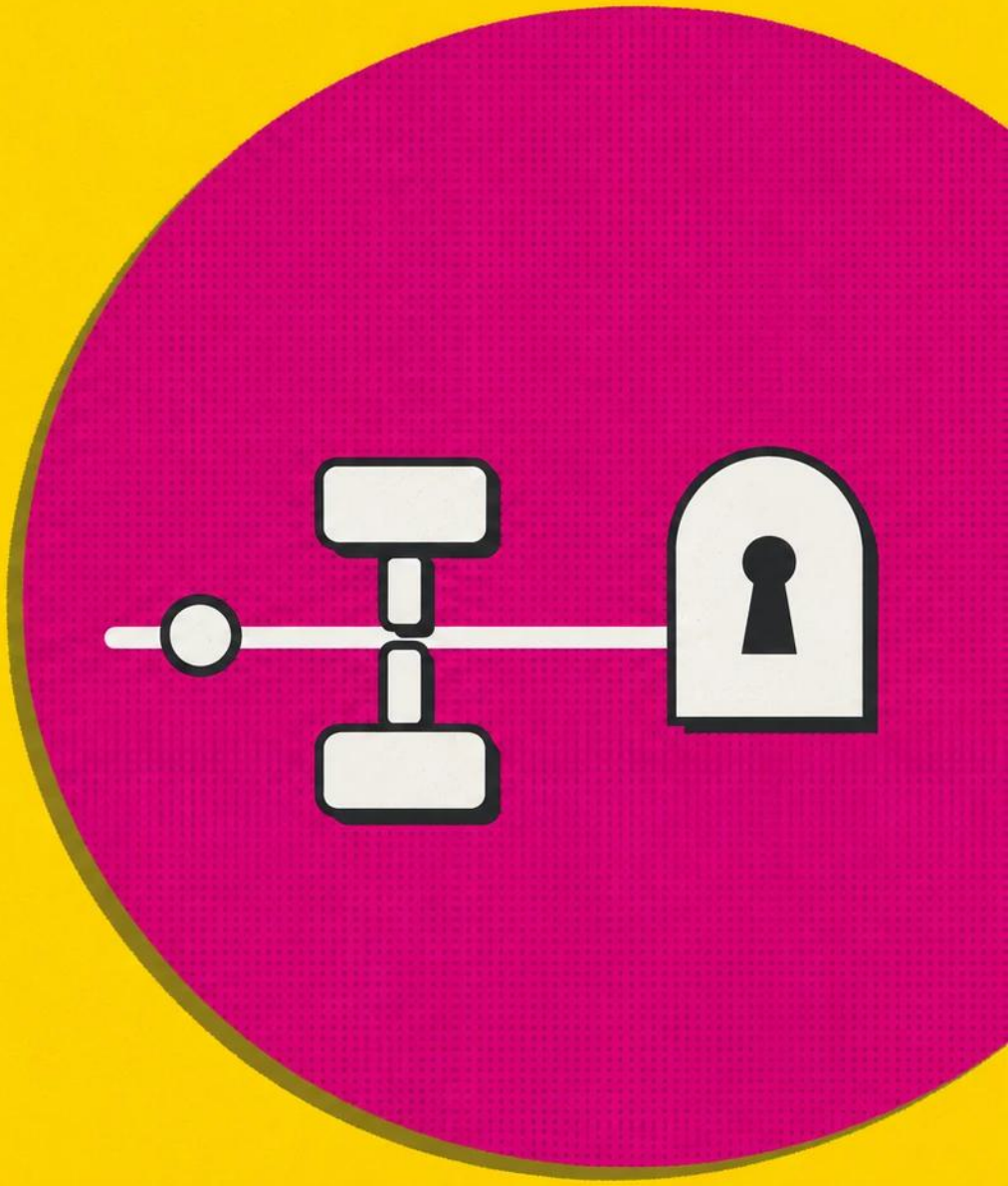
1.10

TEMPS 1 · LE PORTFOLIO · SÉANCE 3

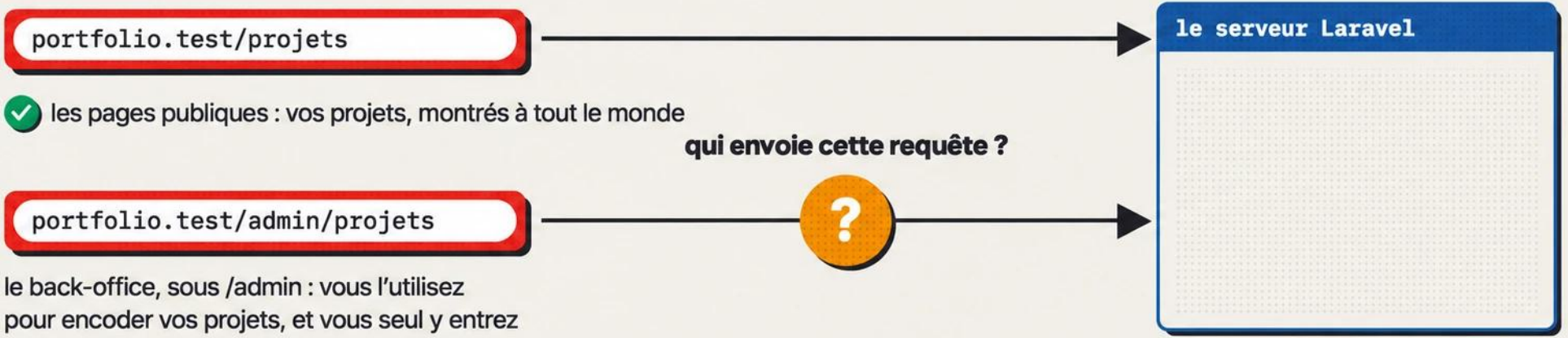
Le back-office

Thibault Six · IFOSUP Wavre · 2026-2027



? Votre back-office sera sous /admin. Comment le site saura-t-il que c'est vous ?

Le back-office n'est qu'à vous : le site doit savoir qui envoie chaque requête.

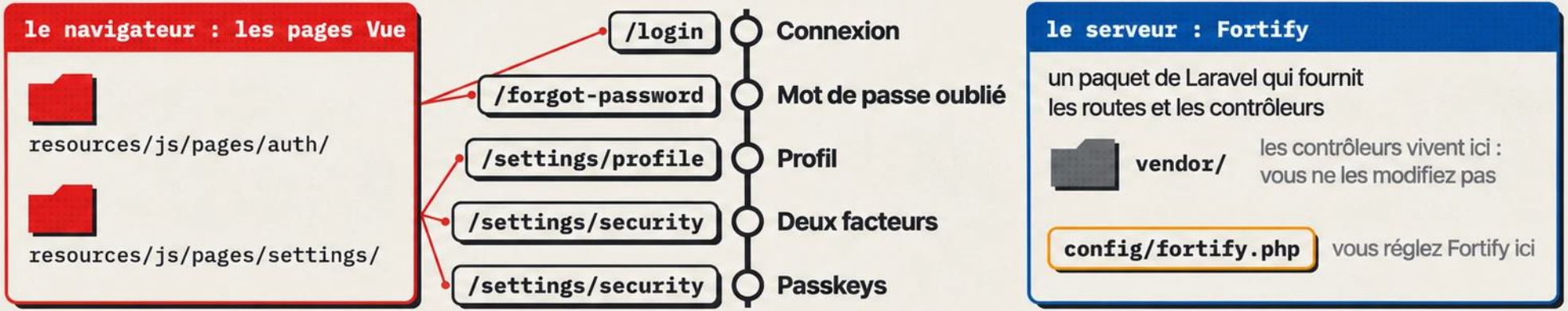


l'authentification : vérifier l'identité d'une personne, par exemple avec une adresse e-mail et un mot de passe

Pour la première adresse, peu importe qui demande. Pour la seconde, Laravel doit reconnaître la personne avant de répondre.

? Que faut-il écrire pour se connecter, retrouver un mot de passe ou gérer son compte ?

Rien : le kit a installé Fortify côté serveur et les pages Vue de connexion et de réglages côté navigateur.



Fortify ajoute ses routes à celles de routes/web.php

```
$ php artisan route:list --path=login
  GET|HEAD login .. login › Laravel\Fortify › AuthenticatedSessionController@create
  POST login .. login.store › Laravel\Fortify › AuthenticatedSessionController@store
  POST passkeys/login .. passkey.login › Laravel\Passkeys › PasskeyLoginController@store
  GET|HEAD passkeys/login/options .. passkey.login-options › Laravel\Passkeys › PasskeyLoginController@index

  Showing [4] routes
```

Lisez une ligne de gauche à droite : la page Vue, l'adresse, Fortify. Le ticket montre les routes de connexion, déclarées par Fortify et pas par vous.

? Personne ne s'inscrira sur votre portfolio. Avec quel compte vous connectez-vous ?

Le seeder crée votre utilisateur ; la fabrique lui donne le mot de passe password et une adresse déjà vérifiée.

database/seeders/DatabaseSeeder.php

```php
User::factory()->create([
    'name' => 'Nora Benali',
    'email' => 'admin@example.com',
]);

$this->call(ProjectSeeder::class);
```

à la place de Test User, test@example.com : votre nom, une adresse inventée suffit en local

`$ php artisan migrate:fresh --seed`

1 2 ✓ votre utilisateur seedé passe directement 3 4

UserFactory

```php
'password' => static::$password ??= Hash::make('password'),
```

le mot de passe password, chiffré avant d'être enregistré
la fabrique remplit aussi email_verified_at

portfolio.test/login

admin@example.com password

MustVerifyEmail → une adresse non vérifiée : renvoyée vers une page qui demande de cliquer sur un lien reçu par e-mail

Suivez la ligne : votre nom dans le seeder, le mot de passe dans la fabrique, la base refaite, la connexion. Avant la mise en ligne, vous choisirez un vrai mot de passe.

? Le kit laisse n'importe qui créer un compte à /register. Qu'est-ce que ça ouvre sur votre portfolio ?

Un visiteur inscrit entrerait dans votre back-office : une ligne en commentaire dans config/fortify.php retire /register.

config/fortify.php

```
'features' => [
    // Features::registration(),
    Features::resetPasswords(),
    Features::emailVerification(),
    // … twoFactorAuthentication et passkeys inchangés
],
```

le tableau features liste les fonctions actives de Fortify

AVANT



un visiteur inscrit entrerait dans votre back-office

APRÈS

```
$ php artisan route:list --path=register
ERROR Your application doesn't have any routes matching the given criteria.
```

Fortify ne déclare plus les routes register et register.store

portfolio.test/register → 404 votre page ErrorPage, dans votre layout public

Une ligne devient un commentaire, la route disparaît, et l'adresse répond par votre page 404 du chapitre 1.08.

Pourquoi npm run build casse-t-il après une seule ligne en commentaire ?

Sans la route, Wayfinder ne génère plus register : les trois fichiers qui l'importent font échouer npm run build.

une ligne en commentaire (1)

```
config/fortify.php
// Features::registration(),
```

la route disparaît (2)

~~/register~~

Wayfinder ne la génère plus (3)

~~register()~~ Ctrl+C, puis relancez composer run dev : Vite surveille routes/, pas config/

(4) **trois fichiers l'importent encore**

```
npm run build
[UNLOADABLE_DEPENDENCY] Could not load ../../../../../resources/js/routes/register
    ┌─[ resources/js/pages/auth/Register.vue?vue&type=script&setup=true&lang.ts:18:23 ]
 18 │ import { store } from "@/routes/register";
    │                        ──────────────────
    │                        └─ No such file or directory (os error 2)
```

✓ **built in** suivie de sa durée

resources/js/pages/auth/Register.vue	`import { store } from "@/routes/register";`	supprimée : plus aucune route ne l'affiche
resources/js/pages/Welcome.vue	`import { register } from "@/routes";`	supprimée : votre page Home l'a remplacée au chapitre 1.04
resources/js/pages/auth/Login.vue	`import { register } from '@/routes';`	la page reste : retirez l'import et le lien « Sign up »

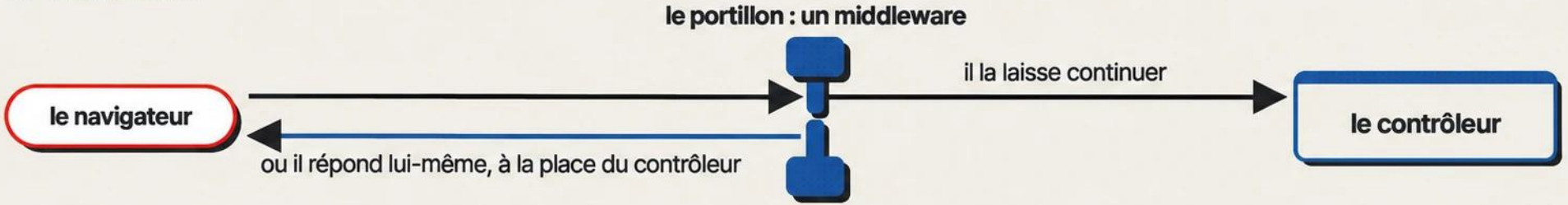
~~`<TextLink :href="register()" :tabindex="5">Sign up</TextLink>`~~

Lisez la chaîne de gauche à droite, puis l'erreur : quel fichier, quelle ligne, quel import. Deux pages se suppriment, la troisième se corrige.

? Qu'est-ce qui empêchera un visiteur qui tape portfolio.test/admin/projets d'arriver dans votre back-office ?

Un middleware contrôle la requête avant le contrôleur ; auth ne laisse passer qu'un utilisateur connecté.

UN MIDDLEWARE



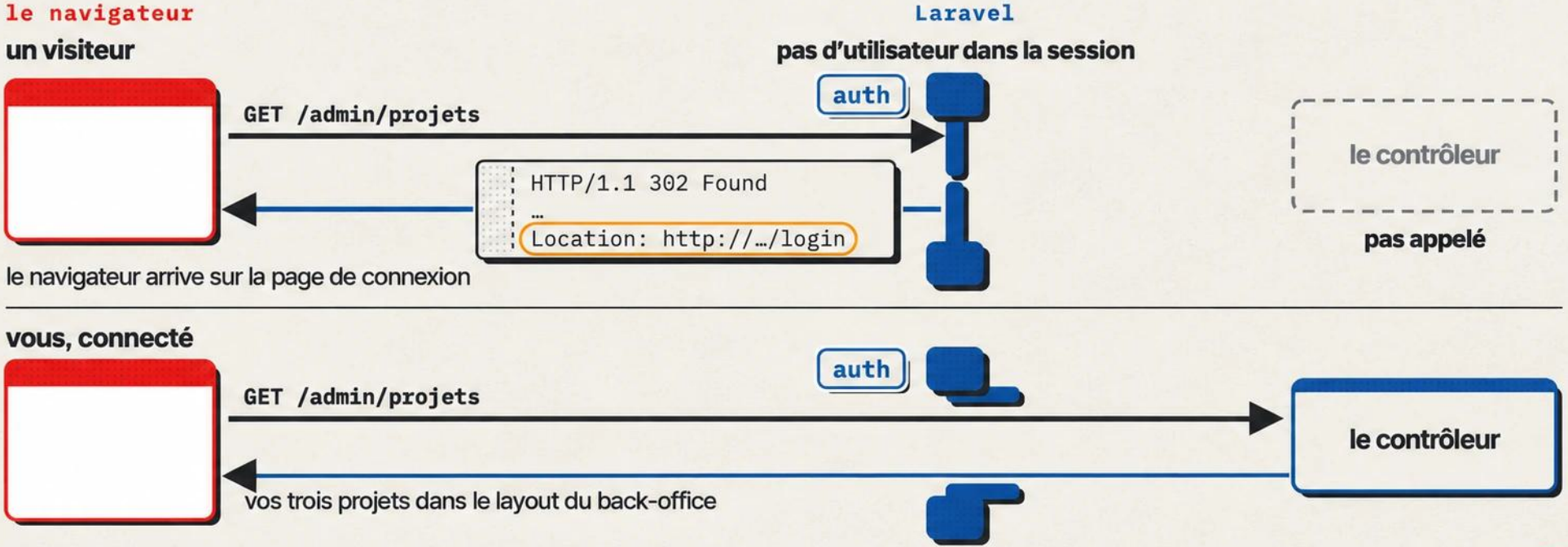
LE MIDDLEWARE AUTH



Le portillon se place sur le **chemin** de chaque route du back-office : aucune requête n'atteint le contrôleur sans passer par lui.

? Un visiteur ouvre /admin/projets. Le contrôleur est-il appelé ?

Sans utilisateur dans la session, auth répond lui-même 302 vers /login : le contrôleur n'est pas appelé.



Même adresse, deux trajets. En haut, le trajet s'arrête au portillon et revient avec une 302 ; en bas, il va jusqu'au contrôleur.

? Comment protéger toutes les pages du back-office sans répéter auth sur chaque route ?

Le groupe donne à chacune de ses routes ses middlewares, un début d'adresse et un début de nom.

routes/web.php

```php
Route::middleware(['auth', 'verified'])->prefix('admin')->name('admin.')->group(function () {
    Route::resource('projets', Admin\ProjectController::class)
        ->except('show')
        ->names('projects')
        ->parameters(['projets' => 'project']);
});
```

1. fait passer chaque route du groupe par les deux middlewares
2. ajoute `admin/` devant chaque adresse du groupe
3. ajoute `admin.` devant chaque nom de route
4. contient les routes concernées

LA LISTE DES PROJETS

adresse **admin/projets**

nom **admin.projects.index**

`use App\Http\Controllers\Admin;` en tête du fichier

Lisez la première ligne dans l'ordre des numéros. Toute route ajoutée dans le groupe hérite des trois réglages.

? L'adresse dit projets, le nom dit projects. D'où viennent {project}, nouveau et modifier ?

names() garde des noms anglais, parameters() nomme le paramètre {project}, resourceVerbs() traduit create et edit.

```
->except('show')
```
retire la fiche : la fiche publique existe déjà

```
$ php artisan make:controller Admin/ProjectController
```

```
->parameters(['projets' => 'project'])
```
nomme le paramètre {project}, comme l'argument `Project $project`

```
->names('projects')
```
garde un nom anglais aux routes, alors que l'adresse est en français

sans cette ligne : {projet}, et la liaison de modèle ne se ferait pas

```
$ php artisan route:list --path=admin
  GET|HEAD admin/projets .. admin.projects.index › Admin\ProjectController@index
  POST admin/projets .. admin.projects.store › Admin\ProjectController@store
  GET|HEAD admin/projets/nouveau .. admin.projects.create › Admin\ProjectController@create
  PUT|PATCH admin/projets/{project} .. admin.projects.update › Admin\ProjectController@update
  DELETE admin/projets/{project} .. admin.projects.destroy › Admin\ProjectController@destroy
  GET|HEAD admin/projets/{project}/modifier .. admin.projects.edit › Admin\ProjectController@edit
```

aujourd'hui, le contrôleur n'a que index()

app/Providers/AppServiceProvider.php · boot()

```
Route::resourceVerbs(['create' => 'nouveau', 'edit' => 'modifier']);
```

traduit `create` et `edit` pour toutes les ressources du site

Suivez chaque flèche d'une ligne de code à sa trace dans route:list. Six routes, pas de show.

? Après la connexion, qu'est-ce qui vous amène sur la liste de vos projets, dans le layout du back-office ?

Fortify vous envoie sur /admin/projets ; index() lit trois colonnes, et le résolveur donne AppLayout à la page.

config/fortify.php

```
'home' => '/admin/projets',
```

avant : /dashboard,
le tableau de bord vide du kit

app/Http/Controllers/Admin/ProjectController.php

```
public function index(): Response
{
    return Inertia::render('admin/projects/Index', [
        'projects' => Project::orderBy('position')->get(['id', 'title', 'slug']),
    ]);
}
```

trois colonnes : la règle des props explicites vaut aussi derrière une connexion

1 2 3 4

resources/js/app.ts

```
case name === 'Dashboard' || name.startsWith('admin/'):
    return AppLayout;
```

aucune ligne de plus : depuis le chapitre 1.07

resources/js/components/AppSidebar.vue

```
title: 'Projets',
href: adminProjectsIndex(),
```



cinq tests du kit attendent encore l'ancienne redirection : remplacez `route('dashboard', absolute: false)` par `route('admin.projects.index', absolute: false)`

Suivez la ligne : la clé home, le contrôleur, le résolveur, l'écran. Le lien Projets de la barre latérale ramène à la même liste.

1.10

? Votre nom s'affiche en bas de la barre latérale. D'où vient-il, et que reçoit le navigateur avec lui ?

auth.user arrive sur chaque page : l'utilisateur connecté sans les colonnes de #[Hidden], ou null pour un visiteur.

```
#[Hidden(['password', 'two_factor_secret', 'two_factor_recovery_codes', 'remember_token'])]
```

app/Http/Middleware/HandleInertiaRequests.php · share()

```
'auth' => [
    'user' => $request->user(),
],
```

extrait

`$request->user()` : l'utilisateur connecté, ou `null` pour un visiteur

le modèle User

- id
- name
- email
- email_verified_at
- created_at
- updated_at
- two_factor_confirmed_at

restent sur le serveur :

- password
- two_factor_secret
- two_factor_recovery_codes
- remember_token

le navigateur, connecté

```
"auth": {
    "user": {
        "id": 1,
        "name": "Nora Benali",
        "email": "admin@example.com",
        "email_verified_at": "2026-10-25T12:35:36.000000Z",
        "created_at": "2026-10-25T12:35:36.000000Z",
        "updated_at": "2026-10-25T12:35:36.000000Z",
        "two_factor_confirmed_at": null
    }
}
```

pour un visiteur

```
"auth":{"user":null}
```

votre nom en bas de la barre latérale

NB Nora Benali

dans une page, `usePage()` : `{{ page.props.auth.user.name }}` ; sur une page publique, testez d'abord `v-if="page.props.auth.user"`

Le kit envoie le modèle entier, à l'inverse de la règle du chapitre 1.07 ; seules les colonnes de #[Hidden] restent sur le serveur. Ce sont vos données, envoyées à vous seul.

1.10

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

À vous, dix minutes : un back-office fermé aux visiteurs, où vous seul entrez, et personne ne peut plus s'inscrire.

À FAIRE

1. Votre nom et votre adresse dans `DatabaseSeeder`, puis `php artisan migrate:fresh --seed`.
2. L'inscription fermée dans `config/fortify.php`, `Register.vue` et `Welcome.vue` supprimées, le lien « Sign up » retiré de `Login.vue`.
3. Le groupe `/admin` dans `routes/web.php`, `Route::resourceVerbs()` dans `AppServiceProvider`, et `Admin\ProjectController` avec `index()`.
4. La page `admin/projects/Index.vue`, le lien « Projets » dans la barre latérale, et home réglé sur `/admin/projets`.

À VÉRIFIER

10 min

- [ ] Sans être connecté, `portfolio.test/admin/projets` mène à la page de connexion.
- [ ] `portfolio.test/register` affiche votre page 404, et la page de connexion n'a plus de lien « Sign up ».
- [ ] Après la connexion, vous arrivez sur `/admin/projets`, avec vos trois projets, et votre nom en bas de la barre latérale.
- [ ] `npm run build` se termine sans erreur.

Quatre étapes à gauche, quatre vérifications à droite ; la première se vérifie en navigation privée.