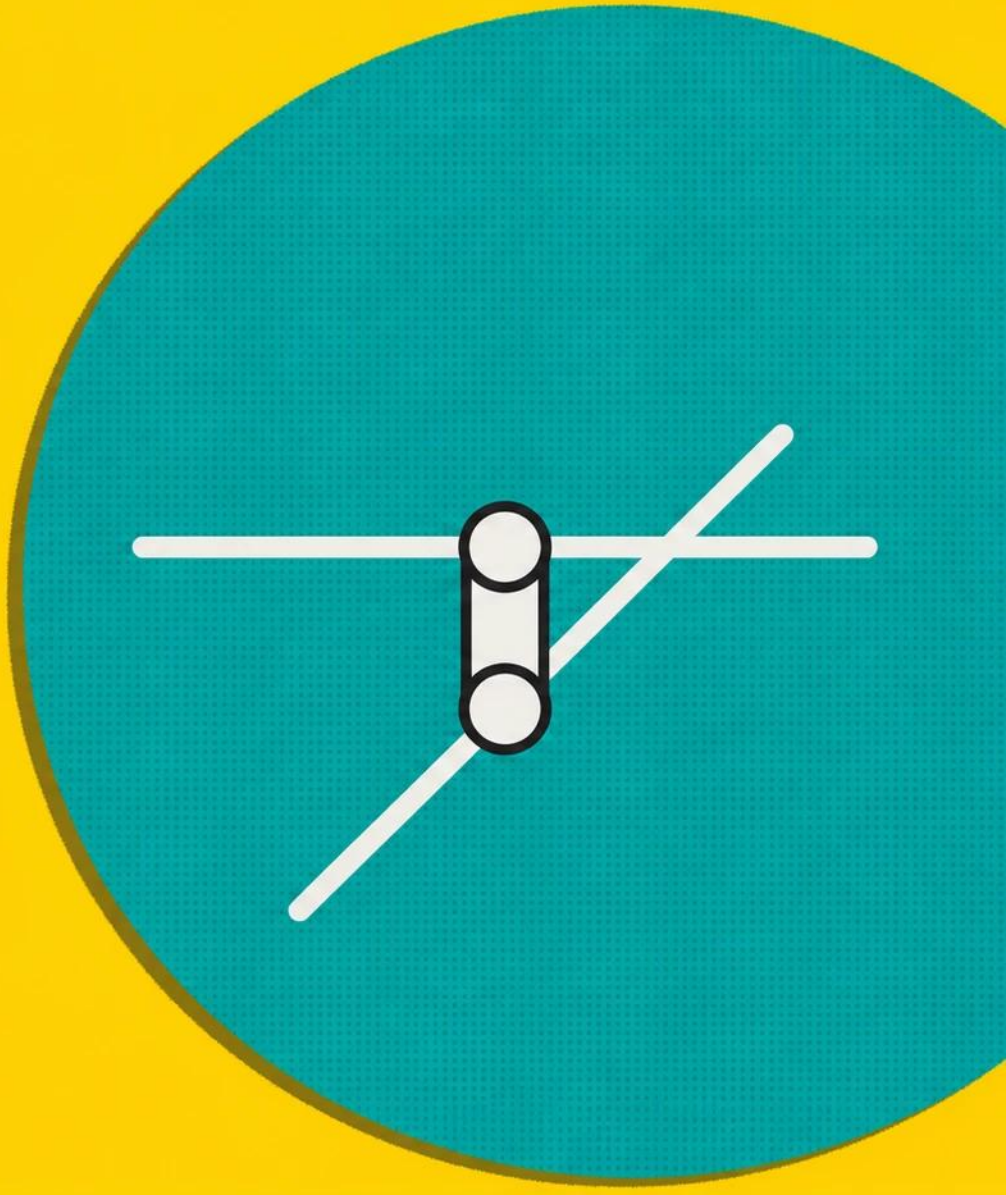
1.03

TEMPS 1 · LE PORTFOLIO · SÉANCE 1

Pourquoi Inertia

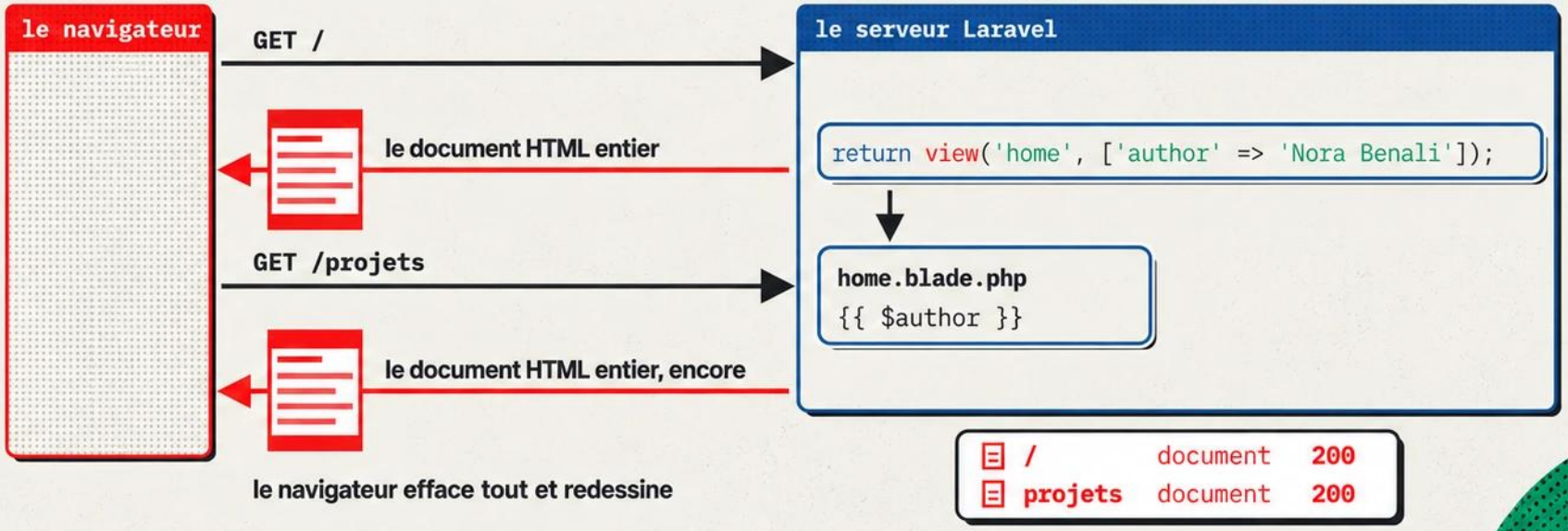
Thibault Six · IFOSUP Wavre · 2026-2027

5XDYN · Projet web dynamique · 1.03



? En Blade, que télécharge le navigateur à chaque clic ?

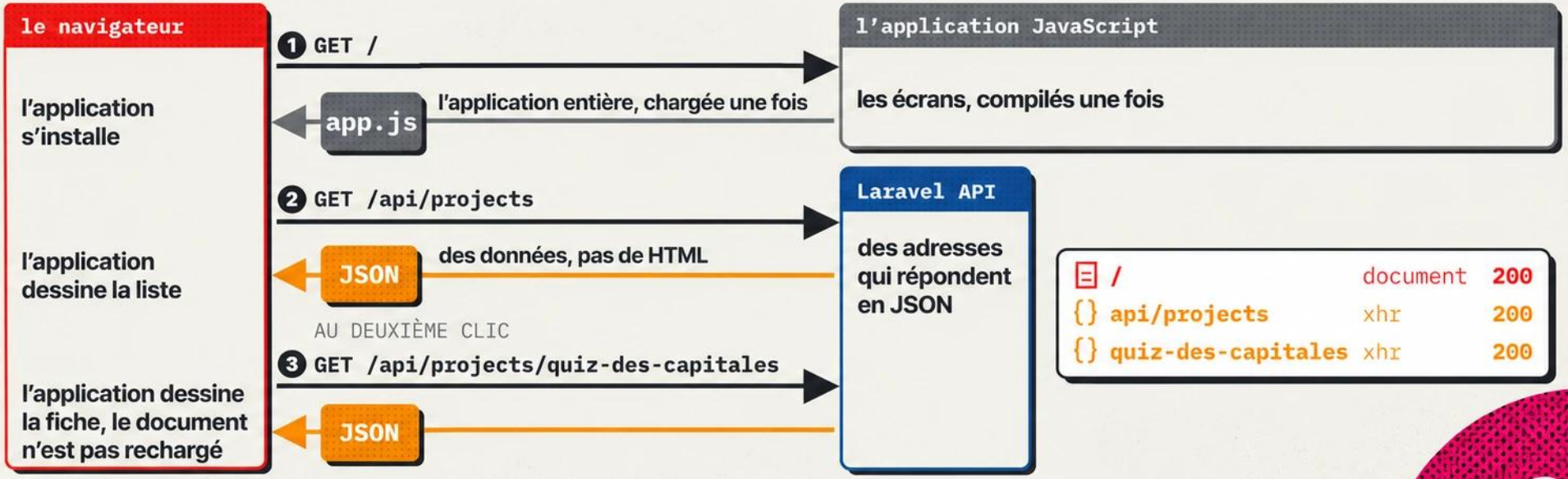
Avec Blade, chaque clic redemande au serveur le document HTML entier.



Suivez les deux allers-retours : à chaque clic, le serveur refait tout le document, même si seul le centre a changé.

? Et si le navigateur dessinait lui-même les écrans ?

Une SPA se charge une fois, puis demande ses données à une API, en JSON, à chaque clic.

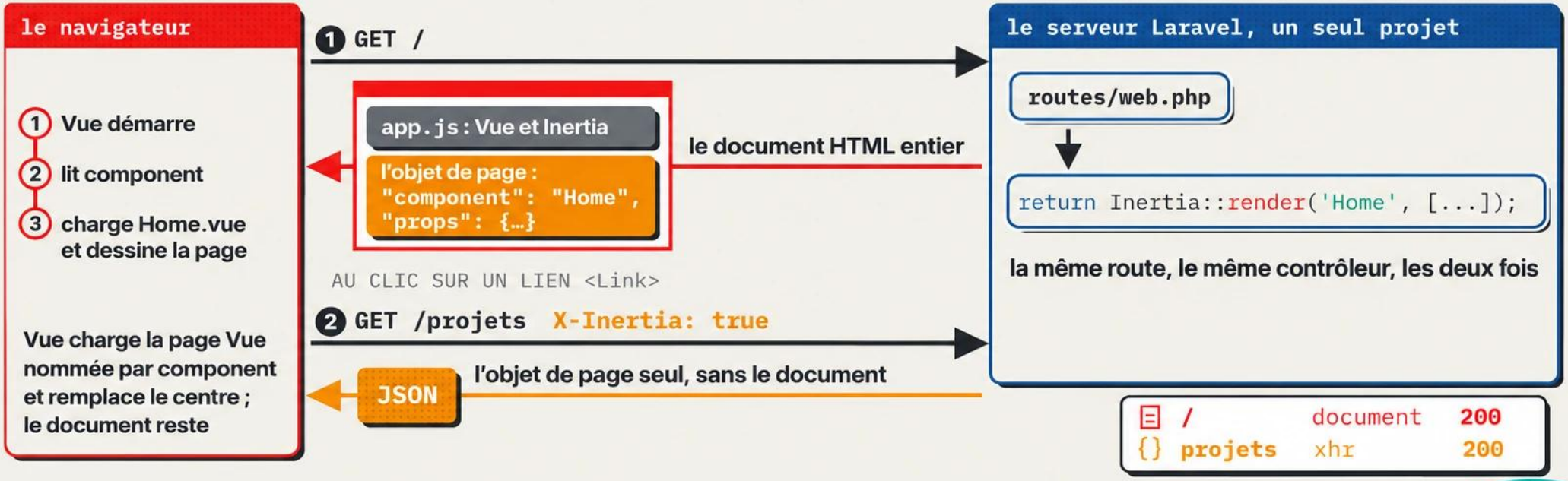


Lisez les trois temps : l'application arrive une fois, puis chaque clic ne demande que des données. Le serveur ne fabrique plus de HTML, le navigateur dessine.

1.03

Et Inertia, comment fait-il les deux depuis un seul projet ?

Inertia envoie d'abord le document entier, puis le JSON de la page, depuis le même projet.



Comparez les deux retours : un document entier la première fois, le JSON de la page ensuite. Les deux fois, c'est Vue qui charge le fichier .vue nommé par component et qui dessine.

? Pourquoi ne pas garder Blade, ou faire une SPA ?

Inertia garde le projet unique de Blade et la navigation d'une SPA.



Deux questions séparées, deux volets : ce qui *arrive* après un clic, puis combien de *projets* on maintient. Inertia prend la bonne colonne des deux.

1.03

? Dans le contrôleur, qu'est-ce qui change ?

Le contrôleur renvoie le nom d'une page Vue et ses props, à la place d'une vue Blade.

Blade le contrôleur, avant

```
return view('home', ['author' => 'Nora Benali']);
```

→ `resources/views/home.blade.php`

le même tableau : ce sont les props

Inertia le contrôleur, après

```
return Inertia::render('Home', ['author' => 'Nora Benali']);
```

→ `resources/js/pages/Home.vue`

`'Home'` désigne le fichier `Home.vue`

Comparez ligne à ligne : même route, même tableau de données ; seul le destinataire change, un fichier .vue à la place d'un .blade.php.

? Au premier chargement, où sont passées les props, et quand Home.vue est-il chargé ?

Le premier document contient l'objet de page : Inertia y lit la page Vue à charger et ses props.




```
"component": "Home"
```
la page Vue à dessiner : resources/js/pages/Home.vue

```
"props": {
  "author": "Nora Benali",
  "errors", "name", "auth", "sidebarOpen" }
```
author vient de la route ; les quatre autres sont ajoutées à toutes les pages

```
"url": "/"
```
l'adresse de la page

```
"version": "4927d70f…"
```
l'empreinte des fichiers compilés par Vite

```
"sharedProps": [
  "errors", "name", "auth", "sidebarOpen" ]
```
les props partagées, ajoutées par le kit et par Inertia

le code source de la page d'accueil, Ctrl+U, recadré

```
<script data-page="app" type="application/json">{"component":"Home","props":{"error
```

❶ le document arrive, avec **app.js** et l'objet de page → ❷ Inertia lit **component** et **props** → ❸ Vue charge **Home.vue** et dessine la page avec ses props

Lisez l'objet de haut en bas, puis la bande du bas : component dit quelle page charger, props ce qu'elle reçoit ; tout visiteur peut lire ces lignes.

1.03

? Au clic suivant, que demande Inertia au serveur ?

Aux clics suivants, la même route renvoie l'objet de page seul, en JSON.



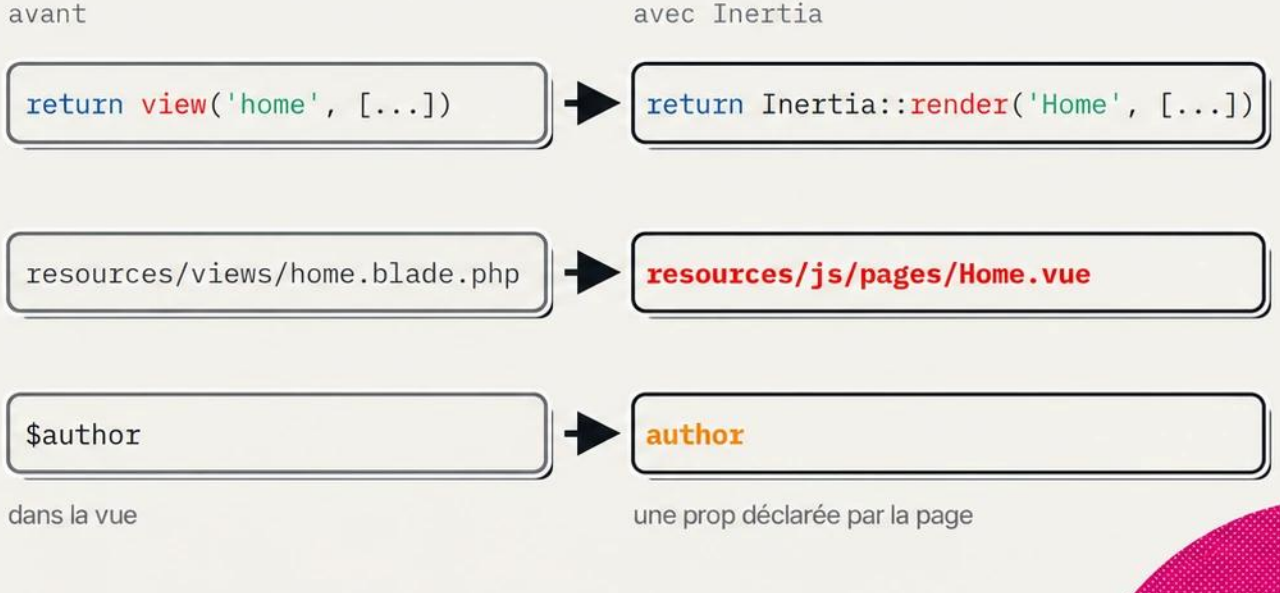
Suivez l'en-tête X-Inertia: true : il ne change pas la route, il change la réponse. Laravel exécute le même contrôleur.

1.03

? Que reste-t-il de ce que vous savez faire avec Laravel ?

Routes, contrôleurs, validation, sessions, Eloquent et tests restent ; seule la fabrication de la page change.

- ✓ app/Http/Controllers/ les sessions
- ✓ app/Http/Requests/ la validation
- ✓ app/Models/ Eloquent
- ✓ database/migrations/
- ✓ database/seeders/
- **resources/js/pages/**
  - **Home.vue** un composant Vue par page
- **resources/views/app.blade.php**
  la seule vue Blade qui reste : la vue racine
- ✓ routes/web.php
- ✓ tests/ Pest



En gris, ce que vous savez déjà faire. En couleur, les deux seuls endroits nouveaux du projet.

? S'il n'y a plus de vues Blade, qui écrit le document HTML ?

Une seule vue Blade reste, la vue racine : elle charge le fichier de la page et y pose l'objet de page.

resources/views/app.blade.php

1 `@vite(['resources/css/app.css', 'resources/js/app.ts', "resources/js/pages/{$page['component']}.vue"])`

vaut Home, donc Vite charge **resources/js/pages/Home.vue**

2 `<x-inertia::app />`

```
<script data-page="app" ...>
l'objet de page de la slide 6
<div id="app">
ici, Vue dessine la page
```

Deux lignes relient le document à Vue : la première charge le fichier de la page, la seconde y pose l'objet de page.

? À quoi ressemble une page écrite en Vue ?

Un composant Vue a deux blocs : le script déclare la prop, le template l'affiche.



Suivez **author** en orange : la clé du tableau PHP, la prop déclarée, les accolades du template, le nom à l'écran.

? Que contient le kit que laravel new installe ?

Le kit réunit sept outils ; vous lisez son TypeScript, vous écrivez vos pages en JavaScript.

Vue 3
dessine les pages dans le navigateur

Inertia 3
relie les contrôleurs aux pages

Tailwind CSS 4
met les pages en forme

shadcn-vue
des composants d'interface copiés chez vous

Fortify
la connexion côté serveur

Wayfinder
des fonctions qui donnent l'adresse de vos routes

Pest
lance les tests, 40 au départ

app.ts

```
<script setup lang="ts">
```

le kit, en TypeScript : vous lisez

Home.vue

```
<script setup>
```

vos pages, en JavaScript : vous écrivez

Deux outils servent dès le chapitre suivant : Inertia pour relier, Vue pour dessiner. Les autres arrivent chapitre par chapitre.

1.03