Trouve ton artisan

Documentation Technique

Région Auvergne-Rhône-Alpes

Version : 1.0.0

Date : Janvier 2026

Auteur : Développeur Web

📋 Table des matières

1. Présentation du projet

Trouve ton artisan est une plateforme web développée pour la région Auvergne-Rhône-Alpes permettant aux particuliers de trouver et contacter des artisans qualifiés dans leur région.

> 🎯 **Mission :** Créer une interface intuitive et accessible permettant de mettre en relation les particuliers avec les artisans de la région, classés par catégories et spécialités.

Contexte

La région Auvergne-Rhône-Alpes souhaite valoriser le savoir-faire de ses artisans locaux en offrant une plateforme moderne et accessible à tous. Ce projet s'inscrit dans une démarche de digitalisation des services régionaux et de soutien à l'économie locale.

Public cible

- **Particuliers** : recherche d'artisans pour des travaux, services ou achats
- **Artisans** : visibilité accrue auprès des habitants de la région
- **Région** : mise en valeur du tissu économique local

2. Objectifs et fonctionnalités

Objectifs principaux

1. Permettre aux particuliers de rechercher un artisan par catégorie ou par nom
2. Consulter une fiche artisan détaillée avec note, spécialité et localisation
3. Contacter l'artisan via un formulaire sécurisé
4. Garantir une accessibilité conforme aux normes WCAG 2.1
5. Assurer un design responsive (Mobile First)

Fonctionnalités implémentées

Frontend

Fonctionnalité	Description	Statut
Page d'accueil	Présentation du service avec section "Comment trouver mon artisan"	✓ Fait
Artisans du mois	Affichage des 3 artisans mis en avant (Top artisans)	✓ Fait
Navigation par catégories	4 catégories : Bâtiment, Services, Fabrication, Alimentation	✓ Fait
Recherche	Recherche d'artisans par nom	✓ Fait
Fiches artisans	Page détaillée avec notation par étoiles, contact, site web	✓ Fait
Formulaire de contact	Envoi d'email sécurisé à l'artisan	✓ Fait
Page 404	Page d'erreur personnalisée	✓ Fait
Mentions légales	Page des mentions légales et CGU	✓ Fait

Backend

Fonctionnalité	Description	Statut
API REST	Endpoints pour catégories, artisans et contact	✓ Fait
Authentification API	Protection par clé API	✓ Fait
Validation	Validation des entrées avec express-validator	✓ Fait
Envoi d'emails	Nodemailer avec support SMTP	✓ Fait
Sécurité	Helmet, CORS, Rate limiting, XSS protection	✓ Fait

3. Technologies utilisées

Stack Frontend

Technologie	Version	Rôle
React.js	18.2	Bibliothèque JavaScript pour l'interface utilisateur
React Router	6.x	Routage côté client (SPA)
Bootstrap	5.3	Framework CSS responsive
Sass/SCSS	1.69	Préprocesseur CSS
Axios	1.6	Client HTTP pour les requêtes API
React Helmet Async	2.0	Gestion du SEO (meta tags)
React Icons	5.0	Bibliothèque d'icônes

Stack Backend

Technologie	Version	Rôle
Node.js	18+	Runtime JavaScript côté serveur
Express.js	4.18	Framework web minimaliste
Sequelize	6.35	ORM pour MySQL
MySQL	8.0+	Base de données relationnelle
Nodemailer	6.9	Envoi d'emails SMTP
Helmet	7.1	Sécurisation des headers HTTP
express-rate-limit	7.1	Protection contre les attaques par force brute
express-validator	7.0	Validation des entrées

4. Architecture du projet

Architecture globale

> Le projet suit une architecture **Client-Serveur** classique avec séparation claire entre le frontend (React) et le backend (Node.js/Express).

Structure des dossiers

Backend

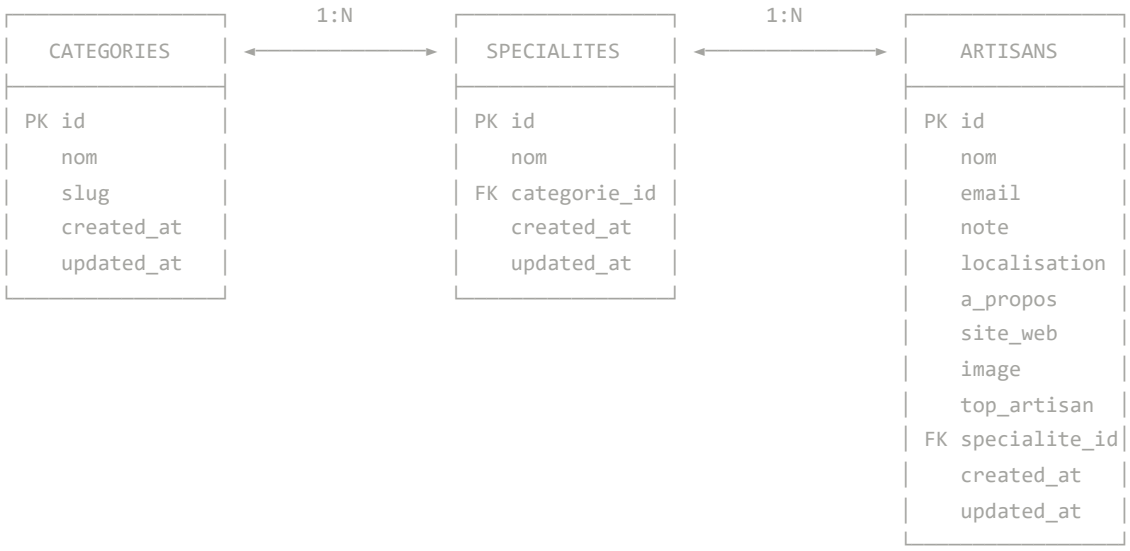
```
backend/
├── config/
│   └── database.js          # Configuration Sequelize/MySQL
├── controllers/
│   ├── artisanController.js # Logique métier artisans
│   ├── categorieController.js # Logique métier catégories
│   └── contactController.js # Logique envoi emails
├── middleware/
│   ├── apiKeyAuth.js        # Authentification par clé API
│   └── validateRequest.js   # Validation des requêtes
├── models/
│   ├── index.js             # Export des modèles
│   ├── Artisan.js           # Modèle Artisan
│   ├── Categorie.js         # Modèle Catégorie
│   └── Specialite.js        # Modèle Spécialité
├── routes/
│   ├── artisanRoutes.js     # Routes /api/artisans
│   ├── categorieRoutes.js   # Routes /api/categories
│   └── contactRoutes.js     # Routes /api/contact
├── scripts/
│   └── sql/
│       ├── create_database.sql  # Création BDD
│       └── seed_database.sql    # Données initiales
├── .env                     # Variables d'environnement
├── package.json
└── server.js                # Point d'entrée
```

Frontend

```
frontend/
├── public/
│   ├── index.html          # Template HTML
│   ├── manifest.json       # PWA manifest
│   └── favicon.ico         # Favicon
├── src/
│   ├── components/
│   │   ├── common/         # Composants réutilisables
│   │   │   ├── ArtisanCard.jsx
│   │   │   ├── Loader.jsx
│   │   │   └── StarRating.jsx
│   │   └── layout/         # Composants de mise en page
│   │       ├── Header.jsx
│   │       └── Footer.jsx
│   ├── pages/
│   │   ├── HomePage.jsx
│   │   ├── ArtisansListPage.jsx
│   │   ├── ArtisanDetailPage.jsx
│   │   ├── LegalPage.jsx
│   │   └── NotFoundPage.jsx
│   ├── services/
│   │   └── api.js          # Client Axios
│   ├── styles/
│   │   ├── _variables.scss  # Variables SCSS
│   │   ├── _mixins.scss     # Mixins SCSS
│   │   └── global.scss      # Styles globaux
│   ├── App.js              # Composant principal
│   └── index.js            # Point d'entrée
├── .env
└── package.json
```

5. Base de données

Modèle Conceptuel de Données (MCD)



Description des tables

Table : categories

Colonne	Type	Contraintes	Description
id	INT	PRIMARY KEY, AUTO_INCREMENT	Identifiant unique
nom	VARCHAR(100)	NOT NULL, UNIQUE	Nom de la catégorie
slug	VARCHAR(100)	NOT NULL, UNIQUE	Slug pour l'URL

Table : specialites

Colonne	Type	Contraintes	Description
id	INT	PRIMARY KEY, AUTO_INCREMENT	Identifiant unique
nom	VARCHAR(100)	NOT NULL	Nom de la spécialité

categorie_id	INT	FOREIGN KEY	Référence vers categories

Table : artisans

Colonne	Type	Contraintes	Description
id	INT	PRIMARY KEY, AUTO_INCREMENT	Identifiant unique
nom	VARCHAR(255)	NOT NULL	Nom de l'artisan
email	VARCHAR(255)	NOT NULL	Email de contact
note	DECIMAL(2,1)	DEFAULT 0	Note moyenne (0-5)
localisation	VARCHAR(255)	NOT NULL	Ville
a_propos	TEXT	-	Description
site_web	VARCHAR(255)	-	URL du site web
image	VARCHAR(255)	-	Photo de l'artisan
top_artisan	BOOLEAN	DEFAULT FALSE	Artisan du mois
specialite_id	INT	FOREIGN KEY	Référence vers specialites

Données initiales

Catégories (4)

- Bâtiment
- Services
- Fabrication
- Alimentation

Artisans (17)

Dont 3 Top artisans mis en avant sur la page d'accueil :

- Au pain chaud (Boulanger) - Montélimar - Note : 4.8
- Chocolaterie Labbé (Chocolatier) - Lyon - Note : 4.9

- Orville Salmons (Chauffagiste) - Evian - Note : 5.0

6. Documentation API

> **Base URL :** `http://localhost:5000/api`
> **Authentification :** Header `x-api-key` requis pour toutes les requêtes

Endpoints Catégories

Méthode	Endpoint	Description
GET	/api/categories	Liste toutes les catégories avec leurs spécialités
GET	/api/categories/:slug	Détails d'une catégorie par son slug
GET	/api/categories/:slug/artisans	Liste des artisans d'une catégorie

Exemple de réponse : GET /api/categories

```
{
  "success": true,
  "count": 4,
  "data": [
    {
      "id": 1,
      "nom": "Bâtiment",
      "slug": "batiment",
      "specialites": [
        { "id": 1, "nom": "Chauffagiste" },
        { "id": 2, "nom": "Electricien" }
      ]
    }
  ]
}
```

Endpoints Artisans

Méthode	Endpoint	Description
GET	/api/artisans	Liste tous les artisans

GET	/api/artisans/top	Liste les artisans du mois (top_artisan = true)
GET	/api/artisans/search?q=query	Recherche d'artisans par nom
GET	/api/artisans/:id	Détails d'un artisan par son ID

Endpoint Contact

Méthode	Endpoint	Description
POST	/api/contact	Envoie un email à un artisan

Corps de la requête POST /api/contact

```
{
  "artisanId": 1,
  "nom": "Jean Dupont",
  "email": "jean.dupont@email.fr",
  "sujet": "Demande de devis",
  "message": "Bonjour, je souhaite..."
}
```

7. Guide d'installation

Prérequis

- **Node.js** version 18 ou supérieure
- **npm** (inclus avec Node.js)
- **MySQL** version 8.0 ou supérieure
- **Git**

Étapes d'installation

1. Cloner le repository

```
git clone https://github.com/votre-username/trouve-ton-artisan.git
cd trouve-ton-artisan
```

2. Installer les dépendances

```
# Backend
cd backend
npm install

# Frontend
cd ../frontend
npm install
```

3. Configurer la base de données

```
# Se connecter à MySQL et exécuter les scripts
mysql -u root -p

SOURCE backend/scripts/sql/create_database.sql;
SOURCE backend/scripts/sql/seed_database.sql;
```

4. Configurer les variables d'environnement

Créer un fichier `.env` dans le dossier backend :

```
PORT=5000
NODE_ENV=development
DB_HOST=localhost
```

```
DB_PORT=3306
DB_NAME=trouve_ton_artisan
DB_USER=root
DB_PASSWORD=votre_mot_de_passe
FRONTEND_URL=http://localhost:3001
API_KEY=votre_cle_api_secrete
```

Créer un fichier `.env` dans le dossier frontend :

```
REACT_APP_API_URL=http://localhost:5000/api
REACT_APP_API_KEY=votre_cle_api_secrete
REACT_APP_SITE_NAME=Trouve ton artisan
```

5. Lancer l'application

```
# Terminal 1 - Backend
cd backend
npm run dev

# Terminal 2 - Frontend
cd frontend
npm start
```

> **✓ Application accessible sur :**
> Frontend : `http://localhost:3001`
> Backend API : `http://localhost:5000/api`

8. Sécurité

Mesures de sécurité implémentées

Mesure	Outil/Méthode	Description
Headers HTTP sécurisés	Helmet.js	Protection contre clickjacking, XSS, sniffing MIME, etc.
CORS	cors middleware	Restriction des origines autorisées
Rate Limiting	express-rate-limit	100 requêtes max par IP toutes les 15 minutes
Validation des entrées	express-validator	Validation et sanitization de toutes les entrées
Protection XSS	xss library	Échappement des caractères dangereux
Protection SQL Injection	Sequelize ORM	Requêtes préparées automatiques
Authentification API	API Key	Clé secrète requise dans les headers

Bonnes pratiques

- Variables d'environnement pour les données sensibles
- Fichiers `.env` exclus du versioning Git
- HTTPS obligatoire en production
- Mots de passe jamais stockés en clair

9. Accessibilité (WCAG 2.1)

Critères respectés

Critère	Description	Implémentation

Navigation au clavier	Tous les éléments interactifs accessibles au clavier	Focus visible, tabindex appropriés
Labels des formulaires	Tous les champs ont un label associé	Attributs for/id, aria-label
Contrastes	Ratio de contraste suffisant	Couleurs conformes WCAG AA
Alternatives textuelles	Images avec attribut alt	Alt descriptifs ou décoratifs
Structure sémantique	Hiérarchie des titres	h1 > h2 > h3 logique
Rôles ARIA	Landmarks et rôles	role="banner", "navigation", "main"
Skip links	Lien d'accès rapide au contenu	Classe .visually-hidden

10. Déploiement

Options de déploiement

Frontend (React)

- **Vercel** : Déploiement automatique depuis GitHub
- **Netlify** : Build automatique et CDN global
- **GitHub Pages** : Hébergement gratuit

Backend (Node.js)

- **Heroku** : PaaS simple avec add-on MySQL
- **Railway** : Déploiement moderne avec base de données intégrée
- **DigitalOcean App Platform** : Infrastructure scalable
- **VPS** : Serveur dédié avec PM2

Base de données

- **PlanetScale** : MySQL serverless
- **AWS RDS** : MySQL managé
- **ClearDB** : Add-on Heroku

Commandes de build

```
# Build du frontend pour la production
cd frontend
npm run build

# Le dossier 'build' contient les fichiers statiques à déployer
```

Variables d'environnement en production

> ⚠ **Important :** En production, assurez-vous de :
>
> - Utiliser une clé API complexe et unique
> - Configurer HTTPS
> - Définir NODE_ENV=production

- Utiliser un service SMTP fiable pour les emails

Trouve ton artisan - Documentation Technique

Région Auvergne-Rhône-Alpes © 2026

Version 1.0.0 - Janvier 2026