

Dossier Projet

ECC RENOVATION



Titre professionnel : Développeur Web et Web Mobile

Estefania Capitão

Introduction et Compétences

- **Présentation :**

Application « deux-en un » (vitrine + gestion) pour une entreprise de rénovation

- **Compétences couvertes :**

- ✓ FRONT END :

- Maquettage
 - Intégration statique
 - Développement dynamique

- ✓ BACK END :

- Base de données relationnelle
 - Accès aux données
 - Composants métier

Contexte et Besoins

- **Objectif principal :**

Moderniser la présence en ligne et optimiser la gestion des devis.

- **Utilisateurs :**

- Visiteur (Prospect) : Particulier ou professionnel cherchant à faire des travaux.
- Client : Client existant pour qui un projet est en cours de chiffrage.
- Administrateur (Mr Capitão) : Gérant de l'entreprise, à l'aise avec les outils numériques de base mais n'est pas technicien.

- **Fonctionnalité Clé :**

Gestion du cycle de vie des devis (CRUD par admin, validation ou refus par client).

Environnement Technique

- Langages :

- ❖ PHP 8.1,
- ❖ SQL,
- ❖ JavaScript (ES6),
- ❖ HTML, SCSS (Sass)

- Conteneurisation :

- ❖ Docker
- ❖ Docker Compose

- Serveur :

- ❖ Apache (dans un conteneur Docker)

- Bases de Données : **Persistance Polyglotte**

- ❖ **MySQL 8.0** (Relationnel : clients, devis)
- ❖ **MongoDB** (NoSQL : messages)

- Outils :

- ❖ VS Code
- ❖ Docker Desktop
- ❖ Composer
- ❖ Git
- ❖ Figma

Méthodologie de Gestion de Version (Git)

- **Workflow Professionnel** : Adoption du "Feature Branch Workflow" pour ne pas travailler directement sur la version stable.
- **Branche main Protégée** : Contient uniquement la version stable et "dockerisée" de l'application (V1.0).
- **Branches de Fonctionnalités** : Tout nouveau développement (ex: page politique de confidentialité) est réalisé sur une branche isolée (ex: feature/politique).
- **Fusion (Merge)** : Intégration dans main uniquement après test et validation.
- **Sécurité** : Fichier .env (mots de passe) exclu du dépôt via .gitignore.

Stratégie de déploiement (Docker)

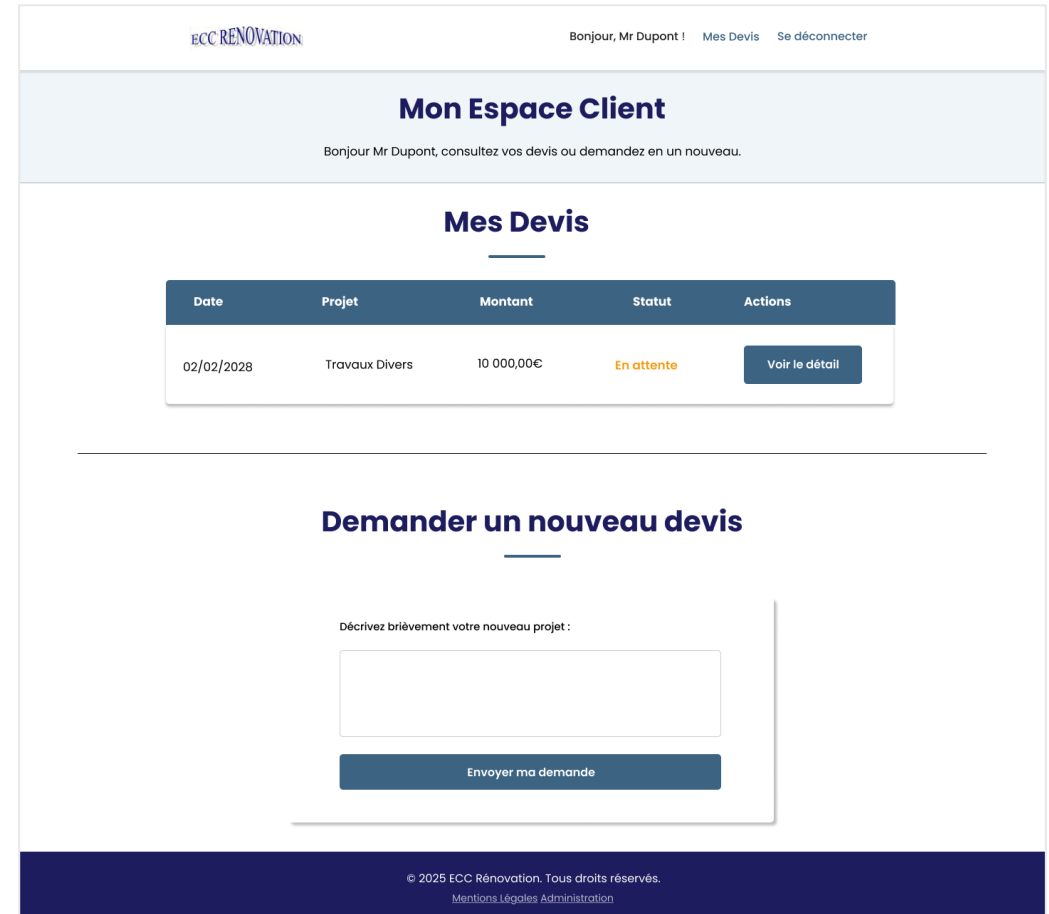
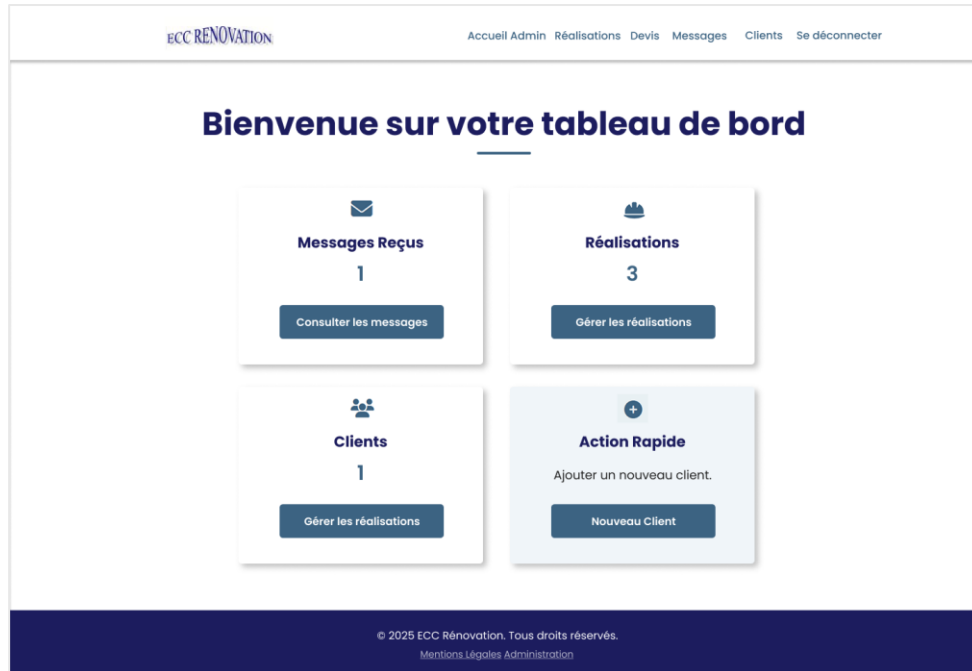
- **Infrastructure Cible** : Serveur VPS (Virtual Private Server) sous Linux (Ubuntu).
- **Processus Simplifié** :
 - Installation de Docker et Git sur le serveur.
 - Clonage du dépôt GitHub.
 - Configuration du fichier .env de production (mots de passe réels).
 - Lancement unique : `docker-compose up -d --build`.

Avantages Clés :

- **Reproductibilité** : L'environnement de production est strictement identique au local (mêmes versions de PHP, MySQL, MongoDB).
- **Portabilité** : L'application tourne dans des conteneurs isolés, indépendants du système hôte.

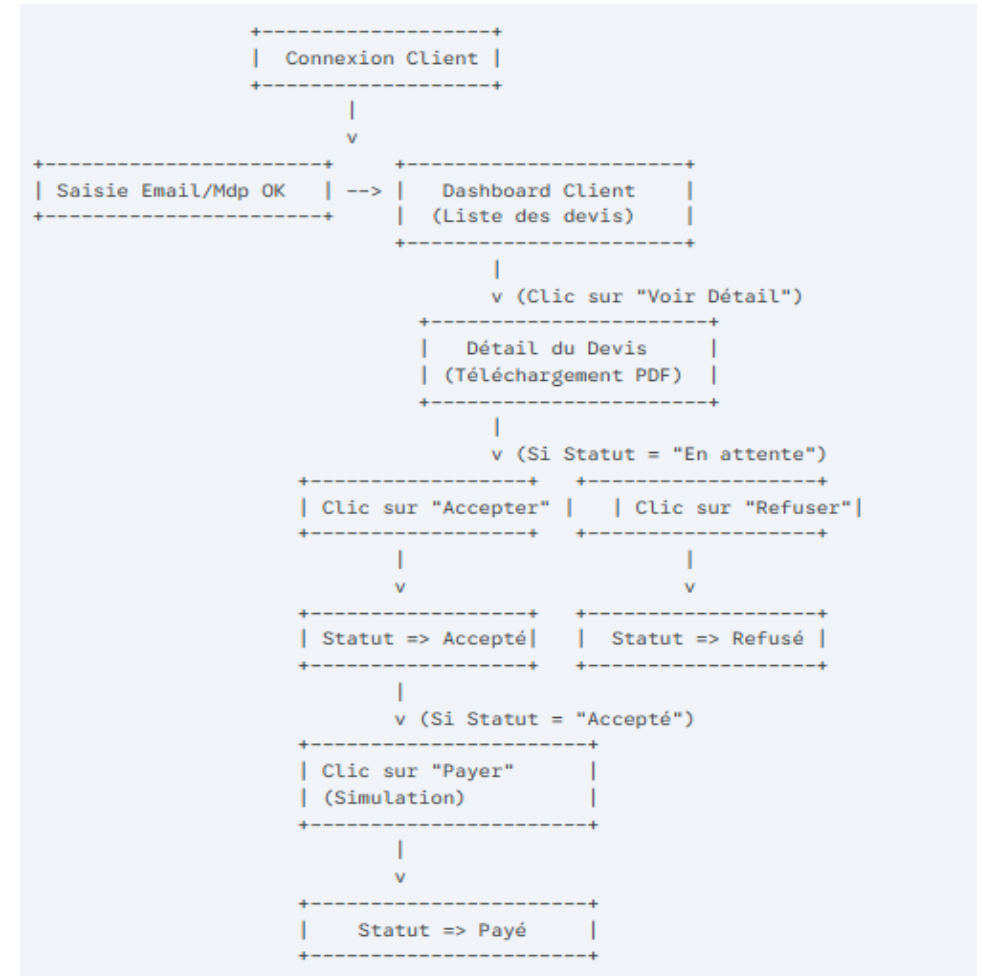
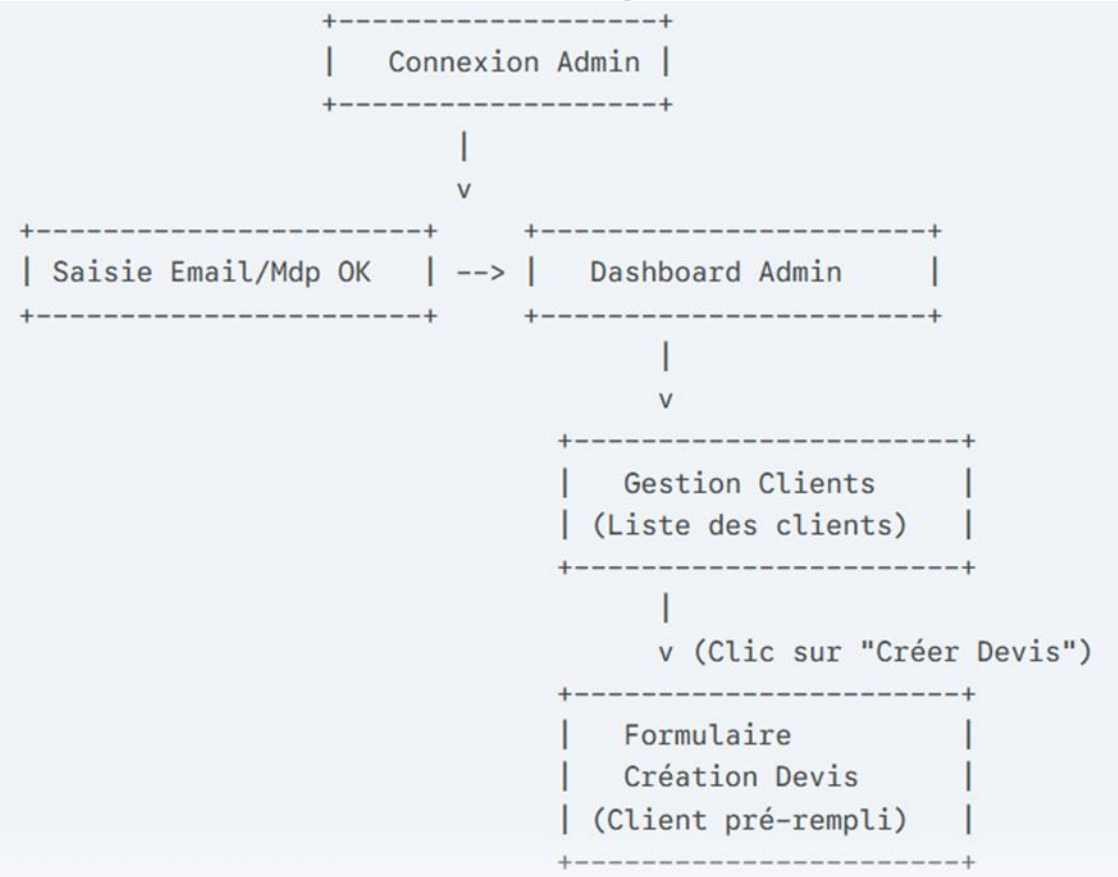
Maquettes et Conception Front-End (1/2)

- Visuels :



Maquettes et Conception Front-End (2/2)

- Schémas de navigation :



Réalisation Front-End (Statique) (1/4)

Grille de services

- Visuel :



Réalisation Front-End (Statique) (2/4)

- **Code HTML (Structure simplifiée) :**

```
<div class="services-grid">  
  <div class="service-card"> ... </div>  
  <div class="service-card"> ... </div>  
  <div class="service-card"> ... </div>  
</div>
```

Réalisation Front-End (Statique) (3/4)

- **Code SCSS (Extrait de `_home.scss`)**

```
.services-grid {  
  display: grid;  
  grid-template-columns: repeat(auto-fit, minmax(280px, 1fr));  
  gap: 30px;  
}
```

Réalisation Front-End (Statique) (4/4)

Explication du choix technique :

- **CSS Grid** → mise en page **automatiquement responsive**.
- `grid-template-columns` → Adaptabilité des cartes au nombre de colonnes idéales selon la taille de l'écran, sans media queries.

Réalisation Front-End (Dynamique)(1/2)

Exemple : Chargement dynamique du portfolio via fetch()

- Extrait **JavaScript** (extrait de js/main.js)

```
fetch('/api/get_projets.php')//a l'intérieur d'une condition if (portfolioGrid)
    .then(response => {
        if (!response.ok) {
            throw new Error('La requête a échoué');
        }
        return response.json();
    })
    .then(data => {
        if (data.length === 0) {
            portfolioGrid.innerHTML = "<p>Aucune réalisation à afficher pour le
moment.</p>";
            return;
        }
    })
```

Réalisation Front-End (Dynamique) (2/2)

- Explications : Comment JavaScript affiche les données de PHP ?

- **JS Demande (Fetch)**

Le JavaScript utilise `fetch()` pour appeler une URL PHP (ex: `/api/projets.php`) de manière **asynchrone** (sans bloquer le navigateur).

- **PHP Répond (JSON)**

Le PHP contacte la base de données, prépare les informations demandées et les renvoie sous forme de **JSON** (un format texte simple que JS comprend).

- **JS Construit & Affiche (DOM)**

Le JavaScript reçoit le JSON, le transforme en objet utilisable.

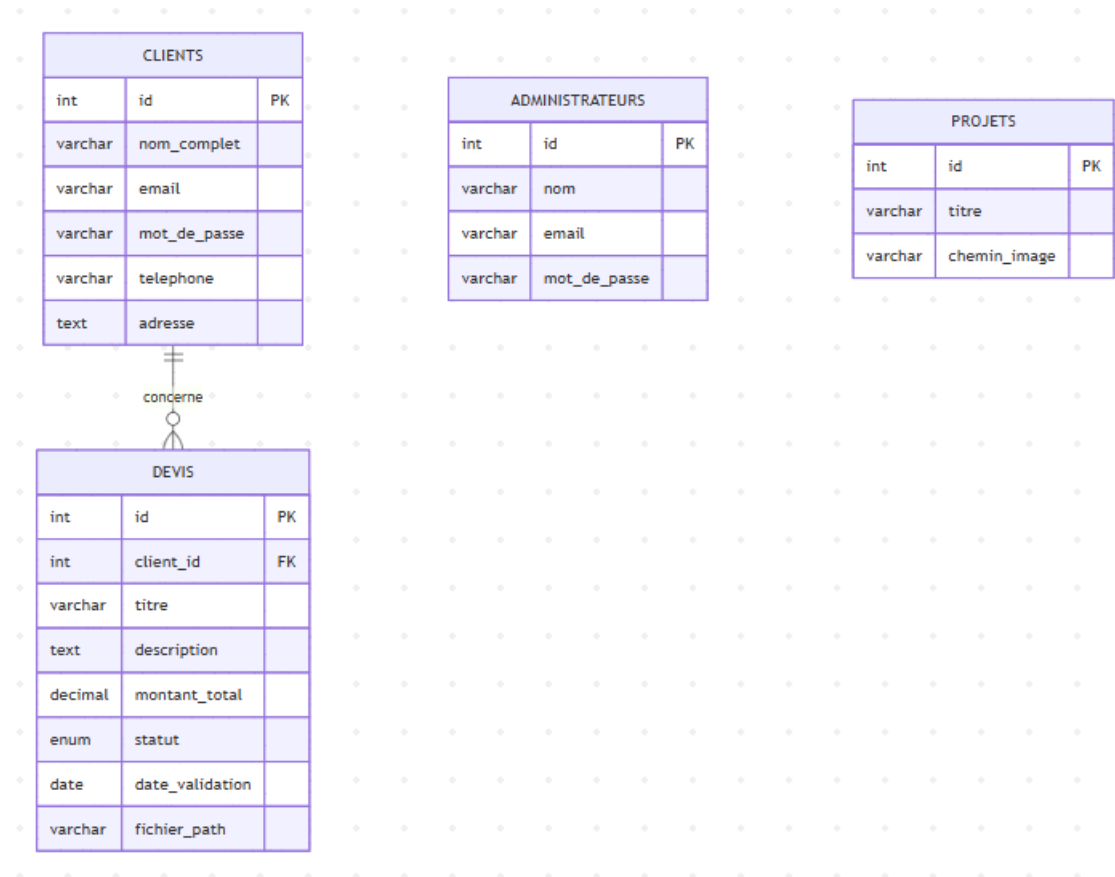
Il **boucle** sur ces données pour créer dynamiquement du code **HTML** (souvent avec `` ${donnée} ``).

Il insère ce nouveau HTML dans la page (avec `element.innerHTML`), mettant à jour l'affichage **sans recharger** la page entière.

Conception Back-End : Bases de données (1/3)

Une base MySQL (Relationnelle) a été conçue pour stocker toutes les données de l'application.

- Visuel :



Conception Back-End (Base de données)

(2/3)

- **Explications :**

La colonne **client_id** dans la table devis sert de **lien** vers la colonne **id** de la table clients. C'est une **clé étrangère**.

Grâce à ce lien, chaque devis est **associé à un client unique**, ce qui garantit que nous savons toujours à qui appartient un devis et permet de retrouver facilement tous les devis d'un client spécifique

Conception Back-End (Bases de données)

(3/3)

- Pour les demandes de contact, une base de données NoSQL de type document (MongoDB) a été choisie. Un message est un **document autonome** : il n'a pas de relations complexes avec d'autres tables. Cette approche offre plus de flexibilité (schema-less) si les champs du formulaire venaient à changer.

Structure d'un document de la collection `demandes_contact` :

```
{
  "_id": "ObjectId('...')"
  "nom": "Jean Dupont (Client existant)",
  "email": "jean.dupont@email.com",
  "message": "Je souhaiterais rénover ma cuisine...",
  "date_soumission": "ISODate('...')"
  "statut": "nouveau"
}
```

Réalisation Back-End : Accès Données (SQL) (1/2)

Toutes les interactions avec la base de données sont effectuées à l'aide de **requêtes préparées** pour prévenir les injections SQL (Sécurité).

- Extrait de code admin/client_actions.php (création nouveau client) :

```
$sql = "INSERT INTO clients (nom_complet, email, telephone,
adresse, mot_de_passe) VALUES (?, ?, ?, ?, ?)";
$stmt = $conn->prepare($sql);
$stmt->bind_param("sssss", $nom_complet, $email,
$telephone, $adresse, $mot_de_passe_hache);
$stmt->execute();
$stmt->close();
header('Location: clients.php');
exit();
}
```

Réalisation Back-End : Accès Données (SQL) (2/2)

- Requête préparée :

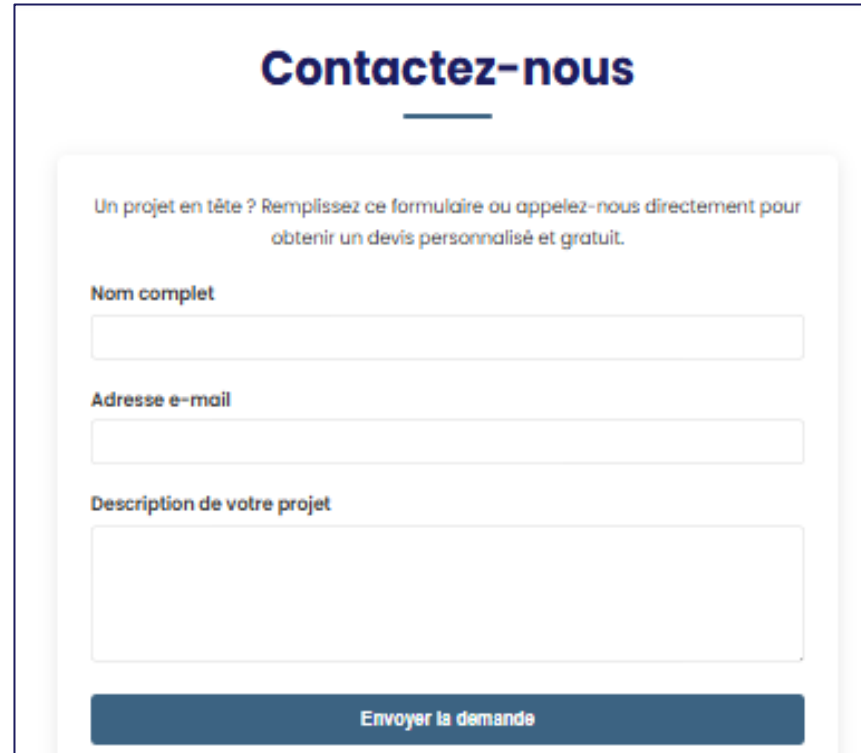
Une requête préparée contre une injection de SQL est une technique où la commande SQL est envoyée au serveur de base de données **séparément** des données de l'utilisateur.

Cela évite qu'un éventuel « pirate » **insère du code SQL malveillant dans les données envoyées par un formulaire** (ex: champ de connexion, recherche) afin de voler des données, modifier des informations ou encore supprimer des tables.

Réalisation Back-End : Accès Données (NoSQL) (1/3)

Composant d'accès NoSQL : Enregistrement d'un message

➤ Visuel du formulaire :

A screenshot of a web form titled "Contactez-nous" in bold dark blue text, underlined. Below the title is a light gray box containing the text: "Un projet en tête ? Remplissez ce formulaire ou appelez-nous directement pour obtenir un devis personnalisé et gratuit." The form has three input fields: "Nom complet" (a single-line text box), "Adresse e-mail" (a single-line text box), and "Description de votre projet" (a multi-line text area). At the bottom of the form is a dark blue button with the white text "Envoyer la demande".

Contactez-nous

Un projet en tête ? Remplissez ce formulaire ou appelez-nous directement pour obtenir un devis personnalisé et gratuit.

Nom complet

Adresse e-mail

Description de votre projet

Envoyer la demande

Réalisation Back-End : Accès Données (NoSQL) (3/3)

Composant d'accès NoSQL : Enregistrement d'un message

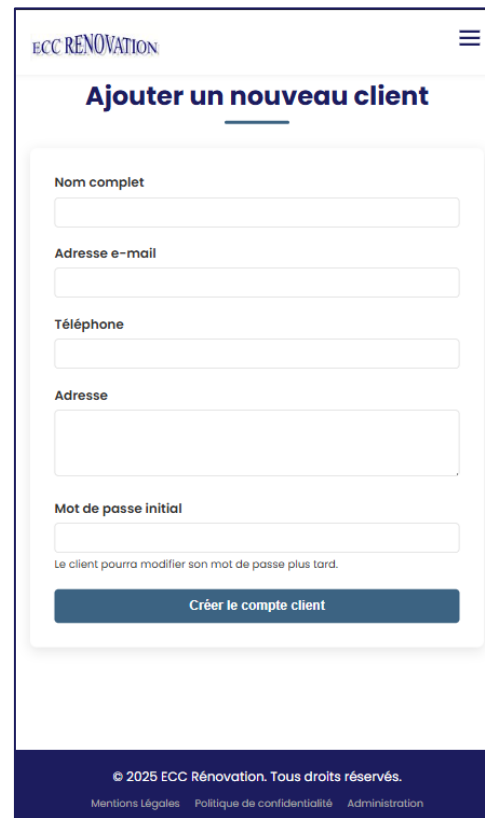
➤ Extrait de Code simplifié de `api/submit_contact.php`:

```
<?php
require_once '../config/mongodb.php'; // On charge la connexion
MongoDB et l'autoloader de Composer

// ...Validation des données...
try {
    // On insère un "document" dans la collection
    $insertResult = $collectionMessages->insertOne([
        'nom' => $nom,
        'email' => $email,
        'message' => $message,
        'date_soumission' => new
MongoDB\BSON\UTCDateTime(), // Format de date NoSQL
        'statut' => 'nouveau'
    ]);
```

Réalisation Back end (Métier) : CRUD de l'élément « Client » - CREATE (1/2)

➤ Visuel (Version Mobile): Page admin/client_form.php



The screenshot displays a mobile application interface for 'ECC RENOVATION'. At the top, the header includes the company logo and a hamburger menu icon. Below the header, the title 'Ajouter un nouveau client' is centered. The main content area contains a form with five input fields: 'Nom complet', 'Adresse e-mail', 'Téléphone', 'Adresse', and 'Mot de passe initial'. A small note below the password field states 'Le client pourra modifier son mot de passe plus tard.' At the bottom of the form is a dark blue button labeled 'Créer le compte client'. The footer of the page contains the copyright notice '© 2025 ECC Rénovation. Tous droits réservés.' and three links: 'Mentions Légales', 'Politique de confidentialité', and 'Administration'.

ECC RENOVATION

Ajouter un nouveau client

Nom complet

Adresse e-mail

Téléphone

Adresse

Mot de passe initial

Le client pourra modifier son mot de passe plus tard.

Créer le compte client

© 2025 ECC Rénovation. Tous droits réservés.

Mentions Légales Politique de confidentialité Administration

Réalisation Back end (Métier) : CRUD de l'élément « Client » - CREATE (2/2)

➤ Extrait de code `admin/client_action.php`

```
if (isset($_POST['action']) && $_POST['action'] == 'add') {  
    $nom_complet = trim($_POST['nom_complet']);  
    $email = trim($_POST['email']);  
    $telephone = trim($_POST['telephone']);  
    $adresse = trim($_POST['adresse']);  
    $mot_de_passe = $_POST['mot_de_passe'];  
    // ...Validation des données, hachage mot de passe  
    $sql = "INSERT INTO clients (nom_complet, email, telephone, adresse,  
        mot_de_passe) VALUES (?, ?, ?, ?, ?)";  
    $stmt = $conn->prepare($sql);  
    $stmt->bind_param("sssss", $nom_complet, $email, $telephone, $adresse,  
        $mot_de_passe_hache);  
    $stmt->execute();  
    $stmt->close();  
}
```

Réalisation Back end (Métier) : CRUD de l'élément « Client » - READ (1/2)

➤ Visuel (Version Mobile): Page admin/client.php

The screenshot shows a mobile application interface for 'ECC RENOVATION'. The title is 'Gestion des Clients'. There is a search bar with the text 'cantais', a 'Rechercher' button, and an 'Ajouter un client' button. Below this, it says 'Résultats pour "cantais"'. A table displays client information: 'Nom' (Julien Cantais), 'Email' (julien.cantais84@gmail.com), and 'Téléphone'. The 'Actions' row contains a 'Modifier' button, a 'Créer devis' button, and a 'Supprimer' link. The footer includes copyright information and links for 'Mentions Légales', 'Politique de confidentialité', and 'Administration'.

Résultats pour "cantais"	
Nom	Julien Cantais
Email	julien.cantais84@gmail.com
Téléphone	
Actions	Modifier Créer devis Supprimer

Réalisation Back end (Métier) : CRUD de l'élément « Client » - READ (2/2)

➤ Extrait de code admin/client.php

```
// Démarrage de session, Connexion BDD, gardien de sécurité

$clients = []; // Initialisation d'un tableau vide pour les clients et du terme
de recherche
$search_term = '';

if (isset($_GET['search'])) {
    if (!empty($search_term)) { // Requête préparée
        $sql = "SELECT id, nom_complet, email, telephone FROM clients WHERE
nom_complet LIKE ? ORDER BY nom_complet ASC";
        $stmt = $conn->prepare($sql);
        $search_pattern = "%" . $search_term . "%"; // Binding
        $stmt->bind_param("s", $search_pattern);
        $stmt->execute(); // Execution de la requête
        $result = $stmt->get_result(); // Récupération des résultats
        $stmt->close(); // fermeture de la requête
```

Réalisation Back end (Métier) : CRUD de l'élément « Client » - UPDATE (1/2)

➤ Visuel (Version Mobile): Page admin/client_edit_form.php

The screenshot shows a mobile web interface for editing a client. At the top, the 'ECC RENOVATION' logo is on the left and a hamburger menu icon is on the right. Below the header, the title 'Modifier le client : Julien Cantais' is centered. The main content area contains a form with the following fields: 'Nom complet' (pre-filled with 'Julien Cantais'), 'Adresse e-mail' (pre-filled with 'julien.cantais94@gmail.com'), 'Téléphone' (empty), 'Adresse' (empty), and 'Nouveau mot de passe (optionnel)' (empty). A note below the password field states: 'Laissez vide pour ne pas changer le mot de passe.' At the bottom of the form is a dark blue button labeled 'Enregistrer les modifications'. The footer of the page is dark blue and contains the copyright notice '© 2025 ECC Rénovation. Tous droits réservés.' and three links: 'Mentions Légales', 'Politique de confidentialité', and 'Administration'.

ECC RENOVATION

Modifier le client : Julien Cantais

Nom complet

Julien Cantais

Adresse e-mail

julien.cantais94@gmail.com

Téléphone

Adresse

Nouveau mot de passe (optionnel)

Laissez vide pour ne pas changer le mot de passe.

Enregistrer les modifications

© 2025 ECC Rénovation. Tous droits réservés.

Mentions Légales Politique de confidentialité Administration

Réalisation Back end (Métier) : CRUD de l'élément « Client » - UPDATE (2/2)

➤ Extrait de code

admin/client_action.php

```
// Vérifie que le formulaire a été envoyé avec l'action "edit"
if (isset($_POST['action']) && $_POST['action'] == 'edit') {
    $client_id = intval($_POST['client_id']);
    $nom_complet = trim($_POST['nom_complet']);
    $email = trim($_POST['email']);
    $telephone = trim($_POST['telephone']);
    $adresse = trim($_POST['adresse']);
    $mot_de_passe = $_POST['mot_de_passe']; // Récupération des données
// ... Validation des données
if (!empty($mot_de_passe)) {
    $mot_de_passe_hache = password_hash($mot_de_passe, PASSWORD_DEFAULT);
    $sql = "UPDATE clients SET nom_complet = ?, email = ?, telephone = ?, adresse
= ?, mot_de_passe = ? WHERE id = ?";
    $stmt = $conn->prepare($sql);
    $stmt->bind_param("ssssi", $nom_complet, $email, $telephone, $adresse,
    $mot_de_passe_hache, $client_id);
} else {
    $sql = "UPDATE clients SET nom_complet = ?, email = ?, telephone = ?, adresse
= ? WHERE id = ?";
    $stmt = $conn->prepare($sql);
    $stmt->bind_param("ssssi", $nom_complet, $email, $telephone, $adresse,
    $client_id);
    $stmt->execute();
    $stmt->close();
}
```

Réalisation Back end (Métier) : CRUD de l'élément « Client » - DELETE (1/2)

➤ Visuel (Version Mobile): Page admin/client.php

ECC RENOVATION

Gestion des Clients

cantais Rechercher Ajouter un client

Résultats pour "cantais"

Nom	Julien Cantais
Email	julien.cantais84@gmail.com
Téléphone	
Actions	Modifier Créer devis Supprimer

© 2025 ECC Rénovation. Tous droits réservés.
[Mentions Légales](#) [Politique de confidentialité](#) [Administration](#)

Réalisation Back end (Métier) : CRUD de l'élément « Client » - DELETE (2/2)

➤ Extrait de code admin/client_action.php

```
if (isset($_GET['action']) && $_GET['action'] == 'delete') {  
    $id = intval($_GET['id']); // Récupère l'ID depuis l'URL et  
    le force à être un nombre entier. (Sécurité)  
  
    $sql = "DELETE FROM clients WHERE id = ?";  
    $stmt = $conn->prepare($sql); // Préparation de la requête  
    $stmt->bind_param("i", $id);  
    $stmt->execute(); // Exécution de la requête  
    $stmt->close(); // Fermeture de la requête
```

Sécurité (1/2)

La sécurité a été une priorité à chaque étape du développement :

- Protection contre les injections de SQL, avec l'utilisation systématique des requêtes préparées pour toutes les requêtes SQL.
- Protection contre les failles XSS (Cross-Site Scripting) avec l'utilisation de la fonction `htmlspecialchars()` avant tout affichage de données provenant de la base de données (noms de clients, messages, titres de devis, etc.)
- Sécurité des mots de passe via l'hachage systématique avec l'algorithme BCrypt (`password_hash()`) et la vérification avec `password_verify()`.

Sécurité (2/2)

- Gestion des accès grâce aux sessions PHP (\$_SESSION) pour séparer les droits.

Des "gardiens de sécurité" sont placés au début de chaque page admin et client pour vérifier si l'utilisateur est connecté et a le bon rôle.

- Sécurisation des identifiants : Les identifiants de la base de données ne sont pas versionnés sur Git. Ils sont stockés dans un fichier .env local, qui est lu par PHP mais ignoré par Git grâce au fichier .gitignore.
- Prise en compte de l'**accessibilité (RGAA)** via l'HTML sémantique et du **RGPD** avec la création d'une page de politique de confidentialité et la sécurisation des données.

Tests et Veille (1/2)

Jeux d'essai : l'acceptation d'un devis par un client.

Scénario	Données en entrée	Résultat Attendu	Résultat Obtenu	Analyse
1. Acceptation	Client connecté, clique sur "Accepter" (Devis ID 1, Statut = 'En attente')	Le statut du devis 1 passe à 'Accepté' dans la BDD.	Le statut est mis à jour à 'Accepté'.	Conforme
2. Refus	Client connecté, clique sur "Refuser" (Devis ID 2, Statut = 'En attente')	Le statut du devis 2 passe à 'Refusé'.	Le statut est mis à jour à 'Refusé'.	Conforme
3. Paiement (simulé)	Client connecté, clique sur "Payer" (Devis ID 1, Statut = 'Accepté')	Le statut du devis 1 passe à 'Payé'.	Le statut est mis à jour à 'Payé'.	Conforme
4. Accès non autorisé	Client B tente d'accéder à voir_devis.php?id=1 (Devis du Client A)	Redirection vers le dashboard.php du Client B.	Redirection vers le dashboard.php.	Conforme

Tests et Veille (2/2)

Veille :

Durant le projet, une veille a été effectuée sur les vulnérabilités de sécurité courantes :

- **Veille de Sécurité (OWASP)** : Consultation du Top 10 (Broken Access Control, Injection, Authentication Failures). **Documentation officielle de PHP** sur la sécurité (notamment les chapitres sur les requêtes préparées et la prévention XSS).
- **Veille Technologique (Docs, STUDI)** : M'a orienté vers l'architecture de **persistance polyglotte (SQL+NoSQL)** et l'adoption de **Docker** pour un environnement reproductible.
- **Outils d'Assistance (IA)** : Utilisé comme "partenaire de réflexion" pour valider l'architecture et accélérer le débogage (conflits de port Docker, erreurs "headers already sent")

Conclusion

Le projet répond aux objectifs et démontre la maîtrise de concepts avancés.

Points forts démontrés :

- ✓ Architecture de persistance polyglotte (SQL + NoSQL).
- ✓ Application entièrement conteneurisée avec Docker. Sécurité rigoureuse (requêtes préparées, htmlspecialchars, hachage, .env).
- ✓ Code maintenable (SCSS Partials, BEM, Workflow Git).
- ✓ Projet fonctionnel, sécurisé et prêt à évoluer (Stripe, notifications...).

Dossier Projet

ECC RENOVATION

Merci pour votre attention