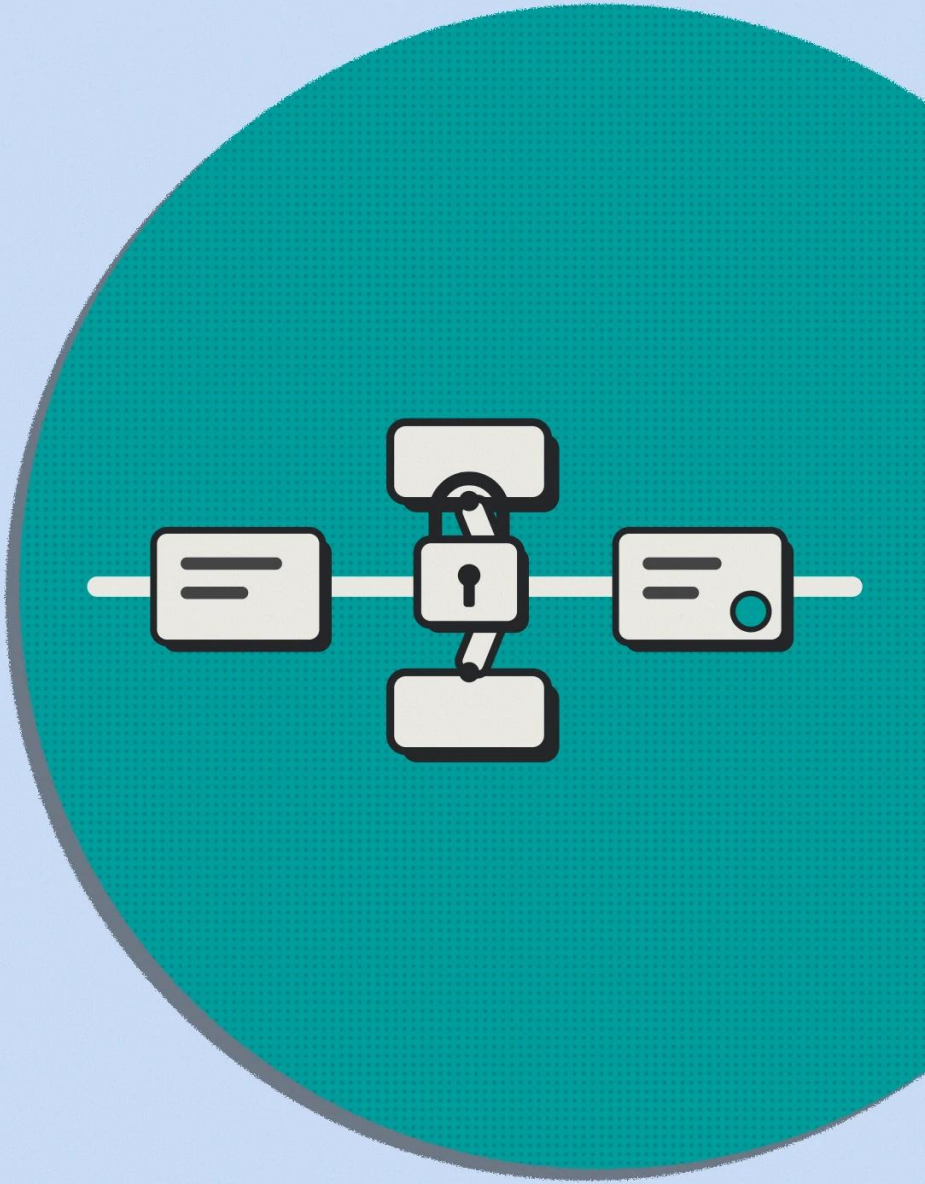
2.09

TEMPS 2 · LE TAMAGOTCHI TROUÉ · SÉANCE 2

L'achat atomique



Thibault Six · IFOSUP Wavre · 2026-2027

? Deux navigateurs confirment au même instant le même transfert entre cartes cadeau. Et le solde ?

Deux confirmations envoyées au même instant ont versé deux fois les mêmes 5 $ sur une carte.

STARBUCKS, 2015



DANS LE TAMAGOTCHI, LE MAGASIN A LA MÊME FORME



En haut, deux confirmations au même instant versent deux fois le même argent. En bas, le magasin du jeu fait les mêmes gestes.

? Le prix vient de la base, la quantité est bornée, le solde contrôlé. Où l'achat peut-il déraper ?

`store()` lit le solde, le contrôle, puis l'écrit : entre-temps, une autre requête peut le lire.

app/Http/Controllers/PurchaseController.php

au début de la requête

- **`$item`** par la liaison de modèle
- **`$user`** par le middleware auth

```
if ($user->balance_cents < $total) {
```

save() envoie à la base le solde calculé en mémoire

```
$user->balance_cents -= $total;
$user->save();
```

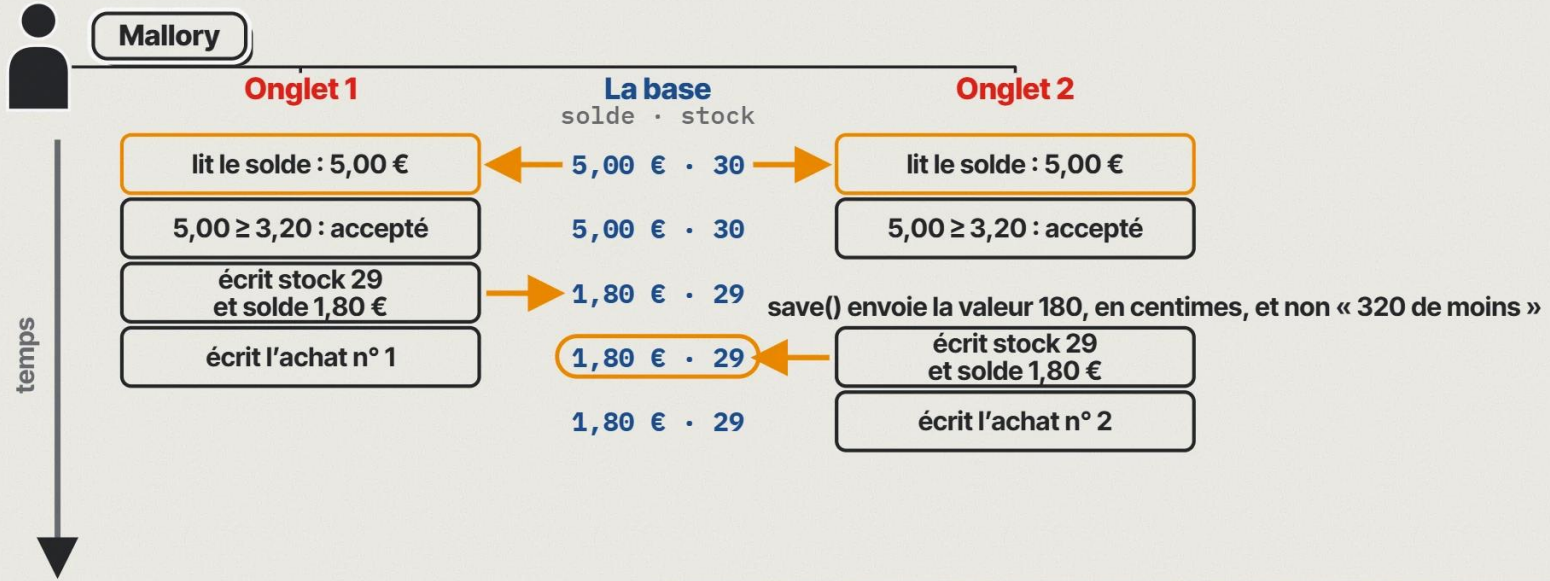


un serveur en ligne traite plusieurs requêtes en même temps, chacune dans son propre processus PHP

Suivez la ligne : lecture, contrôle, écriture. Tant que la première requête n'a pas écrit, une autre peut lire : c'est le trait orange.

? Mallory a 5,00 €, la balle coûte 3,20 €. Deux onglets lisent 5,00 € : que répond chaque contrôle ?

Les deux contrôles disent oui, et chacun écrit 1,80 € : deux balles livrées, une seule payée.



Deux balles livrées, une seule payée.

situation de compétition, race condition en anglais
le résultat dépend de l'ordre dans lequel deux requêtes avancent

Lisez de haut en bas : les deux lectures arrivent avant toute écriture. La seconde écriture pose la même valeur que la première.

? L'achat fait quatre écritures à la suite. Que reste-t-il en base si le serveur s'arrête au milieu ?

On ajoute un `throw` après l'écriture du solde : le serveur répond 500 sans écrire l'achat.

le stock — le solde — l'achat — la jauge de la créature

app/Http/Controllers/PurchaseController.php

```
$user->balance_cents -= $total;
$user->save();

throw new \RuntimeException('Panne au milieu de l\'achat.');

Purchase::create([
```

throw lève une exception, une erreur qui arrête la méthode à cet endroit

```
37        $item->update(['stock' => $item->stock - $quantity]);
38
39        $user->balance_cents -= $total;
40        $user->save();
41
42        throw new \RuntimeException('Panne au milieu de l\'achat.');
43
44        Purchase::create([
```

L'ONGLET RÉSEAU

```
POST /magasin/4/acheter
{"creature_id":"3","quantity":"1"}
Réponse : 500   500
```

RuntimeException

app\Http\Controllers\PurchaseController.php:42

Panne au milieu de l'achat.

500 POST http://tamagotchi.test/magasin/4/acheter

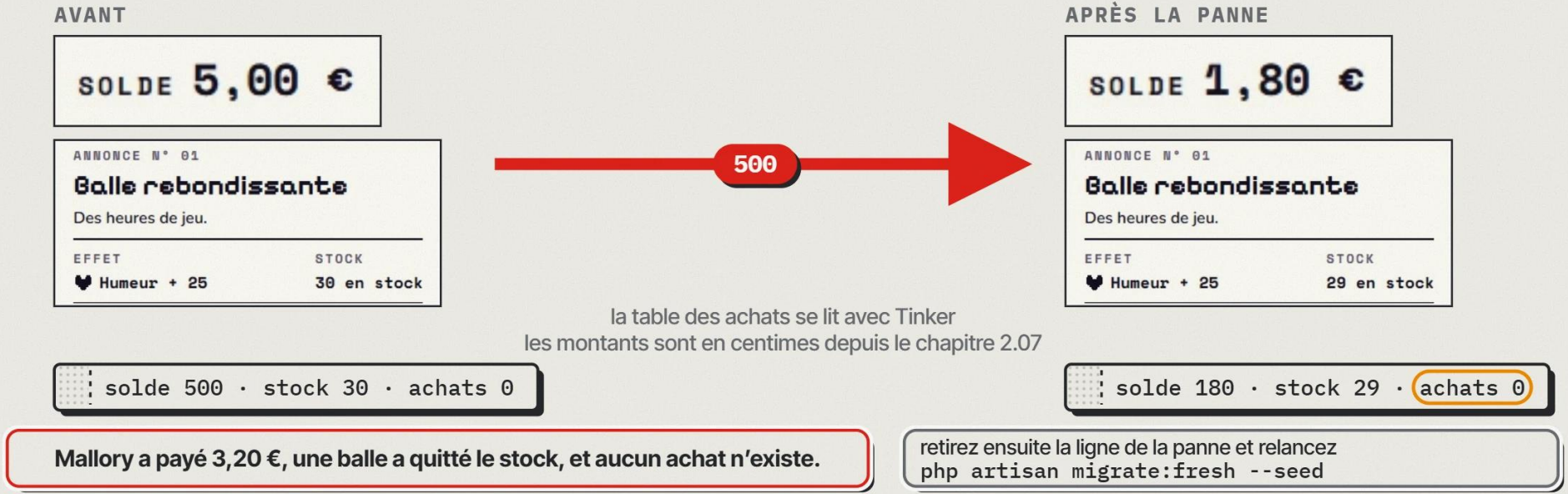
la page d'erreur de Laravel dans une fenêtre, par-dessus le magasin

À gauche, la ligne ajoutée ; à droite, ce que Mallory voit. Les lignes 37 à 40 ont déjà écrit le stock et le solde.

? Mallory recharge le magasin après la panne. Que disent le solde, le stock et la table des achats ?

Après la panne, le solde est à 1,80 € et le stock à 29, mais la table des achats est vide.

5 min



Même page avant et après la panne. Les trois valeurs relevées avec Tinker le confirment : l'argent est parti, l'achat n'existe pas.

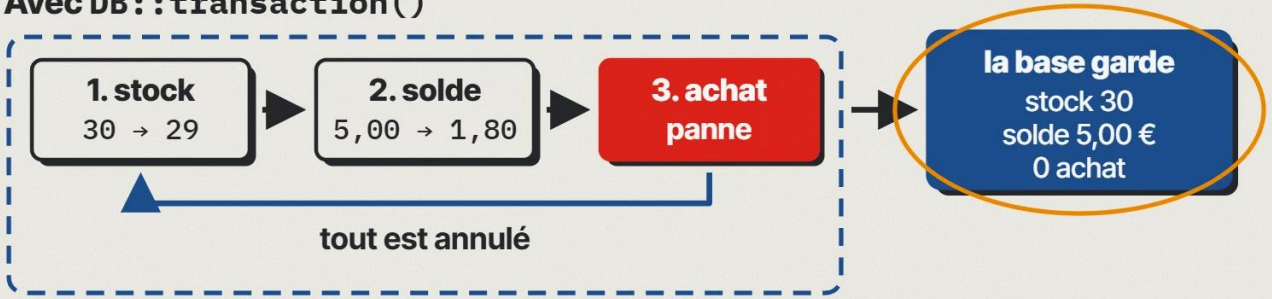
? Un virement retire l'argent d'un compte et l'ajoute sur un autre. Peut-on garder l'un sans l'autre ?

`DB::transaction()` garde toutes les écritures, ou les annule toutes si une exception en sort.

Sans transaction



Avec `DB::transaction()`



transaction : la base garde les écritures en suspens, puis les valide toutes ensemble à la fin, ou les annule toutes

atomique : d'un mot grec qui signifie « qu'on ne peut pas couper »

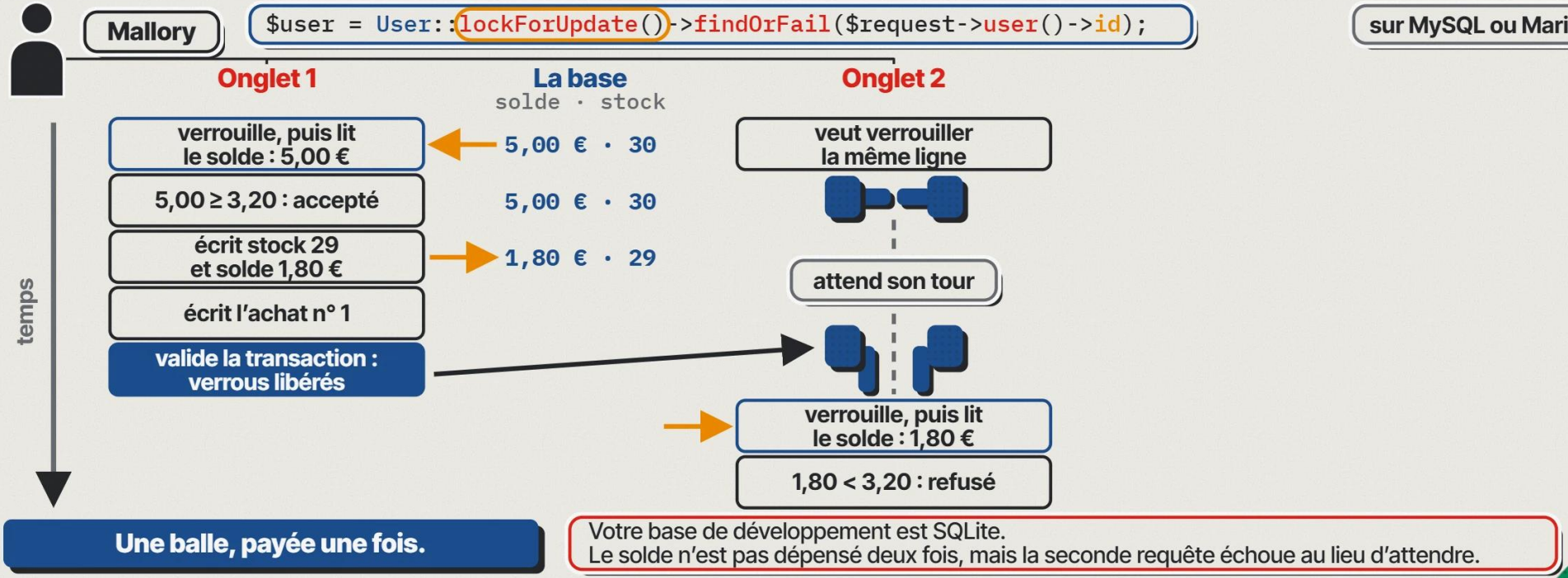
Laravel ouvre une transaction avec `DB::transaction()`, qui reçoit une fonction

elle ne règle pas les deux onglets

Même panne, deux issues. Dans le cadre en tirets, la panne annule tout : la base garde le solde et le stock de départ.

? La transaction répare la panne. Comment empêcher le second onglet de lire 5,00 € lui aussi ?

`lockForUpdate()` verrouille la ligne de la table : l'onglet 2 attend, relit 1,80 € et refuse.



Même frise qu'à la slide 3, avec le verrou : l'onglet 2 attend, lit après l'écriture de l'onglet 1, et son contrôle refuse.

? $item et $user ont été lus avant la transaction. Le verrou sert-il si on contrôle ces valeurs-là ?

Relisez sous verrou, contrôlez les valeurs relues, et refusez en levant une exception.

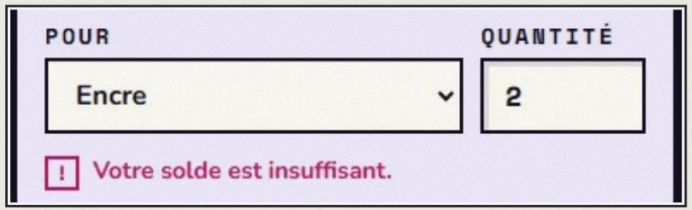
15 min

app/Http/Controllers/PurchaseController.php — extrait

```php
if ($item->stock < $quantity) {
    return back()->withErrors(['quantity' => 'Il n\'en reste pas assez en stock.']);
}

if ($user->balance_cents < $total) {
    return back()->withErrors(['quantity' => 'Votre solde est insuffisant.']);
```

1. lus avant la transaction, sans verrou
2. dans la transaction, `return back()->withErrors(…)` ne fait que sortir de la fonction



le refus passe toujours par le même message

CE QUE VOUS DEVEZ OBTENIR

- [ ] Les écritures du stock, du solde, de l'achat et de la jauge sont dans un seul `DB::transaction()`.
- [ ] Dans la transaction, l'article, le joueur et la créature sont relus avec `lockForUpdate()`.

Le refus doit donc lever une exception. `ValidationException::withMessages()` en fabrique une qui porte le message du champ.

À gauche, les contrôles tels qu'ils sont ; à droite, ce que la correction doit obtenir. Cherchez dix minutes avant d'ouvrir la correction du chapitre.

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

Le test de la panne passe au vert, et la panne rejouée ne coûte plus un centime à Mallory.

À FAIRE

1. `php artisan test --filter=Flaw09`
2. `php artisan test`
3. Remettez la ligne throw dans la fonction de la transaction, entre `$user->save();` et `Purchase::create([`, puis achetez une balle rebondissante comme Mallory. Rechargez le magasin.
4. Retirez alors la ligne de la panne.

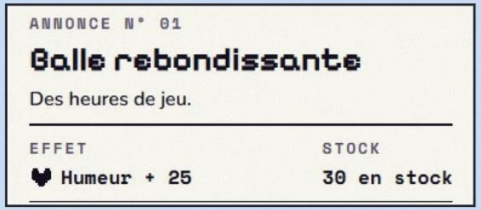
À VÉRIFIER

`Tests:  3 passed (13 assertions)`

`Tests:  1 failed, 88 passed (368 assertions)`

le dernier étant celui du chapitre 2.10

SOLDE 5,00 €



`solde 500 · stock 30 · achats 0`

Un achat se fait dans une transaction : on relit le solde et le stock sous verrou, on les contrôle, puis on écrit tout, ou rien.

Quatre étapes à gauche, trois vérifications à droite. La panne rejouée affiche une erreur, mais le solde et le stock n'ont pas bougé.