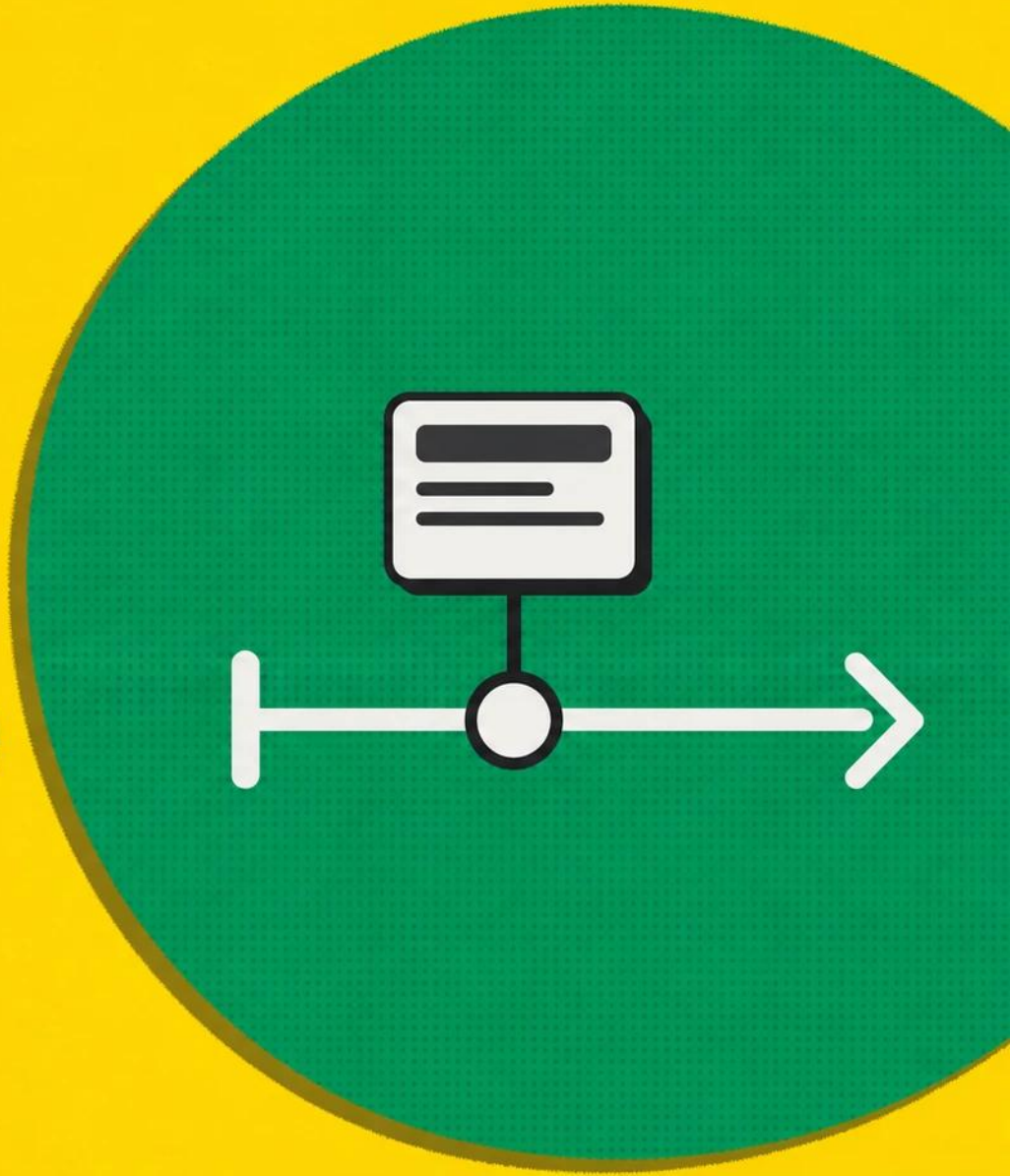
1.04

TEMPS 1 · LE PORTFOLIO · SÉANCE 1

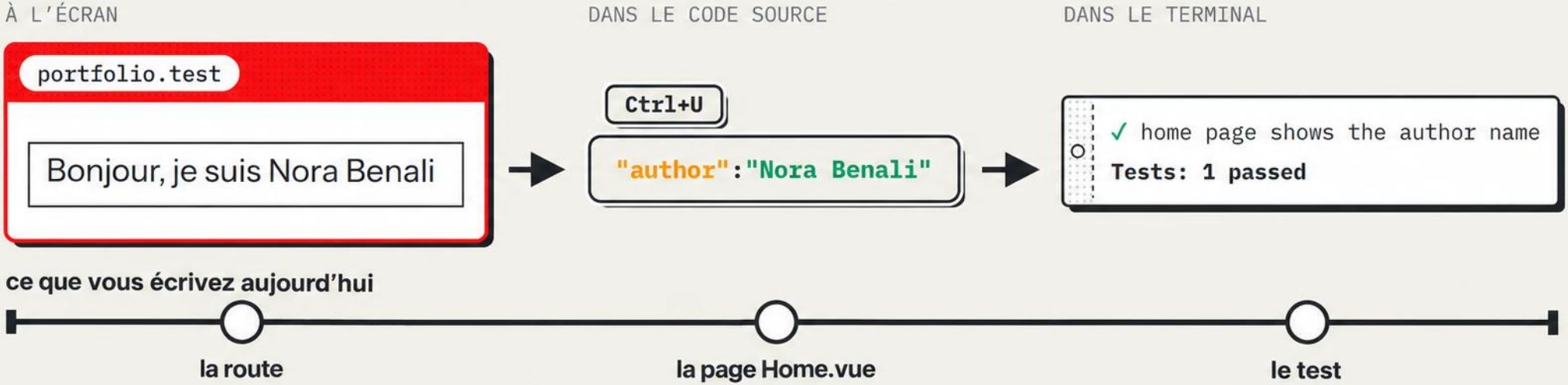
Installer le kit, afficher une page

Thibault Six · IFOSUP Wavre · 2026-2027



? Qu'est-ce qui doit marcher à la fin de ce chapitre ?

À la fin du chapitre, votre portfolio affiche votre nom, reçu du serveur, et un test le vérifie.



Trois preuves, dans cet ordre : à l'écran, dans le code source, dans le terminal. Tout le chapitre mène à ces trois lignes.

? Comment crée-t-on le projet ?

L'installeur crée le projet avec le kit Vue : une commande et sept réponses.

```
$ laravel --version
```
5.31 ou plus

→

```
$ laravel new portfolio --git
```

→  **portfolio/**

database/database.sqlite
public/build/
.git/

les sept questions de l'installeur	
Do you want to use a starter kit?	**Yes**
Which frontend stack should your starter kit use?	**Vue**
Which authentication provider do you prefer?	**Laravel's built-in authentication**
Would you like to add teams support to your application?	**No**
Which testing framework do you prefer?	**Pest**
Do you want to install Laravel Boost to improve AI assisted coding?	**No**
Would you like to run npm install and npm run build?	**Yes**

```
$ laravel new portfolio --vue --pest --npm --database=sqlite --no-boost --git
```

les mêmes réponses en une ligne

Lisez les sept réponses dans l'ordre des questions ; la commande du bas les donne toutes d'un coup.

1.04

Pourquoi faut-il deux terminaux ?

composer run dev tourne dans un second terminal toute la séance, pour que Vite prépare vos pages.

TERMINAL 1

```
$ php artisan …
$ php artisan test
$ git …
```

reste libre :
Artisan, les tests, Git

TERMINAL 2

```
$ composer run dev

  server php artisan serve
  queue php artisan queue:listen
  vite npm run dev
```

laissez-le ouvert toute la séance,
Ctrl+C en partant

prépare vos pages Vue
à chaque modification



.env

```
APP_URL=http://portfolio.test
```

à la place de localhost:8000

Le programme qui compte aujourd'hui est **vite :** il prépare chaque page Vue pour le navigateur à chaque modification. Herd ou Laragon servent toujours portfolio.test.

? Le kit contient des centaines de fichiers. Où travaille-t-on ?

Quatre endroits du projet suffisent aujourd'hui.



En couleur, les quatre endroits où vous irez aujourd'hui ; en orange, le seul fichier que vous créez.

1.04

? Comment votre nom arrive-t-il à l'écran ?

La route passe la prop author à la page Home, qui la déclare puis l'affiche.

routes/web.php

```php
use Inertia\Inertia;

Route::get('/', function () {
    return Inertia::render('Home', [
        'author' => 'Nora Benali', // votre prénom et votre nom
    ]);
})->name('home');
```

le chemin sous resources/js/pages, sans .vue

resources/js/pages/Home.vue

```vue
<script setup>
defineProps({
    author: String,
});
</script>

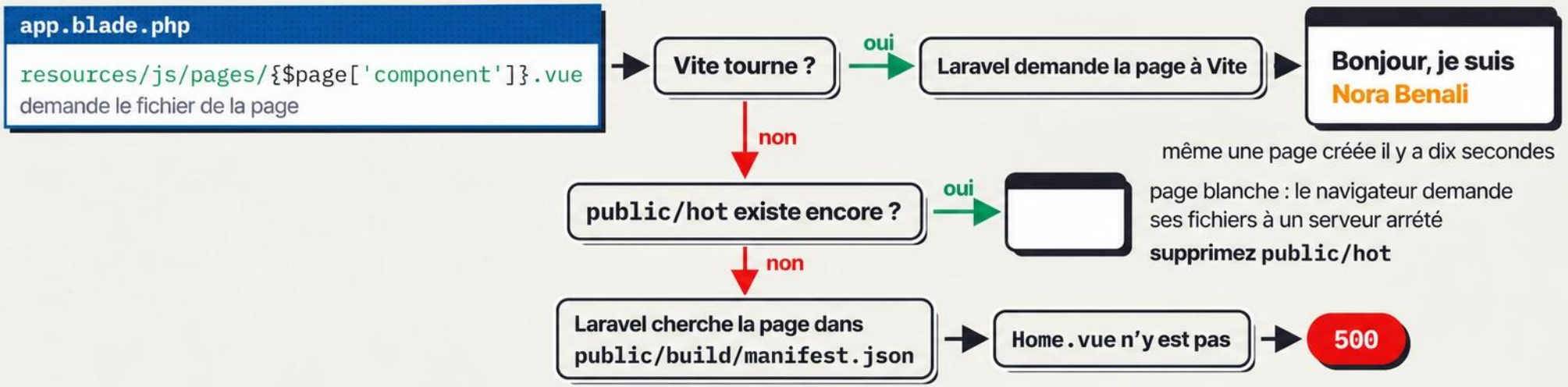
<template>
    <h1>Bonjour, je suis {{ author }}</h1>
</template>
```

Bonjour, je suis Nora Benali

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

? Que se passe-t-il quand `composer run dev` ne tourne pas ?

Sans Vite, une page créée après la dernière compilation fait une erreur 500, et un public/hot oublié fait une page blanche.



la page d'erreur 500 de Laravel, recadrée

Unable to locate file in Vite manifest: resources/js/pages/Home.vue.

la réponse est toujours la même : relancez `composer run dev`

Suivez les deux questions de gauche à droite : chaque branche a sa cause, et la correction est la même.

? La page est blanche et la console affiche une erreur. D'où vient-elle ?

Le kit donne à Home le layout de son back-office, qui lit le nom d'un utilisateur connecté : pour un visiteur, null.

resources/js/app.ts

```
layout: (name) => {
    switch (true) {
        case name === 'Welcome':
            return null;
        case name.startsWith('auth/'):
            return AuthLayout;
        case name.startsWith('settings/'):
            return [AppLayout, SettingsLayout];
        default:
            return AppLayout;
    }
},
```

Home tombe ici

AppLayout la barre latérale du back-office : Platform, Dashboard

lit auth.user.name → **auth.user vaut null** un visiteur n'est pas connecté

TypeError: Cannot read properties of null (reading 'name')

Vue arrête le dessin à la première erreur

la même page compilée par npm run build : la barre latérale apparaît



Lisez la chaîne de gauche à droite : le résolveur, le layout, la lecture d'un nom, la valeur null, l'erreur, la page blanche.

1.04

? Quelle ligne change pour que Home s'affiche seule ?

Une condition de plus dans app.ts, et Home s'affiche seule, sans barre latérale.

app.ts, avant

```
case name === 'Welcome':
    return null;
```

→ page blanche

app.ts, après

```
case name === 'Welcome' || name === 'Home':
    return null;
```

→ Bonjour, je suis Nora Benali

sans barre latérale

la seule ligne du kit que vous changez dans ce chapitre

le layout public de votre portfolio arrive au chapitre 1.08

Comparez les deux case : Home est traitée comme Welcome, aucun layout. C'est la première pause de vérification.

1.04

? Pour afficher votre nom, pourquoi ne pas appeler la prop name ?

Une prop nommée name écrase la prop partagée du kit : le nom de l'application devient votre nom.

app/Http/Middleware/HandleInertiaRequests.php

```php
return [
    ...parent::share($request),
    'name' => config('app.name'),
    'auth' => [
        'user' => $request->user(),
    ],
```

la prop partagée name : le nom de l'application, Laravel

JSON `'author'`

```
"name":"Laravel"
"author":"Nora Benali"
```

JSON `'name'`

~~"name":"Nora Benali"~~

votre prop écrase la prop partagée ;
le logo du back-office afficherait votre nom

Ne donnez jamais à une prop le nom d'une prop partagée : `name`, `auth`, `sidebarOpen`, `errors`.

Comparez les deux JSON : même clé, deux valeurs ; celle de votre route gagne, et tout composant qui lit **name** reçoit votre nom.

1.04

? Comment prouver que la page reçoit la bonne prop, sans ouvrir le navigateur ?

Un test Pest lit l'objet de page et vérifie le composant et la prop ; puis le premier commit.

tests/Feature/HomeTest.php

```php
test('home page shows the author name', function () {
    $this->get('/')
        ->assertOk()
        ->assertInertia(fn (Assert $page) => $page
            ->component('Home')
            ->where('author', 'Nora Benali')
        );
});
```

sans navigateur : le test lit la réponse

l'objet de page

```
"component":"Home"
"author":"Nora Benali"
```

```
$ php artisan test tests/Feature/HomeTest.php
PASS  Tests\Feature\HomeTest
✓ home page shows the author name
Tests:    1 passed (10 assertions)
```

puis le premier commit

« Page d'accueil avec le nom de l'auteur »
dépôt portfolio sur GitHub, vide, sans README

Les deux assertions lisent le même objet de page que vous avez vu avec Ctrl+U ; le test passe parce que composer run dev tourne.

1.04

? Qu'est-ce qui doit être vrai avant de passer au chapitre suivant ?

À vous, quinze minutes : le projet tourne, la page affiche votre nom, le test passe, le commit est sur GitHub.

15 min

- [ ] `laravel --version` affiche 5.31 ou plus.
- [ ] Le projet `portfolio` s'ouvre à l'adresse `http://portfolio.test`.
- [ ] `composer run dev` tourne dans un terminal à part.
- [ ] `http://portfolio.test` affiche « Bonjour, je suis » suivi de votre nom, sans barre latérale.
- [ ] La console des outils de développement ne montre aucune erreur.
- [ ] Le code source de la page contient `"author":` suivi de votre nom.
- [ ] La prop s'appelle `author`, pas `name`.
- [ ] `php artisan test tests/Feature/HomeTest.php` affiche `1 passed`, et `php artisan test` reste vert.
- [ ] Votre commit est fait, et le dépôt est sur GitHub.

si votre projet n'existe pas encore, faites les sections 2 à 10

Cochez dans l'ordre ; les trois dernières cases sont les trois preuves de la première slide.

1.04