

API – Trouve ton artisan

Backend REST développé avec **Node.js**, **Express** et **MySQL**.

Cette API fournit les données nécessaires au frontend :

- catégories
- artisans
- recherche d'artisans
- fiche détaillée d'un artisan



Stack technique

- Node.js
- Express
- MySQL / MariaDB
- mysql2 (Promise API)
- dotenv
- cors
- Swagger UI (documentation API)



Architecture

```
backend/
├── src/
│   ├── app.js # Configuration Express
│   ├── server.js # Point d'entrée du serveur
│   ├── db/
│   │   └── pool.js # Pool de connexions MySQL
│   ├── controllers/ # Logique métier
│   │   ├── artisans.controller.js
│   │   └── categories.controller.js
│   ├── routes/ # Définition des endpoints
│   │   ├── artisans.routes.js
│   │   └── categories.routes.js
│   ├── middlewares/ # Middlewares (validation, erreurs)
│   │   ├── catchAsync.js
│   │   ├── errorHandler.js
│   │   └── validateIdParam.js
│   └── docs/ # Documentation OpenAPI
```

```
├── openapi.js
└── package.json
```

Cette organisation sépare clairement :

- **routes** : définition des endpoints HTTP
- **controllers** : logique métier
- **middlewares** : gestion des erreurs et validation
- **db** : connexion à la base de données
- **docs** : documentation OpenAPI

Architecture backend

```
Client (Frontend / navigateur)
|
▼
Routes (Express)
|
▼
Middlewares
• validation paramètres
• gestion erreurs async
|
▼
Controllers
(Logique métier)
|
▼
Database Layer
(MySQL / MariaDB)
|
▼
Réponse JSON
```

Installation

Depuis le dossier backend :

```
npm install
```

Configuration

Créer un fichier .env basé sur .env.example.

Exemple :

```
PORT=3001
DB_HOST=localhost
DB_USER=app_trouve_artisan
DB_PASSWORD=CHANGE_ME
DB_NAME=trouve_ton_artisan
```

⚠ Le fichier .env ne doit jamais être versionné.

▶ Lancer l'API

En développement :

```
npm run dev
```

En production :

```
npm start
```

♥ Endpoint de test

GET /health Permet de vérifier que l'API est opérationnelle.

Réponse :

```
{
  "status": "API OK"
}
```

📖 Endpoints disponibles

GET /api/categories

Retourne la liste des catégories triées par ordre alphabétique.

GET /api/categories/:id/artisans

Retourne les artisans appartenant à une catégorie donnée.

GET /api/artisans/featured

Retourne jusqu'à **3 artisans mis en avant** (page d'accueil).

Tri :

- note décroissante
 - nom alphabétique
-

GET /api/artisans/:id

Retourne la **fiche détaillée d'un artisan**.

GET /api/artisans?search=motcle

Recherche d'artisans par :

- nom
- ville
- spécialité

Exemple : /api/artisans?search=boulangier



Documentation API

La documentation complète de l'API est disponible via Swagger : <http://localhost:3001/api-docs>

Cette documentation est basée sur la spécification **OpenAPI 3**.



Sécurité

Utilisation d'un **utilisateur MySQL dédié** à l'application.

Principe du **moindre privilège** :

- SELECT
- INSERT
- UPDATE
- DELETE

Séparation des secrets via variables d'environnement.

Aucun mot de passe réel versionné dans le dépôt.



Bonnes pratiques appliquées

Architecture claire :

routes → controllers → base de données

Utilisation de middlewares :

- `catchAsync` : gestion centralisée des erreurs async
 - `errorHandler` : gestion globale des erreurs
 - `validateIdParam` : validation des paramètres d'URL
-

Autres bonnes pratiques :

- pool de connexions MySQL
 - séparation `app` / `server`
 - standardisation charset `utf8mb4`
 - collation `utf8mb4_unicode_ci`
-



Évolutions possibles

- pagination des résultats
 - système de vérification des artisans
 - upload d'image / logo artisan
 - authentification administrateur
-