A. Présentation du projet

Contexte

Le projet **« Trouve ton artisan »** répond au besoin de faciliter la mise en relation entre des particuliers et des artisans ou entreprises locales.
De nombreuses personnes rencontrent des difficultés à trouver rapidement un professionnel fiable selon leur domaine d'activité, leur localisation ou leur spécialité.

Le problème principal à résoudre est donc :

- la dispersion de l'information,
- le manque de visibilité des artisans,
- et l'absence d'une plateforme simple et centralisée.

Objectifs

L'application doit permettre :

- de consulter une liste d'artisans classés par catégories,
- d'afficher les informations détaillées d'un artisan,
- de mettre en avant les meilleurs artisans,
- d'effectuer une recherche rapide par nom ou catégorie,
- d'envoyer un message de contact via un formulaire.

Enjeux

Les enjeux du projet sont :

- **fonctionnels** : proposer une navigation simple et intuitive,
- **techniques** : garantir une architecture fiable et évolutive,
- **utilisateurs** : améliorer l'expérience utilisateur et la confiance,
- **valeur ajoutée** : visibilité accrue pour les artisans locaux.

B. Architecture technique

Schéma global

L'architecture repose sur une **application web full stack** :

- **Frontend** : application React (interface utilisateur)
- **Backend** : API REST Node.js / Express
- **Base de données** : MySQL
- **ORM** : Sequelize
- **Communication** : requêtes HTTP JSON entre frontend et backend

---

Langages et technologies utilisées

- Frontend :
    - JavaScript
    - React
    - React Router
    - Bootstrap
- Backend :
    - Node.js
    - Express
    - Sequelize
    - MySQL
- Autres outils :
    - dotenv
    - CORS

Hébergement et services externes

- Backend prévu pour un déploiement sur **Render**
- Base de données MySQL locale (ou hébergée ultérieurement)
- Pas de service externe tiers (mail, paiement, etc.)

Organisation du code

Le backend est organisé selon une **architecture en couches** :

- config/ : configuration de la base de données
- models/ : modèles Sequelize
- controllers/ : logique des routes
- services/ : logique métier
- routes/ : définition des endpoints
- utils/ : gestion centralisée des erreurs

Cette organisation favorise :

- la lisibilité,
- la maintenabilité,
- l'évolutivité.

Sécurité

- Variables sensibles protégées via .env
- Validation des données côté backend avec Sequelize
- Gestion des erreurs centralisée
- Protection CORS
- Pas d'authentification implémentée à ce stade (choix assumé)

C. Choix techniques

Pourquoi React ?

React a été choisi pour :

- sa logique basée sur les composants,
- sa popularité et sa pérennité,
- sa gestion efficace de l'état,
- la facilité de navigation avec React Router.

Pourquoi MySQL / phpMyAdmin ?

MySQL a été choisi car :

- il est robuste et largement utilisé,
- il s'intègre parfaitement avec Sequelize,
- phpMyAdmin facilite l'administration de la base,
- il est adapté aux données relationnelles structurées.

Compromis et limites

- Absence d'authentification utilisateur
- Pas de tests automatisés
- Envoi de message simulé (sans envoi réel d'e-mail)

Ces choix ont permis de respecter les délais et de garantir un projet fonctionnel.

D. Démarche de conception

Outils utilisés

- **Figma** pour la conception des maquettes
- Visual Studio Code pour le développement
- Git pour le versionnage

Organisation du travail

Le travail a été organisé par étapes :

1. Analyse du besoin
2. Conception des maquettes
3. Mise en place du backend
4. Création du frontend
5. Connexion API
6. Tests et corrections

Les fonctionnalités ont été développées par priorité.

E. Analyse fonctionnelle

User stories principales

- En tant qu'utilisateur, je veux voir les artisans par catégorie.
- En tant qu'utilisateur, je veux consulter la fiche d'un artisan.
- En tant qu'utilisateur, je veux contacter un artisan via un formulaire.
- En tant qu'utilisateur, je veux voir les meilleurs artisans.

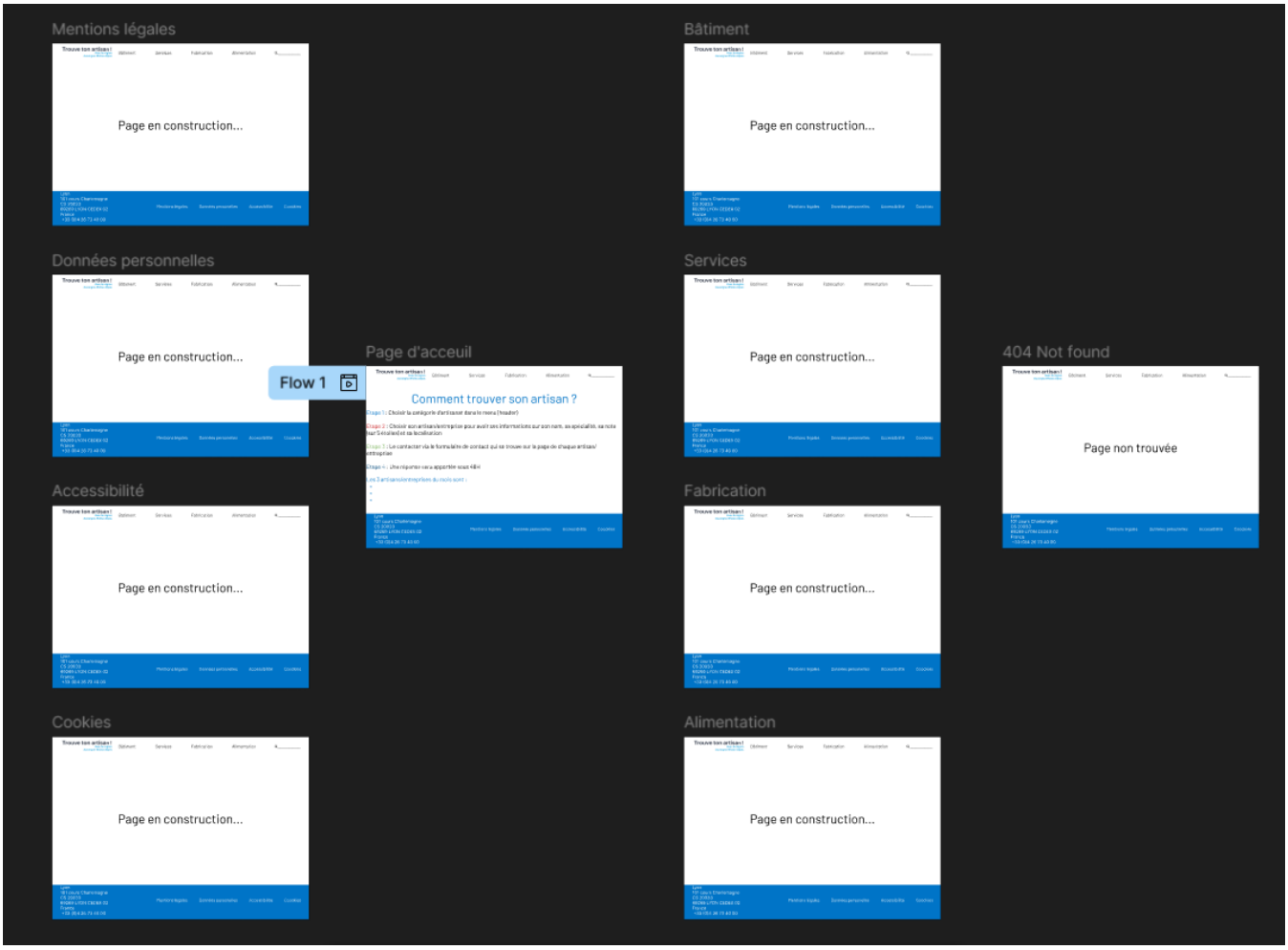
Diagrammes

Les diagrammes utilisés :

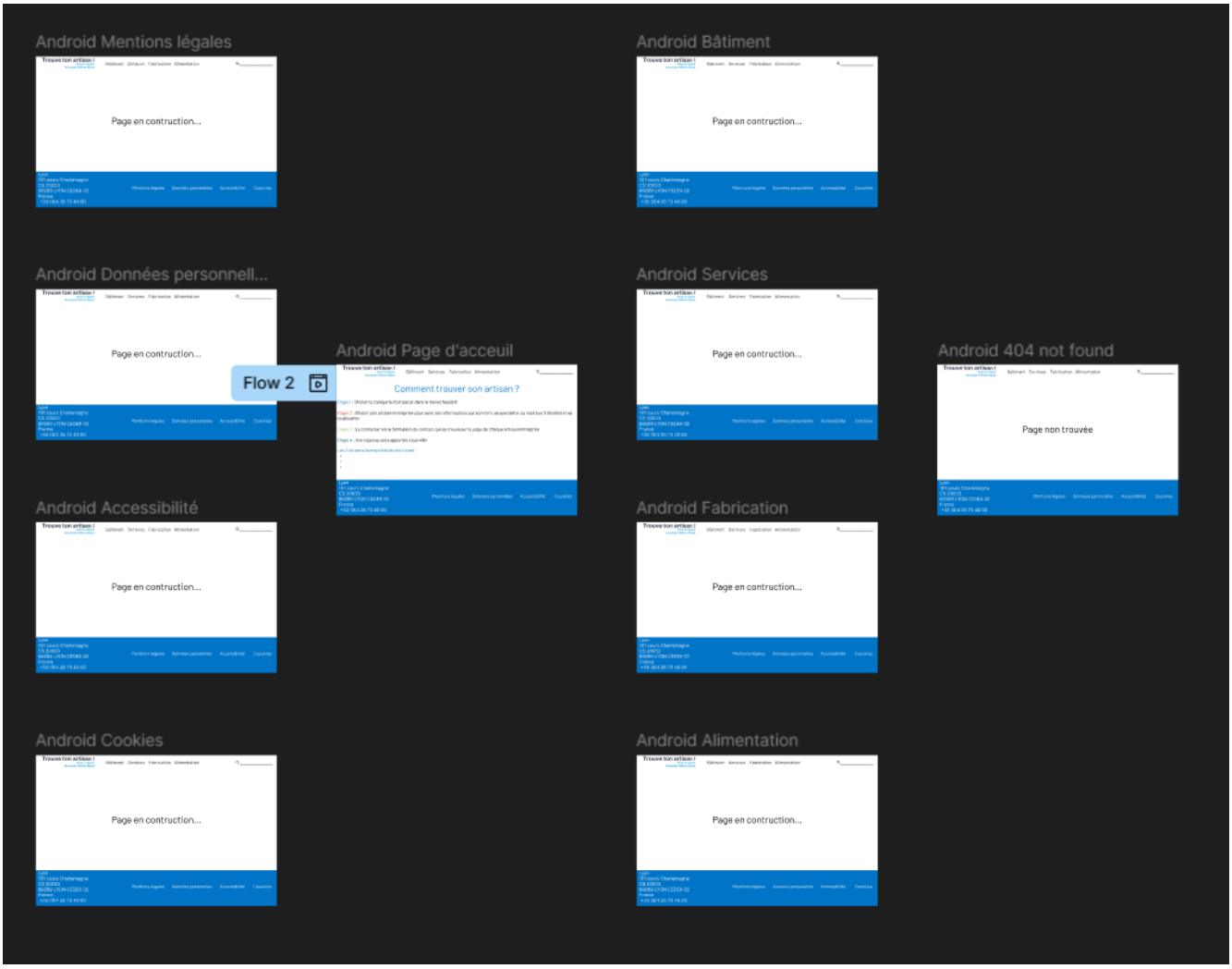
- diagramme de cas d'utilisation (use-case),
- diagramme de séquence pour l'envoi d'un message,
- diagramme d'activité pour la navigation utilisateur.

F. Maquettes Figma

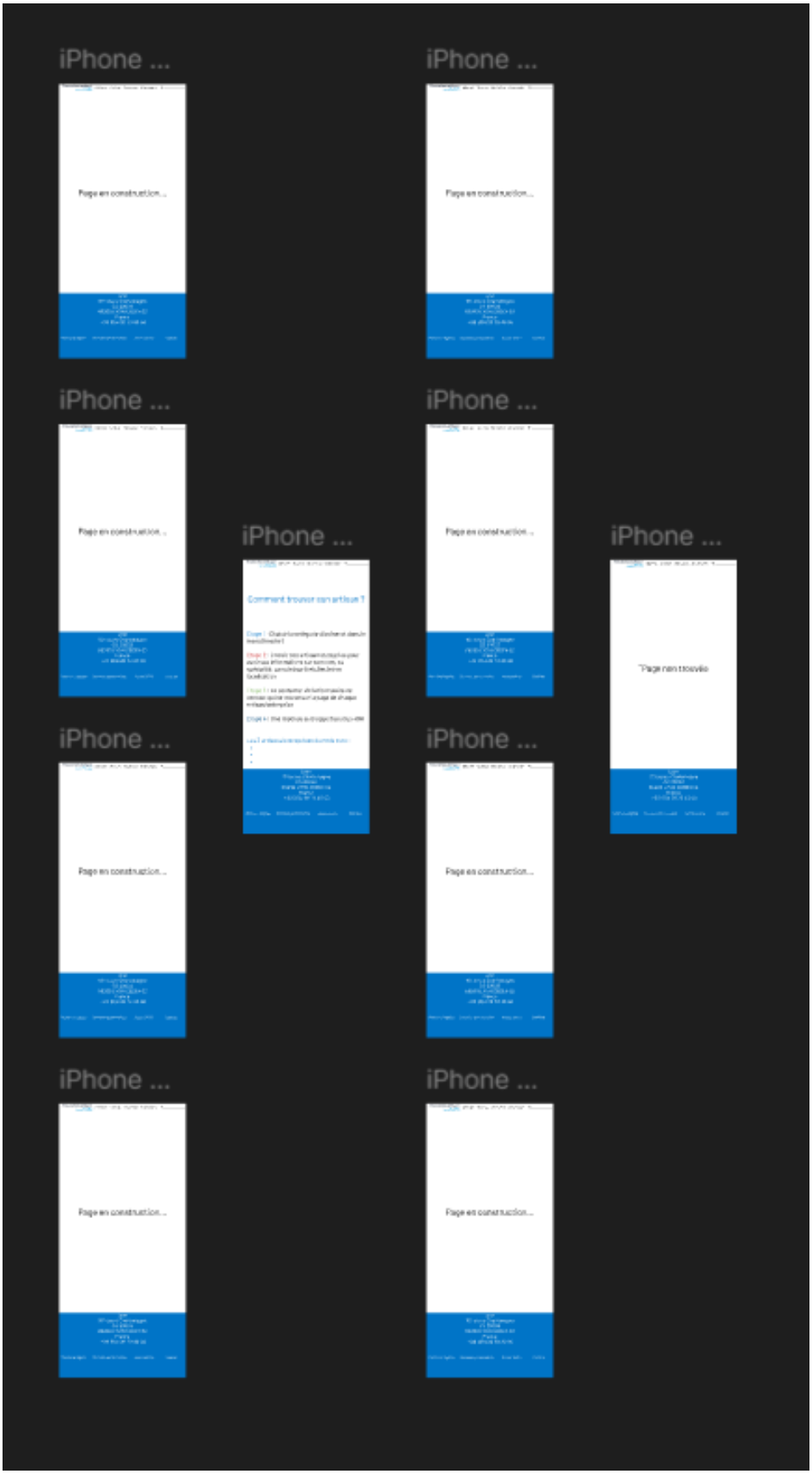
Ordinateur :



Tablette :


Android Mentions légales
Page en contruction...
Android Bâtiment
Page en contruction...
Android Données personnell...
Page en contruction...
Flow 2
Android Page d'acceuil
Comment trouver son artisan ?
Android Services
Page en contruction...
Android 404 not found
Page non trouvée
Android Accessibilité
Page en contruction...
Android Fabrication
Page en contruction...
Android Cookies
Page en contruction...
Android Alimentation
Page en contruction...

Mobile :


iPhone ...
Page en construction...
iPhone ...
Page en construction...
iPhone ...
Page en construction...
iPhone ...
Page en construction...
iPhone ...
Comment trouver son article ?
iPhone ...
Page non trouvée
iPhone ...
Page en construction...
iPhone ...
Page en construction...
iPhone ...
Page en construction...
iPhone ...
Page en construction...

G. Partie développement

Composants importants

- Page d’accueil avec artisans mis en avant
- Composants de liste d’artisans
- Fiche artisan détaillée
- Formulaire de contact
- API REST structurée

Fonctionnalités implémentées

- CRUD complet côté backend
- Filtrage par catégorie
- Route spéciale pour les artisans « top »
- Gestion des erreurs serveur
- Communication frontend / backend

H. Contraintes et solutions

- Problèmes de cohérence des données → normalisation
- Gestion des erreurs Sequelize → handler centralisé
- Problèmes CORS → configuration dédiée

I. Recettage / Tests

Procédure de tests

- Tests manuels fonctionnels
- Vérification des routes API
- Tests de navigation et formulaires

Vérification des besoins

Chaque fonctionnalité a été testée par rapport aux objectifs définis au départ.

Outils de test

- Navigateur
- Console développeur
- Postman pour tester l'API REST

J. Conclusion

Ce qui est terminé

- Application fonctionnelle
- API opérationnelle
- Architecture propre
- Interface responsive

Améliorations possibles

- Authentification utilisateurs
- Tableau de bord administrateur
- Envoi réel d'e-mails
- Tests automatisés
- Déploiement complet en production