2.01

TEMPS 2 · LE TAMAGOTCHI TROUÉ · SÉANCE 1

L'application trouée



Thibault Six · IFOSUP Wavre · 2026-2027

? Cette fois, vous ne construisez rien. Que recevez-vous, et qu'allez-vous en faire ?

Ce jeu fonctionne, mais on l'a percé de failles : vous allez les attaquer, puis les corriger.

une faille : une erreur dans le code qui permet à quelqu'un de faire ce que le site devrait lui refuser

SÉANCE 1 · QUI A LE DROIT ?

- 2.02 — Changer le numéro dans l'adresse
- 2.03 — Un champ de trop dans le formulaire
- 2.04 — Tout ce que la page reçoit est public
- 2.05 — Du code dans un message
- 2.06 — Une recherche qui lit toute la base

SÉANCE 2 · L'ARGENT

- 2.07
- 2.08
- 2.09
- 2.10
- 2.11
- 2.12

Une ligne par séance, une station par chapitre. Le premier chapitre de faille, cerclé, est celui qui suit.

? Vous clonez votre fork. Comment votre copie devient-elle tamagotchi.test, avec ses joueurs ?

Le nom du dossier donne l'adresse `tamagotchi.test`, et `migrate:fresh --seed` rend un jeu neuf.

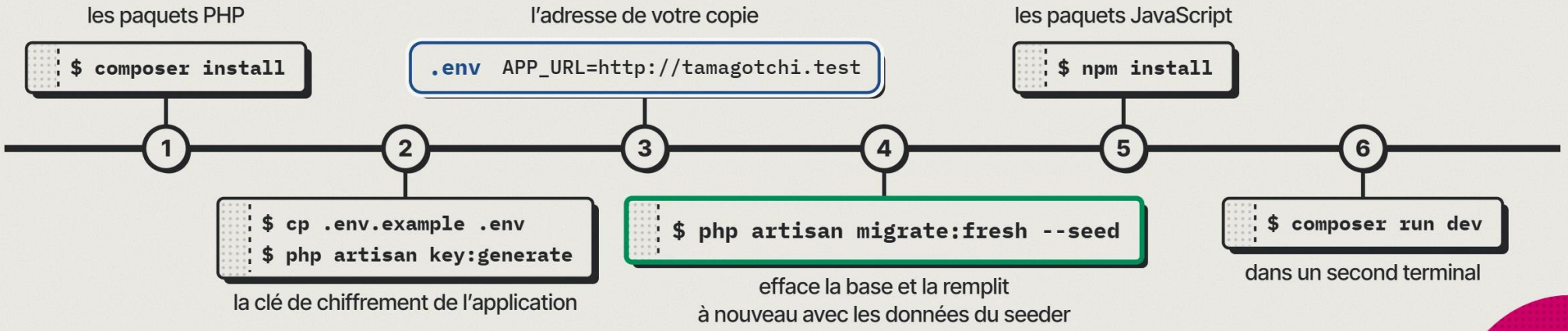
`opmvpc/tamagotchi-troue-26` — « Fork » puis « Create fork » → `votre-compte/tamagotchi-troue-26`

```
$ git clone https://github.com/votre-compte/tamagotchi-troue-26 tamagotchi
$ cd tamagotchi
```

le dernier mot de la commande `git clone` donne son nom au dossier

Votre dossier est relié à votre fork. → `tamagotchi.test`

Herd et Laragon en tirent l'adresse du site



Suivez la ligne de 1 à 6. Retenez la station 4 : après une attaque qui a tout cassé, elle vous rend un jeu neuf.

? Connecté comme Alice, que trouvez-vous dans le jeu, et quelles données chaque page lit-elle ?

Le menu mène à quatre pages : vos créatures, l'enclos d'un joueur, le classement, le magasin.

le solde : l'argent du jeu de la personne connectée

`/dashboard`

Mon tamagotchi



trois jauges de 0 à 100 : nourriture, humeur, énergie

LIT

- `Creature`

`/enclos/{user}`

Enclos



sa devise, ses créatures et son livre d'or

LIT

- `User`
- `Creature`
- `Message`

`/classement`

Classement



range les créatures de tous les joueurs par points

LIT

- `Creature`
- `User`

`/magasin`

Magasin



vend sept articles

LIT

- `Item`
- `Creature`

ÉCRIT

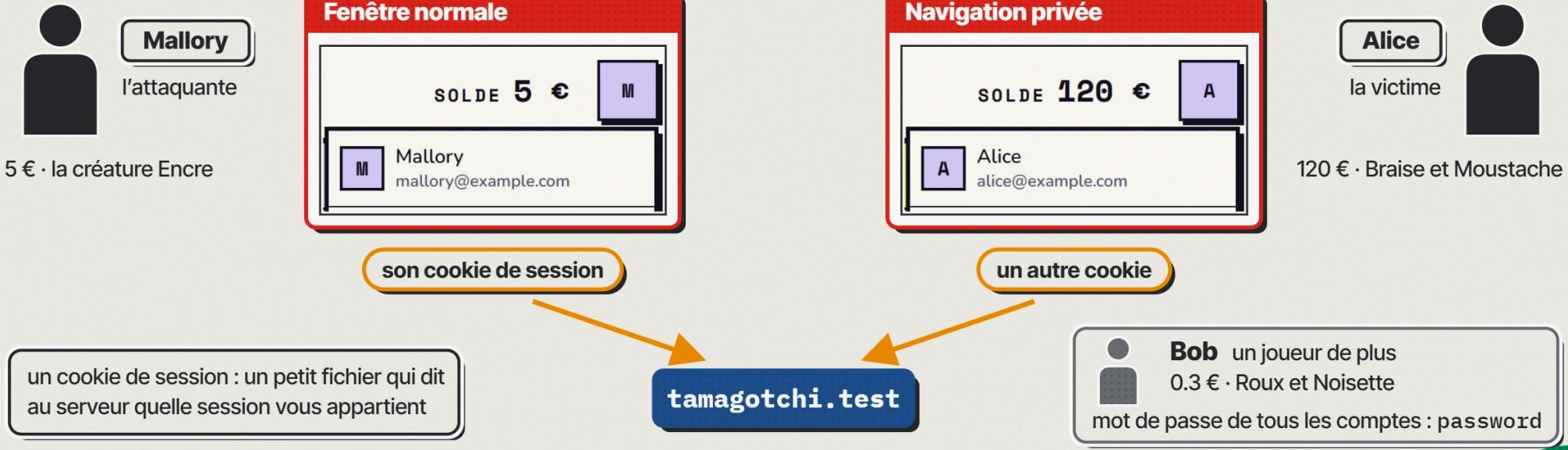
- `Purchase`
- `GiftCode`

Chaque colonne : l'adresse, la page, puis les modèles qu'elle lit. L'achat et le code cadeau, eux, écrivent dans `Purchase` et `GiftCode`.

? Pour voir une attaque des deux côtés, comment être Mallory et Alice en même temps ?

Une fenêtre privée garde ses propres cookies : Mallory dans l'une, Alice dans l'autre.

le menu du joueur, le carré à l'initiale en haut à droite : il affiche le nom et l'adresse e-mail du compte



Chaque fenêtre envoie son propre cookie : le site voit deux joueurs. Le menu du joueur dit qui est connecté dans chaque fenêtre.

? Alice voit Braise et Moustache, Mallory voit Encre. Comment la même page sait-elle qui demande ?

Le jeu sait qui demande grâce à `$request->user()`, mais pas encore s'il en a le droit.

Le Tamagotchi est construit sur le même kit que votre portfolio.

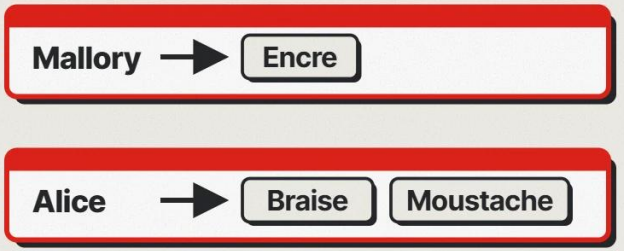


`auth` Toutes les routes du jeu passent par auth.

app/Http/Controllers/CreatureController.php · index() extrait

```php
return Inertia::render('creatures/Index', [
    'creatures' => $request->user()->creatures()
        ->orderBy('name')
        ->get(['id', 'name', 'species', 'food', 'mood', 'energy']),
]);
```

ne lit que les créatures de la personne connectée



✓ **L'application sait donc qui envoie chaque requête.**

? **ce joueur connecté a-t-il le droit de faire ce qu'il demande ?**
les failles de ce temps commencent après cette question

Le même code, deux joueurs, deux listes : la requête porte le joueur connecté. La seconde question, en bas, n'a pas encore de réponse.

? L'attaque marche sur votre copie. Pouvez-vous l'essayer sur le site d'un camarade, pour voir ?

Vous n'attaquez que votre propre copie : ailleurs, la loi punit l'accès sans autorisation.



votre copie, sur votre machine

vous êtes à la fois le propriétaire du site, Alice, Mallory et Bob

Tout y est permis.

~~le site d'un autre étudiant~~

~~celui de l'école~~

~~un site en ligne pour « voir si ça marche aussi »~~

la loi

le Code pénal punit le fait d'entrer sans autorisation dans un système informatique, même sans rien abîmer

la confiance de l'entreprise qui vous engage

elle a besoin de savoir que vous connaissez ces gestes et que vous ne les faites que là où on vous l'a demandé

À gauche, le seul endroit où l'attaque est permise. En bas, les deux raisons : la loi, puis la confiance de ceux qui vous engageront.

? Sur la copie neuve, php artisan test : combien de tests échouent, et à quoi servent-ils ?

24 tests sur 85 sont rouges, rangés par faille ; les 61 verts doivent le rester.

```
$ php artisan test
Tests:    24 failed, 61 passed (290 assertions)
```

tests/Feature/Flaws/

séance 1

2.02 ✗✗✗✗✗✓ the owner still can
il doit le rester après votre correction

2.03 ✗

2.04 ✗

2.05 ✗

2.06 ✗✗

séance 2

2.07 ✗✗✗✗

2.08 ✗✗✗✗✗✗

2.09 ✗✗✗

2.10 ✗

24 rouges
à faire passer au vert

61 verts
qui doivent le rester
42 du kit, 18 du jeu, 1 dans Flaw02

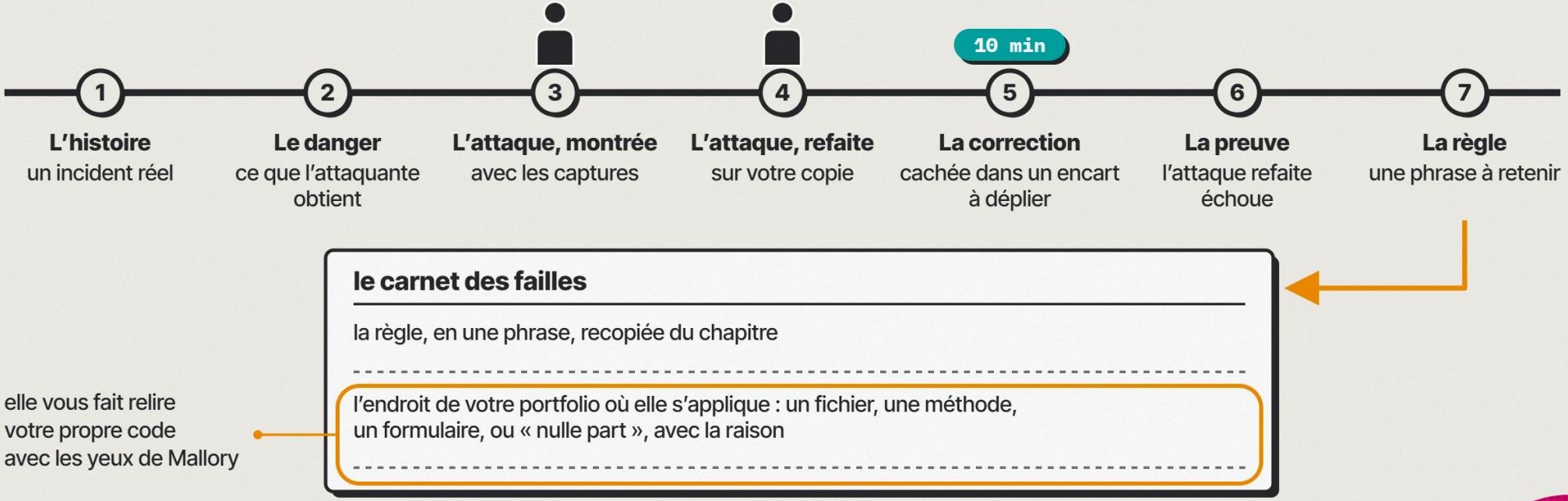
```
$ php artisan test --filter=Flaw02 → Tests:    5 failed, 1 passed (6 assertions)
```

les tests d'une seule faille

Chaque pastille rouge est une vérification qui échoue. Une faille est corrigée quand toute sa rangée est verte, et les 61 verts restent verts.

? Un chapitre de faille suit un même ordre. Par quoi commence-t-il, et par quoi finit-il ?

Un chapitre de faille suit en général sept temps ; sa règle va dans votre carnet des failles.



Lisez la ligne de 1 à 7. Trois chapitres varient cet ordre : le 2.04, le 2.07 et le 2.09. Le carnet vous renvoie à votre portfolio.

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

À vous, dix minutes : votre copie installée, deux joueurs connectés, et 24 tests rouges.

10 min

À FAIRE

1. Le dépôt cloné dans `tamagotchi`, `composer install`, `.env` copié avec `APP_URL=http://tamagotchi.test`, la clé générée.
2. `php artisan migrate:fresh --seed`, `npm install`, puis `composer run dev` dans le second terminal.
3. Mallory connectée dans une fenêtre normale, Alice dans une fenêtre de navigation privée.
4. `php artisan test` dans le premier terminal.

À VÉRIFIER

- [ ] Le menu du joueur affiche Mallory dans une fenêtre, Alice dans l'autre.
- [ ] Depuis le classement, Mallory ouvre l'enclos d'Alice et y lit le message de Bob.
- [ ] `php artisan test` se termine par `24 failed, 61 passed`.
- [ ] `php artisan test --filter=Flaw02` se termine par `5 failed, 1 passed`.

Quatre étapes à gauche, quatre vérifications à droite. Une machine qui n'affiche pas `24 failed, 61 passed` doit être réglée maintenant.