BTS SIO – Option SLAM
Documentation d'épreuve

Mission 5 : Interface de recherche immobilière

Par Baptiste Grimaldi

Ce document est fourni en complément de la fiche E5 correspondante.

E4-Mission5-Interface-recherche-immobilier-Baptiste-Grimaldi

Liens utiles:

Article Interface de recherche immobilière

Check out the portfolio of Baptiste Grimaldi, a full-stack web developer specializing in creating dynamic and responsive websites. Explore his projects and contact him for your next web development project.

https://portfolio.baptistegrimaldi.info/projects/view/interface-de-recherche-immobilere

Article de mon portfolio

SOMMAIRE

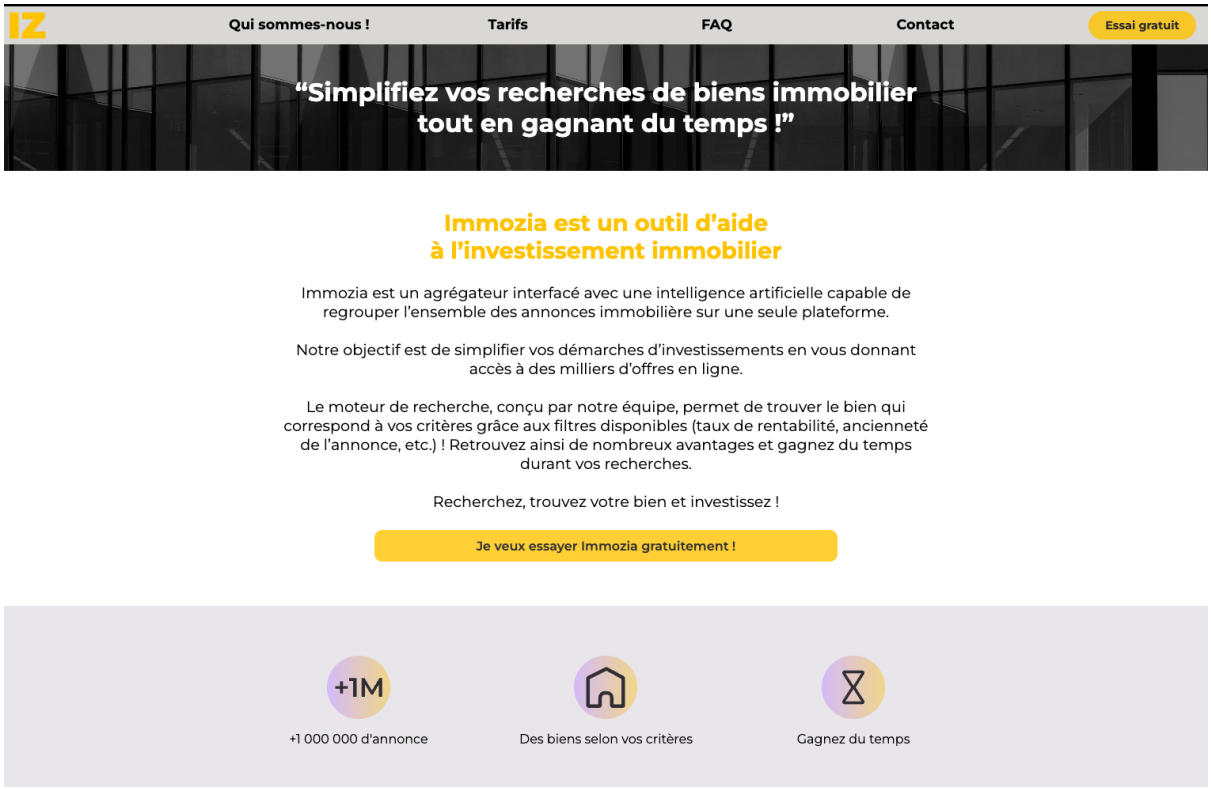
1. Cahier des charges

1.1 Introduction

Dans ce document, nous allons définir les spécifications pour la création d'une interface web en PHP, HTML et JS permettant la recherche de biens immobiliers.

Dans le cadre de mon alternance de deuxième année de BTS SIO option SLAM, j’acompagne un nouveau projet de l’entreprise Oxianet: Immozia.

Immozia est un agrégateur de bien immobiliers. le système parcours le web pour être au courant des annonces le plus vite possible et vous permet d’investir au meilleur moment sur les biens d’une zone.



Landing page actuelle, le reste est en développement et n’est pas publique.

1.2 Expression fonctionnelle du besoin

1.3 Contraintes

Les contraintes pour ce projet incluent :

- L'utilisation des langages de programmation PHP, HTML et JS
- La base de données doit être MySQL
- L'interface web doit être responsive et accessible sur différents appareils
- L'interface web doit être sécurisée et protéger les données des utilisateurs

Les objectifs de la plateforme

- Un gain de temps : Fini les longues heures de recherches à écumer les différents site web.
- Immozia vous permet de gagner du temps en regroupant sur une seule et même plateforme les annonces disponibles sur le marché.

- Des critères poussés: Toujours dans l'objectif de faciliter vos recherches, nous avons pensé les critères de recherche pour que vous puissiez trouver votre bien en seulement quelques clics.
- Des annonces pertinentes : Vous l'avez peut être constaté, en fonction des plateformes, les descriptions peuvent être plus ou moins complète. Ne perdez plus de temps à rassembler les informations, Immozia le fait pour vous.

1.4 Gestion des droits d'accès

La question des droit d’accès est encore un peux tôt pour se poser à cette étape du projet mais il est quand même important de savoir quelle direction prendre.
Le but sera que les droits d’accès changent en fonction de l’abonnement de l’utilisateur.

Aucun diagramme UML n’a encore été fait mais c’est une chose à faire encore.

Malgré tout, une gestion des droits est déjà en place pour le pôle communication de l’entreprise. Une interface de mise en ligne de nouveau articles pour le site type CMS maison à été codé par mes soins. Voir plus.

2. Description des environnements

Le projet sera développé et testé sur les environnements suivants :

- Environnement de développement local
- Environnement de pré-production
- Environnement de production

2.1 Environnement de développement local

L'environnement de développement local pour ce projet est un XAMPP avec accès à une base de données de développement via l'interface de PHPStorm.

> 💡 XAMPP est un environnement de développement web qui permet de créer un serveur web local pour tester et développer des sites web. Il est composé des logiciels suivants: Apache, MySQL, PHP et Perl.

> 💡 PHPStorm est un environnement de développement intégré (IDE) pour PHP. Il offre des fonctionnalités telles que la coloration syntaxique, la correction automatique, le débogage, la navigation de code et la gestion de version. Il est également livré avec une intégration Git et une base de données MySQL pour faciliter la gestion du développement de projet.

2.2 Environnement de pré-production

L'environnement de pré-production est une copie de l'environnement de production pour tester les mises à jour et les nouvelles fonctionnalités avant leur déploiement.

Cet environnement est utilisé pour les tests. Lors de la phase de test, on utilise une instance du projet étant comme en production mais dans un environnement de test.

Lors de la phase de test en développement web, les développeurs utilisent une instance du projet qui est similaire à celle qui sera déployée en production, mais dans un environnement de test. Cette étape est cruciale pour s'assurer que toutes les nouvelles fonctionnalités et les mises à jour de l'interface web fonctionnent correctement et répondent aux exigences spécifiées avant qu'elles ne soient déployées pour les utilisateurs finaux.

Les tests en développement permettent de détecter et de corriger les erreurs avant que l'interface web ne soit déployée en production et disponible pour les utilisateurs finaux. Les développeurs effectuent des tests sur toutes les fonctionnalités de l'interface web pour s'assurer qu'elles fonctionnent correctement. Ils vérifient également que l'interface web est conviviale et facile à utiliser pour l'utilisateur final.

Les tests en développement sont essentiels pour assurer la qualité de l'interface web et minimiser les erreurs qui pourraient nuire à l'expérience utilisateur. Ces tests permettent également de s'assurer que l'interface web est conforme aux exigences et aux spécifications décrites dans le cahier des charges. Tout écart par rapport aux exigences spécifiées est détecté et corrigé avant le déploiement de l'interface web en production.

En somme, la phase de test en développement web est une étape importante dans le processus de développement d'une interface web. Elle permet de s'assurer que toutes les fonctionnalités sont conformes aux exigences spécifiées et qu'elles fonctionnent correctement avant le déploiement en production. Les tests en développement sont effectués pour minimiser les erreurs et assurer l'expérience utilisateur optimale.

2.3 Environnement de production

L'environnement de production est l'environnement final pour le projet, où l'interface web sera déployée et disponible pour les utilisateurs finaux.

Cet environnement n’est jamais directement modifié. C’est l’espace d’action directe avec les utilisateurs finaux.

3. Méthodologie

3.1 Méthodologie et versioning

Bonnes pratiques

- ne commitez que des choses sur les mêmes sujets (style, front, back, etc...)
- Si vous ne pouvez pas écrire de messages de commit concis, cela indique trop de sujets dans le même commit.
- Utilisez un titre et un corps avec seulement la commande commit et en ajoutant une ligne vide entre le titre et le corps.

> git commit
> ::
> TITRE
>
> CORPS
> \# rest comments

Stratégies de branching

- Une convention écrite pour organiser l'équipe.

Deux options principales:

1. Développement **Mainline**:
    - **quelques** branches
    - commits relativement petits
    - normes de test de haute qualité
    -

2. Branches **State**, **Release** et **Feature**
    - **Deux** types de branches différents qui remplissent des types de travail différents
        1. **Long**Running
            - existe tout au long de la vie du projet
            - souvent, ils reflètent les "étapes" de votre cycle de vie de développement
        2. **Short** Running
            - pour les nouvelles fonctionnalités, les corrections de bogues, le refactoring, les expériences
            - sera supprimé après l'intégration (fusion/rebase)

Deux exemples de stratégies de branching

1. GitHub Flow
    - très simple, très léger : seulement long-running
    - branche ("main") + branches de fonctionnalités
2. GitFlow
    - plus de structure, plus de règles
    - long-running : "main" + "develop"
    - de courte durée : fonctionnalités, versions, correctifs

Pour la gestion de version de ce projet, nous avons opté pour une stratégie de branching avec un main, un dev et des branches pour les fonctionnalités. Cette méthode de versioning permet de travailler sur plusieurs fonctionnalités en parallèle tout en minimisant les conflits de code.

L'utilisation de branches pour les fonctionnalités permet de travailler sur des fonctionnalités spécifiques de manière isolée, sans perturber le code principal de l'application. Chaque branche est associée à une fonctionnalité spécifique et est testée avant d'être fusionnée dans la branche de développement (dev).

La branche dev est la branche principale sur laquelle l'équipe travaille. Les fonctionnalités développées dans des branches distinctes sont fusionnées dans la

branche dev une fois qu'elles ont été testées et validées. Cela permet de s'assurer que toutes les fonctionnalités sont en mesure d'interagir correctement les unes avec les autres.

En outre, toutes les fonctionnalités sont vérifiées par une série de tests documentés ainsi que par d'autres développeurs, afin de s'assurer qu'elles répondent aux exigences spécifiées dans le cahier des charges. Cette approche contribue à minimiser les erreurs et à garantir la qualité de l'interface web.

En résumé, la stratégie de versioning avec un main, un dev et des branches pour les fonctionnalités est efficace pour travailler sur plusieurs fonctionnalités en parallèle tout en minimisant les conflits de code. Les fonctionnalités sont testées avant d'être fusionnées dans la branche principale, ce qui permet de garantir la qualité de l'interface web.

3.2 Gestion des tests de la solution

> 💡 Le test est une phase importante dans le développement web car il permet de s'assurer que le site web ou l'application web fonctionne correctement et répond aux exigences et spécifications. Pendant la phase de test, les développeurs et les testeurs identifient et corrigent les défauts ou bugs dans le code, et vérifient que le site web ou l'application web fonctionne comme prévu.

Types de phases de test :

Il existe plusieurs types de tests qui peuvent être effectués pendant la phase de test du développement web, notamment:

1. Test unitaire : Ce type de test se concentre sur les unités ou les composants individuels du code, et est généralement effectué par les développeurs lorsqu'ils écrivent et déboguent le code.
2. Test d'intégration : Ce type de test vérifie que les différents composants ou modules du code fonctionnent correctement ensemble.
3. Test système : Ce type de test consiste à tester l'ensemble du système, y compris tous les composants et modules, pour s'assurer qu'il fonctionne correctement.
4. Test d'acceptation : Ce type de test est généralement effectué par le client ou l'utilisateur final pour vérifier que le site web ou l'application web répond à leurs besoins et exigences.

Il est important de tester soigneusement un site web ou une application web avant sa sortie au public, pour s'assurer qu'il est fiable et fonctionne comme prévu.

Exemple de test d'intégration

Le test d'intégration est un type de test qui vérifie que différents composants ou modules d'un système fonctionnent correctement ensemble.

Voici un exemple de test d'intégration dans un projet de développement web:

> 💡 Imaginez que vous développez une application web qui permet aux utilisateurs de rechercher et d'acheter des produits en ligne. L'application comporte plusieurs composants, notamment une fonction de recherche, un panier d'achat et une passerelle de paiement. Pendant le test d'intégration, vous testerez les interactions entre ces différents composants pour vous assurer qu'ils fonctionnent correctement ensemble.

Exemples de scénarios:

Par exemple, vous pouvez tester les scénarios suivants :

1. Recherche d'un produit : Vous vérifieriez que la fonction de recherche est capable de récupérer et d'afficher les résultats corrects en fonction de la requête

de l'utilisateur.

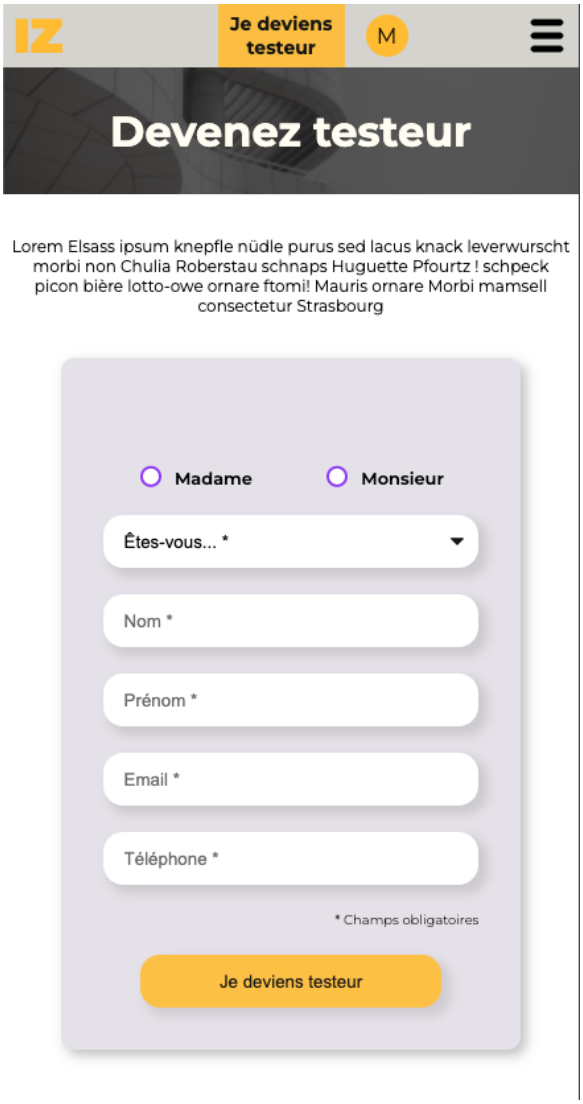
2. Ajout d'un produit au panier : Vous vérifieriez que le panier d'achat est capable d'ajouter le produit correct, et que le prix et la quantité sont affichés correctement.
3. Validation de la commande et paiement : Vous vérifieriez que la passerelle de paiement est capable de traiter correctement le paiement, et que la commande est enregistrée dans le système.

En testant ces scénarios, vous pouvez vous assurer que les différents composants de l'application web fonctionnent parfaitement ensemble et que l'expérience utilisateur est fluide et intuitive.

Exemple de formulaire de test

Ceci est un exemple de test pour Immozia.

Vérification du nombre de champs et de leurs label :

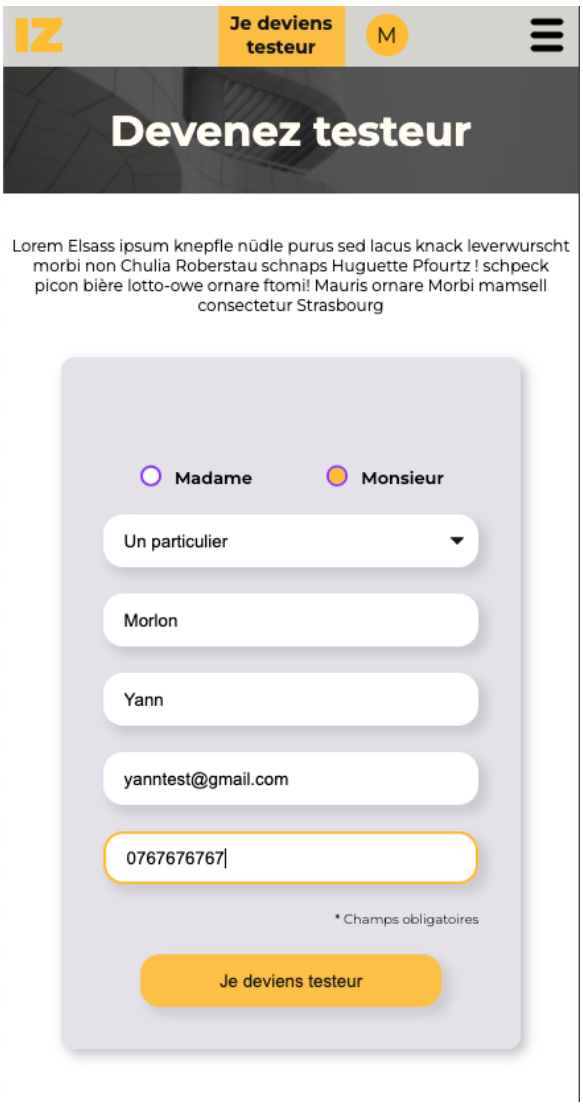


Les champs sont bien tous renseignés, la liste des champs voulu étant :

- ☑ ~~Le sexe~~
- ☑ ~~Le type de personne (particulier ou professionnel)~~
- ☑ ~~Le nom~~
- ☑ ~~Le prénom~~
- ☑ ~~Le mail~~
- ☑ ~~Et le numero de téléphone~~

Test d’envoi avec données vérifiées :

L’image ci-dessous montre un formulaire rempli avec des données valable.



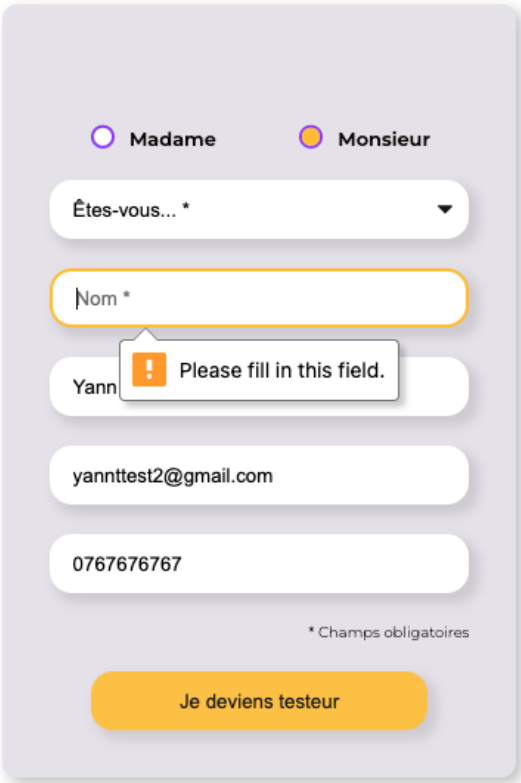
L’image ci-dessous nous montre un message de confirmation de l’envoi des données.

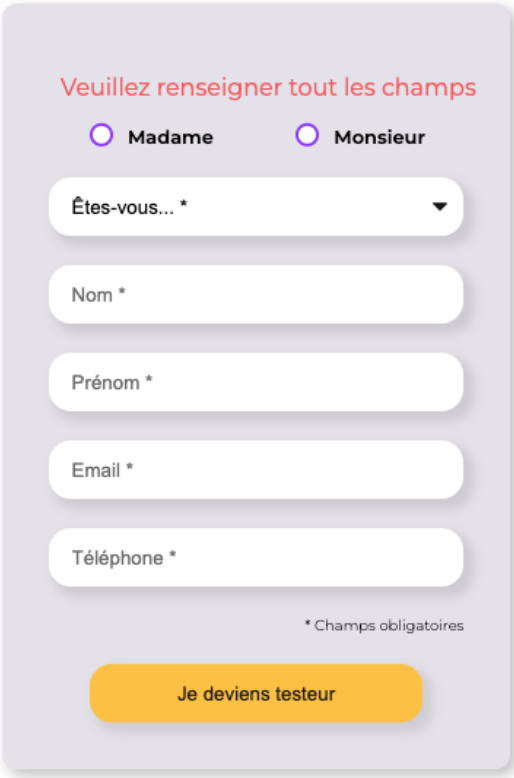
L’image ci-dessous nous montre bien que la base de données a bien reçu les informations rentrées.

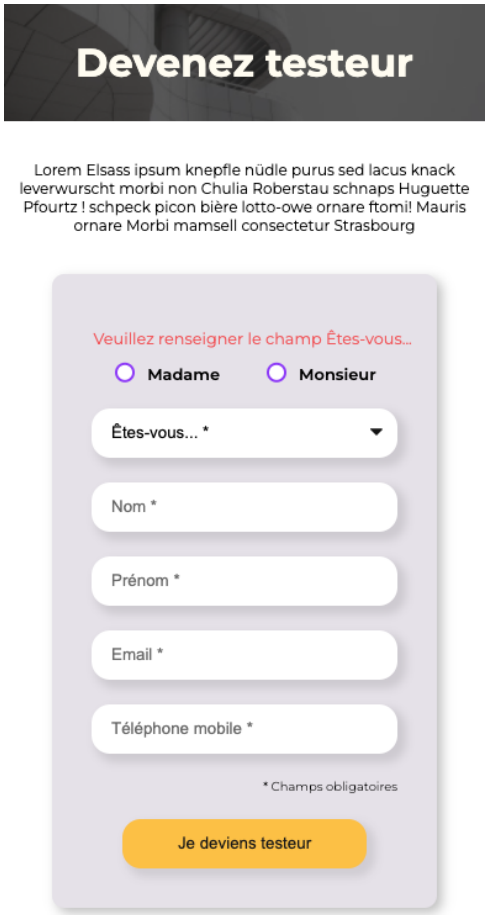
Controle des erreurs :

Plusieurs erreur peuvent survenir voici la liste :

1. Champ non rempli

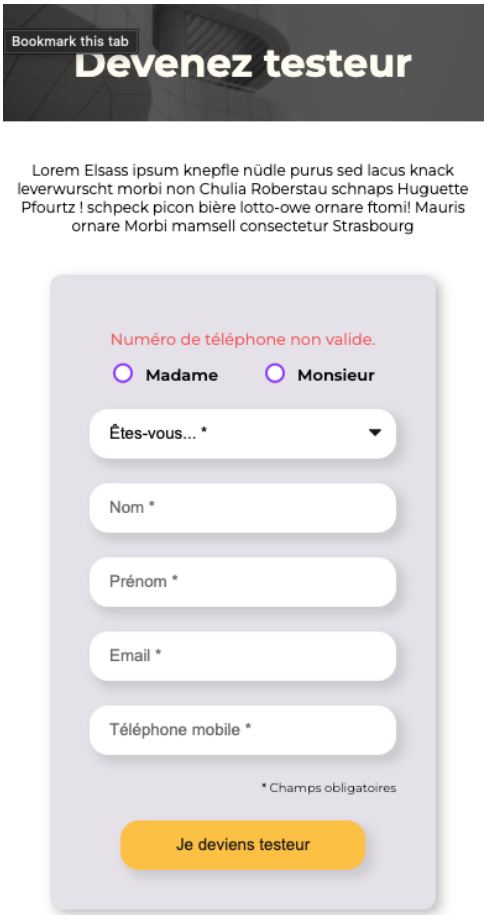






2. Champ mal rempli


Devenez testeur
Lorem Elsass ipsum knepfle nüdle purus sed lacus knack leverwurscht morbi non Chulia Roberstau schnaps Huguette Pfourtz ! schpeck picon bière lotto-owe ornare ftomi! Mauris ornare Morbi mamsell consectetur Strasbourg
Le nom ou prénom ne doit pas contenir de chiffre.
Madame
Monsieur
Êtes-vous... *
Nom *
Prénom *
Email *
Téléphone mobile *
* Champs obligatoires
Je deviens testeur



Resultat du test:

✅ ~~Le formulaire est correctement affiché et tous les champs sont présents et correctement étiquetés.~~

✅ ~~Les données valides sont enregistrées correctement et un message de confirmation est affiché.~~

✅ ~~Les erreurs de saisie et les champs obligatoires manquants sont signalés par des messages d'erreur.~~

3.3 Rédaction de la documentation

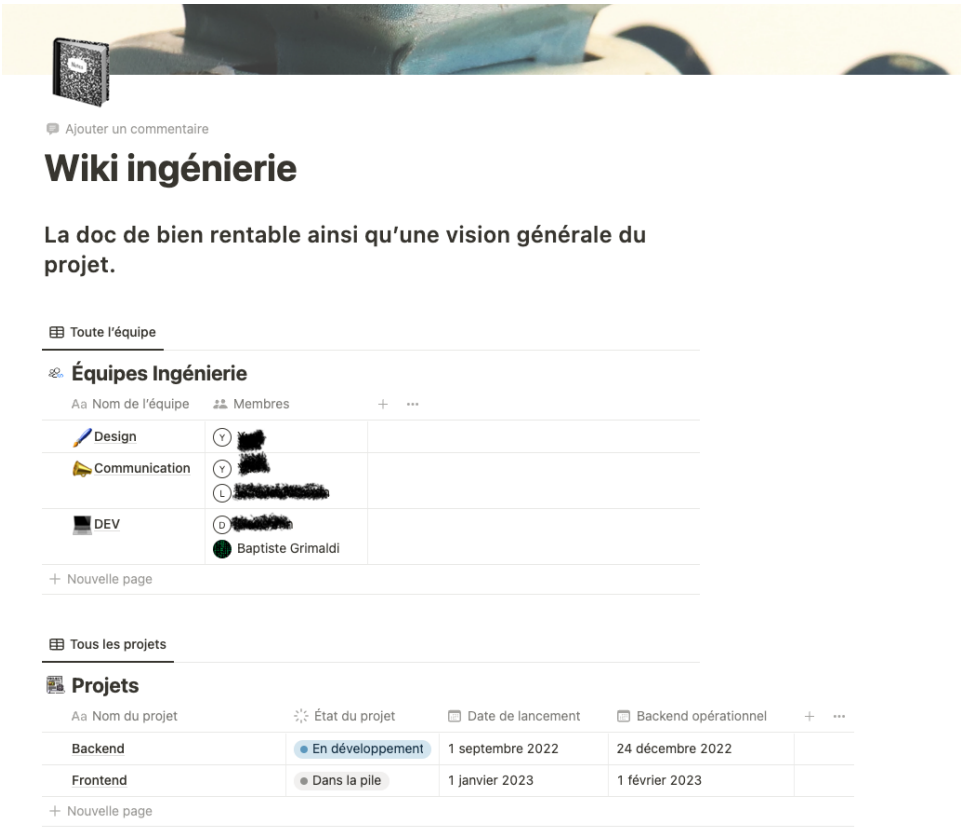
La documentation est maintenu sur un notion de groupe. Cette année, j’ai mis en place un notion pour toute l’équipe. C’est un espace partagé avec des pages pour tout les aspect de l’organisation du projet immozia.

Après chaque phase de développement, la documentation adéquate est rédigée sur le wiki ingénieur du Notion. Cette documentation inclut des informations sur les fonctionnalités développées, les modifications apportées au code et les tests effectués. Elle est organisée de manière cohérente et facile à naviguer, de sorte que l'équipe de développement puisse facilement trouver les informations dont elle a besoin.

Le wiki ingénieur du Notion est un espace partagé pour l'équipe de développement où sont stockées toutes les informations liées au projet. Il s'agit d'un outil de collaboration qui permet à chaque membre de l'équipe de contribuer à la documentation et de la mettre à jour au fur et à mesure que le projet avance. La documentation est rédigée en utilisant une syntaxe de balisage simple pour faciliter la lecture et la compréhension.

La documentation inclut également des captures d'écran et des exemples de code pour illustrer les fonctionnalités développées et les modifications apportées. Cela permet à l'équipe de développement de comprendre rapidement les changements apportés au code et de voir comment les nouvelles fonctionnalités ont été implémentées.

En somme, la rédaction de la documentation est une étape importante du processus de développement. Elle permet à l'équipe de développement de se tenir informée des modifications apportées au code et des nouvelles fonctionnalités développées, et de garantir que toutes les informations pertinentes sont stockées dans un endroit centralisé et facilement accessible.



Extrait du wiki-ingénieurie

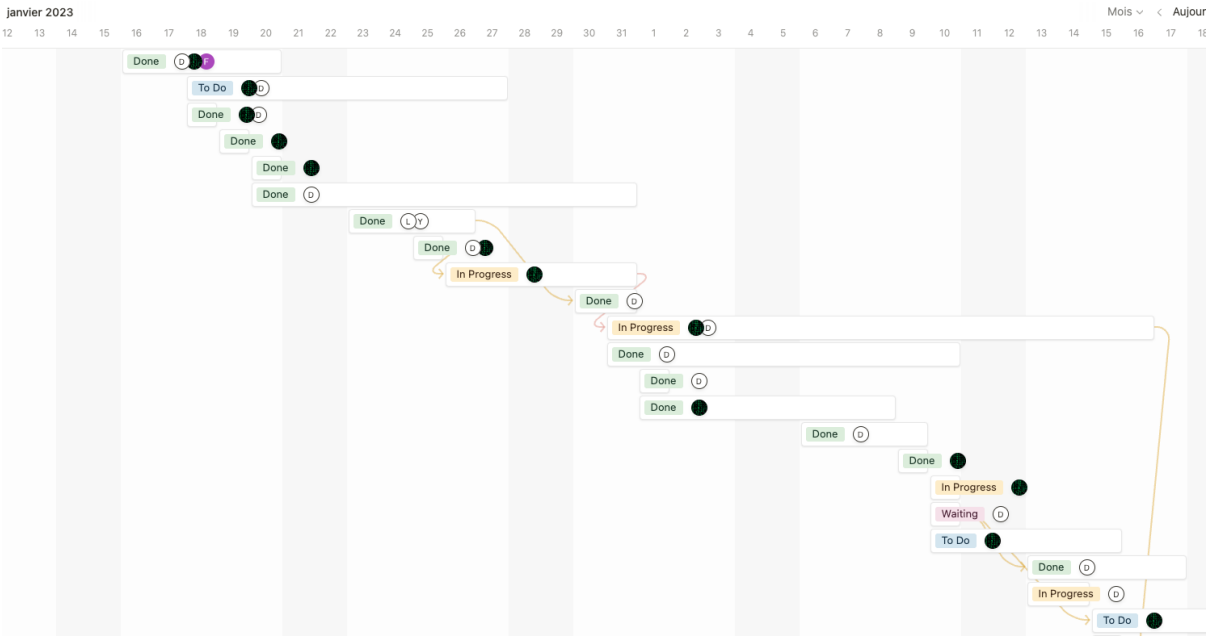
3.4 Gestion de projet

Pour la gestion du projet, l'équipe utilise Notion pour stocker une base de données de tâches. Cette base de données est utilisée pour suivre l'avancement du projet et pour assigner des tâches à différents membres de l'équipe.

Notion permet également d'utiliser différentes visualisations pour la gestion de projet, telles que le tableau Kanban ou le diagramme de Gantt. Le tableau Kanban est utilisé pour suivre l'avancement des tâches, tandis que le diagramme de Gantt est utilisé pour visualiser les dépendances entre les tâches et les jalons du projet.

En utilisant Notion pour la gestion de projet, l'équipe peut travailler de manière collaborative, en temps réel, et avoir une vue d'ensemble de l'avancement du projet. Les mises à jour sont synchronisées en temps réel, ce qui permet à chaque membre de l'équipe de rester informé des dernières modifications apportées au projet.

En somme, Notion est un outil de gestion de projet efficace pour l'équipe de développement d'Immozia. Il permet de stocker une base de données de tâches, de suivre l'avancement du projet et d'utiliser différentes visualisations pour la gestion de projet. En utilisant Notion, l'équipe peut travailler de manière collaborative et avoir une vue d'ensemble de l'avancement du projet.



4. Mise en oeuvre

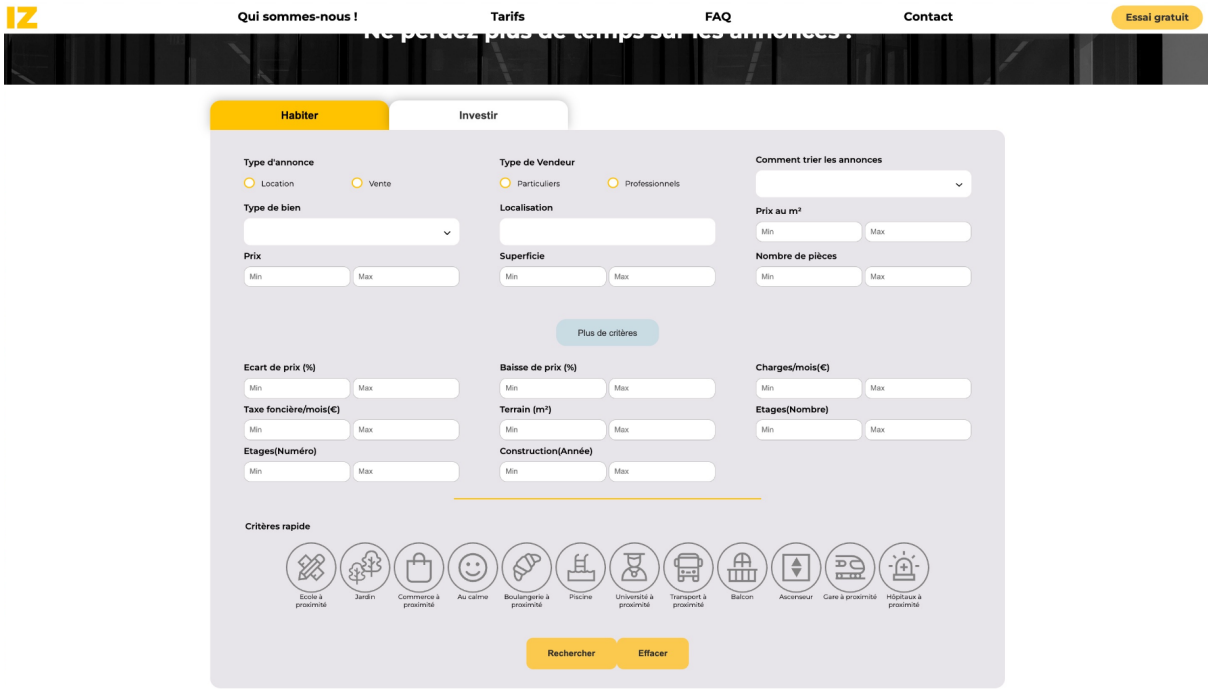
4.1 Conception et développement de l'interface web de recherche immobilière

L’interface en elle même est plutôt simple. Elle reprend les codes de tout les sites de recherches immobiliers en rajoutant plus de précision.

L'interface utilisateur est principalement composée du système de recherche. Il utilise des algorithmes sophistiqués pour filtrer les résultats en fonction des critères spécifiés par l'utilisateur. En plus de cela, il y a également une grande variété d'options de filtrage disponibles pour affiner davantage les résultats. En ce qui concerne l'affichage et la communication des données en php, cela se fait de manière transparente pour l'utilisateur. Les données sont récupérées rapidement et efficacement, puis affichées de manière claire et concise.

Enfin, la mise en forme avec du css permet de donner une apparence attrayante et professionnelle à l'interface utilisateur, tout en améliorant la convivialité globale de l'application.

Première interface



Fonctionnement avec php

Pour boucler sur des données récupérées d'une base de données des biens selon les critères spécifiés par l'utilisateur et les afficher sur l'interface utilisateur, il est recommandé d'utiliser PDO (PHP Data Objects) pour interagir avec la base de données. Cela permet de protéger contre les injections SQL en utilisant des requêtes préparées.

Voici un exemple de code pour récupérer des biens immobiliers à partir d'une base de données MySQL:

```
// Connexion à la base de données
$dsn = 'mysql:host=localhost;dbname=nom_de_la_base_de_donnees';
$user = 'nom_d_utilisateur';
$password = 'mot_de_passe';
$pdo = new PDO($dsn, $user, $password);

// Préparation de la requête SQL pour récupérer les biens
$sql = 'SELECT * FROM biens_immobiliers WHERE type = :type AND ville = :ville';
$stmt = $pdo->prepare($sql);

// Remplacement des paramètres dans la requête SQL
$type = 'appartement';
$ville = 'Paris';
$stmt->bindParam(':type', $type);
$stmt->bindParam(':ville', $ville);

// Exécution de la requête SQL
$stmt->execute();

// Boucle sur les résultats et affichage sur l'interface utilisateur
```

```
while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
    echo $row['nom'] . ' - ' . $row['adresse'] . '
';
}
```

Dans cet exemple, la requête SQL préparée utilise des paramètres nommés (":type" et ":ville") pour spécifier les critères de recherche. Les valeurs de ces paramètres sont remplacées par les variables $type et $ville respectivement, qui sont définies avant l'exécution de la requête.

En utilisant PDO pour interagir avec la base de données, il est facile de protéger contre les injections SQL en utilisant des requêtes préparées. Les données récupérées peuvent ensuite être bouclées et affichées sur l'interface utilisateur en toute sécurité.

⚠️ Notons que dans cette exemple, la connexion à la base de donnée se fait sur le tas. Une meilleure pratique est de gérer la connexion dans un fichier de configuration et de l’appeler quand nécessaire.

Sécurité

⚠️ L'injection SQL est une technique couramment utilisée par les pirates informatiques pour accéder à des données sensibles dans une base de données. Elle consiste à insérer du code SQL malveillant dans une entrée utilisateur, qui est ensuite exécuté par le serveur de base de données.

Pour protéger contre l'injection SQL, il est important de valider et de nettoyer toutes les entrées utilisateur avant de les utiliser dans des requêtes SQL. Les requêtes préparées sont également une bonne pratique pour éviter l'injection SQL, car elles permettent de séparer les données de la requête SQL elle-même.

Optimisation

Quand on parle de millions de données, il est bien compliqué de tout récupéré en une seule fois et de vulgairement enregistrer dans des variable.

- Cela ralenti l’interface utilisateur et réduit la crédibilité de l’entreprise.

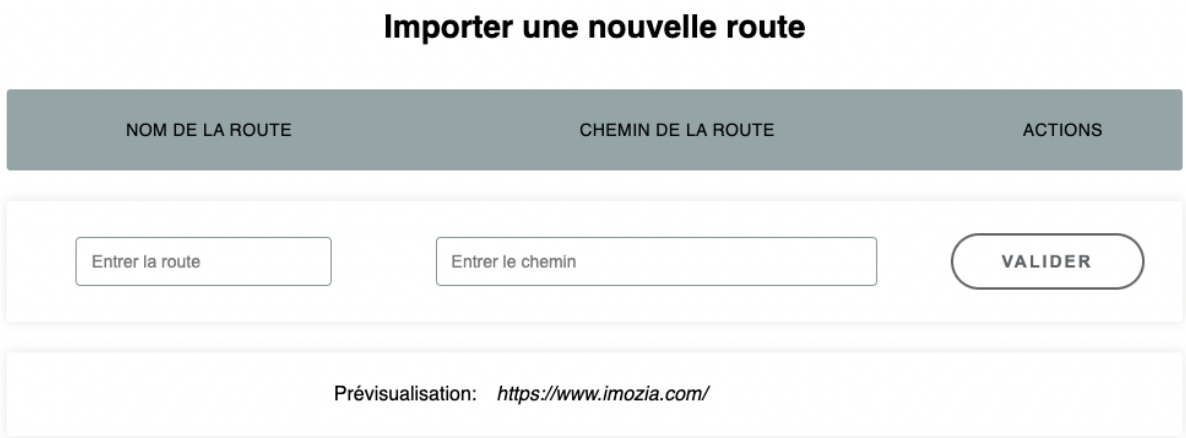
- Php peux nous donner des erreurs car la mémoire tampon de php ne suit plus et est surchargé.

Contre ces problèmes, il existe plusieurs méthodes:

- Pagination : Permet de récupérer les données par blocs, en limitant le nombre de résultats par page.
- Lazy loading : Permet de récupérer les données au fur et à mesure que l'utilisateur fait défiler la page.
- API : Permet de récupérer les données en temps réel via une interface de programmation.

En utilisant ces méthodes, nous avons pu optimiser l'interface web de recherche immobilière pour gérer efficacement un grand nombre de données sans ralentir l'interface utilisateur. Cela permet également de garantir la crédibilité de l'entreprise en fournissant des résultats de recherche précis et rapides.

4.2 L’interface de management des articles immozia.



Une partie de l’interface d’import de nouveaux articles.

Backend

Introduction

Le système d'authentification fonctionne avec le "basic auth" d'apache.

La structure

La structure du backend est la suivante :

▼ 0-com-manager

▼ public

> 💡 La route publique est l'ensemble des ressources auxquelles l'utilisateur peut accéder directement.

▼ views

> 💡 Les "Vues" / pages envoyées à l'utilisateur.

▼ .htaccess

> 💡 Le fichier qui définit le système d'authentification et de routage.

Index.html

Système d'authentification

```
AuthType Basic
AuthName "Protected Area"
AuthUserFile /home/immo/vault/.htpasswd
Require valid-user
```

Le fichier htaccess redirige vers le fichier htpasswd qui contient les authentifications que Apache utilisera lors de la connexion.

> 💡 ⚠️ Le chemin du fichier .htpasswd doit être absolu ⚠️

Frontend

Rendu

Un iframe rend les vues sélectionnées avec le menu latéral.

5. Gestion de la maintenance (corrective / évolutive)

5.1 Mise à jour de la documentation du SI

Pour la gestion de la maintenance corrective et évolutive du SI, il est important de maintenir à jour la documentation du SI. Cela permet de garantir que les développeurs ont accès à toutes les informations nécessaires pour effectuer des modifications et des mises à jour en toute sécurité et en toute efficacité.

En maintenant une documentation à jour, l'équipe de développement peut travailler de manière collaborative et efficace pour résoudre les problèmes et maintenir le SI à jour. Cela permet également de garantir que le système est toujours en mesure de répondre aux besoins de l'entreprise et de ses utilisateurs.

5.2 Évaluation de la qualité de la solution

La qualité de la solution a été évaluée en utilisant une suite de tests a été effectuée pour garantir que toutes les fonctionnalités fonctionnent correctement et que les données sont traitées de manière fiable.

Ensuite, des retours ont été recueillis auprès des futurs utilisateurs, en particulier du pôle de communication, afin de s'assurer que l'interface utilisateur est conviviale et que toutes les fonctionnalités sont utiles et faciles à utiliser.

Enfin, plusieurs versions ont été publiées pour permettre aux utilisateurs de tester la solution et de fournir des commentaires. Ces commentaires ont été pris en compte pour améliorer la solution et garantir qu'elle répond aux besoins des utilisateurs et de l'entreprise.

5.3 Procédure de correction d'un dysfonctionnement

Pour signaler un dysfonctionnement dans l'interface de recherche immobilière, les utilisateurs ont deux options :

1. Utiliser le formulaire de contact disponible sur le site web d'Immozia pour signaler le problème. Ils peuvent fournir des informations détaillées sur le problème rencontré, ce qui aidera l'équipe de développement à diagnostiquer et à résoudre le problème plus rapidement.
2. Ouvrir un ticket sur le dépôt GitHub d'Immozia pour signaler le problème. Cette méthode est particulièrement utile pour les utilisateurs techniques ou les

développeurs qui peuvent fournir des informations détaillées sur le problème et même proposer des correctifs ou des améliorations.

Quelle que soit la méthode utilisée, l'équipe de développement d'Immozia s'engage à diagnostiquer et à résoudre les problèmes signalés dans les plus brefs délais. Les utilisateurs seront informés de l'avancement de la résolution du problème et de la mise à disposition de correctifs ou de mises à jour pour résoudre le problème.

En cas de dysfonctionnement critique, l'équipe de développement prendra des mesures immédiates pour résoudre le problème et en informer les utilisateurs le plus rapidement possible.

6. Bilan du projet

6.1 Validation des exigences point par point

Validation des Exigences

✅ ~~Le formulaire est correctement affiché et tous les champs sont présents et correctement étiquetés.~~

✅ ~~Les données valides sont enregistrées correctement et un message de confirmation est affiché.~~

✅ ~~Les erreurs de saisie et les champs obligatoires manquants sont signalés par des messages d'erreur.~~

Le projet a été un succès, avec une interface de recherche immobilière conviviale et efficace qui répond aux besoins des utilisateurs et de l'entreprise.

La documentation a été maintenue à jour tout au long du projet, ce qui a facilité la maintenance corrective et évolutive du système.

Les tests ont été effectués pour garantir que toutes les fonctionnalités fonctionnent correctement et que les données sont traitées de manière fiable.

Les retours des utilisateurs ont été pris en compte pour améliorer la solution et garantir qu'elle répond aux besoins des utilisateurs et de l'entreprise.

En cas de dysfonctionnement, une procédure claire a été mise en place pour signaler et résoudre le problème. En somme, le projet a été un succès et a permis à l'entreprise de fournir une interface de recherche immobilière efficace et conviviale à ses utilisateurs.

6.2 Axes d'amélioration

Le système d'authentification actuel fonctionne avec le "basic auth" d'Apache. Bien que cela soit suffisant pour une utilisation interne, il peut être amélioré pour offrir une meilleure sécurité.

Il est recommandé de mettre en place un système d'authentification plus robuste, tel que l'authentification à deux facteurs ou l'authentification OAuth. **Ces méthodes offrent une meilleure sécurité en utilisant des méthodes d'authentification plus avancées.**

En outre, il est recommandé de mettre en place une politique de mots de passe forte pour garantir que les utilisateurs utilisent des mots de passe sûrs et qu'ils sont régulièrement mis à jour.

Enfin, il est important de mettre en place des mesures de sécurité supplémentaires, telles que la détection d'intrusion et la surveillance des logs d'activité, pour garantir que le système d'authentification est sécurisé en permanence.

Avis de droits d'auteur

Informations légales sur les droits d'auteur de cette documentation

Par Baptiste Grimaldi