



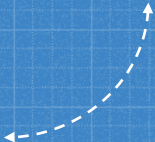
# Créer un blog simple





1

# Installation et configuration requise





# Laravel

- Exécuter la commande suivant pour télécharger la dernière version de laravel

```
AmineDa@DESKTOP-N6KL9B4 MINGW64 ~/Desktop (master)
$ laravel new Dablog

  LARAVEL

Creating a "laravel/laravel" project at "./Dablog"
Info from https://repo.packagist.org: #StandWithUkraine
Installing laravel/laravel (v9.5.0)
- Downloading laravel/laravel (v9.5.0)
- Installing laravel/laravel (v9.5.0): Extracting archive
```

- `composer create-project laravel/laravel nom_de_votre_projet`



## Configuration de la base de données

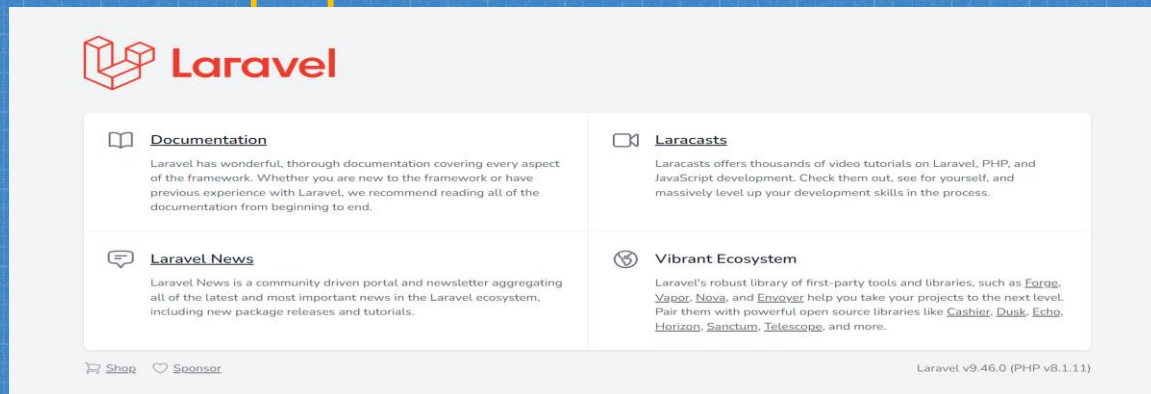
- Ouvrez le fichier `.env` à la racine de votre projet laravel
- Modifiez les paramètres suivant en utilisant les informations de votre base de données :

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=dablog
DB_USERNAME=root
DB_PASSWORD=
```








# Laravel



- Une fois l'installation terminée, accédez au répertoire de votre projet en utilisant la commande `cd: cd nom_nomPjt`
- Vous pouvez maintenant lancer le serveur de développement en exécutant la commande suivante : `php artisan serve`



The screenshot shows the Laravel website homepage. At the top is the Laravel logo, which consists of a red cube icon followed by the word "Laravel" in red. Below the logo is a grid of four featured links, each with an icon and a title. The first link is "Documentation" with a book icon, the second is "Laracasts" with a video camera icon, the third is "Laravel News" with a speech bubble icon, and the fourth is "Vibrant Ecosystem" with a group of people icon. Each link has a short description. At the bottom left are links for "Shop" and "Sponsor", and at the bottom right is the version information "Laravel v9.46.0 (PHP v8.1.11)".

 **Laravel**

 <b>Documentation</b> Laravel has wonderful, thorough documentation covering every aspect of the framework. Whether you are new to the framework or have previous experience with Laravel, we recommend reading all of the documentation from beginning to end.	 <b>Laracasts</b> Laracasts offers thousands of video tutorials on Laravel, PHP, and JavaScript development. Check them out, see for yourself, and massively level up your development skills in the process.
 <b>Laravel News</b> Laravel News is a community driven portal and newsletter aggregating all of the latest and most important news in the Laravel ecosystem, including new package releases and tutorials.	 <b>Vibrant Ecosystem</b> Laravel's robust library of first-party tools and libraries, such as <a href="#">Forge</a> , <a href="#">Vapor</a> , <a href="#">Nova</a> , and <a href="#">Envoyer</a> help you take your projects to the next level. Pair them with powerful open source libraries like <a href="#">Cashier</a> , <a href="#">Dusk</a> , <a href="#">Echo</a> , <a href="#">Horizon</a> , <a href="#">Sanctum</a> , <a href="#">Telescope</a> , and more.

 [Shop](#)  [Sponsor](#)

Laravel v9.46.0 (PHP v8.1.11)



## Installation breeze

- Pour utiliser le package Breeze, vous devrez d'abord l'installer en utilisant Composer. voici comment faire :
- Exécuter les commandes suivante :
- `Composer require laravel/breeze -dev`
- `Php artisan breeze:install`
- `Php artisan migrate`
- `Npm install`
- `Npm run dev`

# Login



Email

exemple@gmail.com

Password

••••••••

☐

Remember me

[Forgot your password?](#)

LOG IN

# Register



Name

Email

Password

Confirm Password

[Already registered?](#)

REGISTER



## MVC

- Créer un modèle qui représente la table blog
- Créer une migration qui décrit les modifications à apporter à votre base de données
- Créer Un contrôleur qui gère les requêtes HTTP
- Créer



## MVC

- Nous utilisons la commande suivante afin de créer model et migration et controller avec une seule commande :
- `Php artisan make:model nom -mcr`
- `-mrc(m=>migration, rc=>resource controller)`

```
AmineDa@DESKTOP-N6KL9B4 MINGW64 ~/Desktop/instance (master)
$ php artisan make:model blog -mcr

  INFO  Model [C:\Users\AmineDa\Desktop\instance\app\Models/blog.php] created successfully.

  INFO  Migration [C:\Users\AmineDa\Desktop\instance\database\Migrations\2023_01_08_132226_create_blogs_table.php] created successfully.

  INFO  Controller [C:\Users\AmineDa\Desktop\instance\app\Http\Controllers/BlogController.php] created successfully.

AmineDa@DESKTOP-N6KL9B4 MINGW64 ~/Desktop/instance (master)
$ |
```



## Table blog

- Un fichier de migration sera créé dans le répertoire «database/migration». Ouvrez ce fichier et ajouter le code suivante dans la méthode «up» :

```
public function up()
{
    Schema::create('blogs', function (Blueprint $table) {
        $table->id();
        $table->string("title");
        $table->string("description");
        $table->integer("id_user");
        $table->timestamps();
    });
}
```

- Et Exécuter la commande **php artisan migrate**





1

# Quatre opérations de base(**CRUD**)





# Les routes de l'application

```
<?php
```

```
use App\Http\Controllers\ProfileController;  
use App\Http\Controllers\BlogController;  
use App\Http\Controllers\welcome;  
use Illuminate\Support\Facades\Route;
```

```
Route::get("/",welcome::class)->middleware("auth");
```

```
Route::get('/dashboard', function () {  
    return view('dashboard');  
})->middleware(['auth', 'verified'])->name('dashboard');
```

```
Route::middleware('auth')->group(function () {  
    Route::get('/profile', [ProfileController::class, 'edit'])->name('profile.edit');  
    Route::patch('/profile', [ProfileController::class, 'update'])->name('profile.update');  
    Route::delete('/profile', [ProfileController::class, 'destroy'])->name('profile.destroy');  
});
```

```
Route::resource("createPost",BlogController::class)->middleware("auth");
```

```
require __DIR__.'/auth.php';
```



## page pour afficher les posts

```
@foreach ($data as $post)
<div class="card m-5">
    <div class="card-header">
        @foreach ($datauser as $user)
            @if($post->id_user == $user->id)
                {{ $user->name }}
            @endif
        @endforeach
    </div>
    <div class="card-body">
        <h5 class="card-title">{{ $post["title"] }} </h5>
        <p class="card-text">{{ $post["description"] }}</p>
    </div>
</div>
@endforeach
```



coding

**post 2**

description post 2

amine

**post 5**

description post 5

amine

**post 5**

description post 3



## Affichage des donnée(CR[ead]UD)

- Afficher les données à partir de contrôleur, vous pouvez utiliser la fonction index:

```
public function index()
{
    //
    $data = Blog::where("id_user",Auth::user()->id)->get();
    return view("home",compact("data"));
}
```

- Et à partir d'un view afficher les donnée de chaque post:



# Affichage des donnée(CR[ead]UD)

- view

```
<div class="home d-flex justify-content-around">
  <div class="allPost ">
    <h1>Your Posts</h1>
    <div class="card" style="width: 30rem; max-height: 80%; overflow-y: auto; position:fixed">
      @foreach ($data as $post)
        <div class="card-body">
          <h5 class="card-title">{{$post["title"]}}</h5>
          <p class="card-text">{{$post["description"]}}</p>
          <div class="d-flex justify-content-around">
            <a href="{{route('createPost.edit',$post->id)}}" class="btn btn-primary w-50 h-50">Edit</a>
            <form action="{{route('createPost.destroy',$post->id)}}" method="post">
              @method("DELETE")
              @csrf
              <button type="submit" class="btn btn-danger">Delete</button>
            </form>
          </div>
        </div>
      </div>
    @endforeach
  </div>
</div>
```



### post 5

description post 3

Edit

Delete

### post 6

description post 6

Edit

Delete

### post 7

description post 7

Edit

Delete



## Stockage des donnée(C[reat]RUD)

- Pour stocker les données vous pouvez utiliser la fonction store et aussi créer
- Un formulaire dans une vue :
- Fonction store:

```
use App\Models\Blog;  
public function store(Request $request)  
{  
    //  
    $data = new Blog();  
    $data->title = $request->title;  
    $data->description = $request->description;  
    $data->id_user = Auth::user()->id;  
    $data->save();  
    return redirect("/home");  
}
```



## Stockage des donnée(C[reat]RUD)

- formulaire :

```
<div class="addPost">
  <h1>Create New Post </h1>
  <form method="post" action={{route("createPost.store")}}>
    @csrf
    <div class="mb-3 " >
      <label for="Title" class="form-label">Title</label>
      <input type="text" class="form-control" id="Title" aria-describedby="emailHelp" name="title">
    </div>
    <div class="mb-3">
      <label for="Description" class="form-label">Description</label>
      <input type="textarea" class="form-control" id="Description" name="description">
    </div>
    <button type="submit" class="btn btn-primary">Post</button>
  </form>
</div>
```



## Your Posts

post 2

description post 2

[Edit](#)[Delete](#)

## Create New Post

Title

Description

[Post](#)

modifier les donnée(CRU[pdate]D)

- Pour modifier les données vous pouvez utiliser la fonction edit et update et aussi créer Un formulaire dans une vue :
- La fonction edit pour find un post et envoyer sur route à la fonction update
- Fonction edit :

```
public function edit($id)
{
    //
    $data = Blog::find($id);
    return view("edit",compact("data"));
}
```



modifier les donnée(CRU[pdate]D)

- Fonction update :

```
public function update(Request $request, $id)
{
    //
    $data = Blog::find($id);
    $data->title = $request->title;
    $data->description = $request->description;
    $data->save();
    return redirect("/home");
}
```

- formulaire

modifier les donnée(CRU[**pdate**]D)

## Formulaire

```
<div class="addPost">
  <form method="post" action="{{route('createPost.update',$data->id)}}" >
    @csrf
    @method("PUT")
    <div class="mb-3">
      <label for="Title" class="form-label" >Title</label>
      <input type="text" class="form-control" value="{{ $data->title }}" id="Title"
        aria-describedby="emailHelp" name="title">
    </div>
    <div class="mb-3">
      <label for="Description" class="form-label">Description</label>
      <input type="textarea" class="form-control" value="{{ $data->description }}"
        id="Description" name="description">
    </div>
    <button type="submit">Edit</button>
  </form>
</div>
```



supprimer les donnée(CRUD[**elete**])

- Pour supprimer les données vous pouvez utiliser la fonction destroy
- Fonction edit :

```
public function destroy($id)
{
    //
    Blog::find($id)->delete();
}
```

(détruire est toujours facile 😊)



# Thanks !

## ANY QUESTIONS?

You can find me at:

@aminedaaboub  
daaboub.am@gmail.com

Website: <https://aminedaaboub.netlify.app/>