FORMATION DEVELOPPEUR WEB ET WEB MOBILE

CENTRE DE FORMATION EUROPEEN

Eléve : GOULJIAR FRED

# DEVOIR DE BILAN

# TROUVE TON ARTISAN

## SOMMAIRE

Contexte du projet :

# I - Présentation

Dans le cadre de sa stratégie de valorisation de l'artisanat local, la **Région Auvergne-Rhône-Alpes** a mandaté la création d'une **plateforme web dédiée aux artisans** de son territoire. Cette région, qui compte près de **221 000 entreprises artisanales** réparties sur **12 départements**, souhaite offrir aux habitants un outil moderne et accessible permettant de **trouver facilement un artisan** selon sa spécialité et de le contacter pour des renseignements, des devis ou des prestations.

Le projet est suivi par les bureaux de la Région situés à **Lyon**, et s'inscrit dans une volonté d'intégration au sein de l'environnement numérique institutionnel existant de la Région :
www.auvergnerhonealpes.fr/contenus/ladministration-regionale

# II - Expression des besoins

La mission est structurée en plusieurs étapes :

1. **Réalisation des maquettes** :
    - Création d'interfaces ergonomiques et modernes via **Figma**, pour **mobile, tablette et ordinateur**, dans une démarche **mobile first**.
    - Validation du design par les équipes de la Région avant le développement.
2. **Développement du site** :
    - Mise en place du **frontend en React.js**, stylisé avec **Bootstrap** et **Sass**, en conformité avec la charte graphique régionale (police *Graphik* et logo fourni).
    - Création d'une **API REST sécurisée** avec **Node.js** et **Express**, interrogeant une base de données **MySQL** via **Sequelize**.
3. **Fonctionnalités attendues** :
    - Affichage dynamique des catégories, spécialités, et fiches artisan,
    - Système de **recherche** par nom,
    - **Formulaire de contact** pour chaque artisan,
    - Respect des **normes d'accessibilité WCAG 2.1** pour tous les publics (jeunes, seniors, personnes en situation de handicap).
4. **Sécurité et hébergement** :
    - L'accès à l'API doit être **restreint à l'application frontend**,
    - Mise en place des **bonnes pratiques de sécurité** (protection des routes, gestion des erreurs, validation des entrées, etc.),

- Le site doit être **hébergé** et mis en ligne en version de démonstration.

## III - Contraintes

1- **Techniques** :

- La base de données MySQL contiendra les données sur les artisans, spécialités, et catégories. Chaque artisan appartient à **une seule spécialité**, et chaque spécialité est liée à **une seule catégorie**.
- L'**API ne devra interroger que cette base de données**, et non d'autres sources.
- L'application devra utiliser exclusivement les technologies imposées : React, Node.js, MySQL, Express, Sequelize, Figma, Git/GitHub.

2- **Accessibilité & ergonomie** :

- Respect des normes **WCAG 2.1** (accessibilité numérique),
- Expérience utilisateur fluide, simple et adaptée à tous profils,
- Site pensé pour un usage quotidien sur **mobile**.

3- **Graphisme** :

- Utilisation de la police **Graphik** (standard de la région),
- Intégration du **logo officiel** fourni,
- Harmonisation avec l'identité visuelle de la Région.

4- **Code & qualité** :

- Code propre, bien commenté, indenté,
- Vérification de la conformité **W3C**,
- Gestion de version via **GitHub**.

# IV - Livrable attendus

### 1. Maquettes UI/UX

- Réalisées sous Figma
- Déclinées pour mobile, tablette et desktop (approche mobile first)
- Conformes à :
    - L'identité visuelle de la Région (police Graphik, logo fourni)
    - L'accessibilité WCAG 2.1
- Validées par l'équipe de la Région avant développement

### 2. Frontend (ReactJS + Bootstrap + Sass)

- Site complet avec les pages suivantes :
    - Page d'accueil avec recherche d'artisan
    - Page "Fiche artisan" comprenant :
        - Nom
        - Image (photo/logo)
        - Note (étoiles)
        - Spécialité
        - Localisation
        - À propos
        - Formulaire de contact (nom, email, objet, message) → envoi d'un mail
        - Lien vers le site web de l'artisan (si existant)
    - Page 404 personnalisée :
        - Image illustrative
        - Texte adapté
        - Gérée via le router
- Fonctionnalités :
    - Affichage dynamique des catégories, spécialités, artisans
    - Système de recherche par nom

- Accessibilité & SEO :
    - Normes WCAG 2.1

**3. Backend (Node.js + Express + Sequelize + MySQL)**

- API REST sécurisée permettant :
    - La récupération des catégories, spécialités, artisans
    - Le traitement des formulaires de contact
- Contraintes :
    - La base de données MySQL est la seule source utilisée
    - L'API doit être protégée (accès restreint à l'application React)

**4. Base de données MySQL**

- Contient :
    - Artisans (1 spécialité par artisan)
    - Spécialités (1 par catégorie)
    - Catégories
- Jeu d'essai fourni à intégrer

**5. Qualité & Hébergement**

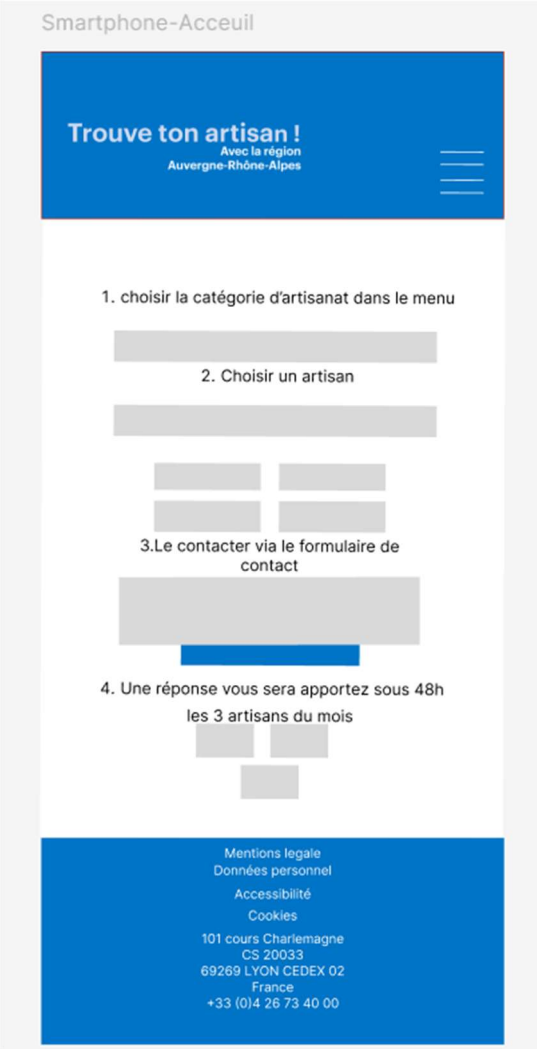
- Code propre, commenté, indenté
- Conformité W3C
- Versionnement avec Git/GitHub
- Hébergement de la version de démonstration en ligne

# V – Maquettes Figma

https://www.figma.com/design/mTtBPHdYOfsfF4RPx2Nfio/Trouve_ton_artisan?m=auto&t=agNJBbWCIHmvmlYt-6

Frames Smartphone

Page d'accueil :



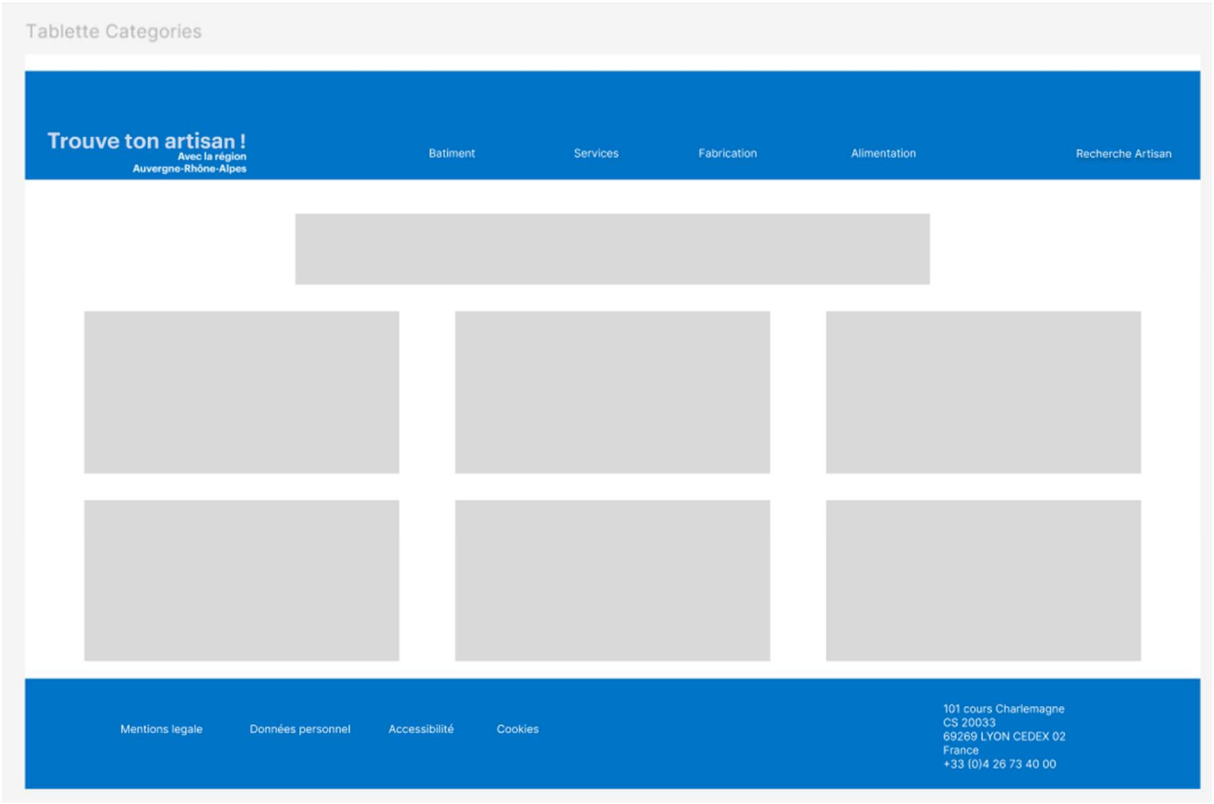
Page Catégorie


Page non trouvé

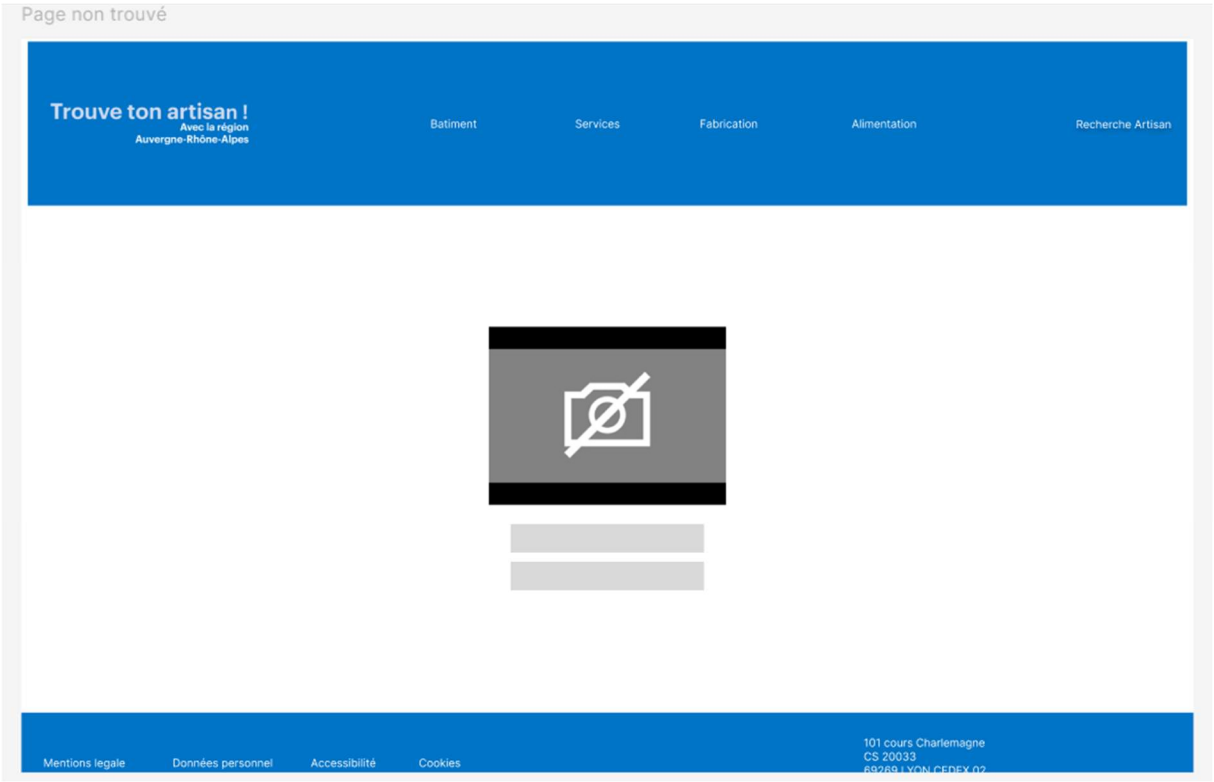
## Frames Tablette

Page d’acceuil


Page catégorie

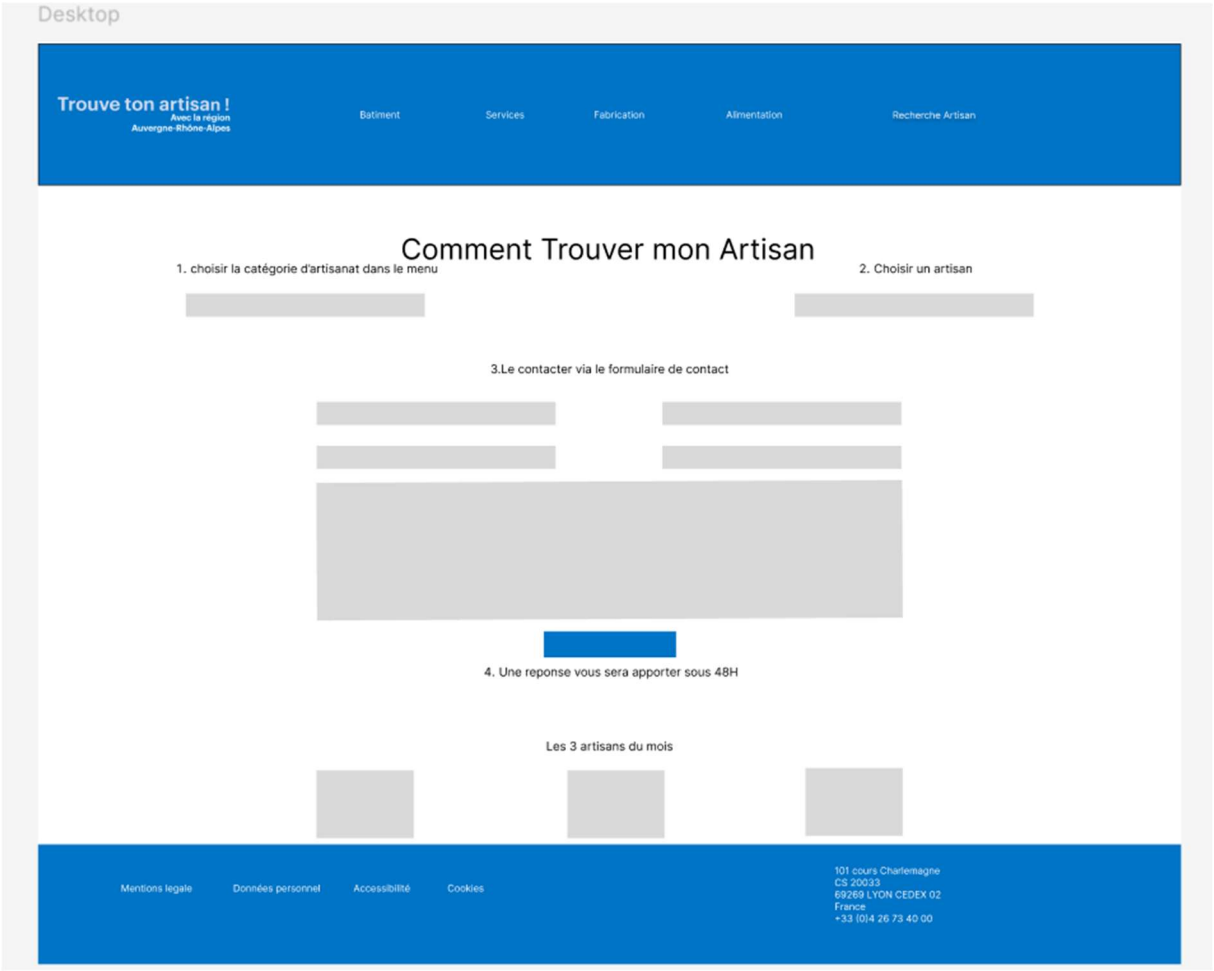



Page non trouvé

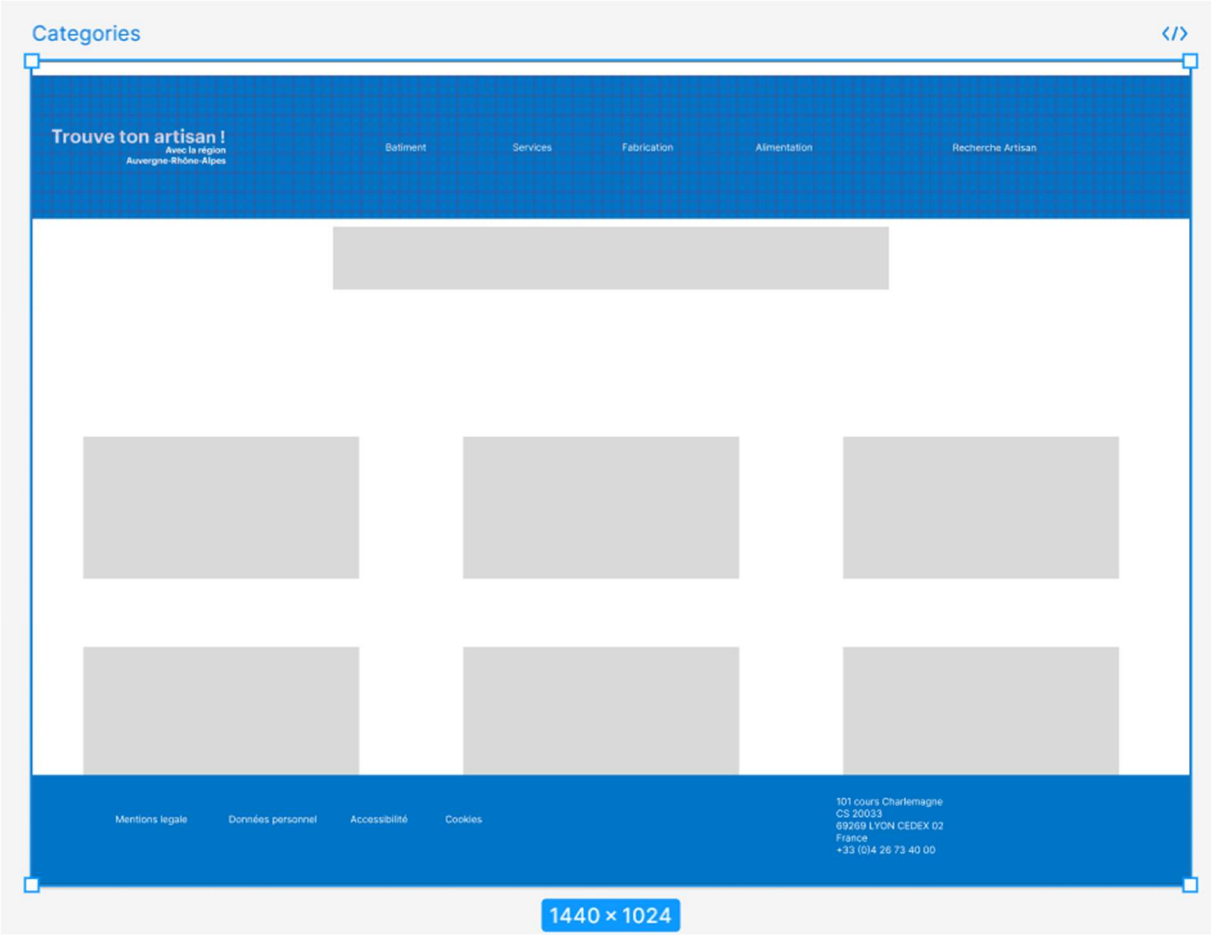


Frames Desktop

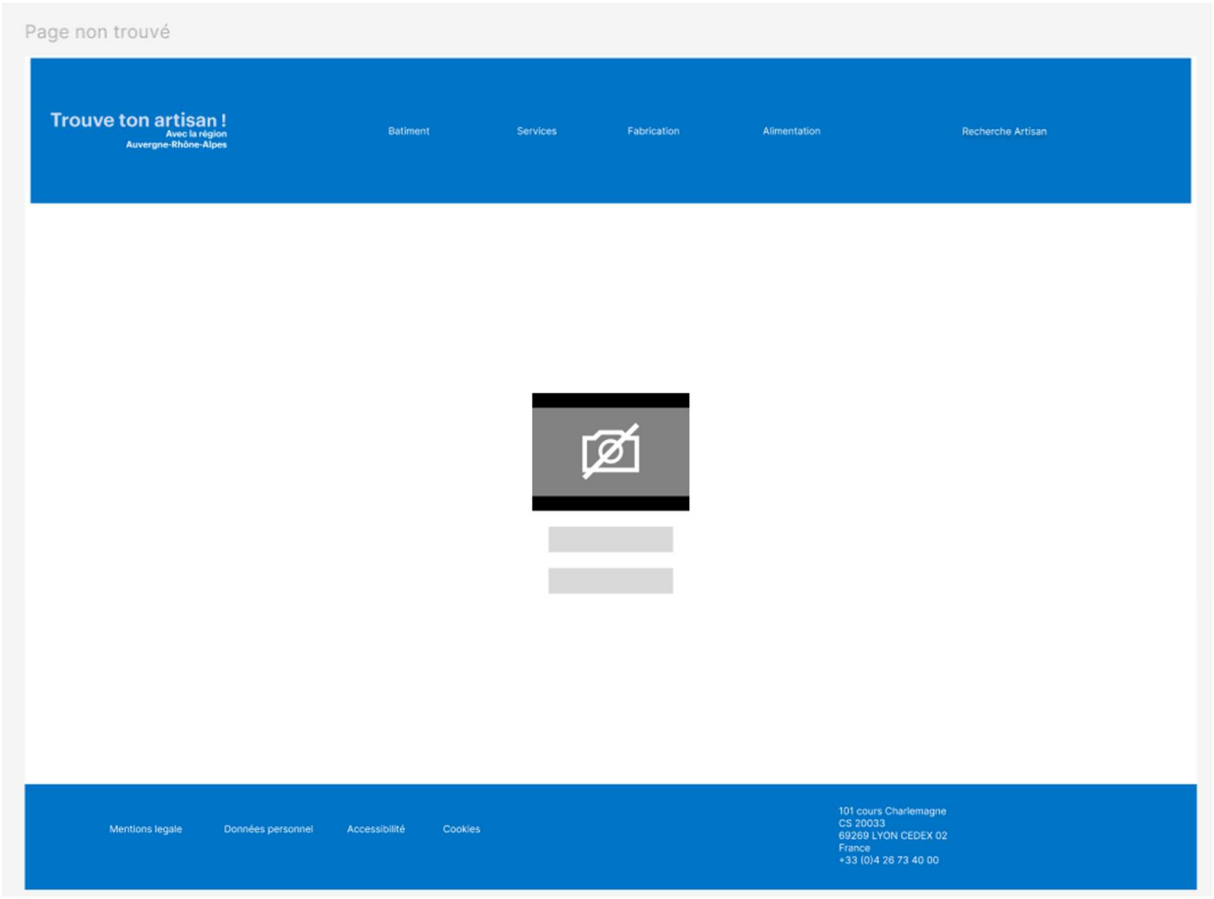
Page accueil




Page catégorie



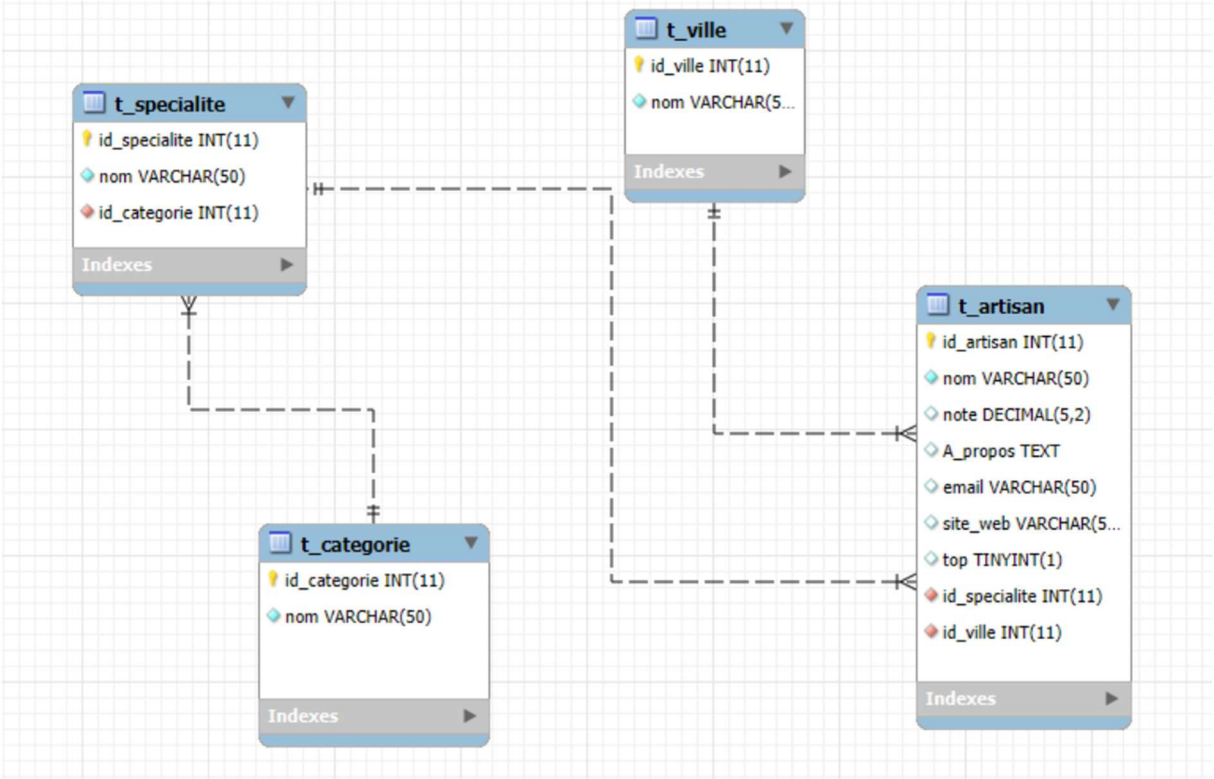

Page non trouvé


Page non trouvé
Trouve ton artisan !
Avec la région
Auvergne-Rhône-Alpes
Batiment
Services
Fabrication
Alimentation
Recherche Artisan
Mentions legale
Données personnel
Accessibilité
Cookies
101 cours Charlemagne
CS 20033
69269 LYON CEDEX 02
France
+33 (0)4 26 73 40 00

# VI-Base de donnée

Base de donnée MySQL

Voici le schéma du MCD




**Tables & Attributs :**

**categories**

| Champ | Type | Contraintes |
|---|---|---|
| id | INT | PK, AUTO_INCREMENT |
| libelle | VARCHAR(100) | NOT NULL |

---

**specialites**

| Champ | Type | Contraintes |
|---|---|---|
| id | INT | PK, AUTO_INCREMENT |
| libelle | VARCHAR(100) | NOT NULL |
| categorie_id | INT | FK → categories.id |

**artisans**

| Champ | Type | Contraintes |
|---|---|---|
| id | INT | PK, AUTO_INCREMENT |
| nom | VARCHAR(100) | NOT NULL |
| note | FLOAT | DEFAULT 0 |
| localisation | VARCHAR(150) | NOT NULL |
| a_propos | TEXT | NULL |
| email | VARCHAR(150) | NOT NULL |
| site_web | VARCHAR(255) | NULL |
| specialite_id | INT | FK → specialites.id |

**Contraintes d'intégrité :**

- specialites.categorie_id → clé étrangère vers categories.id
- artisans.specialite_id → clé étrangère vers specialites.id

## VII – La Sécurité

L'API à été réalisé avec le Framework Express

La connexion à la base de données MySQL est sécurisée

- Les identifiants de connexion sont stockés dans des fichiers .env non versionnés
    - Création du dossier à la racine du projet pour y stocker les variables de la base de donnée (host, port, login etr mot de passe)
- Responsabilisation
    - Création de dossiers séparés (routes/contrôleurs/service) pour responsabilisé chaque élément et pour une gestion plus saine.
    - Gestion des erreurs (try catch)
- Mise en place de middelware

- Création d'un identifiant et d'un mot de passe pour l'accès à la base de donnée trouve_artisan.

- **VIII – La description de la veille, effectuée durant le projet, sur les vulnérabilités de sécurité, description des vulnérabilités éventuellement trouvées et des failles potentiellement corrigées**

Suite bug, j'ai appris et utilisé le pool pour l'accès au serveur car j'avais des difficultés à accéder au résultat des requêtes (perte de connexion de la base de donnée…)

```
// Utilisation de la version promise pour simplifier la gestion async
const promisePool = pool.promise();
promisePool.getConnection()
 .then(conn => {
  console.log('Connexion à la base de données réussie');
  conn.release(); // libérer la connexion immédiatement
 })
 .catch(err => {
  console.error('Erreur de connexion à la base de données', err);
 });

module.exports = promisePool;
```