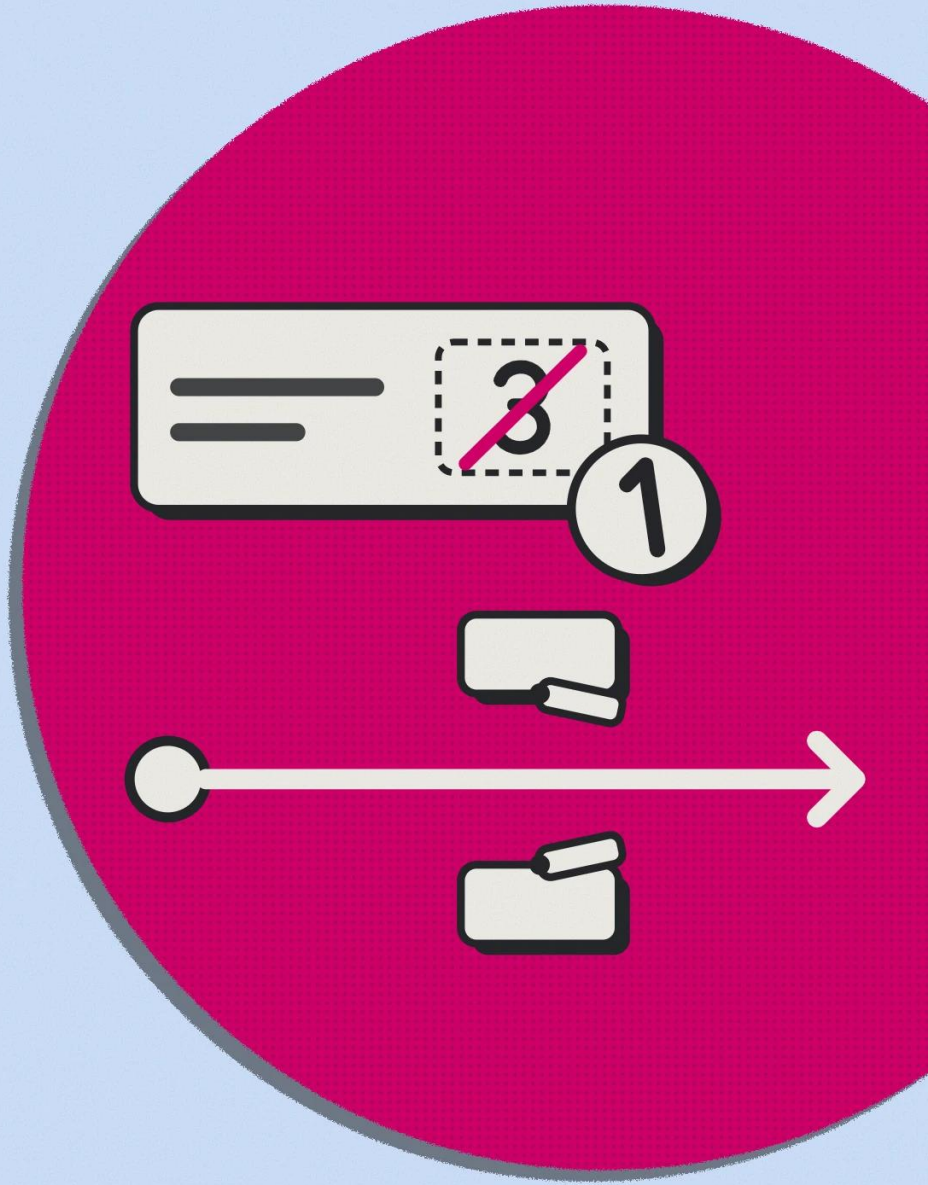
2.02

TEMPS 2 · LE TAMAGOTCHI TROUÉ · SÉANCE 1

Le contrôle d'accès

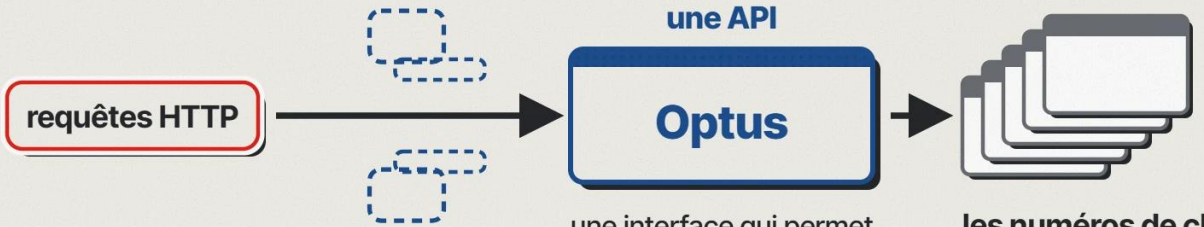
Thibault Six · IFOSUP Wavre · 2026-2027



? Un dossier de client se demande par son numéro. Que se passe-t-il si quelqu'un change ce numéro ?

Chez Optus, un contrôle d'accès contournable a touché environ 9,5 millions de personnes.

rien ne vérifiait que la personne qui demandait un dossier avait le droit de le lire



une interface qui permet à des programmes d'échanger des données

les numéros de clients se suivaient

selon les analyses publiées

SEPTEMBRE 2022

9,5 millions

de personnes touchées

2,1 millions

exposées au vol d'identité par leur permis de conduire ou leur passeport

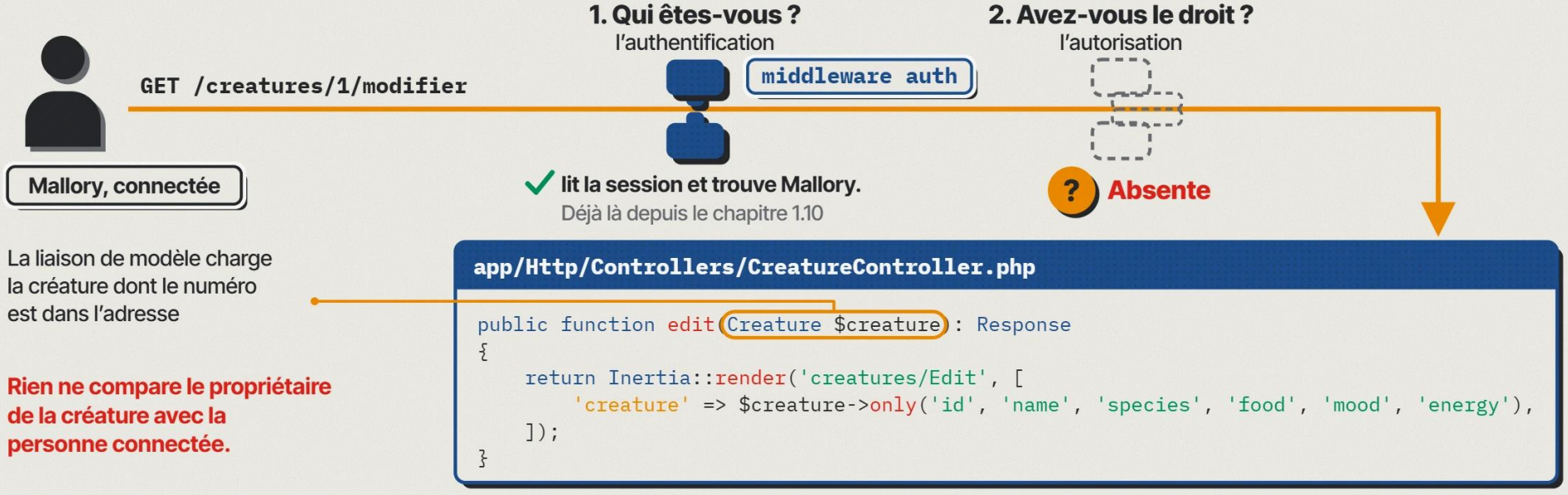
le commissaire australien à l'information

Dans le Tamagotchi, chaque créature a aussi un numéro. Ces numéros se suivent, et ils apparaissent dans l'adresse.

À gauche, ce qui manquait à l'API : une vérification à chaque dossier demandé. À droite, ce que ce manque a coûté.

? Mallory est connectée. Le middleware auth suffit-il pour qu'elle ne touche qu'à ses créatures ?

Le middleware `auth` demande qui vous êtes ; `edit()` ne demande jamais si la créature est à vous.



l'autorisation : vérifier qu'une personne connectée peut faire une action précise sur une donnée précise

La requête passe le premier portillon, puis trouve le second ouvert : `edit()` renvoie la créature demandée, quelle qu'elle soit.

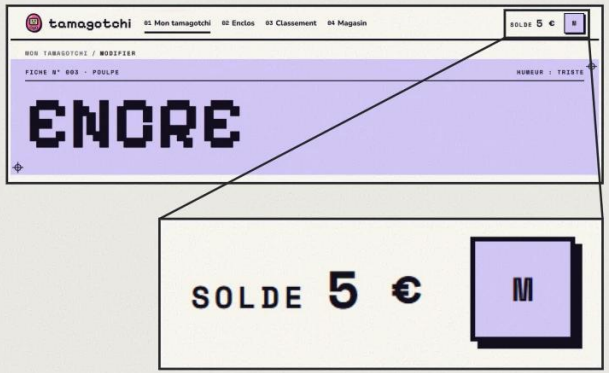
? Aucun lien ne mène Mallory à la page de Braise. Comment l'ouvre-t-elle quand même ?

Mallory remplace 3 par 1 dans l'adresse, et le serveur lui ouvre la page de Braise.

① Trouver un numéro

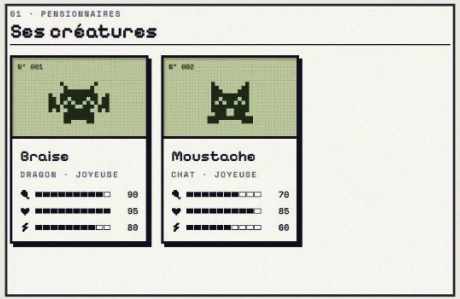
`tamagotchi.test/creatures/3/modifier`

Encre, la créature de Mallory



② L'enclos d'Alice

`/enclos/1`



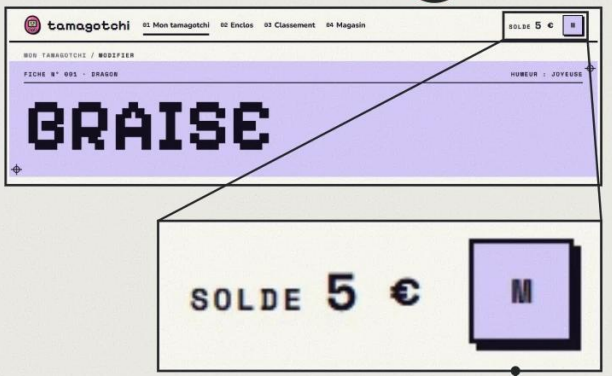
Braise est la créature n° 001, Moustache la n° 002.

Aucun lien « Modifier » n'apparaît sous les vignettes.

③ Changer le numéro

`tamagotchi.test/creatures/1/modifier`

Braise, la créature d'Alice — 200



L'en-tête montre pourtant le solde de Mallory, 5 €, et son initiale.

Lisez les deux adresses : un seul chiffre change. Le haut des deux pages montre le solde et l'initiale de Mallory, connectée.

? Sur la page de Braise, Mallory la renomme, puis clique sur « Jouer ». Le serveur refuse-t-il ?

Le serveur laisse Mallory renommer Braise et jouer avec elle, comme si c'était sa créature.

Renommer

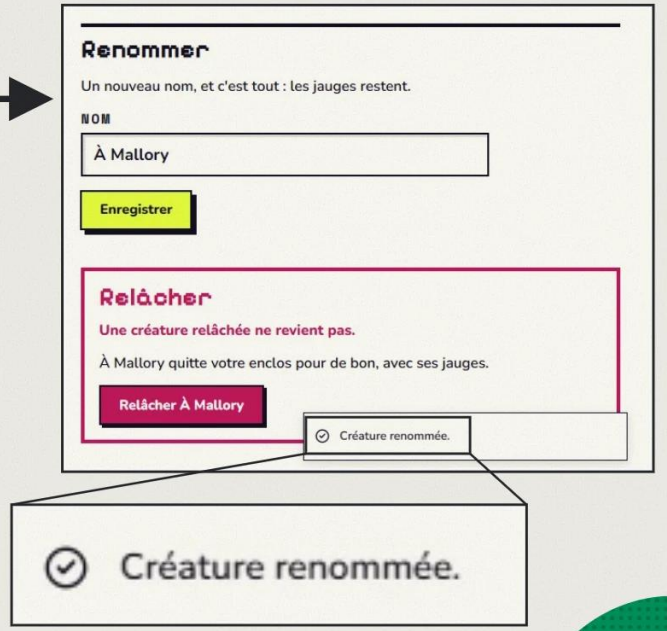
```
POST /creatures/1?_method=PUT
{"name":"À Mallory"}
Réponse : 303, puis la page de modification
```

Jouer

```
POST /creatures/1/jouer
Réponse : 302, retour à la page de modification
```

l'onglet Réseau des outils de développement

Le message « Créature renommée. » s'affiche



Dans l'onglet Réseau, rien ne distingue ces requêtes de celles d'Alice sur sa propre créature : le serveur les accepte.

? La page de Braise n'a pas de touche « Dodo ». Mallory peut-elle quand même faire dormir Braise ?

Mallory tape deux lignes dans la console, et Braise s'endort sans qu'aucun bouton existe.

Console

```
const token = decodeURIComponent(document.cookie.match(/XSRF-TOKEN=([^;]+)/)[1]);
fetch('/creatures/1/dormir', { method: 'POST', headers: { 'X-XSRF-TOKEN': token } });
```

le jeton CSRF : une valeur que Laravel range dans le cookie XSRF-TOKEN et exige avec chaque envoi

Sans ce jeton, la même requête reçoit 419 et rien ne change.

POST	dormir	302
		200

fetch() envoie une requête depuis la page, avec les cookies de la session de Mallory

avant « Jouer »





après une partie et une sieste qu'Alice n'a jamais demandées

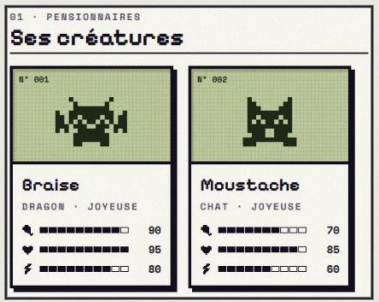
Les deux lignes partent de la page de Mallory, avec son cookie et son jeton. Les jauges changent : le clic sur « Jouer » et la sieste sont passés.

? Mallory clique sur « Relâcher À Mallory ». Que reste-t-il dans l'enclos d'Alice ?

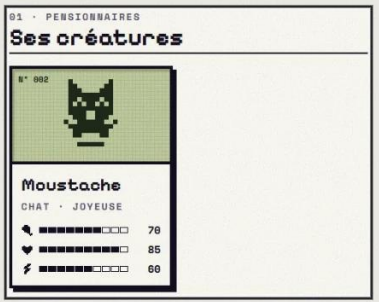
Braise disparaît de l'enclos d'Alice, et cinq routes laissent passer Mallory de la même façon.

```
POST /creatures/1?_method=DELETE
Réponse : 303, puis /dashboard
```

avant



après



Dans l'enclos d'Alice, Braise a disparu.

`php artisan route:list --path=creatures`

- `GET|HEAD creatures/{creature}/modifier`
- `PUT|PATCH creatures/{creature}`
- `POST creatures/{creature}/jouer`
- `POST creatures/{creature}/dormir`
- `DELETE creatures/{creature}`

Cinq routes laissent passer Mallory.

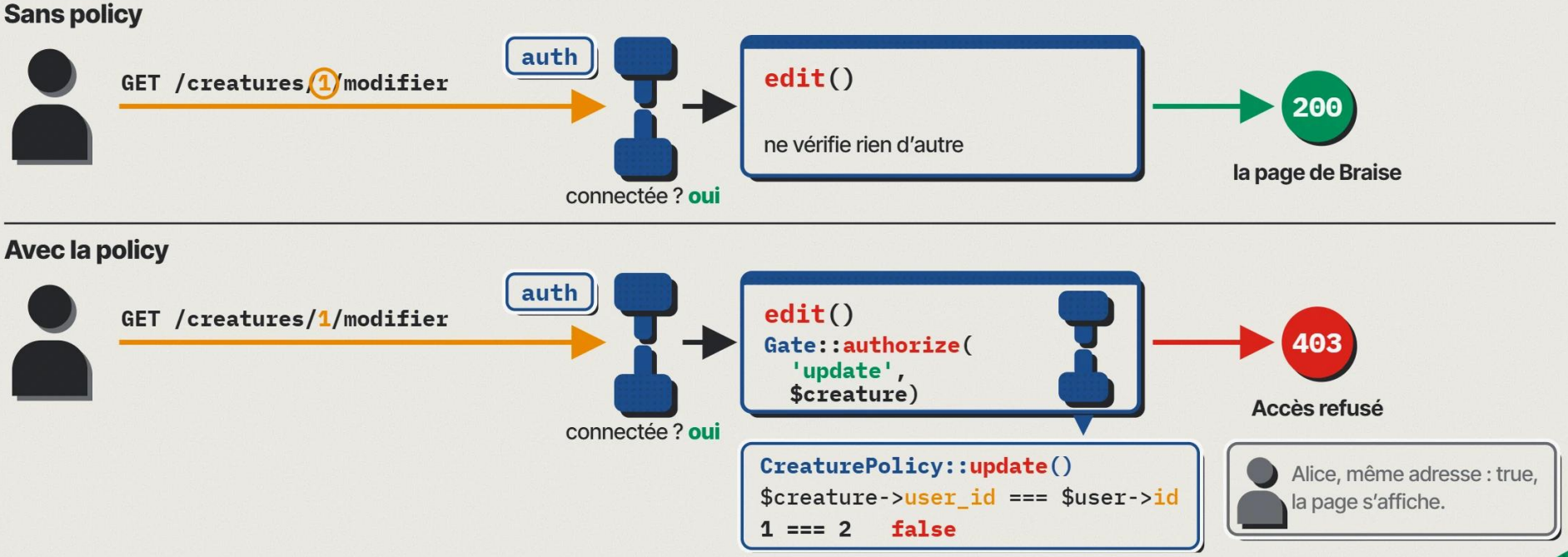
`POST creatures` `GET|HEAD creatures/nouvelle`

ne visent aucune créature existante

En haut, Braise supprimée chez Alice. En bas, cinq routes reçoivent un numéro de créature, et aucune ne vérifie à qui elle est.

? Que faudrait-il ajouter sur le trajet de la requête pour que Mallory reçoive un refus ?

Sans vérification, `edit()` répond 200 ; quand `edit()` consulte une policy, Mallory reçoit 403.



Comparez les deux bandes : un seul objet s'ajoute, le portillon de `Gate::authorize()` dans `edit()`. Il pose la question qui manquait.

? La même vérification serait copiée dans cinq méthodes. Où Laravel range-t-il cette règle ?

Une policy range la règle, une méthode par capacité, et `Gate::authorize()` la consulte.

```
$ php artisan make:policy CreaturePolicy --model=Creature
```

app/Policies/CreaturePolicy.php

```php
public function update(User $user, Creature $creature): bool
{
    return false;
}
```

Il reste à écrire la règle dans la policy.

update() et delete() comparent chacune le propriétaire de la créature avec la personne connectée.

extrait

Laravel la trouve par son nom dans app/Policies

```php
Gate::authorize('update', $creature);
```

en première ligne des cinq actions, avec la capacité du tableau

CE QUE VOUS DEVEZ OBTENIR

Action	Capacité vérifiée
`edit()`	`update`
`update()`	`update`
`play()`	`update`
`sleep()`	`update`
`destroy()`	`delete`

une capacité : le nom d'une action qu'on autorise ou non

```
Tests:    5 failed, 1 passed (6 assertions)
```

Une policy seule ne protège rien.

avec `Gate::authorize()` dans `edit()`

```
× the owner still can
Expected response status code [200] but received 403.
```

Mallory est refusée, mais Alice aussi.

Le premier ticket : la policy seule ne change rien. Le second : consultée mais encore à `false`, elle refuse aussi Alice.

? Qu'est-ce qui doit être vrai à la fin ?

À vous, quinze minutes : `Gate::authorize()` dans cinq actions, six tests verts, Mallory refusée.

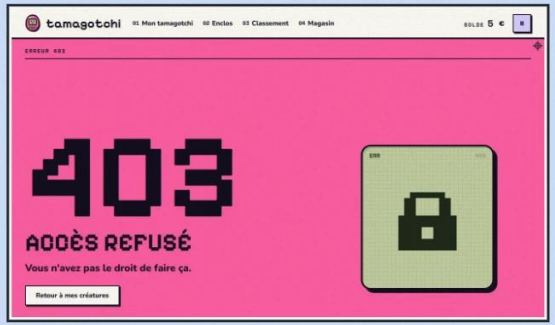
À FAIRE 15 min

1. Dans `CreaturePolicy`, gardez `update()` et `delete()`, et supprimez les cinq autres méthodes.
2. Appelez `Gate::authorize()` en première ligne des cinq actions, avec la capacité du tableau.
3. Affichez un lien « Modifier » sous chaque vignette de l'enclos, seulement si la personne connectée a le droit de modifier la créature.

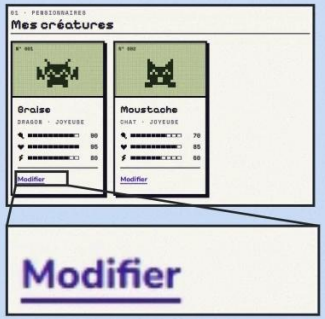
À VÉRIFIER

```
$ php artisan test --filter=Flaw02
Tests:    6 passed (10 assertions) ✓
```

Mallory 403 `tamagotchi.test/creatures/1/modifier`



Alice



LA RÈGLE

Chaque action sur une donnée vérifie côté serveur, avec une policy, que la personne connectée a le droit de la toucher. Un lien masqué ne protège rien.

Le lien « Modifier » n'est que du confort ; la protection, c'est `Gate::authorize()` côté serveur, sur chaque requête.