service: 'api_platform.filter_locator')] private ServiceProviderInterface $filters,
        private AutoMapper $autoMapper,
    ) {}

    public function provide(Operation $operation, array $uriVariables, array $context)
    {
        if ($operation instanceof GetCollection) {
            $query = [];
            foreach ($operation->getFilters() as $filterId) {
                $filter = $this->filters->get($filterId);
                $query = $filter->apply($query, $context);
            }

            $results = $this->getCollection($operation)->find($query)->toArray();

            $results = $this->autoMapper->mapCollection($results, $operation->getClass());

            return $results;
        }
    }
}
```

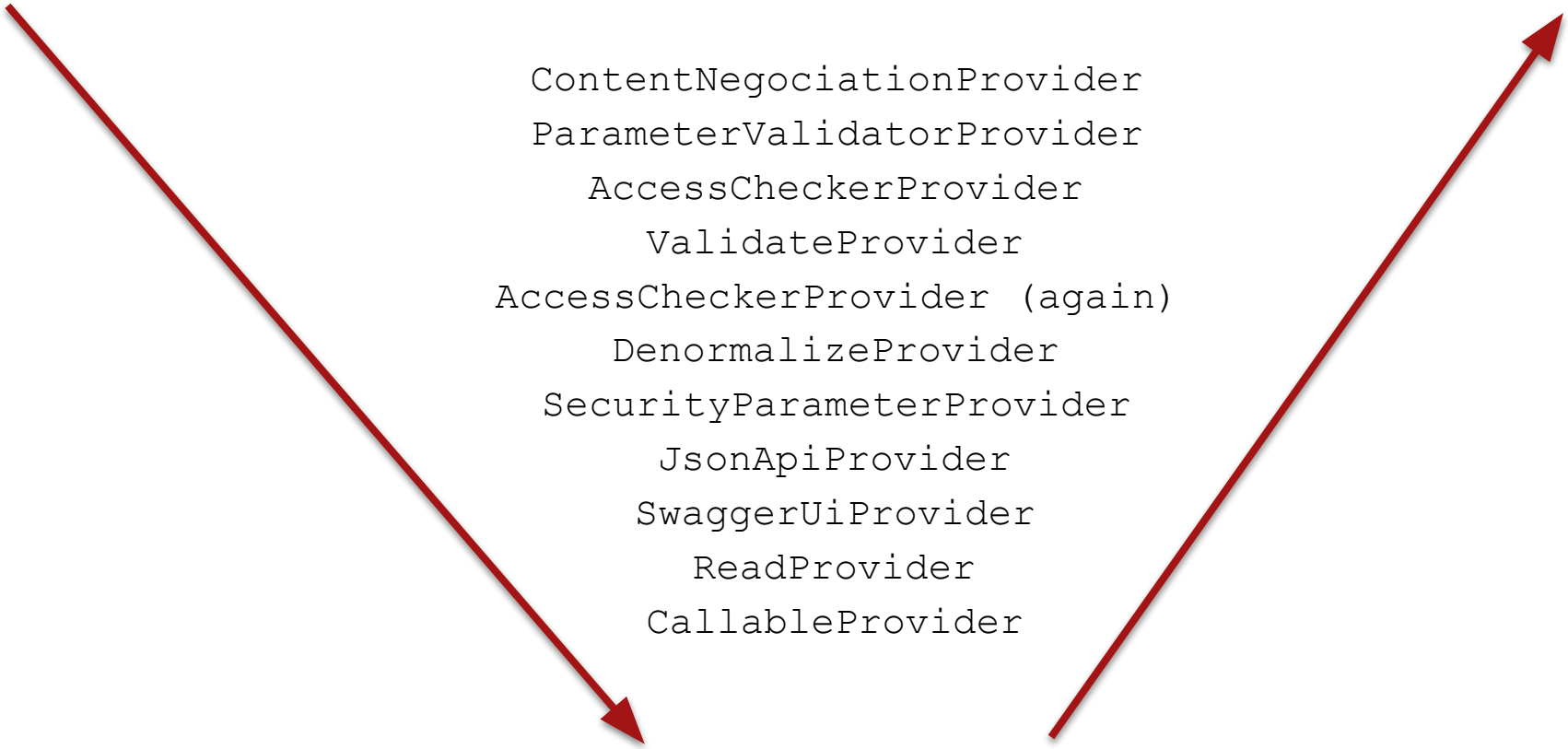
Inject all filter services

Retrieved by internal id

```php
readonly class State implements ProviderInterface
{
    public function __construct(
        private Database $database,
        #[Autowire(service: 'api_platform.filter_locator')] private ServiceProviderInterface $filters,
        private AutoMapper $autoMapper,
    ) {}

    public function provide(Operation $operation, array $uriVariables, array $context)
    {
        if ($operation instanceof GetCollection) {
            $query = [];
            foreach ($operation->getFilters() as $filterId) {
                $filter = $this->filters->get($filterId);
                $query = $filter->apply($query, $context);
            }

            $results = $this->getCollection($operation)->find($query)->toArray();

            $results = $this->autoMapper->mapCollection($results, $operation->getClass());

            return $results;
        }
    }
}
```

Execute the query, map the results ...

API Platform Lasagna



State provider and processor that delegates to Doctrine

```php
#[ApiResource(
    stateOptions: new Options(
        documentClass: RecipeEntity::class,
    ),
    provider: BridgedState::class,
    processor: BridgedState::class,
)]
class RecipeDTO
{
    // …
}
```

Inject Doctrine state provider

```php
class BridgedState implements ProviderInterface
{
    public function __construct(
        private \AutoMapper\AutoMapperInterface $autoMapper,
        private \ApiPlatform\Doctrine\Odm\State\CollectionProvider $collectionProvider,
    ) {}

    public function provide(Operation $operation, array $uriVariables = [], array $context = [])
    {
        $resourceClass = $operation->getClass();
        $doctrineClass = $operation->getStateOptions()->getDocumentClass();
        if ($operation instanceof GetCollection) {
            $results = $this->collectionProvider->provide($operation, $uriVariables, $context);
            assert($results instanceof PaginatorInterface);

            return new TraversablePaginator(
                new \ArrayIterator($this->autoMapper->mapCollection($results, $resourceClass)),
                $results->getCurrentPage(),
                $results->getItemsPerPage(),
                $results->getTotalItems(),
            );
        }
    }
}
```

```php
class BridgedState implements ProviderInterface
{
    public function __construct(
        private \AutoMapper\AutoMapperInterface $autoMapper,
        private \ApiPlatform\Doctrine\Odm\State\CollectionProvider $collectionProvider,
    ) {}

    public function provide(Operation $operation, array $uriVariables = [], array $context = [])
    {
        $resourceClass = $operation->getClass();
        $doctrineClass = $operation->getStateOptions()->getDocumentClass();
        if ($operation instanceof GetCollection) {
            $results = $this->collectionProvider->provide($operation, $uriVariables, $context);
            assert($results instanceof PaginatorInterface);

            return new TraversablePaginator(
                new \ArrayIterator($this->autoMapper->mapCollection($results, $resourceClass)),
                $results->getCurrentPage(),
                $results->getItemsPerPage(),
                $results->getTotalItems(),
            );
        }
    }
}
```

Delegate the query to the Doctrine Provider

```php
class BridgedState implements ProviderInterface
{
    public function __construct(
        private \AutoMapper\AutoMapperInterface $autoMapper,
        private \ApiPlatform\Doctrine\Odm\State\CollectionProvider $collectionProvider,
    ) {}

    public function provide(Operation $operation, array $uriVariables = [], array $context = [])
    {
        $resourceClass = $operation->getClass();
        $doctrineClass = $operation->getStateOptions()->getDocumentClass();
        if ($operation instanceof GetCollection) {
            $results = $this->collectionProvider->provide($operation, $uriVariables, $context);
            assert($results instanceof PaginatorInterface);

            return new TraversablePaginator(
                new \ArrayIterator($this->autoMapper->mapCollection($results, $resourceClass)),
                $results->getCurrentPage(),
                $results->getItemsPerPage(),
                $results->getTotalItems(),
            );
        }
    }
}
```

Profit from all the Doctrine features, filters and pagination

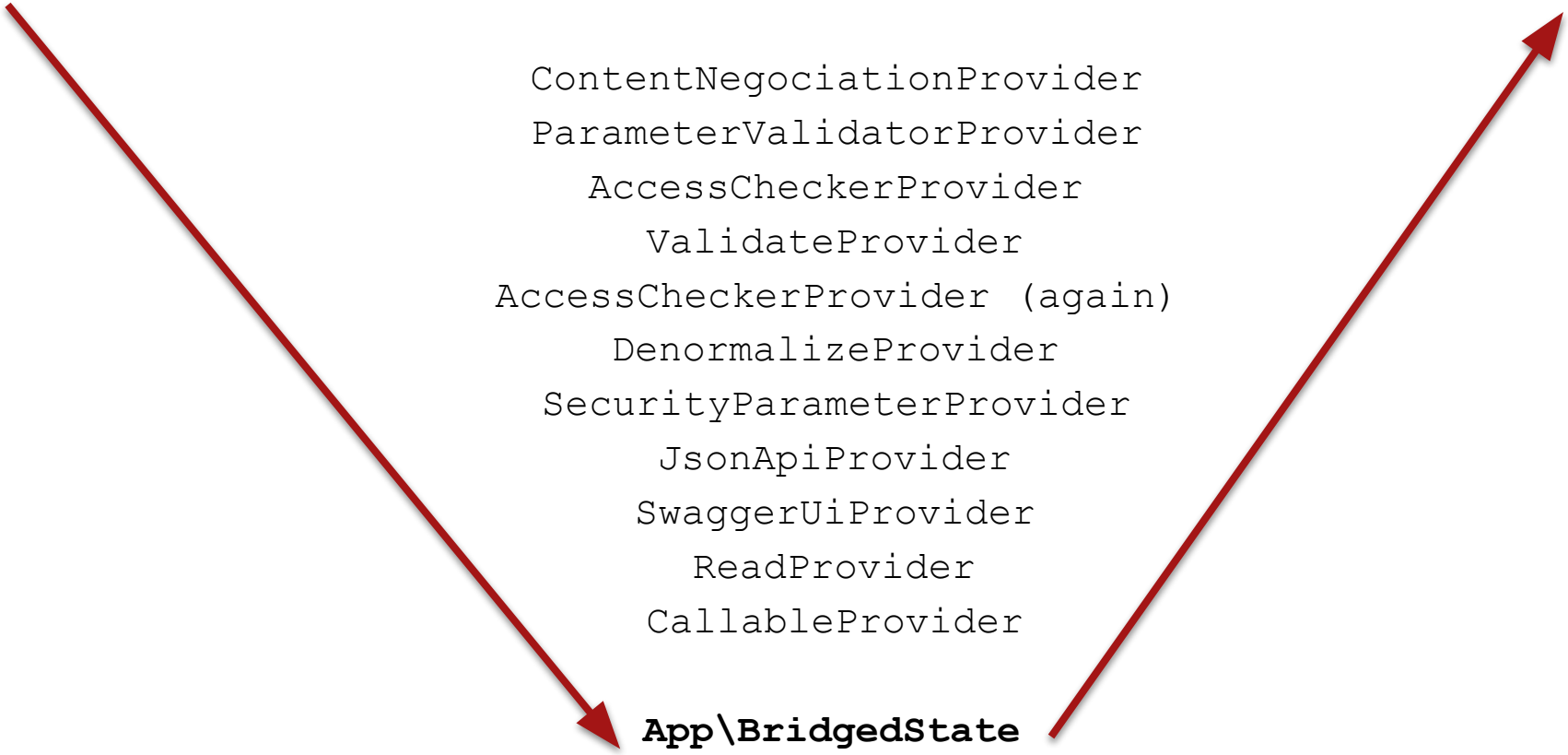
```php
class BridgedState implements ProviderInterface
{
    public function __construct(
        private \AutoMapper\AutoMapperInterface $autoMapper,
        private \ApiPlatform\Doctrine\Odm\State\CollectionProvider $collectionProvider,
    ) {}

    public function provide(Operation $operation, array $uriVariables = [], array $context = [])
    {
        $resourceClass = $operation->getClass();
        $doctrineClass = $operation->getStateOptions()->getDocumentClass();
        if ($operation instanceof GetCollection) {
            $results = $this->collectionProvider->provide($operation, $uriVariables, $context);
            assert($results instanceof PaginatorInterface);

            return new TraversablePaginator(
                new \ArrayIterator($this->autoMapper->mapCollection($results, $resourceClass)),
                $results->getCurrentPage(),
                $results->getItemsPerPage(),
                $results->getTotalItems(),
            );
        }
    }
}
```

Map to the API resource class, to customize the HTTP response

```php
class BridgedState implements ProviderInterface
{
    public function __construct(
        private \AutoMapper\AutoMapperInterface $autoMapper,
        private \ApiPlatform\Doctrine\Odm\State\CollectionProvider $collectionProvider,
    ) {}

    public function provide(Operation $operation, array $uriVariables = [], array $context = [])
    {
        $resourceClass = $operation->getClass();
        $doctrineClass = $operation->getStateOptions()->getDocumentClass();
        if ($operation instanceof GetCollection) {
            $results = $this->collectionProvider->provide($operation, $uriVariables, $context);
            assert($results instanceof PaginatorInterface);

            return new TraversablePaginator(
                new \ArrayIterator($this->autoMapper->mapCollection($results, $resourceClass)),
                $results->getCurrentPage(),
                $results->getItemsPerPage(),
                $results->getTotalItems(),
            );
        }
    }
}
```

Keep the pagination

API Platform Lasagna



Conclusions

1. API Platform if fast for RAD, and flexible for customization,
2. Object Mapping is critical to translate between the database and the API,
3. Store the data the way you use it,
4. Lasagna tastes better than spaghetti