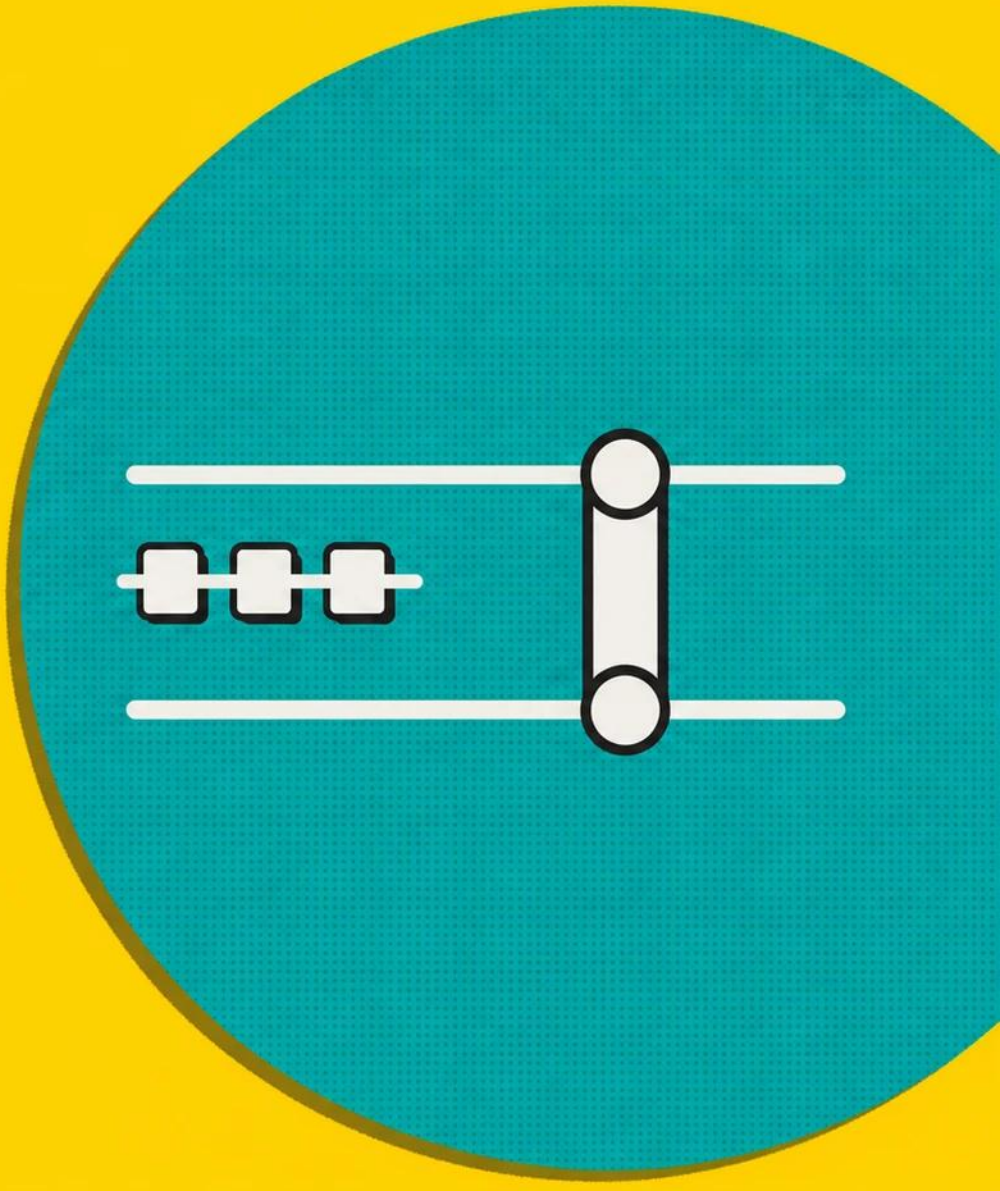
1.06

TEMPS 1 · LE PORTFOLIO · SÉANCE 2

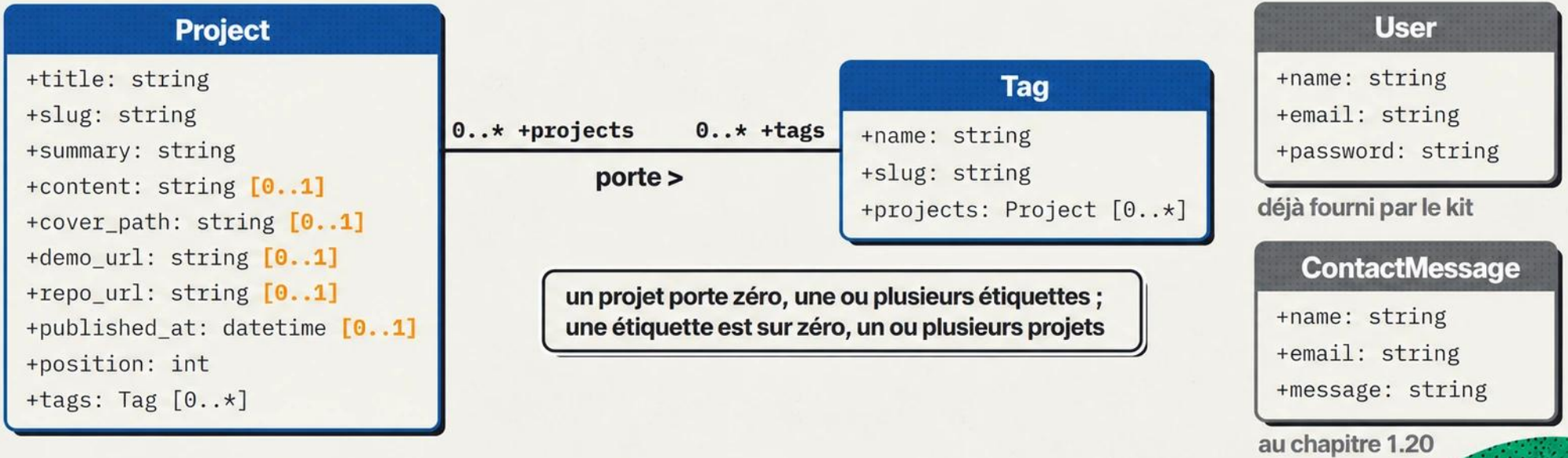
Vos projets dans la base

Thibault Six · IFOSUP Wavre · 2026-2027



? Votre diagramme et celui du cours : où sont les écarts ?

Quatre classes, une relation : Project porte zéro, une ou plusieurs Tag ; chaque écart avec le vôtre a une raison.



Lisez le trait au milieu : deux multiplicités, deux rôles, un verbe et son sens. Les deux classes grises ne sont reliées à rien, et c'est voulu : un seul auteur, des messages sans compte.

? Que devient chaque trait du diagramme dans la base ?

Une classe donne une table, un attribut [0..1] une colonne nullable, et 0..* aux deux bouts une table pivot.

DANS LE DIAGRAMME	DANS LA BASE
la classe `Project`	la table `projects`, au pluriel anglais
`+title: string`	`$table->string('title')`
un attribut marqué `[0..1]`	une colonne `->nullable()`, qui accepte l'absence de valeur
`0..*` aux deux bouts du trait `Project – Tag`	la table pivot `project_tag`
rien	`id, created_at, updated_at` : ajoutées par Laravel

`users` existe déjà : le kit l'a créée, ne la recréez pas

`ContactMessage` attend le formulaire du chapitre 1.20

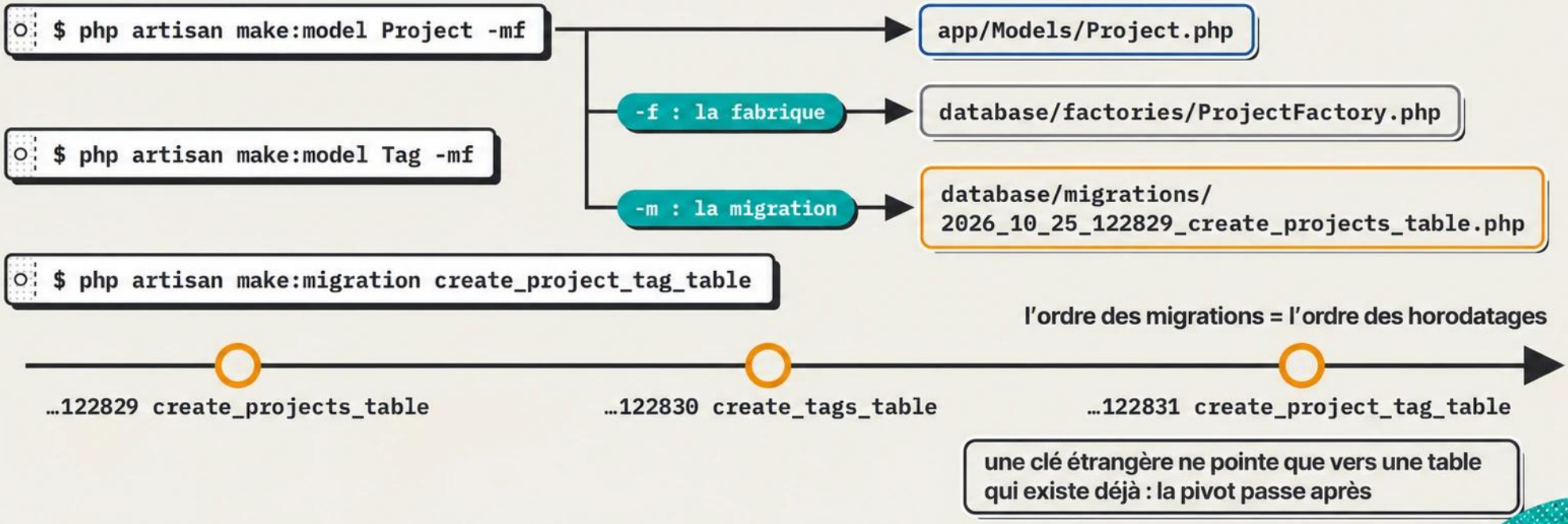
Suivez chaque flèche : c'est le chemin que vous connaissez depuis le festival. Deux classes du diagramme ne demandent rien aujourd'hui.

1.06

03

? Pourquoi créer la migration de la pivot en dernier ?

Une commande crée le modèle, sa migration et sa fabrique ; l'horodatage du fichier fixe l'ordre des migrations.



Lisez les trois tickets, puis la ligne du bas : créée en dernier, la pivot porte l'horodatage le plus récent, et passe donc après projects et tags.

? Comment chaque attribut du diagramme s'écrit-il dans la migration ?

La migration reprend un attribut par ligne ; nullable, unique, 500, longText et default(0) disent le reste.

database/migrations/…_create_projects_table.php

```php
Schema::create('projects', function (Blueprint $table) {
    $table->id();
    $table->string('title');
    $table->string('slug')->unique();
    $table->string('summary', 500);
    $table->longText('content')->nullable();
    $table->string('cover_path')->nullable();
    $table->string('demo_url')->nullable();
    $table->string('repo_url')->nullable();
    $table->timestamp('published_at')->nullable();
    $table->unsignedInteger('position')->default(0);
    $table->timestamps();
});
```

->unique() : le slug est l'adresse du projet, deux projets ne la partagent pas

500 au lieu de 255 : une marge pour un résumé d'une phrase

longText : du HTML, qui peut devenir long

[0..1] dans le diagramme : ce qui peut manquer

->default(0) : un projet créé sans position reçoit 0

Lisez le code ligne à ligne contre le diagramme de la slide 1, puis les quatre détails à droite. content est nullable : au chapitre 1.12, vous enregistrerez un projet avant d'avoir écrit son texte.

1.06

? À quoi ressemble la table qui relie un projet et ses étiquettes ?

La pivot project_tag a deux clés étrangères et une clé primaire composée ; elle n'a ni id, ni timestamps, ni modèle.

…_create_project_tag_table.php

```php
Schema::create('project_tag', function (Blueprint $table) {
    $table->foreignId('project_id')->constrained()->cascadeOnDelete();
    $table->foreignId('tag_id')->constrained()->cascadeOnDelete();
    $table->primary(['project_id', 'tag_id']);
});
```

projects

id	title	slug…
1		
2		

constrained() : la clé étrangère vers projects.id

project_tag

project_id	tag_id
2	1
2	3

vers tags.id

tags

id	name
1	Vue.js
2	Laravel
3	JavaScript

project avant tag : les deux modèles au singulier, dans l'ordre alphabétique

cascadeOnDelete() : le projet disparaît, ses lignes de pivot aussi ; les étiquettes restent

primary([…]) : la même paire ne s'enregistre pas deux fois

Suivez les deux flèches : chaque ligne de la pivot relie un projet à une étiquette, comme artist_concert au festival. Aucun modèle pour elle : les deux modèles la manipulent à travers leur relation.

1.06

? Que doit savoir le modèle pour remplir la table et suivre la relation ?

Fillable, casts() et belongsToMany() : ce qu'on remplit, ce qu'on convertit, ce qui relie les deux modèles.

app/Models/Project.php

```php
#[Fillable(['title', 'slug', 'summary',
    'content', 'cover_path', 'demo_url',
    'repo_url', 'published_at', 'position'])]
class Project extends Model
{
    use HasFactory;

    protected function casts(): array
    {
        return ['published_at' => 'datetime'];
    }

    public function tags(): BelongsToMany
    {
        return $this->belongsToMany(Tag::class);
    }
}
```

imports et commentaires omis

les deux rôles du diagramme : +tags, +projects

app/Models/Tag.php

```php
#[Fillable(['name', 'slug'])]
class Tag extends Model
{
    use HasFactory;

    public function projects(): BelongsToMany
    {
        return $this->belongsToMany(Project::class);
    }
}
```

#[Fillable] : les colonnes que `Project::create([...])` **a le droit de remplir**

casts() : published_at **devient un objet date, pas une chaîne**

belongsToMany(Tag::class) sans autre argument : Laravel déduit project_tag, project_id **et** tag_id

Comparez les deux panneaux : même forme, deux rôles. Vous avez suivi les conventions, donc belongsToMany n'a besoin de rien d'autre.

? À quoi sert une fabrique si le seeder enregistre vos vrais projets ?

La fabrique invente des projets jetables pour les tests ; make() les crée en mémoire, create() les enregistre.

database/factories/ProjectFactory.php

```php
$title = fake()->unique()->sentence(3);

return [
    'title' => $title,
    'slug' => Str::slug($title),
    'summary' => fake()->sentence(12),
    'published_at' => now(),
];
```

extrait : content, demo_url, repo_url et position suivent le même modèle

draft()

```php
return $this->state(['published_at' => null]);
```

un état : la variante brouillon

```
$ php artisan tinker --execute="echo
  App\Models\Project::factory()->draft()->make()->toJson(JSON_PRETTY_PRINT);"
{
    "title": "Nihil dignissimos quos est.",
    "slug": "nihil-dignissimos-quos-est",
    "published_at": null,
    "position": 0
}
```

make() : en mémoire, rien en base ; create() enregistrerait

le faux latin ne doit jamais arriver sur votre site public : il reste dans les tests

Lisez le code, puis le ticket : `fake()` **tire au hasard,** `unique()` **garantit deux titres différents,** `Str::slug()` **fabrique l'adresse. Vos valeurs seront d'autres mots.**

? Comment vos trois fiches projet entrent-elles dans la base ?

Le seeder crée les étiquettes, puis chaque projet avec les textes de sa fiche, et attach() remplit la pivot.

database/seeders/ProjectSeeder.php

```php
$vue = Tag::create(['name' => 'Vue.js', 'slug' => 'vue-js']);
$javascript = Tag::create(['name' => 'JavaScript', 'slug' => 'javascript']);

$quiz = Project::create([
    'title' => 'Quiz des capitales',
    'slug' => 'quiz-des-capitales',
    'cover_path' => 'images/quiz-capitales.jpg',
    'published_at' => now(),
    'position' => 2,
]);
$quiz->tags()->attach([$vue->id, $javascript->id]);
```

extrait : summary, content, demo_url, repo_url et l'étiquette Laravel suivent

`title`, et le slug tiré à la main ← Titre

`summary` ← Résumé

`content`, une rubrique par paragraphe <p> ← Contexte, rôle, problème, choix, résultat, ce que je referais

`cover_path` : images/quiz-capitales.jpg, le chemin sur le disque public, `storage/app/public/` ← Capture d'ensemble

`demo_url`, `repo_url`, ou null ← Liens démo et code

une étiquette créée une fois par `Tag::create()`, reliée par `attach()` ← Étiquettes

`attach([…])` insère une ligne dans `project_tag` par étiquette, comme `guests()->attach()` au festival

une apostrophe dans le texte s'écrit `\'` : `d\'interfaces`

`storage/app/public/` n'est pas versionné : quatre lignes dans son `.gitignore` gardent `images/` ; php `artisan storage:link` le rend visible sous `/storage/`

Suivez les flèches de la fiche vers le code : chaque rubrique de votre devoir a sa colonne, et chaque étiquette est créée une fois avant d'être reliée. Votre seeder enregistre vos trois projets, pas ceux de l'exemple.

Vous changez une migration déjà lancée : que faut-il relancer ?

migrate:fresh --seed vide la base, rejoue les migrations, relance le seeder : à chaque changement.

```
$ php artisan migrate:fresh --seed
2026_10_25_122829_create_projects_table ... DONE
2026_10_25_122830_create_tags_table ... DONE
2026_10_25_122831_create_project_tag_table ... DONE
Database\Seeders\ProjectSeeder ... DONE
```

en production, la commande refuse :

```
DB::prohibitDestructiveCommands(app()->isProduction())
```

les migrations du kit passent en premier, coupées ici



supprime toutes les tables

rejoue toutes les migrations, du kit puis les vôtres

`--seed` : lance `DatabaseSeeder`

la base est recréée à l'identique

Lisez le ticket puis la ligne : la base est vidée à chaque fois, et c'est voulu, vos données de développement viennent du seeder. Une colonne ajoutée à une migration déjà lancée n'apparaît pas sans cette commande.

? Comment voir ce que la base contient, sans page ni back-office ?

Tinker exécute une instruction PHP dans votre application : trois commandes vérifient le compte, l'ordre et les étiquettes.

```
$ php artisan tinker --execute="echo App\Models\Project::count();"
3
```

`count()` : combien de lignes

```
$ php artisan tinker --execute="echo
  App\Models\Project::orderBy('position')->pluck('title')->implode(', ');"
Météo Wavre, Quiz des capitales, Carnet de recettes
```

`pluck('title')` garde une colonne, `implode(', ')` les colle

```
$ php artisan tinker --execute="echo App\Models\Project::firstWhere('slug',
  'quiz-des-capitales')->tags->pluck('name')->implode(', ');"
Vue.js, JavaScript
```

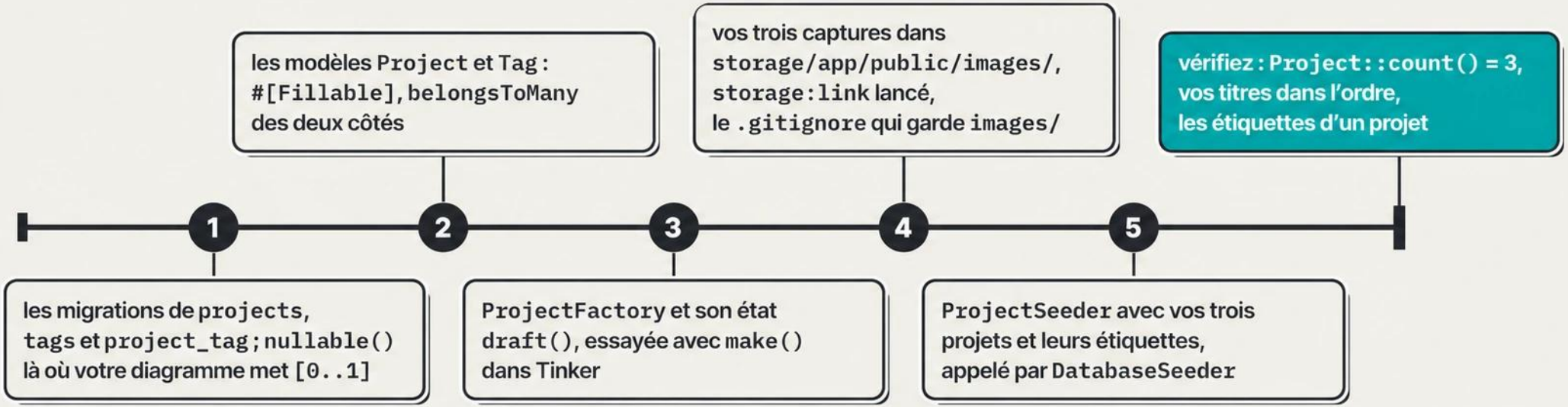
`->tags` sans parenthèses lit les étiquettes reliées dans `project_tag`

guillemets doubles autour, apostrophes dedans : l'inverse perd les guillemets dans PowerShell

Lisez chaque ticket, commande puis réponse : trois projets, vos titres dans l'ordre des positions, les bonnes étiquettes sur un projet. Remplacez le slug par celui d'un des vôtres.

? Par quoi commencez-vous, et comment saurez-vous que c'est fini ?

Cinq étapes, puis les trois commandes Tinker : trois projets, vos titres dans l'ordre, les bonnes étiquettes.



si `migrate:fresh --seed` s'arrête : lisez le nom de la colonne ou de la table dans le message, puis relisez la migration ou le seeder à cet endroit

Suivez les cinq stations ; la slide reste affichée. Au chapitre suivant, vos projets apparaissent dans le navigateur.