1.19

TEMPS 1 · LE PORTFOLIO · SÉANCE 5

Titres, descriptions, Open Graph

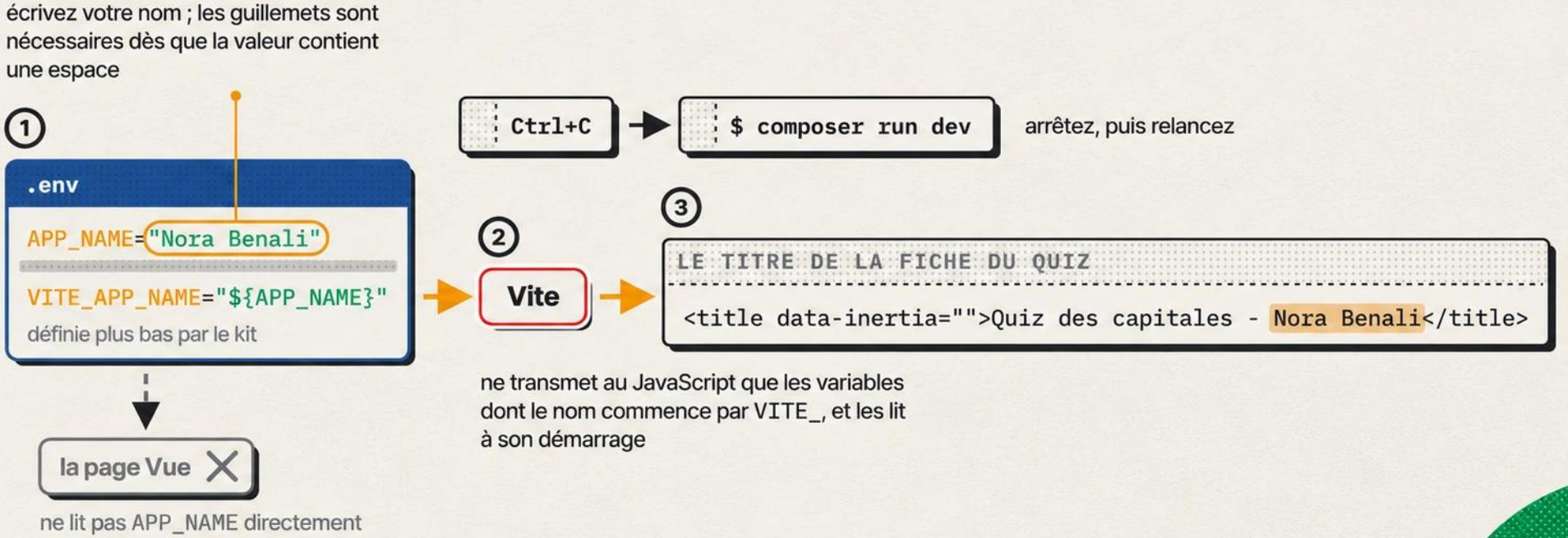
Thibault Six · IFOSUP Wavre · 2026-2027

5XDYN · Projet web dynamique · 1.19



? L'onglet dit « Projets - Laravel ». Où écrire votre nom, pourquoi relancer composer run dev ?

Votre nom s'écrit dans APP_NAME ; la page Vue le lit par VITE_APP_NAME, que Vite ne charge qu'à son démarrage.



Suivez le nom : de .env à Vite, puis jusqu'au titre de l'onglet. Vite ne relit .env qu'à son démarrage.

? Sous un résultat de recherche, une phrase résume la page. Où l'écrire dans une page Vue ?

Chaque page publique écrit sa description dans <Head>, et head-key fait remplacer la balise au lieu de l'empiler.

resources/js/pages/Home.vue

```vue
<Head title="Accueil">
    <meta
        head-key="description"
        name="description"
        content="Portfolio de Nora Benali, développeuse web Laravel et Vue.js à Wavre."
    />
</Head>
```

la balise `<meta name="description">` résume une page en une phrase ; les moteurs de recherche peuvent l'afficher sous le titre

HEAD-KEY : UNE CLÉ PAR BALISE



Inertia remplace la balise qui porte la même clé, au lieu d'en ajouter une à chaque page visitée

DANS LA FICHE

resources/js/pages/projects/Show.vue

```vue
<Head :title="project.title">
:content="project.summary"
```

le titre et la description viennent du projet

Une balise meta par page, dans <Head>. La clé fait qu'en passant d'une page à l'autre, la description est remplacée.

1.19

? Vous collez l'adresse d'une fiche dans LinkedIn. D'où viennent le titre, la phrase et l'image ?

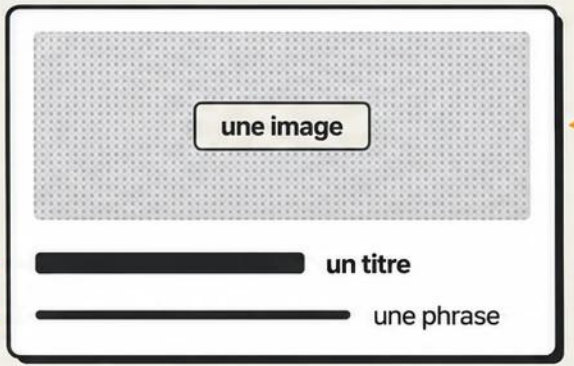
La carte d'un lien partagé vient de trois balises Open Graph dans le <head> : og:title, og:description, og:image.

`<meta property="og:…">` placées dans le <head> de la page

LES TROIS BALISES D'UNE FICHE

`og:title`	le titre du projet
`og:description`	le résumé du projet
`og:image`	l'adresse complète de l'image de couverture

LINKEDIN, WHATSAPP, SLACK, DISCORD



Chaque balise remplit une partie de la carte. Le site les déclare ; l'application qui affiche le lien les lit.

1.19

? Les balises og: sont dans le <Head> de Show.vue. Que reçoit curl, le site une fois compilé ?

Avec composer run dev, curl reçoit les balises og: ; le site compilé ne renvoie plus qu'un titre générique.

curl demande une adresse et affiche la réponse brute du serveur, sans exécuter de JavaScript : il voit ce que voit un robot

AVEC composer run dev — extrait

```
<meta property="og:title" content="Quiz des capitales" data-inertia>
<meta property="og:description" content="Un quiz de géographie en Vue.js avec score, minuteur et classement local." data-inertia>
<meta property="og:image" content="http://127.0.0.1:8000/storage/images/quiz-capitales.jpg" data-inertia>
<title data-inertia="">Quiz des capitales - Nora Benali</title>
```

✓

O /projets/quiz-des-capitales

COMPILÉ AVEC npm run build, SERVI SANS composer run dev, COMME CHEZ OVH

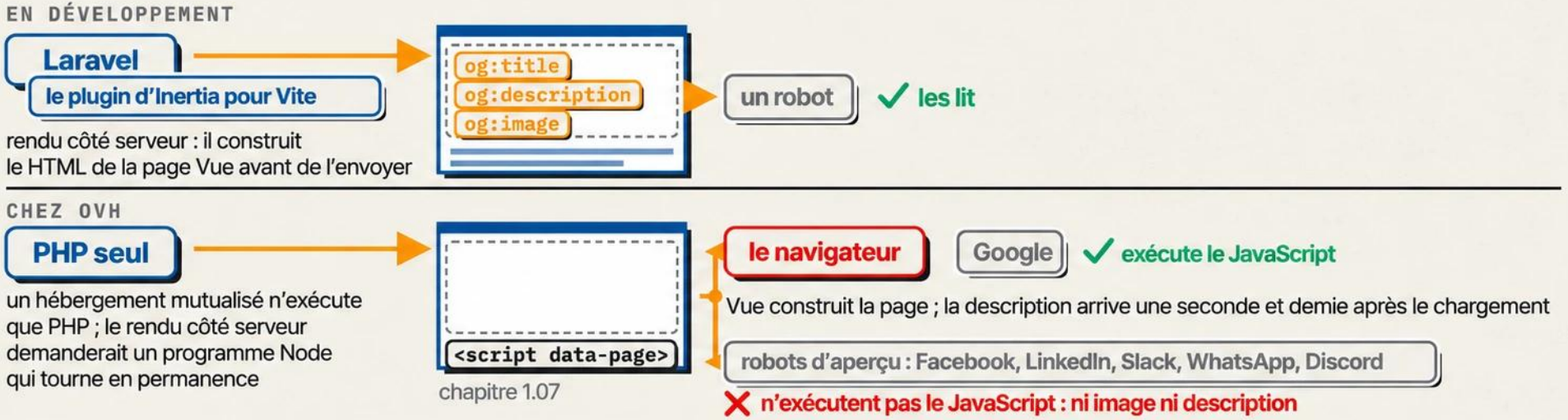
```
<title>Nora Benali</title>
```

✗ ni description, ni Open Graph, et le titre est générique

Même adresse, même code : en haut le site en développement, en bas le site compilé. Il ne reste que le titre générique.

? Pourquoi les balises de <Head> sont-elles dans le HTML en développement, et plus chez OVH ?

Chez OVH, les balises de <Head> n'arrivent qu'avec le JavaScript, que les robots d'aperçu n'exécutent pas.



Une balise qu'un robot doit lire s'écrit en PHP, dans le template Blade.

En haut, le serveur écrit les balises ; en bas, seul le JavaScript les écrit, et les robots d'aperçu ne l'exécutent pas.

? Comment écrire les balises og: et le <title> avant tout JavaScript, sans rendu côté serveur ?

Le contrôleur passe meta au template avec withViewData(), et app.blade.php écrit en PHP les balises og: et le <title>.

`Inertia::render('projects/Show', [...])` — les props partent vers la page Vue → **la page Vue**

ProjectController::show() — extrait

```
])->withViewData([
    'meta' => [
        'title' => $project->title,
        'description' => $project->summary,
        'image' => $project->cover_url,
    ],
]);
```

meta →

resources/views/app.blade.php — juste avant @fonts

```
@isset($meta)                                   ← seulement pour une page qui a passé meta : la fiche
    <meta property="og:title" content="{{ $meta['title'] }}">
    <meta property="og:description" content="{{ $meta['description'] }}">
    @if ($meta['image'])                        ← saute l'image d'un projet qui n'en a pas
        <meta property="og:image" content="{{ $meta['image'] }}">
    @endif
@endisset
```

{{ }} échappe le texte

reste côté serveur, pour le template

retirez les trois og: du <Head> de Show.vue : en développement, elles seraient écrites deux fois

dans <x-inertia::head>

```
<title>{{ isset($meta) ? $meta['title'].' - ' : '' }}{{ config('app.name', 'Laravel') }}</title>
```

le titre de l'onglet existe aussi sans JavaScript

Suivez meta : le contrôleur le passe au template, Blade écrit les balises en PHP, à chaque requête, avant tout JavaScript.

? L'onglet de votre portfolio montre encore le logo de Laravel. Quels fichiers faut-il remplacer ?

Le favicon tient dans trois fichiers de public/ : remplacez-les par les vôtres, en gardant leurs noms.

le favicon : la petite icône de l'onglet et des favoris

resources/views/app.blade.php

```
<link rel="icon" href="/favicon.ico" sizes="any">

<link rel="icon" href="/favicon.svg" type="image/svg+xml">

<link rel="apple-touch-icon" href="/apple-touch-icon.png">
```

public/

- favicon.ico
- favicon.svg
- apple-touch-icon.png

un générateur de favicon en ligne produit les trois à partir d'une image ; les initiales de votre nom dans la couleur de votre planche suffisent

Chaque lien vise un fichier de public/. Gardez les trois noms : les liens restent justes.

08

Votre navigateur affiche les balises og:. Comment savoir ce que reçoit vraiment un robot ?

Compilez avec npm run build, arrêtez composer run dev, supprimez public/hot, puis cherchez og: dans la vue source.

1. `$ npm run build`
2. `Ctrl+C` — arrêtez `composer run dev`
3. ~~`public/hot`~~ — tant qu'il est là, Laravel attend Vite : supprimez-le
4. `portfolio.test` — la vue source d'une fiche : cherchez `og:`

`Ctrl+U`

la vue source de la fiche Quiz des capitales — les trois balises, sans `data-inertia`

```
        <meta property="og:title" content="Quiz des capitales">
<meta property="og:description" content="Un quiz de géographie en Vue.js avec score, minuteur et classement local.">
                <meta property="og:image" content="http://portfolio.test/storage/images/quiz-capitales.jpg">
```



✗ l'inspecteur : il montre la page après le JavaScript

✗ le site en développement : son rendu côté serveur cache le problème

l'adresse de l'image commence par APP_URL : en ligne, ce sera celle de votre site

Les quatre stations font du site local un site comme chez OVH. La vue source montre alors ce que lit un robot.

1.19

Comment vérifier les balises og: à chaque modification, sans ouvrir la vue source à la main ?

Un test Pest reçoit le HTML sans JavaScript, comme un robot, et vérifie avec assertSee que les balises og: y sont.

`Project::factory()->create([...])` le projet Météo Wavre, créé pour le test

tests/Feature/PublicPagesTest.php — extrait

```
$this->get('/projets/meteo-wavre')
    ->assertSee('<meta property="og:title" content="Météo Wavre">', false)
    ->assertSee('<meta property="og:description" content="La météo heure par heure.">', false)
    ->assertSee('/storage/images/meteo-wavre.jpg', false);
```

`false` : cherche le texte tel quel, sans échapper ses guillemets et ses chevrons

le test reçoit le HTML que Laravel envoie : sans rendu côté serveur, sans JavaScript, exactement comme un robot

```
$ php artisan test tests/Feature/PublicPagesTest.php
```

```
  PASS  Tests\Feature\PublicPagesTest
  ✓ the home page shows three published projects
  ✓ a project page has Open Graph tags in the HTML
  ✓ the legal notice page exists

  Tests:    3 passed (21 assertions)
```

Le test demande la fiche comme un robot et cherche les trois balises dans le HTML reçu.

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

À vous, dix minutes : un titre et une description sur chaque page, des balises og: écrites par le serveur sur chaque fiche.

À FAIRE

1. `APP_NAME` porte votre nom.
2. Chaque page publique a une description avec `head-key="description"`.
3. `ProjectController::show()` passe `meta` avec `withViewData()`, et `app.blade.php` écrit les trois balises `og:`.
4. Les trois favicons du kit sont remplacés par les vôtres.

À VÉRIFIER

10 min

- [ ] Chaque onglet affiche « titre de la page - votre nom ».
- [ ] Après `npm run build`, sans `composer run dev`, la vue source d'une fiche contient `og:title`, `og:description` et `og:image`.
- [ ] Le test Open Graph est vert.

Quatre étapes à gauche, trois vérifications à droite ; la deuxième se fait sur le site compilé, sans composer run dev.

1.19