Projet Formulaire Majordhom

Test technique

Développement d'un formulaire de contact responsive et sécurisé

Sommaire

3. Introduction

3.1. Contexte du projet

Le projet Formulaire Majordhom a été réalisé dans le cadre d'un test technique proposé par l'entreprise Majordhom, une agence immobilière spécialisée dans la digitalisation du secteur.

L'objectif est de concevoir et de déployer une application web simple permettant à un visiteur de remplir un formulaire de contact afin de demander une visite, être rappelé ou obtenir des informations supplémentaires sur un bien immobilier.

Ce projet vise à démontrer des compétences techniques en intégration web, développement back-end PHP, gestion de base de données MySQL, et déploiement sous Docker.

3.2. Objectifs du test technique

- Intégrer une maquette fournie en respectant les bonnes pratiques de design et de responsivité.
- Créer un formulaire fonctionnel capable d'enregistrer les données dans une base MySQL.
- Assurer la sécurité du traitement des données (CSRF, honeypot, validation).
- Déployer un environnement complet et reproductible via Docker Compose.
- Rendre le projet disponible sur GitHub public avec un README clair.

3.3. Outils et technologies utilisés

Catégorie	Outils et versions
Langages	HTML5, CSS3, JavaScript, PHP 8.3
Base de données	MySQL 8.4
Outils Dev	Docker Desktop, phpMyAdmin
IDE / Gestion	VS Code, Git, GitHub
Méthodologie	Approche incrémentale, développement modulaire

4. Cahier des charges

4.1. Besoin fonctionnel

L'application doit permettre à un utilisateur de :

- Remplir un formulaire de contact complet (civilité, nom, prénom, email, téléphone, sujet, message, disponibilité).
- Envoyer les données vers un serveur PHP.
- Les enregistrer dans une base de données MySQL.
- Recevoir un message de confirmation une fois la demande validée.

4.2. Exigences principales

Fonctionnelles :

- Envoi sécurisé des données (requêtes POST, validation serveur).
- Vérification des champs obligatoires.
- Gestion d'un message de confirmation ou d'erreur.

Non fonctionnelles :

- Interface responsive et adaptée aux différents écrans.
- Code clair et commenté.
- Exécution fluide et sans erreurs.

4.3. Contraintes techniques et de sécurité

- Déploiement exclusivement sous Docker (Apache + PHP, MySQL, phpMyAdmin).
- Validation et filtrage des entrées utilisateur (CSRF, XSS, SQL Injection).
- Persistance des données via volume Docker.
- Utilisation de PDO pour les connexions sécurisées.

5. Conception technique

5.1. Schéma de base de données

La base contient une table principale leads qui stocke les informations soumises par les visiteurs :

- Données personnelles (nom, prénom, email, téléphone)
- Sujet de la demande
- Message et disponibilités
- Adresse IP et date d'envoi

"Figure 1 – Schéma de la base de données"

5.2. Explication du modèle logique des données

Chaque enregistrement représente une soumission unique de formulaire.

Le champ id sert de clé primaire.

Les autres champs sont définis selon les besoins fonctionnels.

Les champs created_at et ip permettent un suivi et une traçabilité des envois.

6. Modélisation UML

6.1. Diagramme de cas d'utilisation

Ce diagramme montre les interactions entre le Visiteur (acteur principal) et le Système PHP/MySQL :

- Remplir le formulaire
- Envoyer la demande
- Valider et enregistrer les données
- Afficher le message de confirmation

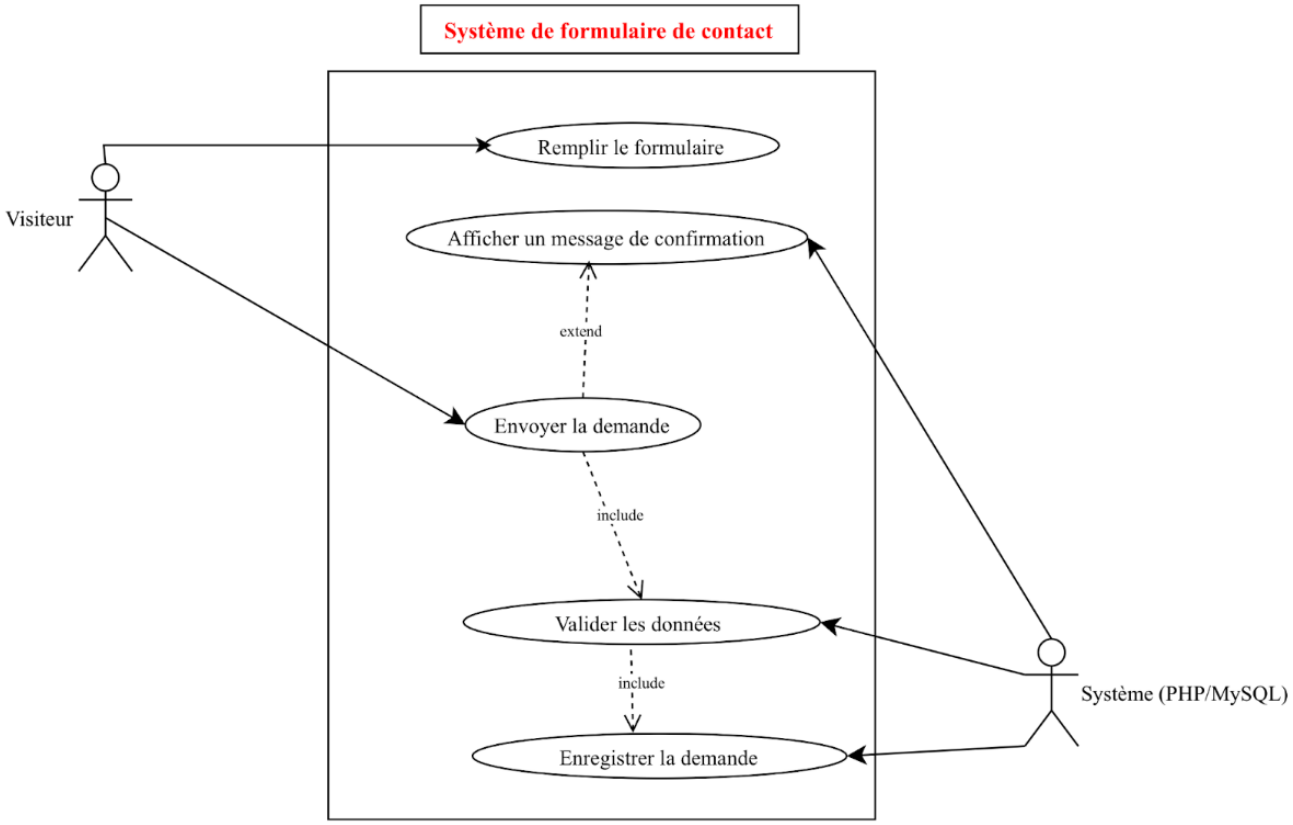


Figure 2 – Diagramme de cas d'utilisation

6.2. Diagramme de séquence

Le diagramme illustre la communication entre :

- Le Visiteur,
- Le Navigateur,
- Le Serveur PHP,
- Et la Base MySQL.

On y voit le processus d'envoi, de validation et de réponse.

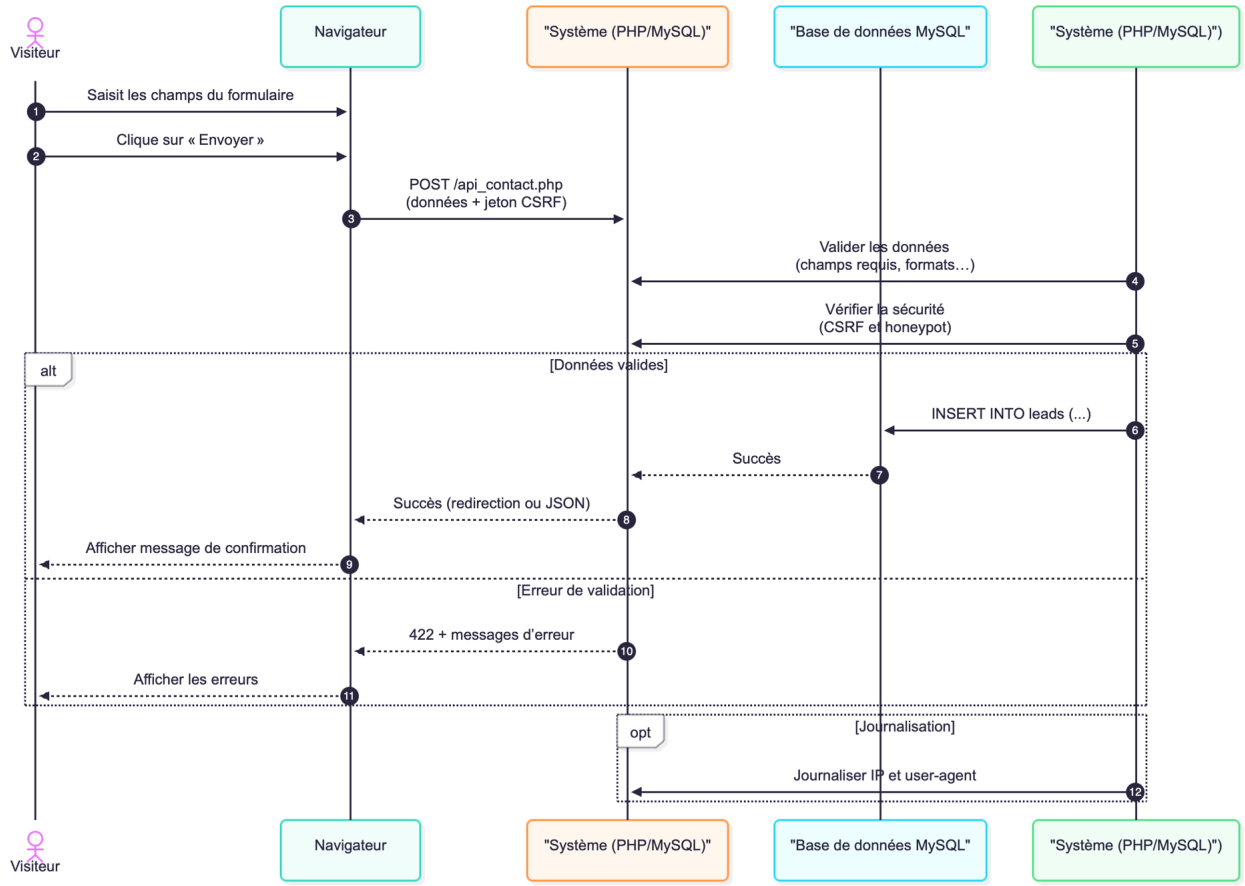
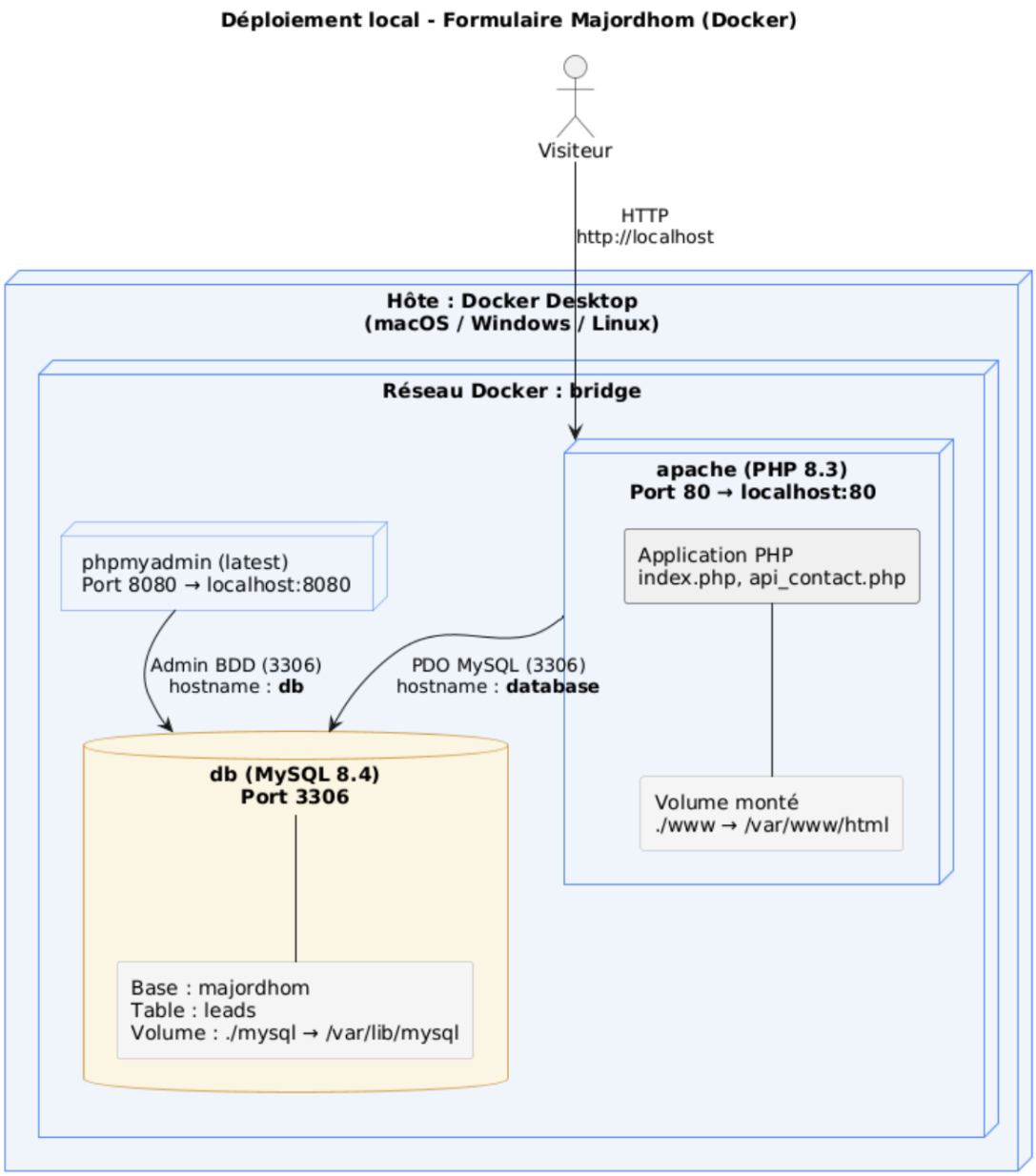


Figure 2 - Diagramme de séquence : processus d'envoi du formulaire

6.3. Diagramme de déploiement (Docker)

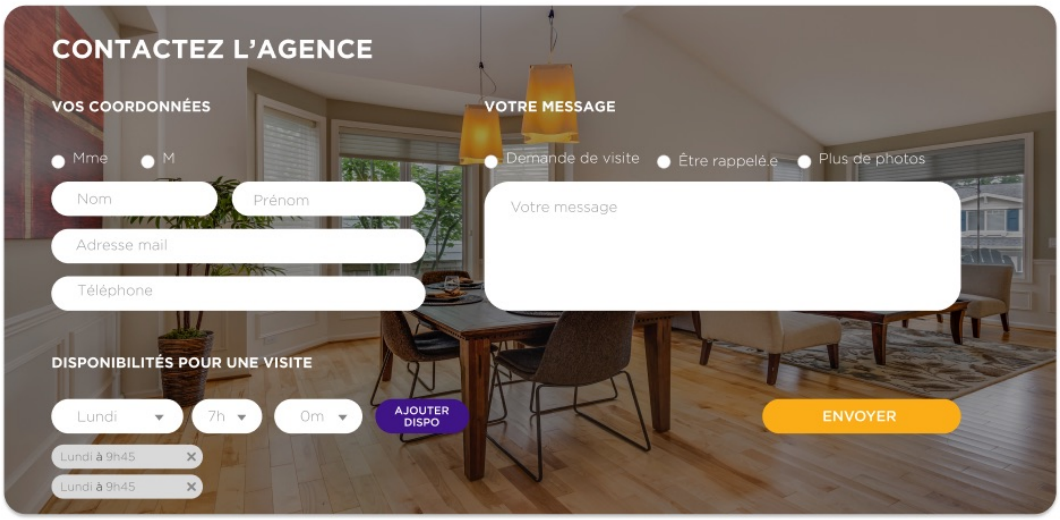
Ce schéma représente le déploiement de l'application dans un environnement Docker composé de :

- Un conteneur apache (PHP 8.3)
- Un conteneur mysql (8.4)
- Un conteneur phpMyAdmin (admin BDD)



7. Maquette et intégration

7.1. Présentation visuelle du formulaire



7.2. Principes d'intégration et résponsivité

La maquette a été intégrée à l'aide de HTML/CSS purs avec une structure flexible.

Le design s'adapte automatiquement à toutes les résolutions (mobile, tablette, desktop).

Le code respecte les normes d'accessibilité et les standards W3C.

8. Analyse et retour d'expérience

8.1. Difficultés rencontrées

- Configuration initiale de Docker et communication entre conteneurs.
- Validation et envoi du formulaire (gestion des erreurs côté client et serveur).
- Structure du projet et gestion des chemins relatifs.

8.2. Solutions et apprentissages

- Création d'un docker-compose.yml clair et stable.
- Mise en place de validations via PHP et JavaScript.
- Apprentissage de la logique de réseau Docker (bridge, ports, volumes).

8.3. Compétences développées

- Déploiement d'un environnement web complet sous Docker.
- Bonne compréhension de la structure client / serveur.
- Application des bonnes pratiques de sécurité web.
- Organisation et documentation d'un projet technique.

9. Conclusion

9.1. Bilan du projet

Le projet a permis de réaliser une application web fonctionnelle et sécurisée intégrant tous les aspects essentiels du développement web moderne : front-end, back-end, base de données et conteneurisation.

9.2. Perspectives et améliorations possibles

- Ajouter un système de mails automatiques (confirmation utilisateur).
- Mettre en place une interface d'administration pour la gestion des leads.
- Déployer le projet sur un hébergeur cloud (AWS, Render, ou Netlify + backend Docker).