



Q E

ChatGPT ▾

Partager



ChatGPT

Explorer les GPT

7 jours précédents

Réponse à candidature stage

To-Do List Docker

2024

Liste d'anime populaires

Liste de films populaires

j'ai comme travail à faire : une todo list avec docker, aide moi à le faire en te basant sur mes compétences... étant un débutant, je voudrais le plus de détails possible

#### 💡 Mémoire mise à jour

On va construire une todo list en utilisant Docker, en intégrant des technologies que tu maîtrises (Laravel, PostgreSQL, etc.). Voici le plan :

### 1. Choix de la stack

On va utiliser :

- **Backend** : Laravel (PHP) pour gérer les tâches.
- **Base de données** : PostgreSQL pour stocker les tâches.
- **Frontend** : Blade (le moteur de templating de Laravel).
- **Docker** : Pour isoler l'application et gérer ses dépendances.



### 2. Installation et configuration de Docker

Si Docker n'est pas encore installé :

- Télécharge Docker Desktop
- Vérifie l'installation avec `docker -v`

Crée un **Dockerfile** pour l'image Laravel :

```
dockerfile Copier Modifier  
# Utilisation de l'image PHP avec Apache  
FROM php:8.2-apache  
  
# Installation des dépendances  
RUN apt-get update && apt-get install -y \  
    libpq-dev \  
    unzip \  
    && docker-php-ext-install pdo pdo_pgsql  
  
# Installation de Composer  
COPY --from=composer:latest /usr/bin/composer /usr/bin/composer  
  
# Copie du projet Laravel dans le conteneur  
WORKDIR /var/www/html  
COPY . .  
  
# Installation des dépendances Laravel  
RUN composer install  
  
# Autoriser l'écriture dans le stockage Laravel  
RUN chown -R www-data:www-data /var/www/html/storage /var/www/html/bootstrap/cache  
  
CMD [ "apache2-foreground" ]
```



### 3. Configuration de Docker Compose

Crée un fichier **docker-compose.yml** pour orchestrer Laravel et PostgreSQL :

```
yaml Copier Modifier  
version: '3.8'  
  
services:  
  app:  
    build: .  
    container_name: todo-laravel  
    ports:
```

```
- "8000:80"
volumes:
- ./var/www/html
depends_on:
- db
environment:
- APP_ENV=local
- APP_DEBUG=true
- APP_KEY=base64:randomkey
- DB_CONNECTION=pgsql
- DB_HOST=db
- DB_PORT=5432
- DB_DATABASE=todolist
- DB_USERNAME=postgres
- DB_PASSWORD=secret

db:
image: postgres:15
container_name: todo-postgres
restart: always
environment:
POSTGRES_DB: todolist
POSTGRES_USER: postgres
POSTGRES_PASSWORD: secret
ports:
- "5432:5432"
volumes:
- pgdata:/var/lib/postgresql/data

volumes:
pgdata:
```

#### 4. Migrations et Modèle Laravel

Dans ton projet Laravel, exécute :

```
sh
↓ Copier Modifier
php artisan make:model Task -m
```

Modifie database/migrations/xxxx\_create\_tasks\_table.php :

```
php
↓ Copier Modifier
public function up()
{
    Schema::create('tasks', function (Blueprint $table) {
        $table->id();
        $table->string('title');
        $table->boolean('completed')->default(false);
        $table->timestamps();
    });
}
```

Applique la migration :

```
sh
↓ Copier Modifier
php artisan migrate
```

#### 5. Routes et Contrôleur

Crée un contrôleur :

```
sh
↓ Copier Modifier
php artisan make:controller TaskController --resource
```

Modifie app/Http/Controllers/TaskController.php :

```
php
↓ Copier Modifier
```

```

use App\Models\Task;
use Illuminate\Http\Request;

class TaskController extends Controller
{
    public function index()
    {
        return view('tasks.index', ['tasks' => Task::all()]);
    }

    public function store(Request $request)
    {
        Task::create($request->validate(['title' => 'required']));
        return redirect()->route('tasks.index');
    }

    public function update(Task $task)
    {
        ↓
        $task->update(['completed' => !$task->completed]);
        return redirect()->route('tasks.index');
    }

    public function destroy(Task $task)
    {
        $task->delete();
        return redirect()->route('tasks.index');
    }
}

```

Ajoute les routes dans **routes/web.php** :

```

php
Copier Modifier
use App\Http\Controllers\TaskController;
Route::resource('tasks', TaskController::class);

```

## 6. Vue Blade

Dans **resources/views/tasks/index.blade.php** :

```

blade
Copier Modifier
<!DOCTYPE html>
<html lang="fr">
<head>
    <title>Todo List</title>
</head>
<body>
    <h1>Ma Todo List</h1>

    <form action="{{ route('tasks.store') }}" method="POST">
        @csrf
        ↓
        <input type="text" name="title" required>
        <button type="submit">Ajouter</button>
    </form>

    <ul>
        @foreach($tasks as $task)
            <li>
                <form action="{{ route('tasks.update', $task) }}" method="POST" style="display: inline-block; margin-right: 10px;">
                    @csrf @method('PUT')
                    <input type="checkbox" onChange="this.form.submit()" {{ $task->completed ? 'checked' : '' }}>
                </form>
                {{ $task->title }}
                <form action="{{ route('tasks.destroy', $task) }}" method="POST" style="display: inline-block; margin-left: 10px;">
                    @csrf @method('DELETE')
                    <button type="submit">Supprimer</button>
                </form>
            </li>
        @endforeach
    </ul>

```

```
</body>  
</html>
```

## 7. Lancer l'application

Démarre Docker Compose :

```
sh  
docker-compose up -d
```

Copier

Modifier

Accède à ton app sur <http://localhost:8000/tasks>.



C'est une première version simple et fonctionnelle ! Tu peux améliorer l'UI avec Bootstrap ou ajouter du JavaScript pour une meilleure expérience utilisateur.

je voudrais encore plus de détails

D'accord ! Je vais détailler chaque étape en expliquant pourquoi on fait chaque action et comment tout fonctionne ensemble.

## 📌 Objectif



Créer une application de gestion de tâches (Todo List) avec Laravel et PostgreSQL, entièrement contenue dans des containers Docker.

## 🛠 1. Prérequis

Avant de commencer, assure-toi d'avoir installé :

- Docker et Docker Compose ([Téléchargement](#))
- PHP 8+, Composer, PostgreSQL et Laravel si tu veux tester hors Docker.

Vérifie que Docker est bien installé avec :

```
sh  
docker -v  
docker-compose -v
```

Copier

Modifier

## 🏗 2. Initialisation du projet Laravel

On commence par créer un nouveau projet Laravel dans un dossier `todo-app` :

```
sh  
composer create-project --prefer-dist laravel/laravel todo-app  
cd todo-app
```

Copier

Modifier

Ajoute les dépendances pour PostgreSQL :

```
sh  
composer require doctrine/dbal
```

Copier

Modifier

Ajoute Laravel Sail (facultatif si tu veux tester avec Sail au lieu de Docker Compose) :

```
sh
```

```
composer require laravel/sail --dev  
php artisan sail:install
```

Copier

Modifier

## 3. Création du Dockerfile

Le Dockerfile sert à définir l'environnement pour l'application Laravel.

Dans le dossier `todo-app`, crée un fichier `Dockerfile` :

```
dockerfile
```

Copier

Modifier

```
# Image de base : PHP avec Apache  
FROM php:8.2-apache

# Installer les extensions nécessaires  
RUN apt-get update && apt-get install -y \  
    libpq-dev \  
    unzip \  
    && docker-php-ext-install pdo pdo_pgsql

# Installer Composer  
COPY --from=composer:latest /usr/bin/composer /usr/bin/composer

# Définir le répertoire de travail  
WORKDIR /var/www/html

# Copier tout le projet dans le conteneur  
COPY . .

# Installer les dépendances Laravel  
RUN composer install

# Changer les permissions pour Laravel  
RUN chown -R www-data:www-data /var/www/html/storage /var/www/html/bootstrap/cache

# Exposer le port Apache  
EXPOSE 80

# Démarrer Apache  
CMD ["apache2-foreground"]
```

Explications :

1. On utilise une image PHP 8.2 avec Apache.
2. On installe les extensions nécessaires pour PostgreSQL (`pdo_pgsql`).
3. On installe Composer (gestionnaire de dépendances PHP).
4. On copie le projet Laravel dans le conteneur.
5. On installe les dépendances Laravel (`composer install`).
6. On donne les bons droits aux dossiers `storage` et `bootstrap/cache`.
7. On expose le port 80 pour accéder à Laravel via Apache.
8. On lance Apache en mode foreground.

## 4. Configuration de Docker Compose

Crée un fichier `docker-compose.yml` à la racine du projet :

```
yaml
```

Copier

Modifier

```
version: '3.8'

services:
  app:
```

```

build: .
  container_name: todo-laravel
  ports:
    - "8000:80"
  volumes:
    - .:/var/www/html
depends_on:
  - db
environment:
  - APP_ENV=local
  - APP_DEBUG=true
  - APP_KEY=base64:randomkey
  - DB_CONNECTION=pgsql
  - DB_HOST=db
  - DB_PORT=5432
  - DB_DATABASE=todolist
  - DB_USERNAME=postgres
  - DB_PASSWORD=secret

db:
  image: postgres:15
  container_name: todo-postgres
  restart: always
  environment:
    POSTGRES_DB: todolist
    POSTGRES_USER: postgres
    POSTGRES_PASSWORD: secret
  ports:
    - "5432:5432"
  volumes:
    - pgdata:/var/lib/postgresql/data

volumes:
  pgdata:

```

#### Explication des services :

- **app** : Conteneur Laravel basé sur le Dockerfile.
  - Monte le dossier local dans `/var/www/html` (code accessible à l'intérieur du conteneur).
  - Lie le port 80 du conteneur au port **8000** sur la machine.
  - Dépend de la base de données PostgreSQL.
- **db** : Conteneur PostgreSQL
  - Crée une base de données `todolist` avec l'utilisateur `postgres` et le mot de passe `secret`.
  - Utilise un volume `pgdata` pour la persistance des données.

## 5. Création de la base de données

Modifie `.env` pour configurer PostgreSQL :

env	 Copier  Modifier
DB_CONNECTION=pgsql DB_HOST=db DB_PORT=5432 DB_DATABASE=todolist DB_USERNAME=postgres DB_PASSWORD=secret	

Puis génère la clé de l'application Laravel :

sh	 Copier  Modifier
php artisan key:generate	

## 6. Création du modèle et de la

# migration

On va créer une table `tasks` pour stocker les tâches.

Génère un modèle avec migration :

```
sh
php artisan make:model Task -m
```

Modifie `database/migrations/xxxx_create_tasks_table.php` :

```
php

public function up()
{
    Schema::create('tasks', function (Blueprint $table) {
        $table->id();
        $table->string('title');
        $table->boolean('completed')->default(false);
        $table->timestamps();
    });
}
```

Applique la migration :

```
sh
php artisan migrate
```

## 🎯 7. Développement du contrôleur

Génère un contrôleur :

```
sh
php artisan make:controller TaskController --resource
```

Modifie `app/Http/Controllers/TaskController.php` :

```
php

use App\Models\Task;
use Illuminate\Http\Request;

class TaskController extends Controller
{
    public function index()
    {
        return view('tasks.index', ['tasks' => Task::all()]);
    }

    public function store(Request $request)
    {
        Task::create($request->validate(['title' => 'required']));
        return redirect()->route('tasks.index');
    }

    public function update(Task $task)
    {
        $task->update(['completed' => !$task->completed]);
        return redirect()->route('tasks.index');
    }

    public function destroy(Task $task)
    {
        $task->delete();
        return redirect()->route('tasks.index');
    }
}
```

## 🔗 8. Définition des routes

Ajoute les routes dans `routes/web.php` :

```
php  
use App\Http\Controllers\TaskController;  
Route::resource('tasks', TaskController::class);
```

## 🎨 9. Création de la vue Blade

Crée le fichier `resources/views/tasks/index.blade.php` :

```
blade  
  
<!DOCTYPE html>  
<html lang="fr">  
<head>  
<title>Todo List</title>  
</head>  
<body>  
<h1>Ma Todo List</h1>  
  
<form action="{{ route('tasks.store') }}" method="POST">  
    @csrf  
    <input type="text" name="title" required>  
    <button type="submit">Ajouter</button>  
</form>  
  
<ul>  
    @foreach($tasks as $task)  
        <li>  
            <form action="{{ route('tasks.update', $task) }}" method="POST" style="display: inline-block;">  
                @csrf @method('PUT')  
                <input type="checkbox" onChange="this.form.submit()" {{ $task->completed ? 'checked' : '' }}>  
            </form>  
            {{ $task->title }}  
  
            <form action="{{ route('tasks.destroy', $task) }}" method="POST" style="display: inline-block;">  
                @csrf @method('DELETE')  
                <button type="submit">Supprimer</button>  
            </form>  
        </li>  
    @endforeach  
</ul>  
</body>  
</html>
```

## 🚀 10. Démarrage du projet

Lance l'application avec :

```
sh  
docker-compose up -d
```

Puis accède à l'application :

🔗 <http://localhost:8000/tasks>

```
S C:\Users\melih\Documents\IUT\S4\P_A\VIRTU.TODO LIST\todo-app> php artisan migrate
```

```
Illuminate\Database\QueryException
```

```
SQLSTATE[08006] [7] could not translate host name "db" to address: Unknown host (Connection: pgsql, SQL: select exists (select 1 from pg_class c, pg_namespace n where n.nspname = current_schema() and c.relname = 'migrations' and c.relkind in ('r', 'p') and n.oid = c.relnamespace))
```

```
at  
vendor\laravel\framework\src\Illuminate\Database\Connection.php:823  
819 |     $this->getName(), $query, $this->prepareBindings($bindings), $e  
820 |     );  
821 | }  
822 | ↓  
→ 823 |     throw new QueryException(  
824 |         $this->getName(), $query, $this->prepareBindings($bindings), $e  
825 |     );  
826 | }  
827 | }
```

```
1  
vendor\laravel\framework\src\Illuminate\Database\Connectors\Connector.php:66
```

```
PDOException::("SQLSTATE[08006] [7] could not translate host name "db" to address: Unknown host")
```

```
2  
vendor\laravel\framework\src\Illuminate\Database\Connectors\Connector.php:66
```

```
PDO::__construct(sql:host=db;dbname='todolist';port=5432;client_encoding='utf8';sslmode='prefer', "postgres", Object(SensitiveParameterValue), [])
```

L'erreur `could not translate host name "db" to address: Unknown host` indique que Laravel ne peut pas résoudre le nom d'hôte `db`, qui est censé être le conteneur PostgreSQL. Cela peut être dû à plusieurs raisons :

## Vérifications et solutions

### 1 Vérifier si Docker tourne bien

Exécute la commande suivante pour voir si les conteneurs sont bien lancés :

```
sh  
Copier Modifier
```

```
docker ps
```

Si `db` (PostgreSQL) ne s'affiche pas, cela signifie que le conteneur n'est pas démarré.

 Solution : Redémarrer les conteneurs avec :

```
sh  
Copier Modifier
```

```
docker-compose down  
docker-compose up -d
```

Puis réessaye `docker ps` pour voir s'il tourne bien.

### 2 Vérifier si Laravel tourne bien à l'intérieur du conteneur

Essaie de rentrer dans le conteneur Laravel et exécuter la migration depuis là :

```
sh  
Copier Modifier
```

```
docker exec -it todo-laravel bash
```

Puis exécute :

```
sh  
php artisan migrate
```

Si ça marche depuis l'intérieur du conteneur, cela signifie que Laravel ne trouve pas **db** quand il est lancé sur ta machine locale.

### 3 Forcer la récréation du conteneur PostgreSQL

Parfois, Docker ne met pas à jour correctement les services. Essaye ceci :

```
sh  
docker-compose down -v  
docker-compose up --build -d
```

Explication :

- `down -v` supprime aussi les volumes (⚠ cela efface les données de PostgreSQL).
- `up --build -d` reconstruit les conteneurs.

Après cela, réessaie :

```
sh  
docker exec -it todo-laravel php artisan migrate
```

### 4 Tester la connexion à la base de données

Depuis le conteneur Laravel, exécute cette commande pour tester la connexion PostgreSQL :

```
sh  
psql -h db -U postgres -d todolist
```

Si ça ne marche pas, PostgreSQL n'est pas bien configuré.

### 5 Vérifier le fichier `.env` de Laravel

Ouvre ton `.env` et vérifie ces lignes :

```
env  
DB_CONNECTION=pgsql  
DB_HOST=db  
DB_PORT=5432  
DB_DATABASE=todolist  
DB_USERNAME=postgres  
DB_PASSWORD=secret
```

Puis exécute cette commande pour vider le cache de configuration :

```
sh  
php artisan config:clear  
php artisan cache:clear
```

Et réessaie la migration :

```
sh  
php artisan migrate
```

Avec ces étapes, PostgreSQL devrait être détecté correctement. Tiens-moi au courant si l'erreur persiste !



ChatGPT peut faire des erreurs. Envisagez de vérifier les informations importantes.

