

Tpi

« Caves »



1	Introduction.....	3
1.1	Cadre, description et motivation	3
1.2	Organisation.....	3
1.3	Objectifs	3
1.3.1	Gestion des publicités conviviale :	3
1.3.2	Gestion sécurisée et flexible des informations utilisateur :.....	4
1.3.3	Affichage et gestion détaillés des annonces :	4
1.3.4	Visualisation et filtrage anonymes des annonces :	4
1.3.5	Facilité de communication et de messagerie :	4
1.3.6	Sécurité et protection des données :	4
1.3.7	Panier d'achat	4
1.4	Planification initiale	5
2	Analyse	6
2.1	Fonctions prévues	6
2.2	Méthode de projet.....	7
2.3	Planification	7
2.4	Maquettes.....	11
2.5	User Cases	22
3	Implémentation	24
3.1	Choix techniques.....	24
3.2	Modèle conceptuel de données	26
3.3	Modèle Logique de données	28
3.4	Diagramme de Flux	30
3.5	Ergonomie de site (Bastien et Scapin)	33
3.6	Points techniques spécifiques.....	34
3.6.1	Enregistre les images	34
3.6.2	Utilisation et fonctionnalité de la carte.....	35
3.6.3	filtres.....	43
3.7	Bugs.....	50
4	Tests.....	50
4.1	Stratégie de test.....	50
5	Conclusions.....	53
5.1	Objectifs atteints.....	53
5.2	Objectifs non-atteints	53
5.3	Suites possibles au pour le projet	53
6	Annexes	53
6.1	Journal de travail.....	53
7	SOURCES.....	53

1 Introduction

1.1 Cadre, description et motivation

Cette application web que nous avons décidé de réaliser a été conçue comme un point de rencontre entre acheteurs et vendeurs. Étant donné que chaque acheteur et vendeur est un utilisateur de l'application, ils seront classés comme utilisateurs et pourront exécuter des fonctions telles que l'ajout d'annonces, l'affichage d'annonces, la saisie et la modification d'informations personnelles et la modification des annonces qu'ils ont placées.

1.2 Organisation

Chef de projet : Mr. Pascal BENZONANA, pascal.benzonana@eduvaud.ch
Expert 1 : Mr. Ernesto MONTEMAYOR, ernesto@bati-technologie.ch
Expert 2 : Mr. Grégory CHARMIER, gregory.charmier@eduvaud.ch

Elève 1 : Yasin Salih Akyüz, yasin.akyuz@eduvaud.ch

	Yasin Salih Akyüz
Partie administration	X
Partie BDD	X
Partie Backend	X
Partie Frontend	X
Partie Panier	X
Partie Carte	X
Mise en page générale	X

1.3 Objectifs

- L'utilisateur doit s'inscrire avec succès en saisissant ses informations personnelles.
- L'utilisateur peut se connecter avec les informations qu'il a saisies lors de l'inscription.
- L'utilisateur doit pouvoir saisir une annonce sur sa propre page.
- Les informations personnelles et les annonces doivent être téléchargées dans la base de données et peuvent être modifiées par l'utilisateur.
- Les utilisateurs doivent pouvoir voir l'annonce.
- L'utilisateur peut rechercher des annonces depuis la section de recherche.
- L'utilisateur connecté peut filtrer les annonces selon certains critères.
- Les annonces sont affichées sur la carte selon leurs adresses, et l'utilisateur connecté est affiché différemment sur la carte.
- Les annonces sur la carte sont également affectées par les options de recherche et de filtrage.
- L'utilisateur peut ajouter le produit qu'il souhaite au panier et le produit ajouté au panier sera retiré du stock pendant un certain temps.
- Les utilisateurs peuvent s'envoyer des messages après s'être connectés.

1.3.1 Gestion des publicités conviviale :

Les utilisateurs doivent pouvoir créer et publier facilement des publicités à partir de leur propre profil. Lors de la création d'une annonce, les utilisateurs doivent disposer d'options d'édition de texte enrichi

(par exemple, gras, italique, listes, titres, etc.).

1.3.2 Gestion sécurisée et flexible des informations utilisateur :

Les informations personnelles des utilisateurs (nom, e-mail, mot de passe, etc.) doivent être stockées en toute sécurité et les utilisateurs doivent pouvoir mettre à jour ces informations à tout moment.

Lors des mises à jour de mots de passe, les anciens mots de passe doivent être protégés et les nouveaux mots de passe doivent être hachés et stockés en toute sécurité.

1.3.3 Affichage et gestion détaillés des annonces :

Les utilisateurs doivent disposer d'une interface qui permettra une visualisation détaillée des publicités publiées. Les informations telles que la description du produit, le prix, les images et les coordonnées doivent être accessibles sur les pages de détails de l'annonce.

Les utilisateurs doivent pouvoir gérer leurs propres publications et les modifier ou les supprimer à tout moment.

1.3.4 Visualisation et filtrage anonymes des annonces :

Les utilisateurs doivent pouvoir consulter les publications sans se connecter. Ils doivent cependant se connecter pour poster ou communiquer.

Diverses options doivent être proposées pour faciliter le filtrage des annonces, par exemple la fourchette de prix, la catégorie, l'emplacement, etc.

1.3.5 Facilité de communication et de messagerie :

Un système de messagerie doit être prévu entre les utilisateurs pour communiquer avec les annonceurs. Ce système devrait permettre aux utilisateurs de communiquer directement depuis la page de détail de l'annonce.

1.3.6 Sécurité et protection des données :

Des mesures appropriées doivent être prises pour la sécurité des informations et des publicités des utilisateurs, et la sécurité des bases de données doit être assurée.

Il est particulièrement important que les informations sensibles (par exemple les mots de passe) soient stockées et traitées correctement.

1.3.7 Panier d'achat

Les utilisateurs pourront ajouter des produits au panier grâce au bouton « Ajouter au panier » qui devient visible une fois connecté. Dans ce cas, le produit peut être affiché dans le panier, la quantité peut être sélectionnée en fonction de l'état du stock et le prix total changera à mesure que le nombre de produits dans le panier augmentera.

1.3.8 Messages

Les messages envoyés par les utilisateurs doivent être reçus et enregistrés par un système de messagerie actif, puis transmis avec succès au destinataire. L'utilisateur doit également pouvoir voir les messages passés et les conditions nécessaires doivent être remplies pour que la conversation se poursuive sans interruption.

1.4 Planification initiale

Le projet débutera le mardi 30 avril à 8h00 et se terminera le mercredi 29 mai à 11h35. La planification du projet, comme décrit dans le cahier des charges, comprend les phases d'analyse, d'implémentation, de tests et de documentation.

Pour le suivi du projet, la période a été divisée en sprints, avec un total de quatre sprints définis.

Projet		Planification											
Caves			30.04.24	06.05.24	17.05.24	24.05.24	31.05.24	07.06.24	14.06.24	21.06.24			
Total													
Prévu	90 h 00		29 h 25	24 h 40	19 h 05	16 h 50							
Akyüz	16 h 55		16 h 55										
		SEM	18	19	20	21	22	23	24	25			
1 Analyse										19 h 40			
										4 h 10			
	11 - Analyse	Prévu	1h00	1h00	0h30								
		Akyüz	0h30										
	12 - Planification	Prévu	1h00	0h30	0h30								
		Akyüz	0h15										
	13 - Créez le fichier	Prévu	0h30										
		Akyüz	0h10										
	14 - Icescrum & Github	Prévu	3h30	1h10	1h00	2h30							
		Akyüz	3h15										
	15 - Recherches	Prévu	2h00	2h00	2h00	0h30							
		Akyüz											
		Prévu											
		Akyüz											
2 Implémentation										30 h 15			
										6 h 05			
	21 - MCD-MLD	Prévu	1h00	2h00	0h45	0h20							
		Akyüz											
	22 - BDD-SQL	Prévu	1h45	1h00	0h45	0h20							
		Akyüz	1h30										
	23 - Balsamiq	Prévu	2h00	0h30	0h30	0h20							
		Akyüz	2h00										
	24 - Backend	Prévu	4h00	7h00	6h00	3h00							
		Akyüz	2h35										
	25 - Frontend	Prévu	1h00	3h00	1h30	2h00							
		Akyüz	0h45										
	26 - Map	Prévu	4h00	1h00	0h30	1h00							
		Akyüz	2h00										
3 Tests										10 h 35			
3 Tests													
	31 - Use-cases	Prévu	2h00	1h00	0h30	1h00							
		Akyüz											
	32 - Tests	Prévu	0h30	0h30	0h35	0h30							
		Akyüz											
	33 - Résultats des tests	Prévu	0h30	0h30	1h00	2h00							
		Akyüz											
		Prévu											
		Akyüz											
		Prévu											
		Akyüz											
4 Documentation										14 h 30			
										3 h 55			
	41 - Journal de travail	Prévu	0h40	0h30	0h30	0h30							
		Akyüz	0h20										
	42 - Rédaction de la documentation	Prévu	3h00	2h30	2h00	2h20							
		Akyüz	1h30										
	43 - Contacts avec le chef de projet et les experts	Prévu	1h00	0h30	0h30	0h30							
		Akyüz	2h05										

SEM 1	04:00	SPRINT-1	23 h 25
	04:00		
	06:20		
SEM 2	04:00	SPRINT-2	24 h 40
	04:00		
	06:20		
SEM 3	04:00	SPRINT-3	19 h 05
	06:20		
	04:45		
SEM 4	04:00	SPRINT-4	16 h 50
	06:20		
	04:45		
SEM 5	06:20		
	04:00		
	01:45		
Total	90:00		90 h 00
Par semaine Nb heures Nb périodes			
Lundi	06:20		8
Mardi	04:00		5
Mercredi	04:00		5
Jeudi	06:20		8
Vendredi	04:45		6
Total	25:25		32

Intégration de la page d'accueil et de la carte :

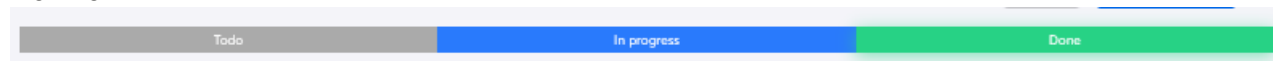
Les produits ajoutés par les utilisateurs sont répertoriés sur la page d'accueil. De plus, grâce à l'intégration cartographique, les utilisateurs peuvent voir les produits à proximité sur la carte en fonction de leur emplacement.

Système de messagerie :

Les utilisateurs peuvent utiliser un système de messagerie pour contacter les annonceurs. Il existe un formulaire de contact pour contacter l'annonceur sur la page de détail de l'annonce.

2.2 Méthode de projet

Le projet sera géré selon la méthode **Agile**, en utilisant une approche combinée de Scrum et de Kanban. Une approche basée sur le sprint, typique de Scrum, sera adoptée pour optimiser le processus de développement du projet. Chaque sprint visera à atteindre des objectifs fixés sur une période de temps spécifique. Parallèlement, nous intégrerons la méthode **Kanban** pour améliorer la visibilité et la fluidité du workflow.



Ce système Kanban consistera à utiliser un tableau Kanban pour visualiser toutes les tâches du projet, les classer en différentes catégories (par exemple, à faire, en cours, terminé) et limiter le travail en cours pour assurer une gestion continue et efficace des ressources. Cette combinaison des deux méthodes permettra une flexibilité maximale et une amélioration continue tout au long du cycle de développement du projet.

2.3 Planification

Des outils appropriés seront utilisés pour planifier et suivre le projet.

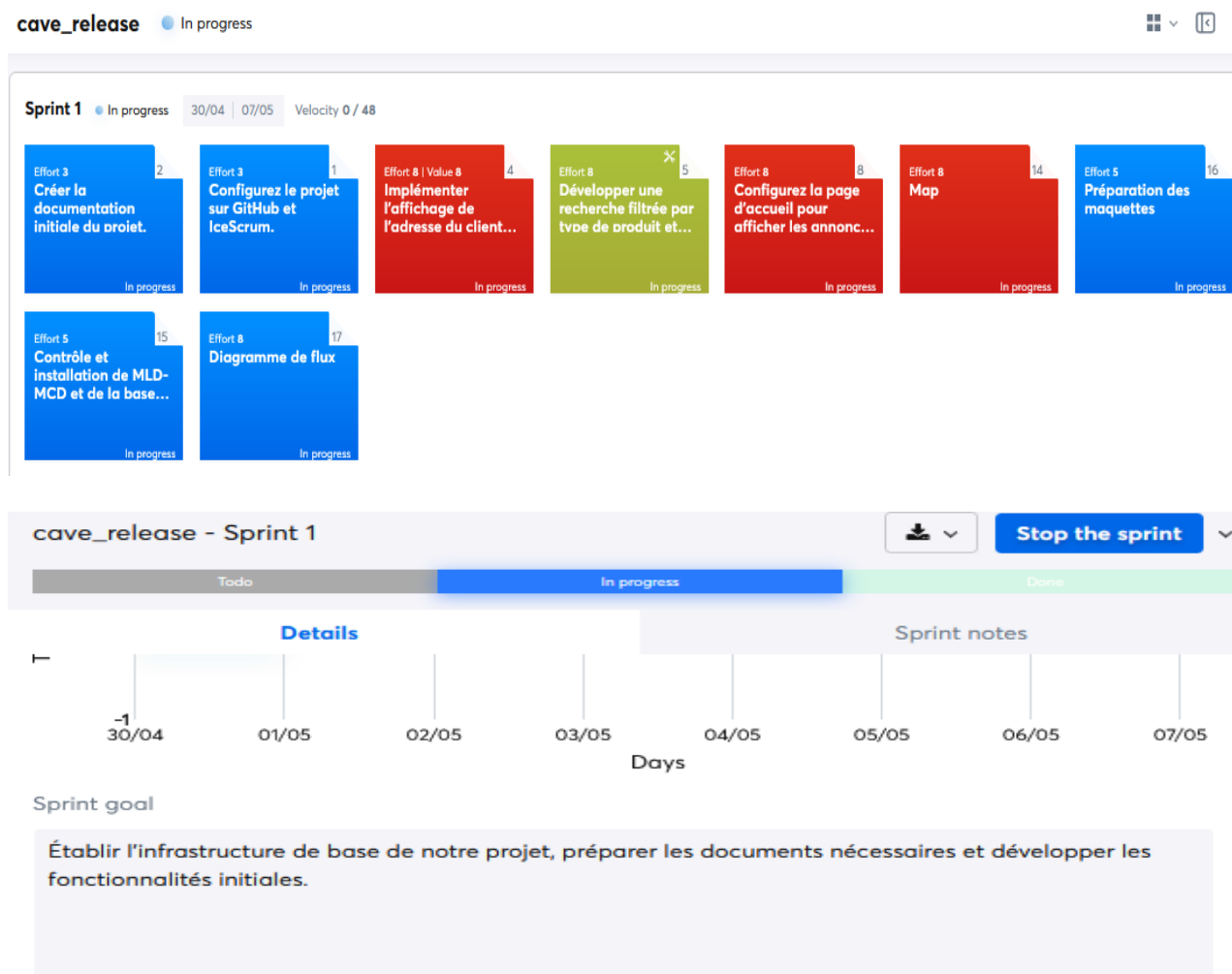
La répartition du travail et le calendrier seront déterminés et suivis tout au long de la gestion du projet. De plus, des étapes et des sprints importants seront déterminés et des efforts seront déployés pour atteindre ces objectifs.

J'utilise "Ice Scrum" pour suivre et planifier le projet. Pendant que nous suivons l'avancement du projet avec des fonctionnalités telles que le sprint, la tâche, le test, nous rendons le suivi du projet plus détaillé.

- Sprint 1 (30.04-07.05)

- Dans la phase de développement du site web, les travaux ont commencé sur la préparation des documents, la réalisation du projet, et l'établissement de la base de données, parmi d'autres tâches nécessaires et supportives.
- Pour contrôler, organiser et suivre facilement la progression du projet, un dossier local et un dossier GitHub ont été créés, et une structure de suivi avec iceScrum a été mise en place.
- Pendant le sprint-1, un test-task a été ajouté à chaque histoire pour faciliter le suivi, des mises à jour ont été faites sur le tableau des tâches et un objectif de sprint a été créé.
- Des maquettes suggérant l'apparence des pages web ont été préparées en utilisant le programme Balsamiq.

- À l'aide de Draw.io, un tableau "MCD" (Modèle Conceptuel de Données) a été détaillé, travaillant en profondeur sur les relations de référence entre les tables prévues pour la base de données.
- Le tableau "MLD" (Modèle Logique de Données) a été finalisé sur "MySQL Workbench", avec l'ajout de détails tels que clé primaire, clé étrangère, null, etc., et les liens entre les tables ont été revus.
- Les opérations de base de données ont été achevées avec l'utilisation de "HeidiSQL", les tables ont été établies, et des opérations d'INSERT et de SELECT de test ont été effectuées pour vérifier le fonctionnement des tables et leurs interconnexions.
- La page d'accueil a été organisée pour être utilisable pour les opérations de backend.
- Mapbox GLJS a été choisi pour l'utilisation de la carte et ajouté à la page d'accueil pour améliorer l'expérience utilisateur.
- Un diagramme de flux a été préparé avec le programme Draw.io et sera mis à jour tout au long des sprints.
- Des options de filtrage et de recherche ont été ajoutées au projet, offrant de nouvelles fonctionnalités aux utilisateurs.
- Des options ont été ajoutées pour que l'utilisateur puisse voir les publicités et leur emplacement sur la carte après le processus de connexion, et pour voir les informations sur la publicité lorsqu'il clique sur les publicités.





- Sprint 2 (08.05-16.05)

Au cours du deuxième sprint, des efforts seront déployés pour augmenter les fonctionnalités visuelles et utilisables de l'application par les utilisateurs et pour garantir que les instructions soient explicatives pour l'utilisateur.

- De plus, augmenter l'expérience de vente de l'utilisateur et rendre les fonctions du panier fonctionnelles seront un objectif distinct et important.
- Pendant le sprint-2, un test-task a été ajouté à chaque histoire pour faciliter le suivi, des mises à jour ont été faites sur le tableau des tâches et un objectif de sprint a été créé.
- L'accent sera mis sur l'affichage des publicités et des utilisateurs connectés sur la carte.



Les opérations que l'utilisateur effectue lors de l'utilisation de la fonction panier doivent également avoir un effet sur la base de données. Il ne sera peut-être pas possible de terminer le sprint dans le cadre des études menées dans ce contexte.

Les éléments du sprint qui ne pourront pas être complétés seront transférés au sprint suivant et le suivi se poursuivra.

- Sprint 3 (17.05-23.05)

- La mise en place d'un système de messagerie sera la tâche la plus importante de ce sprint.
- Les progrès réalisés depuis le début de la candidature seront testés et ajoutés au document.

Sprint 3 ● Todo 17/05 | 23/05 Planned velocity 18

Effort 13 7

Mettre en place un système de messagerie entre...

Planned

Effort 5 9

Commencez à tester les fonctionnalités développées jusau'...

Planned

Bien que le 3ème sprint ait été planifié comme suit, puisque la fonction Panier n'a pas pu être complétée dans le temps imparti, le travail s'est poursuivi pendant le 3ème sprint et s'est achevé au sprint 3.

Sprint 3 ● Done 17/05 | 23/05 Velocity 31 / 39

✓ | Effort 13 | Value 11 10

Mettre en place un système de réservation de...

☑ 4 ☰ 3

Done

✓ | Effort 13 | Value 12 7

Mettre en place un système de messagerie entre...

☑ 4 ☰ 4

Done

✓ | Effort 5 9

Commencez à tester les fonctionnalités développées jusau'...

☑ 8 ☰ 2

Done

À la suite des tests effectués à la fin du Sprint-3, une déficience a été observée uniquement dans le processus de messagerie, et cette déficience sera mentionnée dans la section bug. Outre cette lacune, les tests ont été réalisés avec succès par une tierce personne.

- Sprint 4 (24.05-29.05)

Sprint 4 ● Todo 24/05 | 29/05 Planned velocity 21

Effort 8 11

Finaliser les ajustements de l'interface utilisateur.

Planned

Effort 8 12

Effectuer des tests approfondis, y compris des tests...

Planned

Effort 5 13

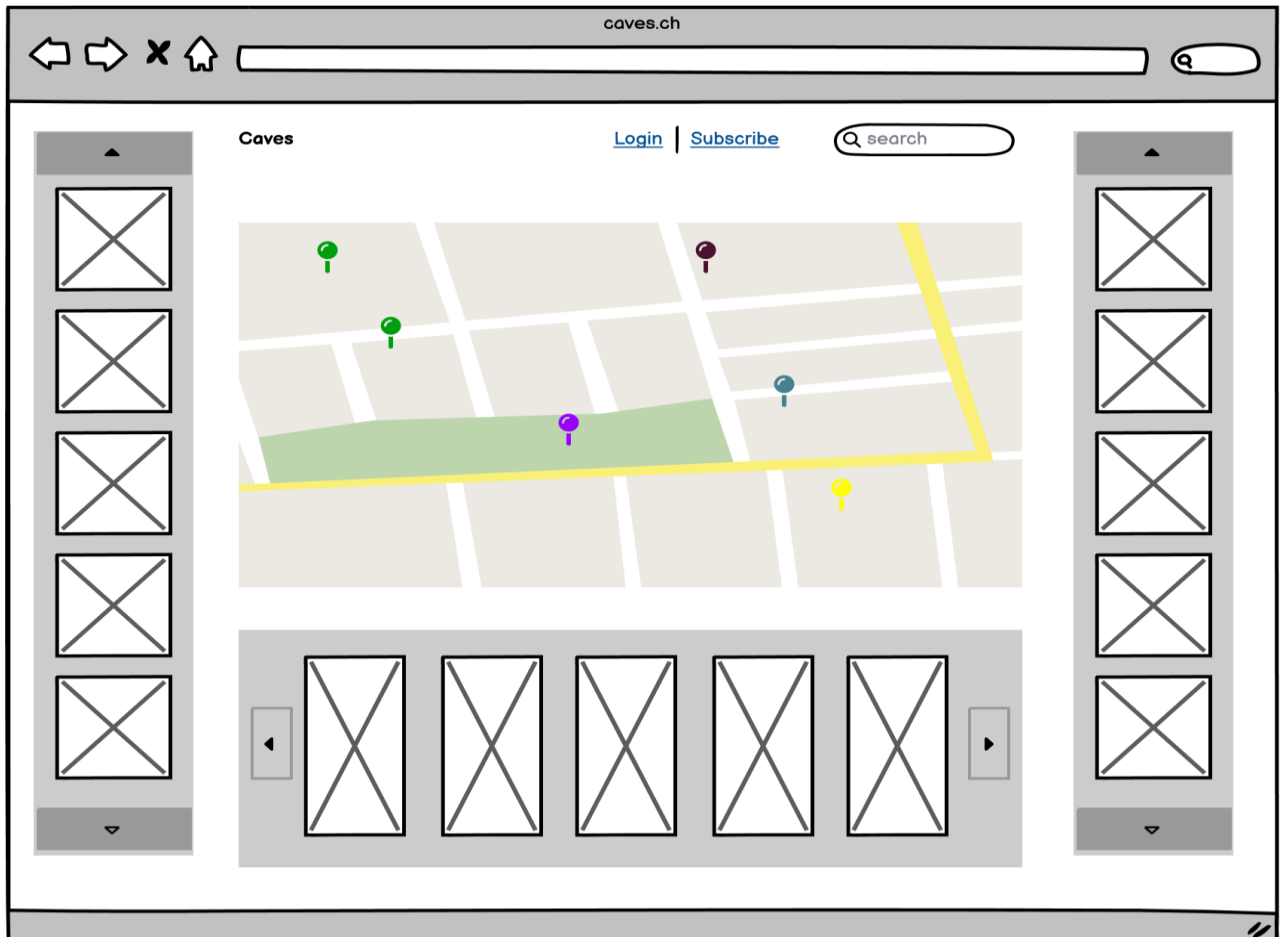
Préparer la documentation finale du projet et l...

Planned

2.4 Maquettes

- Lorsque l'utilisateur ouvre la page web, il sera accueilli par la page « PAGE ACCUEIL » qui apparaîtra comme suit.

Sur cette page, l'utilisateur pourra s'inscrire et se connecter, rechercher des produits. Cependant, bien qu'il puisse voir les annonces, il devra se connecter pour entrer en contact avec l'annonceur.



Dans cette page, si l'utilisateur est connecté ou non, l'utilisateur n'a pas accès aux mêmes actions :

- ⇒ Non-connecté :
 - Login
 - Subscribe
 - Les Annonces
- ⇒ Connecté :
 - Logout
 - Filtrer
 - Messages
 - Mypage
 - Les Annonces

Avant de vous connecter, vous devez vous inscrire en saisissant vos informations d'utilisateur. Lorsque vous cliquez sur le bouton S'abonner, une fenêtre contextuelle s'ouvrira et l'utilisateur saisira et enregistrera ses informations. Ensuite, en se connectant, il pourra modifier ses informations personnelles, publier une annonce et contacter d'autres annonceurs.

Lorsque l'utilisateur souhaite s'inscrire, il cliquera sur le bouton "Subscribe", et à ce moment-là, une page ou une fenêtre pop-up s'ouvrira pour entrer les informations de l'utilisateur.

Lors de la saisie des informations, l'utilisateur verra un écran d'avertissement lui indiquant de saisir son adresse e-mail et son numéro de téléphone conformément au type de données défini dans la base de données. Ainsi, il pourra compléter le processus d'inscription avec succès.

A Web Page

https://

Caves Login

Nom de l'entreprise (facultatif)

Nom Prénom

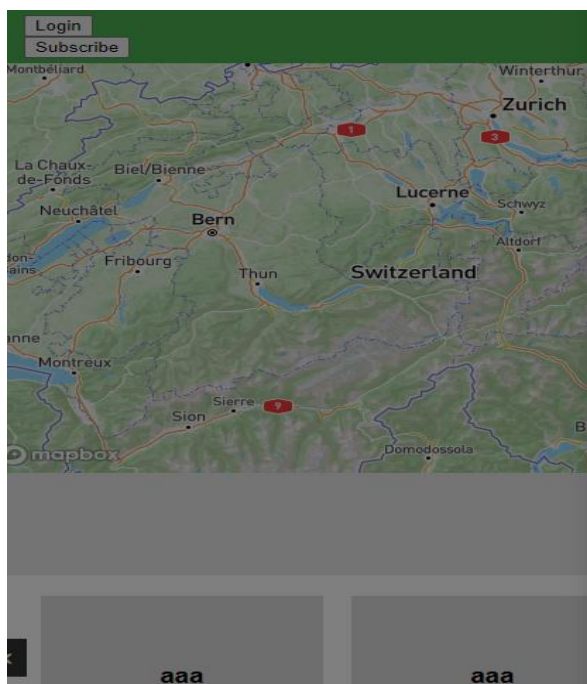
e mail Phone

Rue et N° Code Postal

Ville Canton

Password Confirm Password

Subscribe



Subscribe

Nom de l'entreprise (facultatif)

Name

Firstname

E mail

Numéro de téléphone

Adresse

Numéro du bâtiment

Code postal

Ville

Canton

Le mot de passe

Confirmez le mot de passe

Subscribe

- Après avoir terminé le processus d'inscription, l'utilisateur pourra se connecter à sa page utilisateur en utilisant l'adresse e-mail enregistrée et le mot de passe choisi.

Il aura alors la possibilité de publier des annonces, de modifier les annonces publiées, de modifier ses informations personnelles, ainsi que d'envoyer et de recevoir des messages.

The screenshot shows a web browser window titled 'A Web Page'. The address bar contains 'https://'. The main content area of the browser displays a web page with the following elements: a header 'Caves' on the left and a 'Subscribe' button on the right; a form with two input fields labeled 'E mail' and 'Password'; and a 'Login' button centered below the input fields.



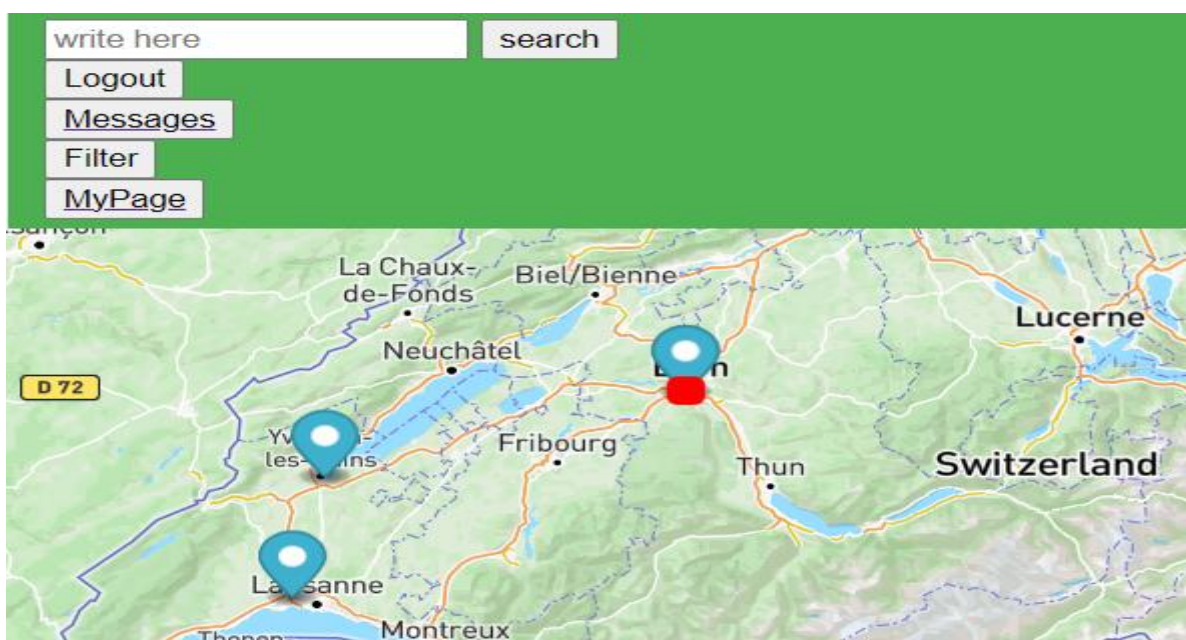
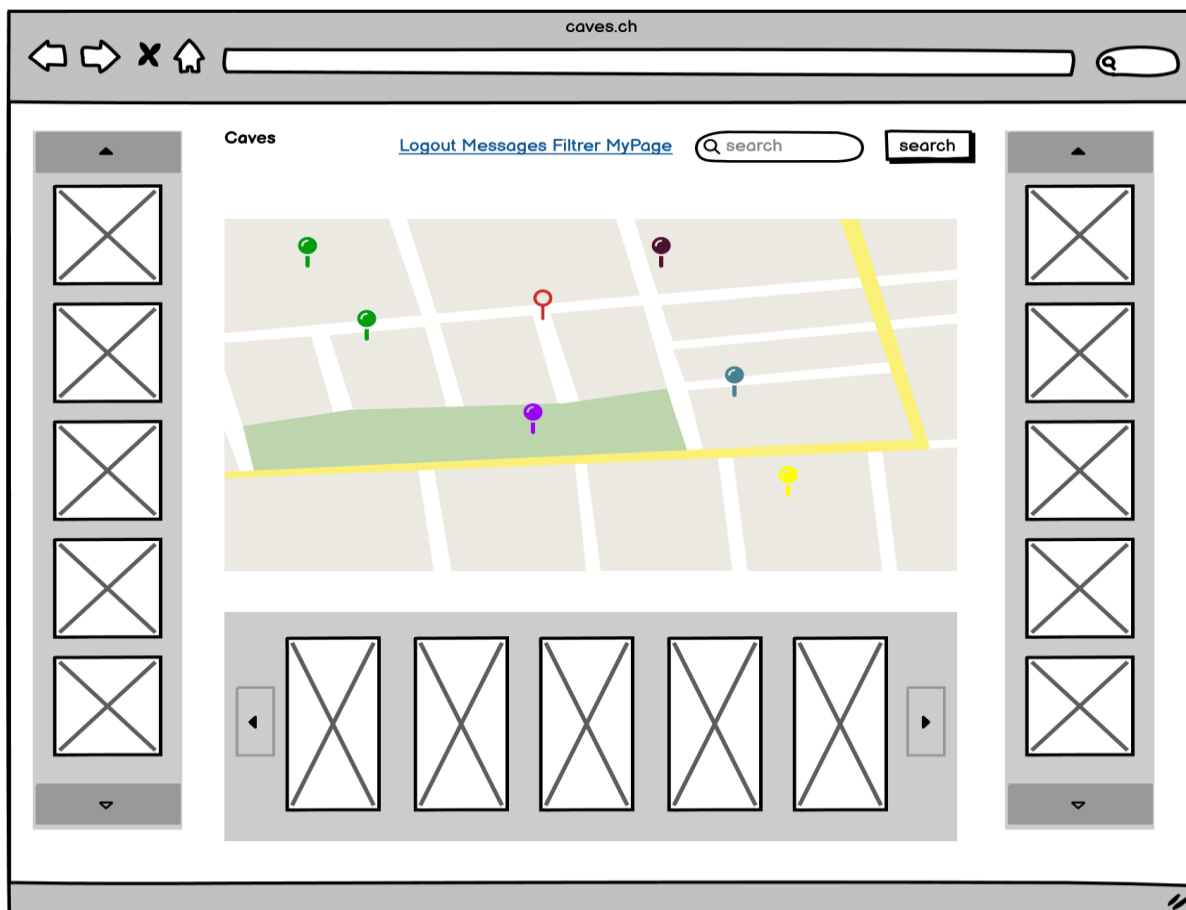
Login

The image shows a clean, modern login form. It has a title 'Login' at the top. Below the title are two input fields: 'E-mail' and 'Password'. At the bottom of the form is a large green button with the text 'Login' in white.

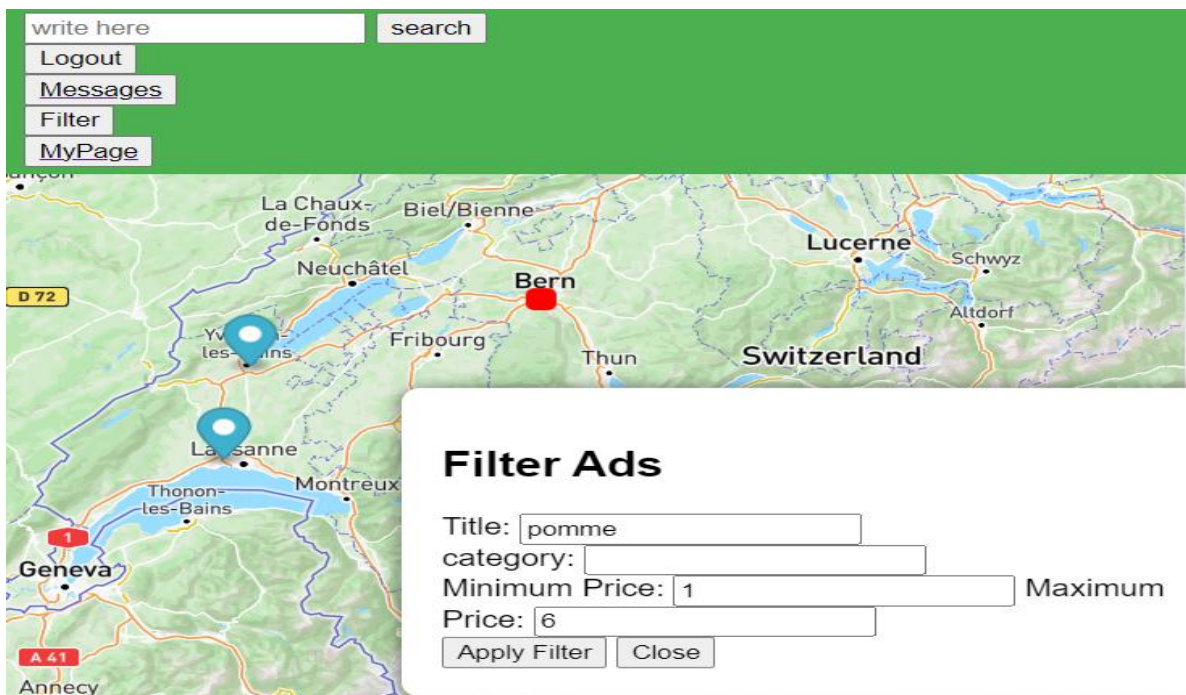
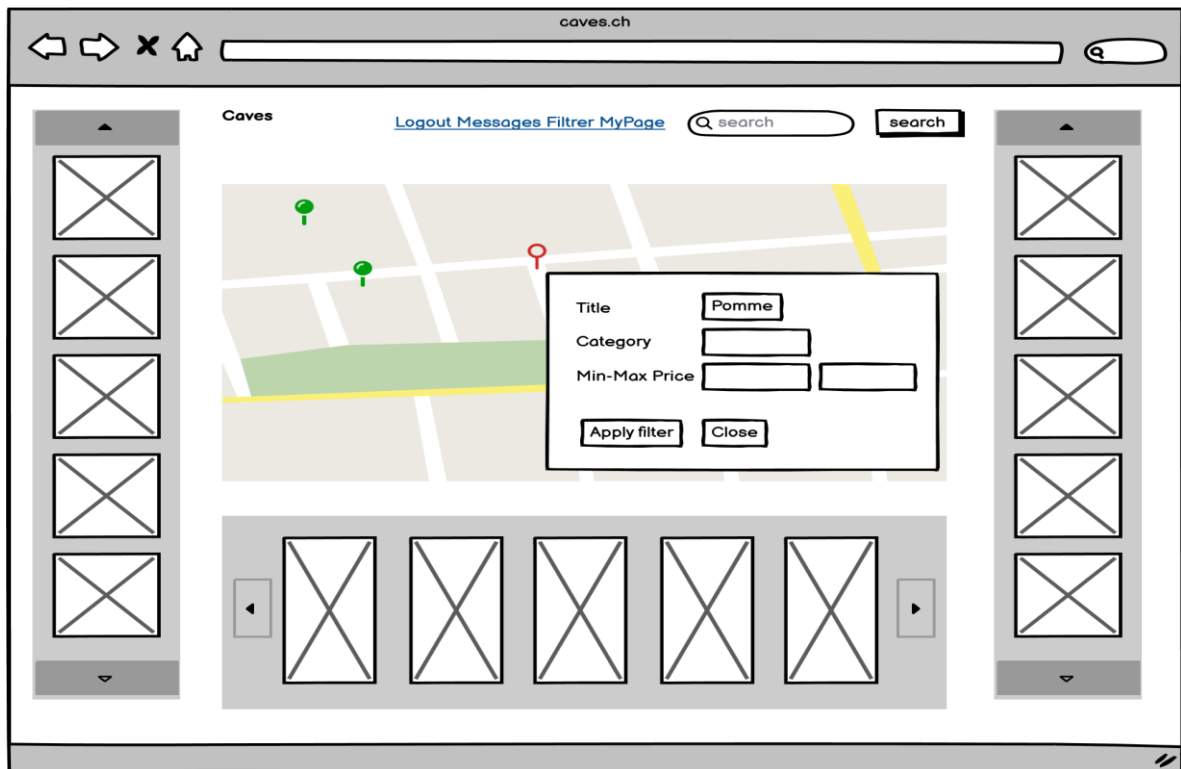
- Après l'enregistrement de l'utilisateur, les boutons sur la page d'accueil changeront pour offrir différentes fonctionnalités.

En cliquant sur le bouton MyPage, l'utilisateur sera redirigé vers sa page personnelle. En cliquant sur le bouton de déconnexion, il pourra se déconnecter.

Le bouton Filtrer permettra de réaliser des recherches détaillées, et le bouton messages permettra à l'utilisateur de voir ses messages.



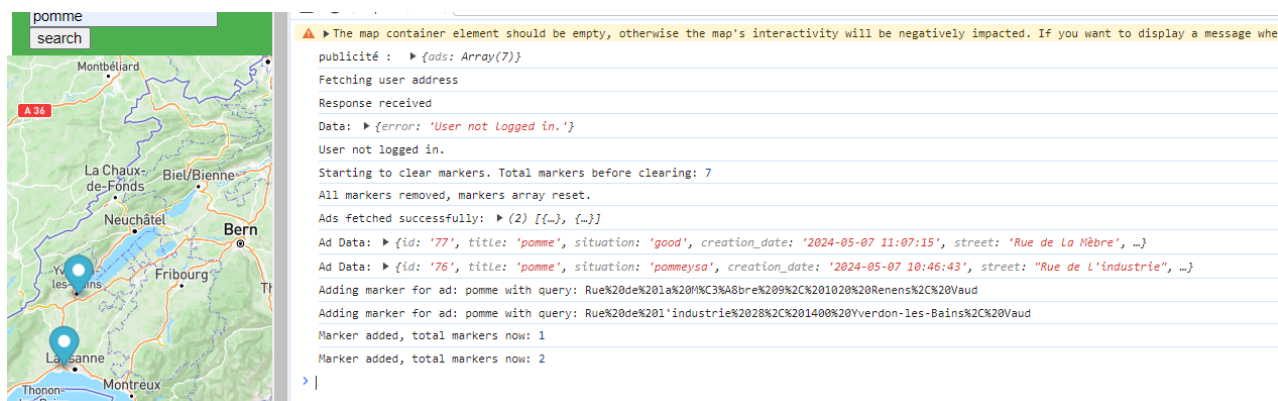
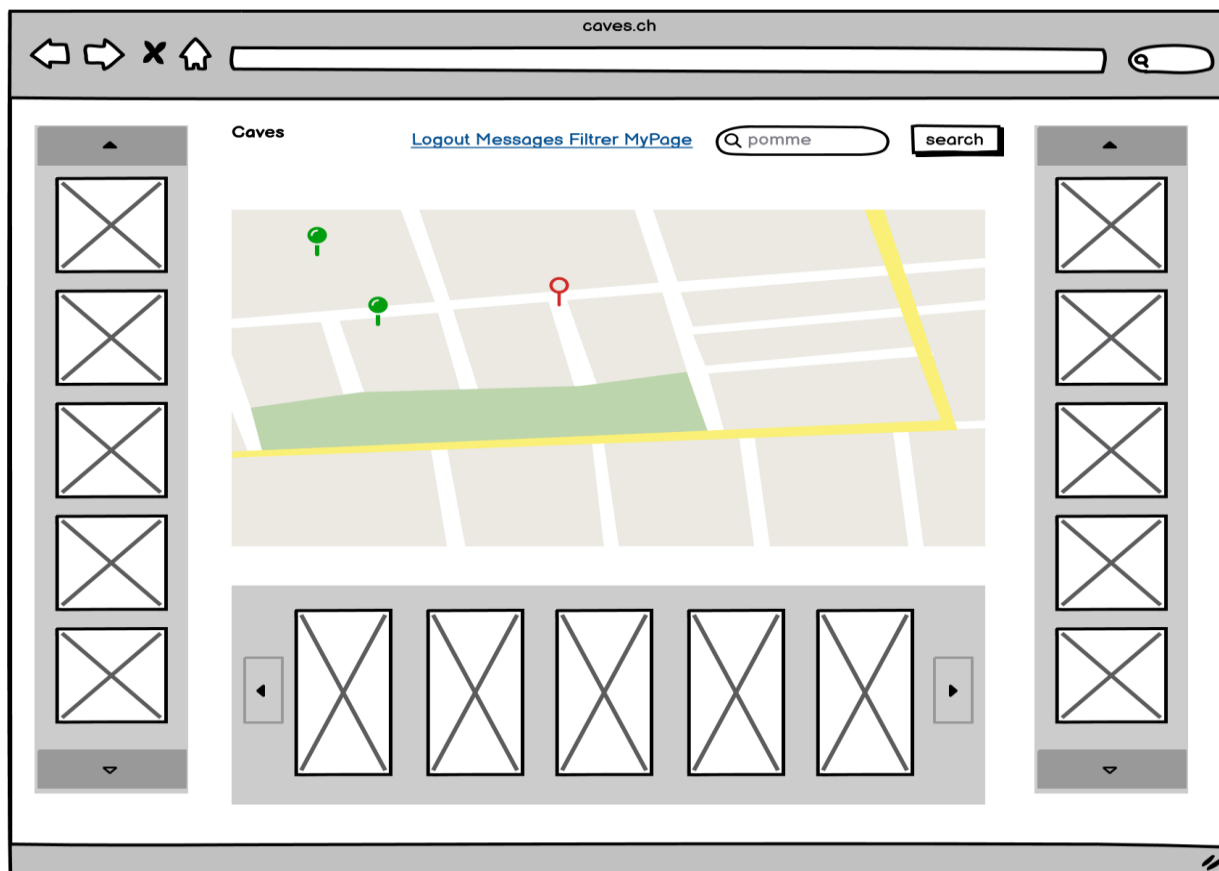
- L'utilisateur pourra également effectuer les opérations de filtrage qu'il jugera nécessaires en sélectionnant l'option de filtrage sur cette page. Dans l'exemple, le filtrage sous l'en-tête « titre » est affiché. À ce stade, lorsque l'on clique sur le bouton « Appliquer le filtre », les publicités affichées sur cette page s'affichent en fonction du résultat du filtrage.



```
Starting to clear markers. Total markers before clearing: 1 accueil.js:42
All markers removed, markers array reset. accueil.js:45
Ads fetched successfully: ▶ (2) [{...}, {...}] accueil.js:59
Ad Data: accueil.js:75
▶ {id: '76', title: 'pomme', situation: 'pommeysa', creation_date: '2024-05-07 10:46:43', street: "Rue de L'industri
e", ...}
Ad Data: accueil.js:75
▶ {id: '77', title: 'pomme', situation: 'good', creation_date: '2024-05-07 11:07:15', street: 'Rue de La Mèbre', ...}
Adding marker for ad: pomme with query: Rue%20de%20l'industrie%2028%2C%201400%20Yverdon-les- accueil.js:92
Bains%2C%20Vaud
Adding marker for ad: pomme with query: Rue%20de%20la%20M%C3%A8bre%209%2C%201020%20Renens%2C%20Vaud accueil.js:92
Marker added, total markers now: 1 accueil.js:107
Marker added, total markers now: 2 accueil.js:107
```


Comme on le voit dans cet exemple, les marqueurs sur la carte changent avec la fonction de filtrage, et les annonces sur la page d'accueil sont également affectées par les options de filtrage.

- Search



Dans cet exemple, nous voyons que la carte et les annonces dans l'application sont instantanément affectées par l'opération de recherche, et que les annonces sur la page changent en conséquence.

- La page "MyPage" est une page personnelle où l'utilisateur peut effectuer et organiser ses transactions en détail, unique pour chaque utilisateur.
Sur cette page, l'utilisateur peut modifier toutes ses informations avec des redirections, poster de nouvelles annonces et voir les annonces qu'il a postées.
Cette page, que nous pourrions appeler une page de modification, permettra à l'utilisateur d'interagir de multiples manières dans les étapes futures et restera ouverte à des ajouts.
Par exemple, l'option de supprimer le compte de l'utilisateur pourrait être ajoutée à cette page

plus tard.

My Page

Edit Personal Information

Name	first Name	Company Name	Email	Phone	Street	Building Number	Postal Code
City	Canton	new password	Save Changes				

Post an Ad

Ad Title	situation	fruit
product name:	Urün Adi	
price:		
Stock:		
images:	Sélect. fichiers	Aucun fichier choisi
Post Ad		

Ads List

mais2

bon



Product Name: mais fruit

Price: 1

Stock: 4

A Web Page

← → × 🏠
https://mypage
🔍

Edit Personal Informations

Nom	<input type="text"/>	Prénom	<input type="text"/>
e mail	<input type="text"/>	Phone	<input type="text"/>
Rue et N°	<input type="text"/>	Code Postal	<input type="text"/>
Ville	<input type="text"/>	Canton	<input type="text"/>
Password	<input type="text"/>	Save Changes	

Post an Ad

<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>
Images : <input type="button" value="select fichier"/>	<input type="button" value="PostAd"/>

My Ads

Item One

Item Two

ItemThree

Item Four

L'utilisateur pourra à ce stade consulter ses informations personnelles et apporter les modifications souhaitées. Lors de la modification, les anciennes données seront préservées dans les champs laissés vides. Le nouveau mot de passe sera également enregistré dans la base de données sous forme de hachage.

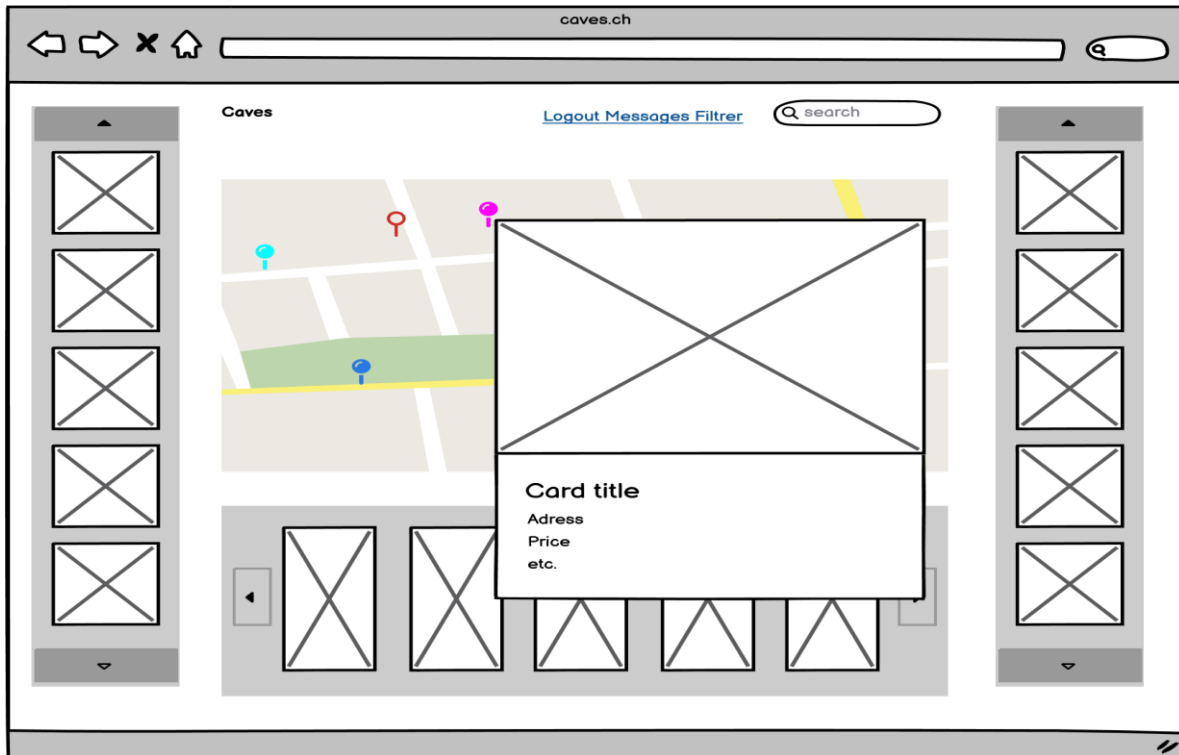
Dans la section de publication d'annonce, il pourra publier une annonce détaillée en saisissant les informations nécessaires. Il pourra également sélectionner plusieurs images et les enregistrer dans la base de données sous forme d'URL.

Il pourra consulter les annonces qu'il a publiées dans la section Liste des annonces et, à l'avenir, effectuer des opérations de modification et de suppression sur ces annonces.

- Sur la page d'accueil (après la connexion), l'utilisateur peut voir les annonces qui ont été publiées précédemment.

Lorsque l'utilisateur passe la souris sur une annonce, un pop-up apparaîtra, fournissant des informations succinctes sur celle-ci.

Avec ces informations, si l'utilisateur le souhaite, il pourra accéder à une page de détails de l'annonce pour voir plus de détails.



- La page de détails de l'annonce s'ouvre lorsque l'utilisateur clique sur une annonce.

Sur cette page, l'utilisateur peut voir les informations détaillées de l'annonce et, s'il est connecté, peut envoyer un message à l'autre utilisateur qui a publié l'annonce.

carrot



disponible

carrot

Price: 3

Stock: 4

Address: Rue de l'industrie 28, 1400 Yverdon-les-Bains, Vaud

Creation Date: 07/05/2024

0

carrot



disponible

carrot

Price: 3

Stock: 4

Address: Rue de l'industrie 28, 1400 Yverdon-les-Bains, Vaud

Creation Date: 07/05/2024

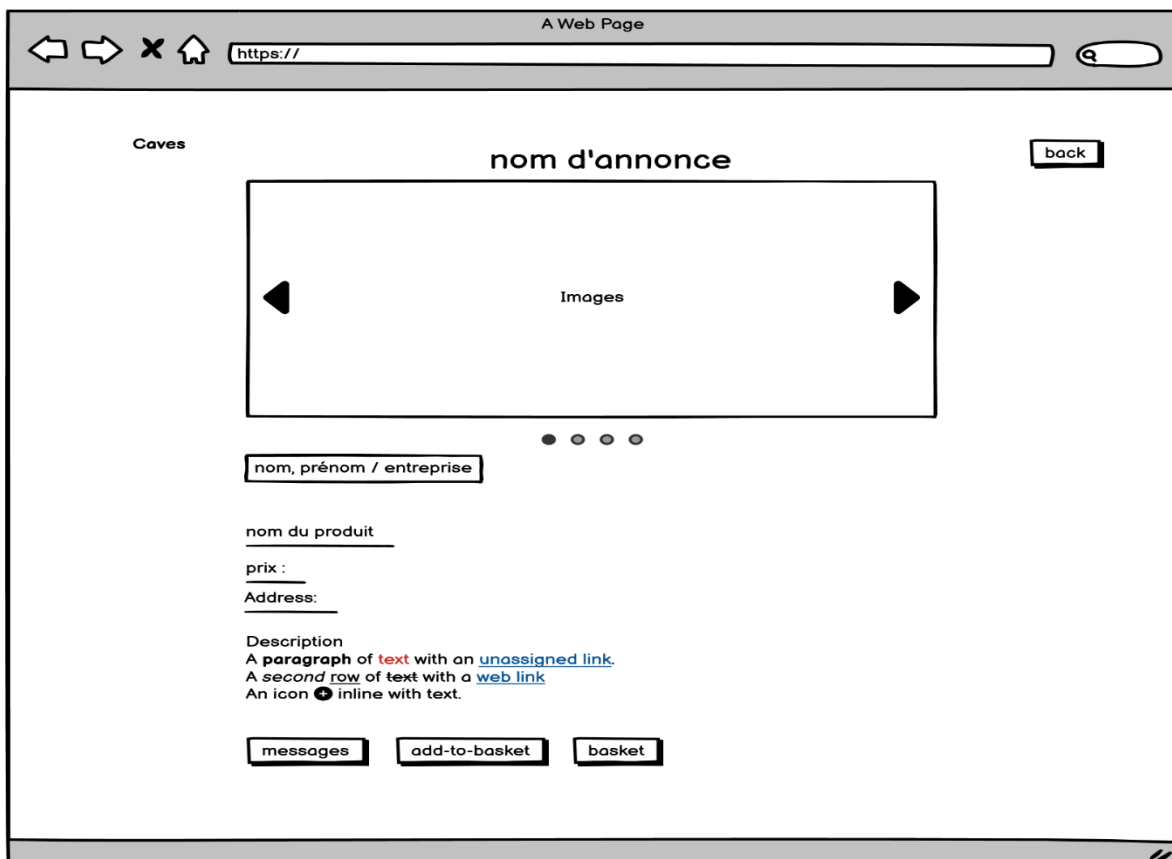
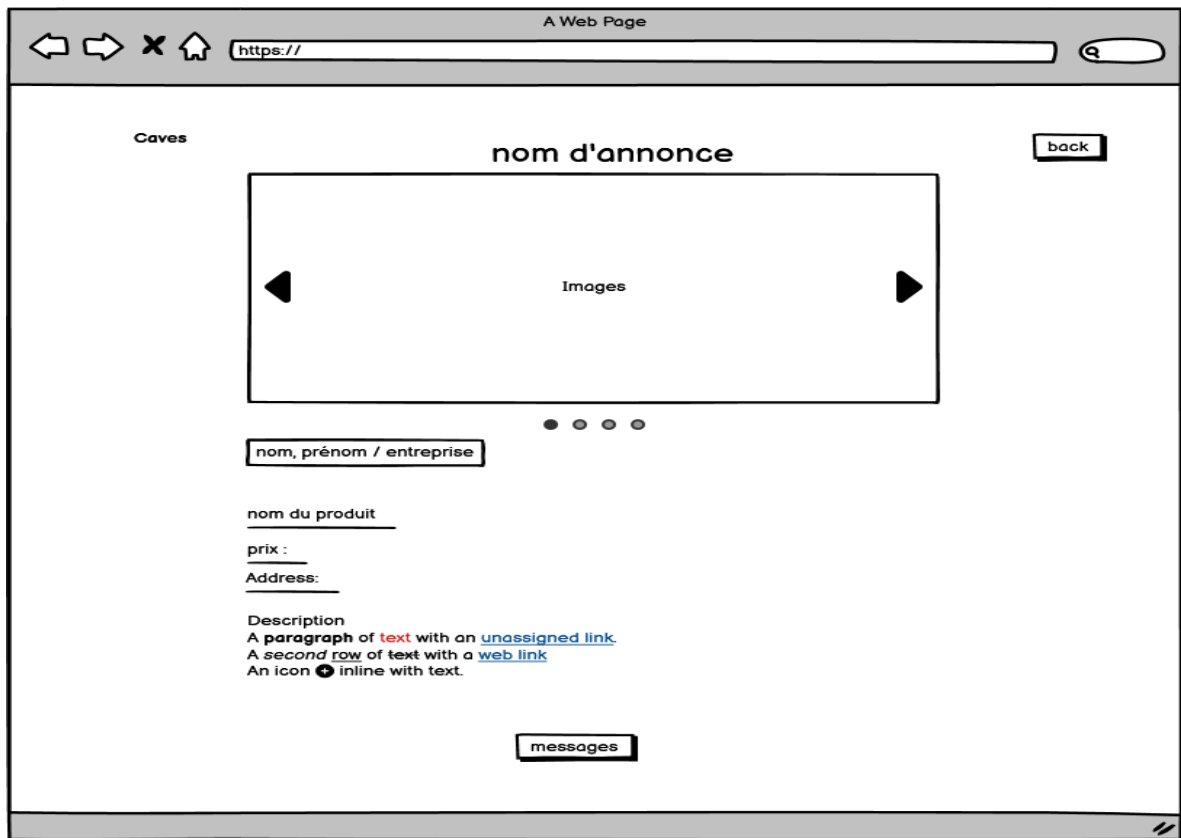
0

Messages

Add To Cart

Panier(0)

Ces deux captures d'écran prises à partir de l'application montrent que la page de détails est différente selon que l'utilisateur s'est connecté ou non. Pour que l'utilisateur puisse envoyer des messages ou ajouter des articles au panier, il est d'abord nécessaire qu'il se connecte.



- Un utilisateur qui s'est connecté et qui visualise une annonce peut poser directement des questions à l'utilisateur ayant publié l'annonce, permettant ainsi une communication directe entre eux sans avoir à partager leurs informations de contact personnelles dans l'annonce.

Ce système est également une caractéristique importante pour protéger les données personnelles de chaque utilisateur, évitant la nécessité de divulguer des informations de contact personnelles dans les annonces.

A Web Page

https://

Caves

messages

back

une petit information sur l'annonce

prix

expéditeur du message

destinataire du message

Ce produit est-il toujours disponible?

Envoyer

- Après avoir effectué la connexion, un des boutons qui apparaît est le bouton « Mes messages ».

En cliquant sur ce bouton, l'utilisateur peut voir tous les messages qu'il a envoyés et reçus en un seul endroit, et il a également la possibilité de supprimer l'historique des messages qu'il souhaite.

A Web Page

https://

Caves

MESSAGES

une petit information sur l'annonce

Supprimer

- Une fois que l'utilisateur s'est connecté et a accédé à la page de publicité, il verra le bouton Ajouter au panier. Lorsque vous cliquez sur ce bouton, le produit sera ajouté au panier.

Lorsque vous cliquez sur le bouton du panier, la page du panier s'ouvrira et les détails du panier seront affichés.




Image Name

Name Product

Quantity

Prix :




Image Name

Name Product

Quantity


Prix :

CheckOut

EmptyCart

Une fois que le produit est dans le panier, il sera réservé pour l'utilisateur.

Cette période de réservation est prédéfinie et peut être augmentée ou diminuée. Lorsque le temps est écoulé, le panier sera réinitialisé et le produit sera retiré du panier.



pomme - Quantity: 4 - Price: 12 TL

emptyCart checkout

DevTools is now available in French!

Always match Chrome's language Switch DevTools to French Don't show ag

Elements Console Sources Network Performance

top Filter Def

Cart timeout started. Cart will be cleared in 30 seconds.

2.5 User Cases

« Use Case » Les cas d'utilisation (scénarios d'utilisation) mentionnés dans les tableaux définissent les comportements et conditions attendus pour diverses situations d'interaction des utilisateurs avec une plateforme web. Ces informations, fournies sur deux pages, couvrent les fonctions de base que les utilisateurs peuvent exécuter sur le site web et présentent les scénarios et conditions possibles pour chacune de ces fonctions.

Le premier tableau contient les scénarios définis pour la page d'accueil du site, qui est le point de départ de l'accès des utilisateurs. Ces scénarios incluent des processus tels que l'affichage du formulaire d'inscription et la création d'un nouvel utilisateur. Pour chaque scénario, les actions requises (par exemple, cliquer sur un bouton), les conditions nécessaires à leur réalisation et les résultats attendus (succès ou échecs) sont spécifiés.

Le deuxième tableau traite des scénarios après que l'utilisateur se soit connecté au système. Ces scénarios incluent des fonctions telles que la mise à jour d'informations, la publication d'annonces, la fermeture de session et la gestion des messages. Les actions nécessaires, les conditions et les résultats possibles pour chaque scénario sont également listés en détail.

Objectif du Use Case	Action	Conditions	Scénario condition remplie	scénario echec condition
Page principale (non-connecté)				
Visionner des publicités	Cliquez sur les annonces	as pouvoir contacter l'annonceur sans se conn	Affichage des informations sur les annonces via la base de données	
Afficher formulaire d'inscription	Click sur bouton "subscription"		Affichage du formulaire d'inscription	
Créer un nouvel utilisateurs	Click sur bouton "Subscribe"	Ne pas avoir de compte avec la même adresse email	Inscription de l'utilisateur dans la base de données	Message d'erreur s'affiche "un compte est déjà lié à cette adresse email" Avertissement si les informations saisies ne correspondent pas aux types de données. Avertissement si les informations saisies ne correspondent pas aux types de données. Email, postal code, building number.
			Session utilisateurs "ouverte", avec ses données chargées dans le sessionstorage	Avertissement que les deux mots de passe saisis ne correspondent pas.
Fermer le formulaire d'inscription	Cliquez n'importe où sur l'écran		Fermeture du formulaire d'inscription	
Afficher formulaire de login	Click sur bouton "LOGIN"		Affichage du formulaire de login	
Connecter un utilisateurs	Click sur bouton "LOGIN" du formulaire login	Avoir déjà créé un compte	bouton "login" se transforme en bt "logout"	Avertissement "@" si les informations de email sont mal saisies
			un bouton "mypage" apparaît	Avertissement donné si toutes les cases ne sont pas remplies
		valeurs email & password correctes	un bouton "messages" apparaît	Un message d'erreur s'affiche : "vos identifiants sont incorrects"
		Tout les champs du formulaires sont remplis	un bouton "filter" apparaît	
Fermer le formulaire de login	Cliquez n'importe où sur l'écran		Fermeture du formulaire de login	

Page principale (connecté)				
Page personnelle Click bt " MyPage"	Click bt "Save Changes"	De nouvelles données ont-elles été saisies conformément aux types de données	Inscription des informations	Avertissement "@" si les informations de email sont mal saisies
			Si aucune information n'est saisie, les anciennes informations sont conservées.	Avertissement si les informations saisies ne correspondent pas aux types de données. Avertissement si les informations saisies ne correspondent pas aux types de données. Email, postal code, building number.
	Click bt "Post Ad"	Recevez un avertissement si les statuts de prix et de stock restent vides	La publicité est enregistrée dans la base de données.	Si les informations sur les prix et les stocks sont incomplètes, l'annonce ne sera pas enregistrée dans la base de données.
			Les photos sont enregistrées dans la base de données sous forme d'URL.	
				L'utilisateur peut voir les annonces qu'il a publiées sur sa page.

Se déconnecter	click bt "LOGOUT"	Avoir déjà créé un compte et être connecté dessus	bouton "LOGOUT" se transforme en bouton "LOGIN"	Le bouton est invisible
			bouton "MyPage" disparaît	
			bouton "Messages" disparaît	
			bouton "Filter" disparaît	
			Les produits dans le panier sont supprimés.	
			Le bouton d'adresse sur la carte de l'utilisateur disparaît.	
			sur la page de détails de l'annonce deviennent	

Page Messages	Click bt " Messages"	Avoir déjà créé un compte et être connecté dessus	L'utilisateur peut voir et supprimer tous les messages précédents.	
		L'utilisateur peut cliquer sur le bouton Supprimer les messages		

Option de filtrage	Click bt " Filter"	Avoir déjà créé un compte et être connecté dessus		
		L'utilisateur dispose de plusieurs options pour filtrer.	L'utilisateur peut voir les annonces filtrées sur la carte et sur la page d'accueil en fonction des options de filtrage.	
		L'utilisateur peut filtrer en fonction des données du tableau dans la base de données.		
	Click bt "apply filter"	Ce bouton permet de filtrer en fonction des critères saisis par l'utilisateur.		
	Click bt "close"	Ce bouton ferme la fenêtre pop-up de filtrage.		
Option de search	Click bt "Search"	Chaque utilisateur peut effectuer une recherche même s'il ne se connecte pas.	Les produits recherchés sont recherchés via le titre et amenés à la page d'accueil. De plus, les publicités sur Ahrita sont concernées par cette recherche.	

3 Implémentation

3.1 Choix techniques

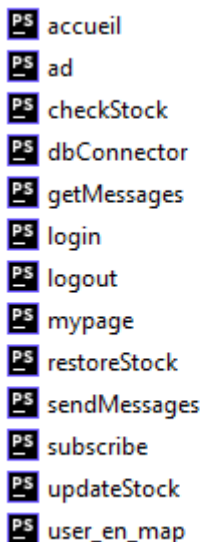
- Frontend



PS accueil
ad
PS ad
index
PS map
messages
PS messages_panel
myMessages
PS myMessages
mypage
PS mypage
PS panier
panier
PS panier
PS script
PS sendMessages
PS style
PS user_en_map

Développé en utilisant HTML, CSS et JavaScript. L'interface utilisateur est enrichie d'éléments dynamiques. Par exemple, des formulaires ont été utilisés pour mettre à jour les informations personnelles des utilisateurs ou pour publier de nouvelles annonces.

- Backend

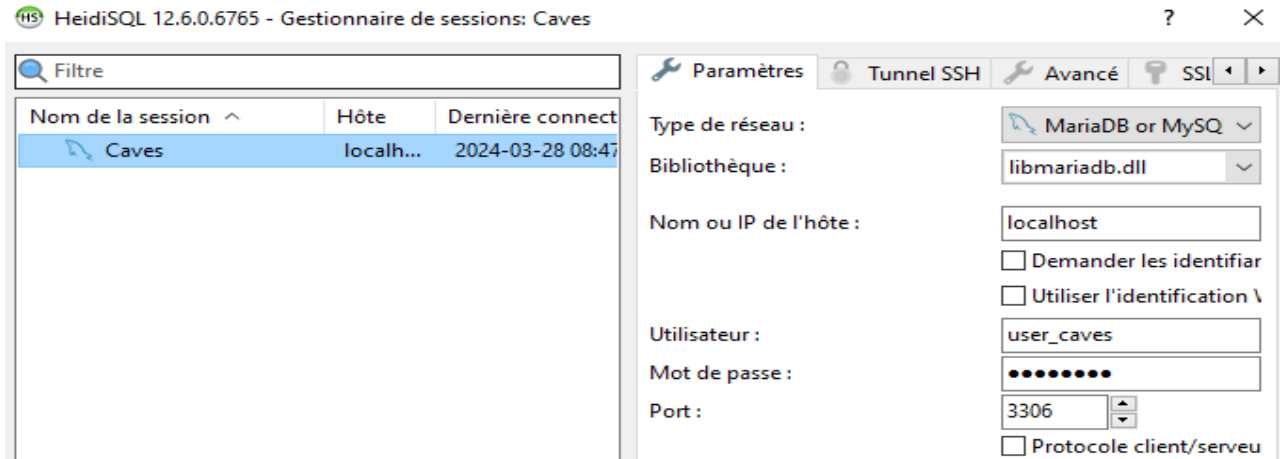


PS accueil
PS ad
PS checkStock
PS dbConnector
PS getMessages
PS login
PS logout
PS mypage
PS restoreStock
PS sendMessages
PS subscribe
PS updateStock
PS user_en_map

Développé en langage PHP. Les transactions utilisateur, les interactions avec la base de données et les opérations côté serveur sont effectuées dans cette couche. Des fonctions telles que le traitement des données, la vérification des utilisateurs et le traitement des publications sont exécutées ici.

- Système de base de données

La base de données MySQL a été utilisée pour le projet.



Tableaux : La base de données est conçue à partir de divers tableaux tels que les utilisateurs, les publicités, les produits, les adresses et les photos. Le modèle relationnel a été utilisé pour maintenir l'intégrité des données et assurer leur cohérence.

caves		240.0 KiB
★	announcements	48.0 KiB
	categories	32.0 KiB
	comments	32.0 KiB
	messages	64.0 KiB
	photos	32.0 KiB
	products	16.0 KiB
	users	16.0 KiB

Nom	Lignes	Taille
announcements	8	48.0 KiB
categories	6	32.0 KiB
comments	0	32.0 KiB
messages	0	64.0 KiB
photos	8	32.0 KiB
products	8	16.0 KiB
users	2	16.0 KiB

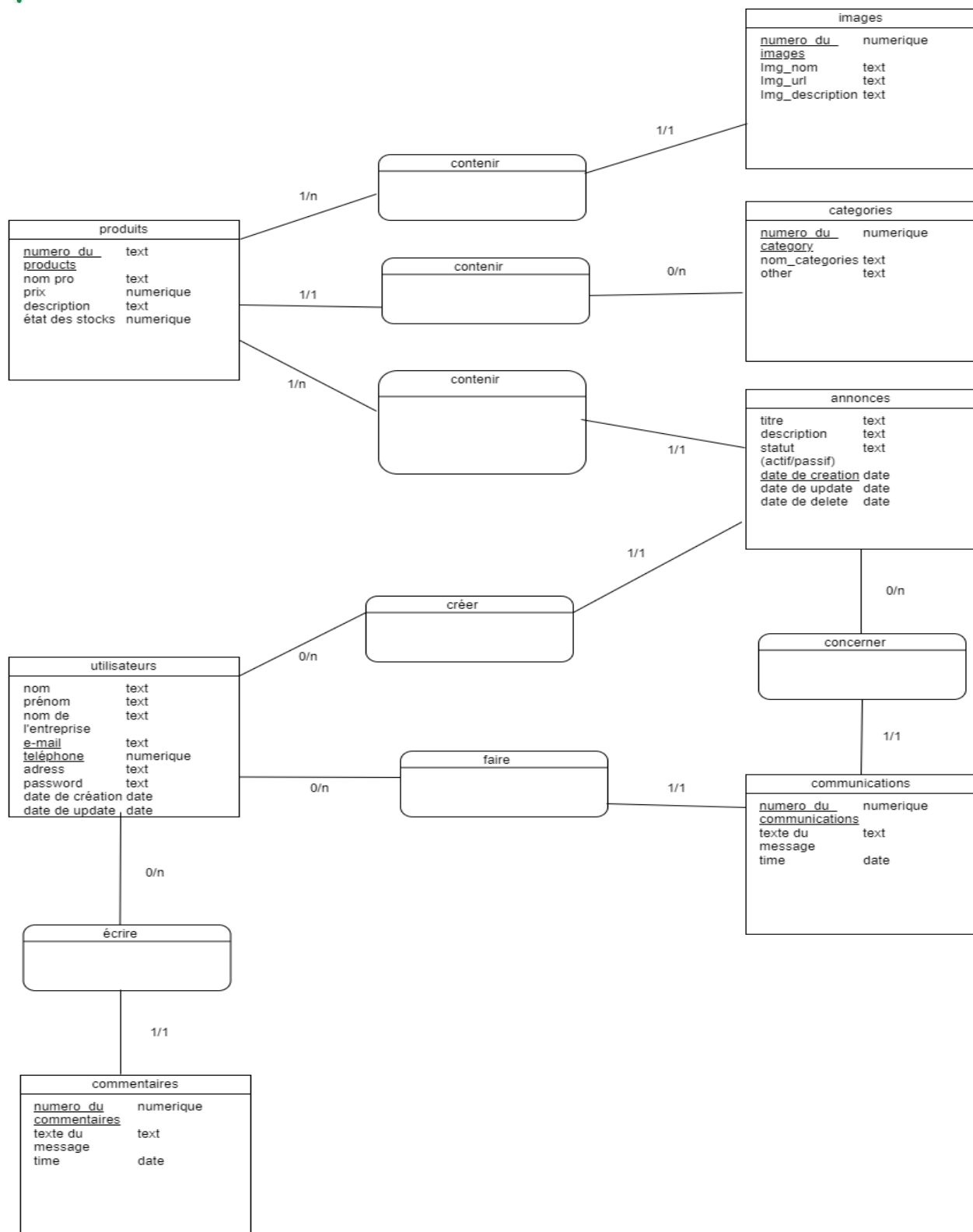
Après l'installation de la base de données, des efforts ont été déployés pour organiser la fonction de messagerie et la table des messages a été mise à jour comme indiqué ci-dessous.

#	Nom	Type de données	Taille/Ensem...	Non signé	NULL autorisé	ZEROFILL	Par défaut
1	id	INT	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT
2	buyer_id	INT	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
3	seller_id	INT	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
4	announcement_id	INT	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
5	message_text	VARCHAR	255	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
6	sent_time	TIMESTAMP		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	'CURRENT_TIMESTAMP'

```
CREATE TABLE IF NOT EXISTS `mydb`.`messages` (
  `id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `buyer_id` INT UNSIGNED NOT NULL,
  `seller_id` INT UNSIGNED NOT NULL,
  `announcement_id` INT UNSIGNED NOT NULL,
  `message_text` VARCHAR (255) NOT NULL,
  `sent_time` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
```

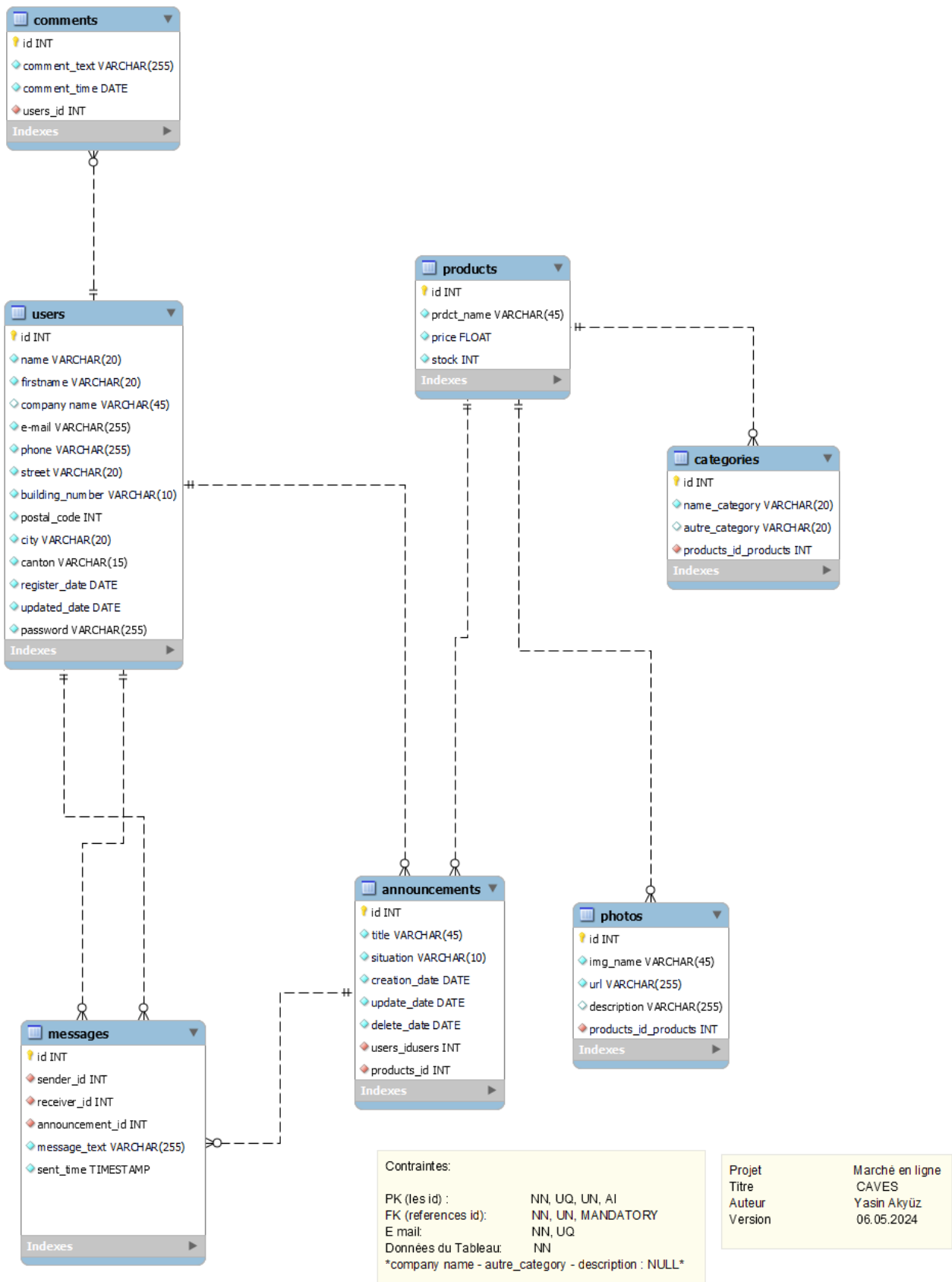
```
PRIMARY KEY (`id`),
INDEX `fk_messages_sender_id_idx` (`buyer_id` ASC) VISIBLE,
INDEX `fk_messages_receiver_id_idx` (`seller_id` ASC) VISIBLE,
INDEX `fk_messages_announcement_id_idx` (`announcement_id` ASC) VISIBLE,
CONSTRAINT `fk_messages_sender`
  FOREIGN KEY (`buyer_id`)
    REFERENCES `mydb`.`users` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
CONSTRAINT `fk_messages_receiver`
  FOREIGN KEY (`seller_id`)
    REFERENCES `mydb`.`users` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
CONSTRAINT `fk_messages_announcement`
  FOREIGN KEY (`announcement_id`)
    REFERENCES `mydb`.`announcements` (`id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

3.2 Modèle conceptuel de données



Projet	Marché en ligne
Titre	CAVES
Auteur	Yasin Akyüz
Version	06.05.2024 V1.0

3.3 Modèle Logique de données



Le schéma de la base de données est constitué de tables qui définissent diverses entités et les relations entre elles. Les entités de la base de données sont listées comme suit : produits, users,

comments, photos, catégories, annonces, et messages. Chaque table est équipée de différentes contraintes nécessaires pour assurer l'intégrité des données.

- Tables et Relations :

Chaque table possède une clé primaire (id) définie avec des contraintes telles que NN (Not Null), UQ (Unique), UN (Universal), AI (Auto-Incrément), ce qui assure que chaque enregistrement de la table soit identifié par une clé unique et auto-incrémentée.

Les clés étrangères (FK) se basent sur les id référencés et comprennent des contraintes NN (Not Null), UN (Universal), MANDATORY, indiquant que les références aux tables liées sont obligatoires et que chaque référence doit être unique.

Les adresses e-mail sont définies comme NN (Not Null) et UQ (Unique), signifiant que chaque utilisateur doit avoir une adresse e-mail unique.

Il est indiqué que certains champs peuvent être NULL, ce qui montre que ces zones sont optionnelles.

```
CREATE TABLE IF NOT EXISTS `mydb`.`messages` (
  `id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `buyer_id` INT UNSIGNED NOT NULL,
  `seller_id` INT UNSIGNED NOT NULL,
  `announcement_id` INT UNSIGNED NOT NULL,
  `message_text` VARCHAR (255) NOT NULL,
  `sent_time` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  INDEX `fk_messages_sender_id_idx` (`buyer_id` ASC) VISIBLE,
  INDEX `fk_messages_receiver_id_idx` (`seller_id` ASC) VISIBLE,
  INDEX `fk_messages_announcement_id_idx` (`announcement_id` ASC) VISIBLE,
  CONSTRAINT `fk_messages_sender`
    FOREIGN KEY (`buyer_id`)
      REFERENCES `mydb`.`users` (`id`)
      ON DELETE CASCADE
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_messages_receiver`
    FOREIGN KEY (`seller_id`)
      REFERENCES `mydb`.`users` (`id`)
      ON DELETE CASCADE
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_messages_announcement`
    FOREIGN KEY (`announcement_id`)
      REFERENCES `mydb`.`announcements` (`id`)
      ON DELETE CASCADE
      ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

- Normalisation et Critères BCNF :

Les tables conformes à la 1NF montrent une structure où les valeurs des colonnes sont atomiques et où il n'y a pas de groupes répétitifs.

Les conditions 2NF et 3NF sont supposées être remplies en considérant que tous les champs, à l'exception des clés étrangères, dépendent de la clé de la table.

La BCNF exige que tous les déterminants soient une partie de la clé candidate. Selon les informations

indiquées dans le schéma, chaque colonne id sert de clé primaire pour la table et les clés étrangères se réfèrent correctement aux clés des tables concernées.

users - Table										
Table Name: users										
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
name	VARCHAR(20)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
firstname	VARCHAR(20)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
company name	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
e-mail	VARCHAR(255)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
phone	VARCHAR(255)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
street	VARCHAR(20)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
building_number	VARCHAR(10)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
postal_code	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
city	VARCHAR(20)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
canton	VARCHAR(15)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
register_date	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
updated_date	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
password	VARCHAR(255)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

announcements - Table										
Table Name: announcements										
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
title	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
situation	VARCHAR(10)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
creation_date	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
update_date	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
delete_date	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
users_idusers	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
products_id	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Nous devons spécifier des contraintes lors de la saisie des données dans le tableau MLD. Vous pouvez voir des exemples de contraintes sélectionnées à la fois dans la description en bas du tableau mld et dans le tableau des utilisateurs et annonces donné en exemple.

3.4 Diagramme de Flux

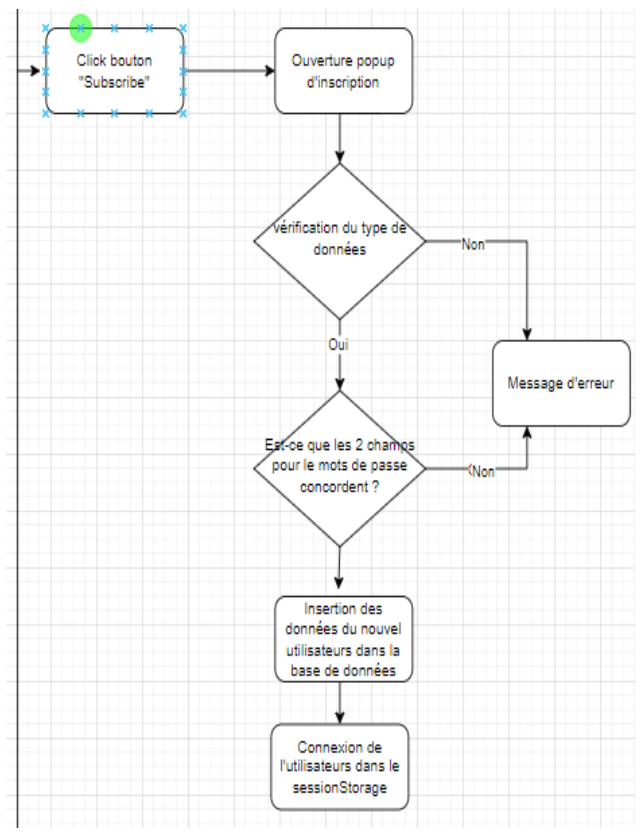
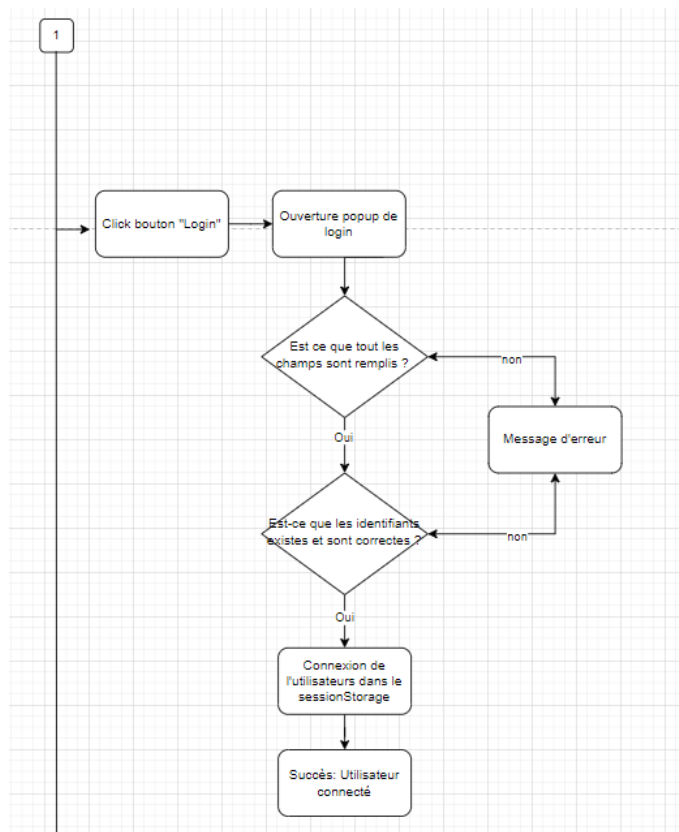
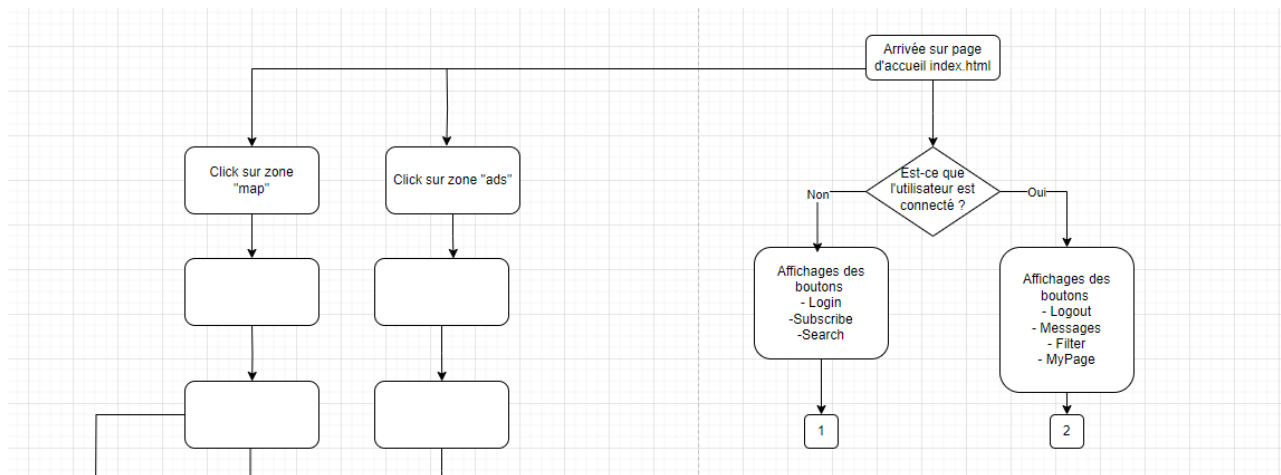
Ce diagramme de flux illustre comment les utilisateurs peuvent effectuer diverses opérations sur la plateforme, étape par étape. Il est composé de processus qui définissent les actions des utilisateurs et comment le système y répond.

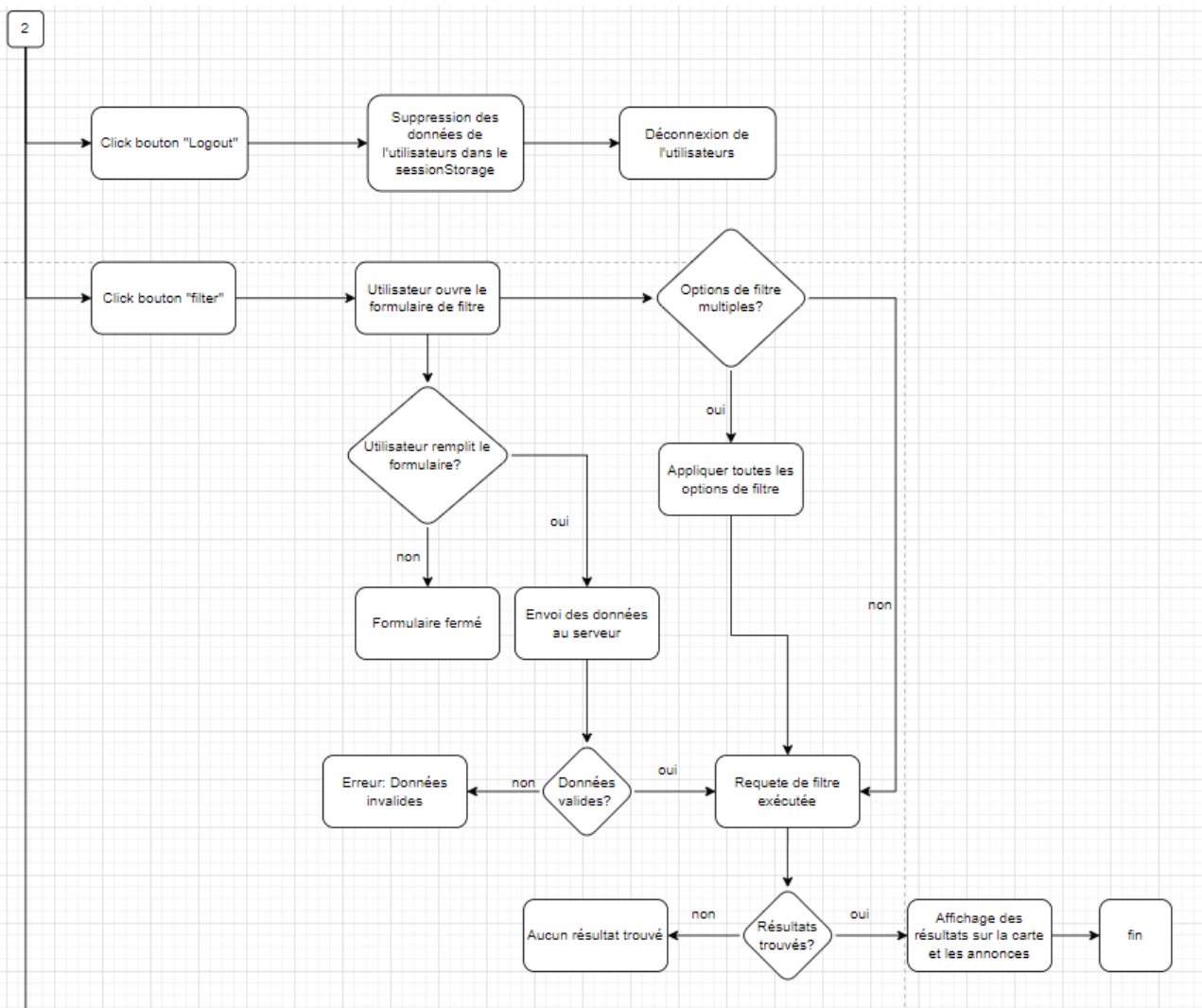
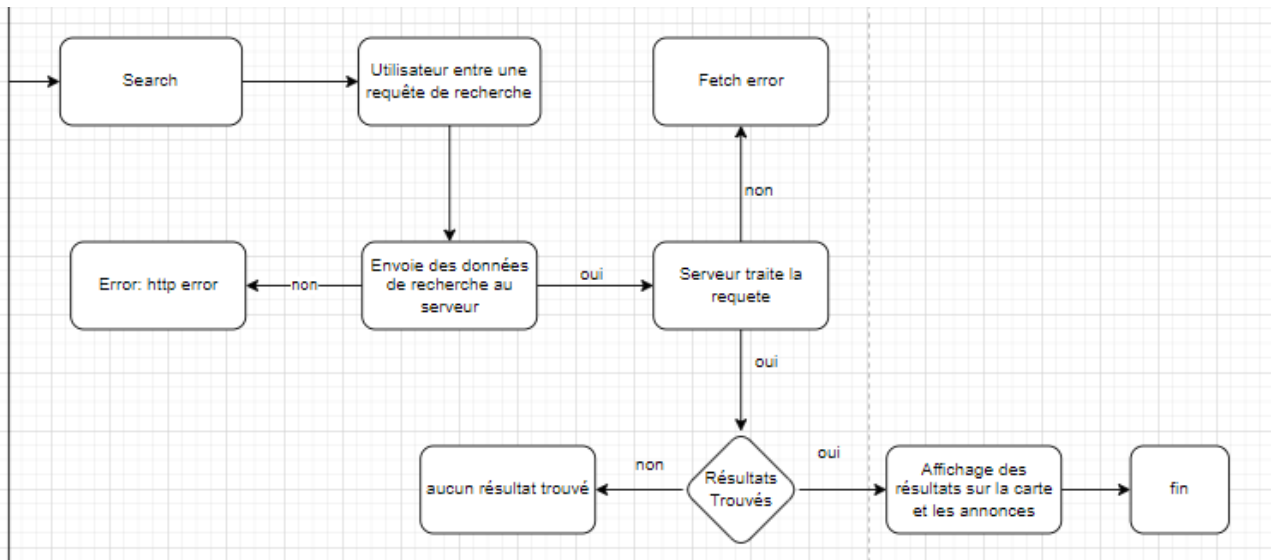
Pour décrire les processus affichés dans le diagramme par exemple :

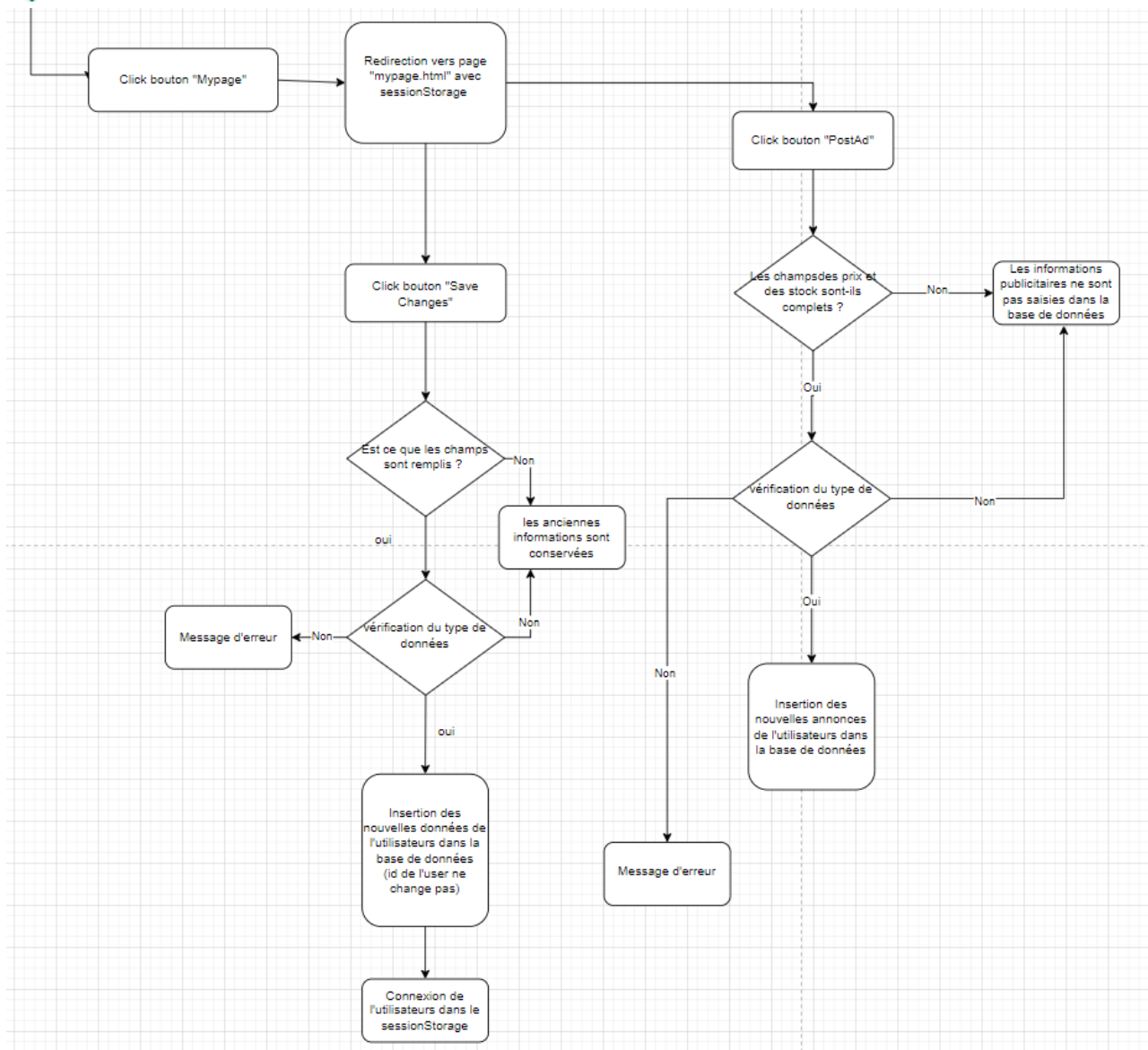
- L'utilisateur entre sur le site et peut voir divers boutons : connexion (login), inscription, etc.
- L'utilisateur démarre le processus de connexion en cliquant sur le bouton "Login".
- Si les informations de l'utilisateur sont correctes, le système reconnaît l'utilisateur et ouvre la session. Sinon, il répond avec un message d'erreur.

- Une fois la session ouverte, diverses options sont proposées à l'utilisateur : mise à jour des informations personnelles, envoi de messages, déconnexion, etc.
- L'utilisateur peut fermer la session en cliquant sur le bouton "Log out".

Les actions de l'utilisateur et les réponses du système dépendent de certaines conditions, telles que la véracité du nom d'utilisateur et du mot de passe, le remplissage des champs du formulaire et l'exactitude des informations dans la base de données. Chaque étape dirige à la suivante en fonction des conditions positives ou négatives.







3.5 Ergonomie de site (Bastien et Scapin)

3.6 Points techniques spécifiques

3.6.1 Enregistre les images

La manière dont les photos sont stockées et gérées sur les sites Web dépend souvent des besoins et des exigences de performances de l'application. Dans votre cas, vous envisagez de stocker des photos sous forme de texte dans la base de données plutôt que de les stocker sous forme de fichiers physiques. Les deux méthodes présentent des avantages et des inconvénients :

Stockage sur le système de fichiers :

Avantages :

- Le système de fichiers est généralement optimisé pour stocker des fichiers multimédias volumineux.
- Le serveur Web peut servir directement des fichiers statiques, ce qui est souvent plus rapide que d'extraire des données de la base de données.
- Il permet de déplacer facilement les images vers des services tels que CDN (Content Delivery Network).

Désavantages :

- S'il existe de nombreux fichiers, le système de fichiers peut devenir difficile à gérer.
- Les noms et chemins de fichiers doivent être conservés dans la base de données, ce qui peut entraîner des problèmes de synchronisation.
- Les opérations de sauvegarde et de migration peuvent devenir plus complexes.

Stockage dans la base de données (par exemple sous forme de texte BLOB ou encodé en base64) :

Avantages :

- Les métadonnées et le contenu des fichiers sont conservés au même endroit, ce qui peut faciliter la gestion.
- La sauvegarde et la migration peuvent être plus simples car toutes les données sont consolidées dans la base de données.

Désavantages :

- Les bases de données ne sont généralement pas optimisées pour stocker de grandes quantités de données binaires.
- Vous devez écrire du code supplémentaire pour servir le contenu du fichier, car le serveur Web ne peut pas le servir directement.
- La charge sur la base de données augmente, ce qui peut entraîner des problèmes de performances et d'évolutivité.

L'idée de stocker du texte dans la base de données est généralement privilégiée pour les petits fichiers ou un petit nombre de fichiers multimédias. Mais dans un système à grande échelle, il est plus courant de conserver physiquement les fichiers et leurs références dans la base de données.

Caves

caves

240.0 KiB

announcements

48.0 KiB

categories

32.0 KiB

comments

32.0 KiB

messages

64.0 KiB

photos

32.0 KiB

products

16.0 KiB

users

16.0 KiB

information_schema

mydatabase

caves.photos: 9 ligne(s) au total (exact)

#	id	img_name	url	descrip...	products_id_products
1	55	pomme.png	/Frontend/images/pomme.png		77
2	56	carrot.png	/Frontend/images/carrot.png		78
3	57	champignons_coop.jpg	/Frontend/images/champignons_coop.jpg		79
4	58	mais_coop.jpg	/Frontend/images/mais_coop.jpg		80
5	59	mais_coop.jpg	/Frontend/images/mais_coop.jpg		81
6	60	carrot_2_ysn.jpg	/Frontend/images/carrot_2_ysn.jpg		82
7	61	carrot_fruits_migros.jpg	/Frontend/images/carrot_fruits_migros.jpg		83
8	62	pomme.jpg	/Frontend/images/pomme.jpg		84
9	63	mais_coop.jpg	/Frontend/images/mais_coop.jpg		86

- Comment traiter le dossier sur internet

```
$finfo=newfinfo(FILEINFO_MIME_TYPE);  
$fileMimeType = $finfo->file($_FILES['url']['tmp_name'][$i]);
```

```
$finfo = new finfo(FILEINFO_MIME_TYPE);:
```

Crée une instance de la classe `finfo` et définit cette instance en mode de détermination de type MIME. La classe `finfo` est utilisée pour vérifier le type MIME des fichiers.

```
$fileMimeType = $finfo->file($_FILES['url']['tmp_name'][$i]); ;
```

En utilisant le nom temporaire du fichier téléchargé, détermine le type MIME de ce fichier. Le chemin `$_FILES['url']['tmp_name'][$i]` indique le chemin où le fichier téléchargé via le formulaire est temporairement stocké sur le serveur.

- L'extension PHP finfo doit être activée en PHP. ->php.ini

```
912     ;extension=ftp
913     extension=fileinfo
914     ;extension=gd2
```

3.6.2 Utilisation et fonctionnalité de la carte

Pourquoi une carte ?

Il a été ajouté au projet pour faciliter les fonctions de l'utilisateur sur l'application et augmenter l'intérêt pour l'application en offrant une expérience différente.

Sur cette carte, l'utilisateur peut visualiser les annonces et obtenir des petites informations en cliquant sur les annonces. Si l'utilisateur souhaite voir les détails de l'annonce, il peut accéder aux détails de l'annonce via les boutons sur la carte.

Il est également affecté par les processus de filtrage et de recherche de la carte et affiche les publicités des utilisateurs sur la carte en fonction des détails de la recherche.

Puisqu'il sera plus facile pour l'utilisateur de voir quelle annonce est la plus proche de lui en regardant les annonces, une fonctionnalité a été ajoutée pour permettre à l'utilisateur connecté de voir sa position sur la carte.

- Sélection de la carte et critères de sélection.

Il est possible de travailler avec de nombreux types de cartes et d'intégrer ces cartes dans la page. Tout d'abord, la première chose qui me vient à l'esprit est d'utiliser Google Maps. Cependant, en termes d'utilisation, j'ai décidé d'essayer un nouveau système et d'utiliser le système de cartographie "MapBox GL JS", qui est un système de cartographie différent que nous concevrons en fonction du budget et de la manière dont nous l'utilisons.

Mapbox a gagné en popularité lorsque Google a décidé de monétiser son API Maps et a augmenté ses prix de plus de 1 400 %. Cela signifie que l'API Google Maps n'est pas la meilleure option pour de nombreuses grandes entreprises, et elles commencent à chercher d'autres options.

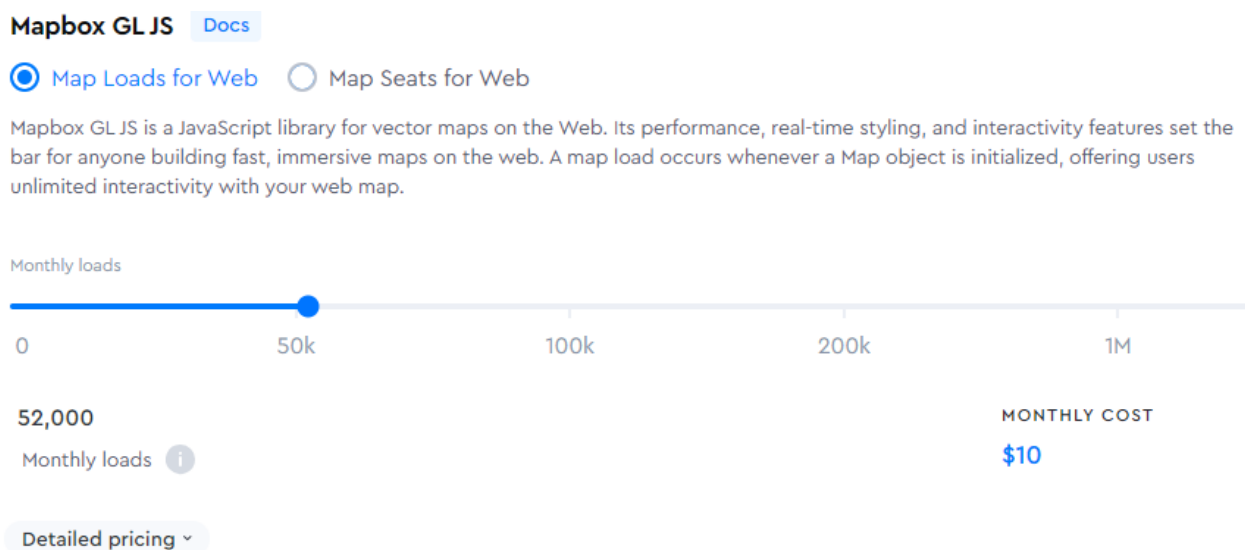
Actuellement, des applications comme Shopify, Facebook, Pinterest, Foursquare, DHL, et Airbnb sont toutes prises en charge par Mapbox. Cela peut être attribué au fait que le modèle tarifaire est plus attractif pour un grand nombre de demandes de cartes.

- Tarifs

Les deux services sont gratuits jusqu'à un certain point, puis commencent à être facturés. Le modèle de tarification de ces API est un peu complexe, le modèle de tarification de l'API Google est basé sur l'utilisation ou les demandes tandis que l'API Mapbox propose différentes options de tarification pour les utilisateurs individuels.

Mapbox est gratuit pour jusqu'à 25000 utilisateurs et 50000 installations Web; Jusqu'à 28 000 installations par mois sont gratuites lorsque vous payez 5\$ pour 1000 entre 50001 et 100000, et le prix baisse à mesure que le volume augmente.

Google Maps est plus cher, Dynamic Maps coûtant 7\$ pour 1000 requêtes.



- Services associés

L'API Google Maps comprend divers services tels que Images satellite et StreetView. Toutes ces fonctionnalités contribuent à l'utilité de cette API cartographique. Mapbox doit encore développer et ajouter des plugins.

Mapbox vise à fournir une plateforme open source et utilise les données d'OpenStreetMap. Les données cartographiques et les services API ne sont pas aussi riches que Google Maps.

- Option de recherche

Google Maps et Mapbox disposent tous deux de fonctionnalités de suggestion automatique dans la fonction de recherche. Ces deux API permettent aux utilisateurs de rechercher différents emplacements en utilisant des coordonnées (latitude et longitude) ou en saisissant le nom du lieu dans la barre de recherche.

Bien que la fonction de recherche soit entièrement gratuite dans MapBox, la fonction de recherche dans GoogleMaps est également payante après un certain point.

- Fonctionnalités des marqueurs de carte

GoogleMaps a des fonctionnalités de style plus limitées que MapBox.

L'API Google Maps n'est pas en mesure d'afficher un grand nombre de marqueurs à la fois et peut également présenter des problèmes de performances de l'interface utilisateur, tels que la vitesse et la fluidité du rendu lors du zoom sur les cartes.

En revanche, Mapbox est plus performant car il utilise des tuiles vectorielles et fournit une API Mapbox GL JS très flexible.

- Personnalisation/Thème

Google Maps et Mapbox soutiennent l'idée de personnalisation, mais la personnalisation est la principale force de Mapbox.

On peut dire que Google Maps est un peu rigide ou moins flexible en termes de personnalisation. Google Maps vous oblige à utiliser sa couche de base par défaut, mais Mapbox n'a pas une telle restriction.

Mapbox fournit également un éditeur de carte intégré appelé TileMill ; Il s'agit d'un service payant mais aura accès aux bibliothèques de cartes et à la possibilité de télécharger des styles personnalisés. Mapbox est allé un peu plus loin à cet égard. Il ouvre toute votre plateforme à la personnalisation.

Vous pouvez utiliser Mapbox Studio, un outil puissant avec lequel vous pouvez personnaliser votre carte et créer des ensembles de données réutilisables. Vous pouvez obtenir des détails très détaillés avec la sculpture et créer des designs uniques.

Par exemple, nous pouvons ajouter un bouton à la carte et passer à l'autre conception de carte que nous avons conçue. Nous pouvons nommer cette carte « Batman », définir un fond noir et afficher l'emplacement de l'utilisateur avec le symbole "Batman". Ou nous pouvons concevoir une carte rouge et blanche et la nommer « My Wonderful Switzerland ».

Principe de Fonctionnement

Pour mieux comprendre comment ces fonctionnalités sont mises en œuvre dans le projet, examinons quelques aspects techniques clés :

- AccessToken

Il s'agit d'une clé d'authentification requise pour accéder aux services Mapbox.

Il surveille l'utilisation de l'API et garantit que les demandes proviennent d'un utilisateur ou d'une application autorisée. Pour authentifier notre application sur Mapbox, nous devons inclure notre AccessToken dans chaque requête API.

À ce stade, j'ai également déterminé le style de la carte, saisi les coordonnées du pays, spécifié l'emplacement qu'elle afficherait lors de sa première ouverture, et ajusté le niveau de zoom de la carte à travers le pays.

```
mapboxgl.accessToken = 'pk.eyJ1IjoieWFzaW5ha3l1eiIsImEiOiJjbHRjcHh4bXowMm40MmtvOXJtdDRzMm1lIn0.s
const map = new mapboxgl.Map({
  container: 'map',
  // Nous pouvons choisir notre style MapBox ou le préparer à Mapbox Studio et l'ajouter nous-
  style: 'mapbox://styles/mapbox/streets-v12',
  center: [8.2275, 46.8182],
  zoom: 6
});
```

- Affichage des Annonces sur la Carte avec Mapbox GL JS

Pour mieux comprendre comment les annonces sont affichées comme des boutons sur la carte à l'aide de Mapbox GL JS, nous pouvons décomposer le processus étape par étape. Cela inclut la récupération des données, le géocodage des adresses, la création des marqueurs, et l'ajout des pop-ups interactifs.



```
publicité : {ads: Array(7)}
  ads: Array(7)
    0: {
      building_number: "28"
      canton: "Vaud"
      city: "Yverdon-les-Bains"
      creation_date: "2024-05-07 10:46:43"
      id: "76"
      name_category: "fruits"
      photo_url: "/Frontend/images/pomme.jpg"
      postal_code: "1400"
      product_name: "pomme"
      product_price: "3"
      product_stock: "3"
      situation: "pommesa"
      street: "Rue de l'industrie"
      title: "pomme"
    }
    1: {id: '72', title: 'mais', situation: 'disponible', creation_date: '2024-05-07 09:22:46', street: 'Aarberggasse', ...}
```

- Récupération des Données d'Annonces

Le processus commence par une requête HTTP envoyée au script accueil.php sur le serveur backend. Ce script est responsable de fournir les données des annonces, typiquement en format JSON, qui inclut des détails essentiels tels que l'adresse, le prix, le titre, et d'autres métadonnées relatives à chaque annonce.

```
fetch( input: '/Backend/accueil.php') Promise<Response>
  .then(response : Response => response.json()) Promise<any>
  .then(data => {
    data.ads.forEach(ad => {
      // Processus pour chaque annonce
    });
  })
  .catch(error => console.error('Error:', error));
```

- Géocodage des Adresses

Pour chaque annonce récupérée, l'adresse est transformée en coordonnées géographiques (latitude et longitude) grâce à l'API de géocodage de Mapbox. Ce processus est essentiel pour pouvoir placer précisément chaque annonce sur la carte. L'adresse est d'abord encodée pour être compatible avec l'URL et ensuite une requête de géocodage est envoyée.

```
const query = encodeURIComponent(`${ad.street} ${ad.building_number}, ${ad.postal_code} ${ad.city}, $`
fetch(`https://api.mapbox.com/geocoding/v5/mapbox.places/${query}.json?access_token=${mapboxgl.access`
  .then(response => response.json())
  .then(data => {
    if (data.features.length > 0) {
      const coordinates = data.features[0].center;
```

Carte avant et après connexion

Fetching user address

Response received

Data: ▼ {error: 'User not logged in.'} ⓘ
error: "User not logged in."
▶ [[Prototype]]: Object

Login successful

Fetching user address

Response received

Data: ▼ {street: 'Aarberggasse', building_number: '53', postal_code: '3011', city: 'Bern', canton: 'Bern'} ⓘ
building_number: "53"
canton: "Bern"
city: "Bern"
postal_code: "3011"
street: "Aarberggasse"
▶ [[Prototype]]: Object

Geocoding data:



Geocoding data:

```
{type: 'FeatureCollection', query: Array(5), features: Array(1), attribution: 'NOTICE: © 2024 Mapbox and its suppliers. All right. are.'}
  attribution: "NOTICE: © 2024 Mapbox and its suppliers. All rights reserved. Use of this data is subject to the Mapbox Terms of :
  features: Array(1)
    0:
      address: "53"
      center: Array(2)
        0: 7.44191
        1: 46.949535
        length: 2
      [[Prototype]]: Array(0)
      context: Array(6)
        0: {id: 'neighborhood.2173996', mapbox_id: 'dXJuOm1ieHBsYzpjU3dz', text: 'Obere Altstadt'}
        1: {id: 'postcode.7024172', mapbox_id: 'dXJuOm1ieHBsYzpheTRz', text: '3011'}
        2: {id: 'locality.9128492', mapbox_id: 'dXJuOm1ieHBsYzppMG9z', wikidata: 'Q56448261', text: 'Rotes Quartier'}
        3: {id: 'place.2353196', mapbox_id: 'dXJuOm1ieHBsYzpjK2dz', wikidata: 'Q70', text: 'Bern'}
        4: {id: 'region.9260', mapbox_id: 'dXJuOm1ieHBsYzpkQ3c', wikidata: 'Q11911', short_code: 'CH-BE', text: 'Bern'}
        5: {id: 'country.8748', mapbox_id: 'dXJuOm1ieHBsYzpjJXc', wikidata: 'Q39', short_code: 'ch', text: 'Switzerland'}
        length: 6
      [[Prototype]]: Array(0)
      geometry:
        coordinates: (2) [7.44191, 46.949535]
        type: "Point"
        [[Prototype]]: Object
        id: "address.5713260724030922"
        place_name: "Aarberggasse 53, 3011 Bern, Switzerland"
        place_type: ["address"]
        properties: {accuracy: 'rooftop', mapbox_id: 'dXJuOm1ieGFkcjpkYTgzYWUwMS1iM2MxLTQxNGQ0tOGNjZC1jMjE2Y2I0NzNmOGI'}
```

Une fois l'utilisateur connecté, le bouton rouge qui apparaît sur la page d'accueil indique le profil de l'utilisateur. On peut observer que MapBox prend l'adresse, la transforme en coordonnées et l'ajoute à la carte.

Si vous souhaitez afficher l'emplacement de l'adresse ci-dessus sur des cartes, vous pouvez utiliser ces coordonnées. « 46°94'95.5"N 7°44'19.1"E »

- Création des Marqueurs

Une fois les coordonnées obtenues, un marqueur est créé pour chaque annonce. Mapbox GL JS permet d'ajouter des marqueurs personnalisés à des positions spécifiques sur la carte. Chaque marqueur est ensuite associé à un popup qui contient les détails de l'annonce. Ce popup est conçu pour s'afficher lorsque l'utilisateur clique sur le marqueur.

Ajout d'annonces ;

```
const newMarker = new mapboxgl.Marker()
  .setLngLat(coordinates)
  .setPopup(new mapboxgl.Popup({offset: 25}).setHTML(descriptionHTML))
  .addTo(map);
markers.push(newMarker);
}
```

Ajout de l'utilisateur connecté ;

```
if(userMarker){
  userMarker.setLngLat(coordinates);
}else {
  const el : HTMLDivElement = document.createElement( tagName: 'div');
  el.className = 'user-marker';
  el.style.backgroundColor = 'red';
  el.style.width = '15px';
  el.style.height = '15px';
  el.style.borderRadius = '30%';
  userMarker = new mapboxgl.Marker(el)
    .setLngLat(coordinates)
    .addTo(map);
}
console.log('Marker added');
} else {
```

Le contenu HTML du popup est conçu pour être interactif. Lorsque l'utilisateur clique sur le popup, il est redirigé vers une page de détails spécifique à l'annonce (ad.html), ce qui améliore l'expérience utilisateur en fournissant des informations plus détaillées sur chaque annonce.

Ce processus entier permet de transformer des données statiques d'annonces en éléments interactifs sur une carte dynamique, rendant l'interface utilisateur plus attrayante et informative. L'utilisation de Mapbox GL JS pour ces tâches assure une intégration fluide et efficace, profitant de ses capacités de personnalisation et de sa performance robuste.

Utilisation Sur L'application

- Affichage de publicités

Les annonces saisies par les utilisateurs et l'utilisateur connecté sont affichées avec des boutons sur la carte, comme indiqué ci-dessus. Des précautions ont été prises pour s'assurer que le bouton de l'utilisateur n'était pas le même que le bouton des publicités. L'utilisateur qui clique sur les annonces pourra accéder aux détails de l'annonce en cliquant sur le bouton.

- Gestion des Marqueurs et Annonces

Dans mon application, les fonctionnalités de recherche et de filtrage interagissent étroitement avec la carte Mapbox GL JS pour afficher efficacement les annonces pertinentes sous forme de marqueurs. Ce processus dynamique permet une interaction utilisateur fluide et met à jour les annonces et les marqueurs en temps réel selon les critères spécifiés.

- Filtrage et recherche sur la carte

En cas de filtrage et de recherche, les boutons sur la carte seront également affectés par ces opérations et les résultats seront affichés avec de nouveaux boutons sur la carte.

Nettoyage des Marqueurs et Annonces Existantes :

Avant de charger de nouvelles annonces basées sur les critères de recherche ou de filtrage, je commence toujours par nettoyer les marqueurs et les annonces existants. Cela évite l'encombrement sur la carte et permet une meilleure visibilité des nouvelles données pertinentes.

```
function clearAdsAndMarkers() : void {  
  
    const adsContainer : HTMLElement = document.getElementById( 'ads-container' );  
    adsContainer.innerHTML = '';  
  
    console.log("Starting to clear markers. Total markers before clearing:", markers.length);  
    markers.forEach(marker => marker.remove());  
    markers = [];  
    console.log("All markers removed, markers array reset.");  
    map.jumpTo({center: map.getCenter()}); // Recentrer la carte  
}
```

Récupération des Annonces Filtrées :

Après avoir nettoyé la carte et l'interface utilisateur, j'envoie une requête au backend avec les

paramètres de recherche ou de filtrage. Cette requête est conçue pour récupérer les annonces filtrées qui doivent être affichées.

```
async function fetchAdsFiltered(params) : Promise<void> {
  await clearAdsAndMarkers(); // Effacez les publicités et les marqueurs existants avant d'en récupérer
  const queryParams : string = new URLSearchParams(params).toString();
  try {
    const response : Response = await fetch(input: `/Backend/accueil.php?${queryParams}`);
    if (!response.ok) throw new Error(`HTTP error! Status: ${response.status}`);
    const result = await response.json();
    console.log("Ads fetched successfully:", result.ads);
    displayAds(result.ads); // Afficher de nouvelles annonces
    markers.forEach(marker => marker.remove()); // Supprimer les marqueurs existants
    markers = []; // Réinitialiser le tableau de marqueurs
    result.ads.forEach(ad => {
      addMarker(ad); // Add new markers for each ad
    });
  }
}
```

Affichage des Annonces et Ajout de Nouveaux Marqueurs :

Chaque annonce récupérée est affichée dans le conteneur des annonces sur ma page web, et je crée et ajoute un nouveau marqueur sur la carte pour chaque annonce.

```
function displayAds(ads) : void {
  const adsContainer : HTMLElement = document.getElementById( elementId: 'ads-container');
  ads.forEach(ad => {
    console.log("Ad Data:", ad);
    const adElement : HTMLDivElement = document.createElement( tagName: 'div');
    adElement.innerHTML = `
```

```
function addMarker(ad) : void {
  const fullAddress : string = `${ad.street} ${ad.building_number}, ${ad.postal_code} ${ad.city}, ${ad.canton}`;
  const query : string = encodeURIComponent(fullAddress);
  console.log("Adding marker for ad:", ad.title, "with query:", query);
  fetch( input: `https://api.mapbox.com/geocoding/v5/mapbox.places/${query}.json?access_token=${mapboxgl.accessToken}`
    .then(response : Response => response.json()) Promise<any>
    .then(data => {
      if (data.features.length > 0) {
        const coordinates = data.features[0].center;
```

```
const marker = new mapboxgl.Marker()
  .setLngLat(coordinates)
  .setPopup(new mapboxgl.Popup({offset: 25}).setHTML(descriptionHTML))
  .addTo(map);
markers.push(marker); // Cette étape est très importante, on ajoute les marqueurs à
console.log("Marker added, total markers now:", markers.length);
```

Comme le montre l'exemple ci-dessous, les fonctions cartographiques sont directement affectées par les cas d'utilisation des options de recherche et de filtrage.

Les publicités sont supprimées, de nouvelles publicités sont trouvées, toutes les opérations d'ajout sont répétées et ajoutées à la carte comme un nouveau bouton.

```
publicité : ▶ {ads: Array(7)} sc
Starting to clear markers. Total markers before clearing: 7 ac
All markers removed, markers array reset. ac
Ads fetched successfully: ac
▼ (2) [{...}, {...}]
  ▶ 0: {id: '76', title: 'pomme', situation: 'pommeysa', creation_date: '2024-05-07 10:46:43', street: 'Rue de l'industrie', ...}
  ▶ 1: {id: '77', title: 'pomme', situation: 'good', creation_date: '2024-05-07 11:07:15', street: 'Rue de la Mère', ...}
    length: 2
  ▶ [[Prototype]]: Array(0)
Ad Data: ac
  ▶ {id: '76', title: 'pomme', situation: 'pommeysa', creation_date: '2024-05-07 10:46:43', street: 'Rue de l'industrie', ...}
Ad Data: ▶ {id: '77', title: 'pomme', situation: 'good', creation_date: '2024-05-07 11:07:15', street: 'Rue de la Mère', ...} ac
Adding marker for ad: pomme with query: Rue%20de%20l'industrie%2028%2C%201400%20Yverdon-les-Bains%2C%20Vaud ac
Adding marker for ad: pomme with query: Rue%20de%20la%20Mère%2029%2C%201020%20Renens%2C%20Vaud ac
Marker added, total markers now: 1 acc
Marker added, total markers now: 2 acc
```



Ces intégrations des fonctionnalités de recherche et de filtrage garantissent une interaction dynamique et efficace avec la carte Mapbox GL JS dans mon projet. Elles permettent de gérer l'affichage et la mise à jour des annonces et des marqueurs en fonction des critères de l'utilisateur, assurant ainsi une expérience utilisateur améliorée et réactive.

3.6.3 Fonctions de Filtrage et de Recherche

Chaque utilisateur peut voir les publicités sur l'application et cliquer sur les publicités pour voir les détails de la publicité.

Cependant, s'il existe des milliers d'annonces, l'utilisateur doit pouvoir trouver l'annonce qu'il recherche selon les conditions de recherche.

Dans ce contexte, les opérations de filtrage et de recherche ajoutées au projet augmenteront l'expérience utilisateur et apporteront l'une des contributions les plus importantes au fonctionnement de l'application. Ce travail que nous avons fait sur le projet

- Processus de filtration

Un écouteur d'événement de clic est affecté à l'élément filter-btn. Lorsque ce bouton est cliqué, la visibilité de l'élément appelé filtre-popup est activée et un formulaire de filtrage est présenté à l'utilisateur.

Un écouteur d'événement d'envoi est affecté au formulaire filter-form. Lorsque le formulaire est envoyé, les valeurs de titre, de nom de catégorie (name_category), de prix minimum et de prix

maximum saisies par l'utilisateur sont prises en compte et ces valeurs sont données en tant que paramètres à la fonction `fetchAdsFiltered` et les publicités sont extraites en les filtrant via l'API.

```
document.getElementById( elementId: 'filter-btn').addEventListener( type: 'click', listener: function() : void {
    document.getElementById( elementId: 'filter-popup').style.display = 'block'; // Popup à göster
});
```

Ce code écoute l'événement de clic sur l'élément avec l'identifiant 'filter-btn' et affiche un formulaire de filtrage ('filter-popup') pour l'utilisateur.

- Traitement du Formulaire de Filtrage et Récupération des Annonces

```
document.getElementById( elementId: 'filter-form').addEventListener( type: 'submit', listener: async function(e :
    e.preventDefault();
    // Obtenir des critères de filtrage
    const title = document.getElementById( elementId: 'title').value;
    const categoryName = document.getElementById( elementId: 'name_category').value;
    const minPrice = document.getElementById( elementId: 'min-price').value;
    const maxPrice = document.getElementById( elementId: 'max-price').value;
    await fetchAdsFiltered( params: {
        title: title,
        name_category: categoryName,
        min_price : minPrice,
        max_price : maxPrice
    });
```

Ce fragment intercepte la soumission du formulaire 'filter-form', évite le rechargement de la page, récupère les valeurs des champs du formulaire, et appelle `fetchAdsFiltered` avec ces valeurs comme paramètres pour filtrer les annonces.

- Est-il nécessaire que tous les champs de filtre soient remplis ?

Il n'est pas obligatoire de remplir tous les champs du formulaire de filtre. Les utilisateurs peuvent remplir uniquement les champs pertinents pour appliquer le filtre désiré. Cette flexibilité est gérée là où la fonction `fetchAdsFiltered` est appelée. L'objet des paramètres de filtre (`{ title, name_category, min_price, max_price }`) accepte que certaines de ces valeurs puissent être vides, et la requête API est dynamiquement construite avec ces valeurs.

```
async function fetchAdsFiltered(params) : Promise<void> {
    await clearAdsAndMarkers(); // Effacez les publicités et les marqu
    const queryParams : string = new URLSearchParams(params).toString();
    try {
```

Ce code convertit les valeurs obtenues du formulaire de filtre en paramètres de requête URL. Si une valeur est absente, le paramètre correspondant ne figure pas dans la chaîne de requête.

- Comment récupérons-nous les prix min-max de la base de données ?

Les prix minimum et maximum sont récupérés des champs de saisie dans le formulaire de filtrage. L'utilisateur spécifie ces valeurs, qui sont ensuite passées comme paramètres à la requête SQL pour filtrer les annonces selon ces limites de prix.

La condition `WHERE` dans la requête SQL utilise ces valeurs pour limiter les résultats aux annonces dont les prix se situent dans l'intervalle spécifié.

- Opérations de Recherche

```
document.getElementById( 'search-form').addEventListener( type: 'submit', listener: function(e : S
    e.preventDefault();
    const searchTerm = document.getElementById( 'search-input').value;
    fetchAdsFiltered( params: { title: searchTerm })
});
```

Ce code gère la soumission du formulaire de recherche. Il empêche le comportement par défaut de soumission et utilise le terme de recherche pour récupérer les annonces via fetchAdsFiltered.

- Comment récupérons-nous les critères de recherche et de filtre de la base de données?

Les critères de recherche et de filtre sont les valeurs entrées par l'utilisateur via le formulaire. Ces valeurs sont passées en paramètres à la fonction fetchAdsFiltered et sont traitées à l'intérieur de cette fonction avec URLSearchParams, devenant ainsi une partie de la requête envoyée à la base de données. L'API reçoit ces paramètres de requête et exécute la requête SELECT appropriée dans la base de données pour extraire les annonces nécessaires.

La requête est utilisée pour sélectionner des annonces en fonction des filtres spécifiés.

```
$query = "SELECT a.id, a.title, a.situation, a.creation_date, u.street, u.building_number,
u.postal_code, u.city, u.canton, p.prduct_name as product_name, p.price as product_price,
p.stock as product_stock, ph.url as photo_url, c.name_category
FROM annoucement a
JOIN users u ON a.users_idusers = u.id
```

La requête SQL de base inclut des opérations de jointure qui combinent les tableaux d'annonces, d'utilisateurs, de produits, de catégories et de photos. Les conditions requises pour le filtrage (WHERE) sont ajoutées à la requête en fonction des paramètres fournis par l'utilisateur.

Si l'utilisateur a fourni un paramètre de filtre spécifique, ces paramètres sont ajoutés à la requête selon le cas :

```
if (!empty($title)) { $query .= " AND a.title LIKE :title"; }
if (!empty($nameCategory)) { $query .= " AND c.name_category LIKE :nameCategory"; }
if (!empty($minPrice)) { $query .= " AND p.price >= :minPrice"; }
if (!empty($maxPrice)) { $query .= " AND p.price <= :maxPrice"; }
```

Dans le backend, ces paramètres sont récupérés de la requête HTTP et utilisés pour construire une requête SQL conditionnelle qui filtre les annonces. Les valeurs pour les prix minimum et maximum sont particulièrement importantes pour garantir que les annonces retournées correspondent à la fourchette de prix définie par l'utilisateur.

- Nettoyage & Récupération des Annonces et des Marqueurs

Après le processus de recherche et de filtrage comme expliqué dans la section carte, les annonces seront supprimées et rappelées de la base de données selon les critères de filtrage et de recherche.

```
function clearAdsAndMarkers() : void {
    const adsContainer : HTMLElement = document.getElementById( 'ads-container');

    async function fetchAdsFiltered(params) : Promise<void> {
        await clearAdsAndMarkers(); // Effacez les publicités et les marqueurs
```

- Comparaison et exemple de démonstration des fonctionnalités de recherche et de filtrage

Remplir chacune des options de filtrage lors de la candidature vous permettra de retrouver directement le produit selon les critères de recherche. Par exemple, lorsque nous recherchons le produit « pomme » dans la section de recherche, nous voyons deux publicités. Si nous spécifions l'un de ces produits dans la catégorie fruits et l'autre dans la catégorie légumes dans la section filtre, il listera un produit chacun en fonction du contenu « fruit » et « légume » que nous avons écrit dans la section catégorie.

Lorsque nous recherchons « pomme » dans le processus de filtrage et de recherche, nous obtenons les mêmes résultats sur l'écran de la console ;

```
Ads fetched successfully: ▶ (2) [{...}, {...}]
```

```
Ad Data:
```

```
▶ {id: '76', title: 'pomme', situation: 'pommeysa', creation_date: '2024-05-07 10:46:43', street: 'Rue de L'industrie', ...}
```

```
Ad Data: ▶ {id: '77', title: 'pomme', situation: 'good', creation_date: '2024-05-07 11:07:15', street: 'Rue de La Mère', ...}
```

Filter Ads

Title:

category:

Minimum Price:

Maximum Price:

- Si nous effectuons la recherche que nous avons définie comme "fruit" dans l'option de filtrage, nous obtiendrons tous les produits avec les critères de fruits saisis dans l'annonce.

On peut voir le résultat ici en comparant les "ad_id" des produits. Les produits de la catégorie fruits sont uniquement les produits avec l'ad_id 76 et 78.

```
Ad Data: ▶ {id: '78', title: 'mais2', situation: 'bon', creation_date: '2024-05-07 11:10:36', street: 'Aarbergergasse', ...}
```

```
Ad Data:
```

```
▶ {id: '76', title: 'pomme', situation: 'pommeysa', creation_date: '2024-05-07 10:46:43', street: 'Rue de L'industrie', ...}
```

- Le prix de l'un de ces deux serpents, dont la catégorie est "fruit", est de 1 et le prix de l'autre est de 3, donc si nous fixons le prix maximum à 2, nous verrons une annonce suite à nos critères de filtrage.

Filter Ads

Title:

category:

Minimum Price:

Maximum Price:



Starting to clear markers. Total markers before clearing: 2

All markers removed, markers array reset.

Ads fetched successfully: ▼ [{"...}] ⓘ

```
▼ 0:
  building_number: "53"
  canton: "Bern"
  city: "Bern"
  creation_date: "2024-05-07 11:10:36"
  id: "78"
  name_category: "fruit"
  photo_url: "/Frontend/images/mais_coop.jpg"
  postal_code: "3011"
  product_name: "mais fruit"
  product_price: "1"
  product_stock: "9"
  situation: "bon"
  street: "Aarberggasse"
  title: "mais2"
  ► [[Prototype]]: Object
  length: 1
  ► [[Prototype]]: Array(0)
```

Ad Data: ► {id: '78', title: 'mais2', situation: 'bon', creation_date: '2024-05-07 11:10:36', street: 'Aarberggasse', ...}

Adding marker for ad: mais2 with query: Aarberggasse%2053%2C%203011%20Bern%2C%20Bern

Marker added, total markers now: 1

3.6.4 Fonctionnalité du Panier - Description Détaillée

La fonctionnalité panier de notre projet est une fonctionnalité importante qui permet aux utilisateurs d'ajouter, de gérer et d'acheter efficacement des produits dans le panier. Cette section décrit les fonctionnalités détaillées de notre système de panier et comment ces fonctions sont mises en œuvre.

Aperçu des fonctionnalités de Panier

- Ajout de produits au panier.
- Voir le contenu du panier.
- Mettez à jour les quantités de produits dans le panier.
- Retrait de produits du panier.
- Vider automatiquement le panier après l'inactivité.
- Restauration des stocks de produits.

Ajout de Produits au Panier

Lorsqu'un utilisateur examine un produit et souhaite l'acheter, la fonction addToCart est activée. Cette fonction récupère les informations du produit, met à jour le panier et actualise le nombre d'articles dans le panier sur la page. Le processus d'ajout au panier vérifie que la quantité demandée par l'utilisateur ne dépasse pas le stock disponible.

```
function addToCart(productId) : void {
  const quantity : number = parseInt(document.getElementById( elementId: 'product-quantity').value, radix: 10);
  if (quantity < 1) {
    alert("Vous devez ajouter au moins un produit!");
    return;
  }

  let cart = JSON.parse(sessionStorage.getItem( key: 'cart')) || [];
  const existingProductIndex : number = cart.findIndex(item : T => item.id === productId);
  const existingQuantity : any | number = existingProductIndex !== -1 ? cart[existingProductIndex].quantity : 0;
  const totalQuantity = existingQuantity + quantity;

  checkStock(productId, totalQuantity);
}
```

- Visualisation et Mise à Jour du Contenu du Panier

Les utilisateurs peuvent visualiser et mettre à jour les quantités des produits dans leur panier. La fonction `displayCartItems` met à jour dynamiquement le contenu du panier, permettant aux utilisateurs d'avoir des informations précises sur les produits qu'ils envisagent d'acheter.

```
function displayCartItems() : void {
  let cart = JSON.parse(sessionStorage.getItem( key: 'cart')) || [];
  const cartContainer : HTMLElement = document.getElementById( elementId: 'cart-items');
  cartContainer.innerHTML = ''; // Effacer le contenu précédent
  if (cart.length === 0) {
    cartContainer.innerHTML = 'Votre panier est vide.';
    return;
  }
  let html : string = '<ul>';
  cart.forEach(item : T => {
    html += '<li>'
      
```

- Vidage Automatique du Panier et Restauration des Stocks

Les fonctions `startCartTimeout` et `clearCart` gèrent le vidage automatique du panier après une période d'inactivité et restaurent les stocks des produits. Cette gestion aide à optimiser l'inventaire et à prévenir la réservation excessive des produits.

```
function clearCart() {
  sessionStorage.removeItem('cart');
  displayCartItems();
  restoreStocks();
}
```

```
function startCartTimeout() {
  setTimeout(() => {
    clearCart();
  }, 30000);
}
```



- pomme - Quantity: 1 - Price: 3 CHF

VIDER LE PANIER

PAGE DE PAIEMENT

Le délai d'attente du panier a commencé. Le panier sera effacé dans 30 secondes.

>

Vider le panier pour cause d'inactivité.

Updating cart count...

- Détail Technique de la Communication entre Frontend et Backend pour la Gestion du Panier

Dans notre projet, la gestion du panier utilise intensivement la communication entre le frontend et le backend pour assurer une mise à jour précise et en temps réel des stocks et des données du panier.

Lorsque les utilisateurs interagissent avec le panier, notamment lors de l'ajout ou de la suppression de produits, le frontend envoie des requêtes au backend en utilisant l'API fetch. Ces requêtes permettent de communiquer des informations essentielles telles que l'ID du produit et la quantité concernée pour la mise à jour des stocks.

Vérification du stock

```
function checkStock(productId, quantity) : void {
  fetch( input: `/Backend/checkStock.php?productId=${productId}&quantity=${quantity}` )
    .then(response : Response => response.json()) Promise<any>
    .then(data => {
      if (data.error) {
        alert(data.error);
      }
    })
}
```

Mise à jour du stock après confirmation

```
function updateStock(productId, quantity) : void {
  fetch( input: `/Backend/updateStock.php?productId=${productId}&decreaseAmount=${quantity}` )
    .then(response : Response => response.json()) Promise<any>
    .then(data => {
      if (data.error) {
        alert(data.error);
      } else {
        addProductToCart(productId, quantity);
        alert("Le produit a été ajouté au panier et le stock a été mis à jour.");
      }
    })
}
```

Restauration Automatique des Stocks

Quand le panier est vidé automatiquement après une période d'inactivité, ou manuellement par l'utilisateur, il est crucial de restaurer les stocks des produits qui étaient réservés dans le panier. Le processus est géré par une fonction restoreStocks qui envoie une requête au fichier updateStock.php sur le serveur via fetch. Voici comment cela fonctionne techniquement :

```
function restoreStock(productId, quantity) : void {
  fetch( input: `/Backend/restoreStock.php?productId=${productId}&restoreAmount=${quantity}` )
    .then(response : Response => response.json()) Promise<any>
}
```

```
$productId = isset($_GET['productId']) ? $_GET['productId'] : die(json_encode(['error' => 'productId is required']));
$restoreAmount = isset($_GET['restoreAmount']) ? $_GET['restoreAmount'] : die(json_encode(['error' => 'restoreAmount is required']));
```

Dans l'exemple ci-dessus, le produit nommé pomme, qui a été ajouté au panier et qui a un stock de 1, a été rechargé dans la base de données après avoir été retiré du stock pendant un certain temps.

products	16.0 KiB	6	82	carrot	3	7
users	16.0 KiB	7	83	carrot_fruits	3	4
information_schema		8	84	pomme	3	0
products	16.0 KiB	6	82	carrot	3	7
users	16.0 KiB	7	83	carrot_fruits	3	4
information_schema		8	84	pomme	3	1
mydatabase		9	85	pomme	2	2

Ces détails techniques montrent l'importance de la communication entre le frontend et le backend dans la gestion du panier. Elle permet de s'assurer que l'expérience utilisateur est fluide, que les données de stock sont exactes, et que les opérations de panier sont traitées de manière efficace et sécurisée. La capacité à ajuster le stock en temps réel et à restaurer correctement les stocks assure que notre système est robuste et réactif aux actions de l'utilisateur.

3.6.5 Fonction de messagerie

3.7 Bugs

4 Tests

4.1 Stratégie de test

L'utilisateur sera confronté à de nombreuses options lors de l'utilisation de l'application.

L'utilisateur devra saisir les informations correctes et agir conformément à notre système établi.

Dans ce contexte, il sera de notre devoir de guider l'utilisateur et d'obtenir des données saines.

Il est nécessaire de tester à la fois en termes de guidage de l'utilisateur et de fonctionnalité de l'application.

Vous pouvez trouver ces résultats de tests ci-dessous ;

TEST		
Travaux à réaliser	Réalise	Parties manquantes
Avertissement que l'utilisateur doit renseigner les informations lors du processus d'inscription.	Un avertissement apparaîtra à l'écran : Ne le laissez pas vide.	
Plusieurs personnes peuvent-elles s'inscrire avec la même adresse e-mail ?	Si le même e-mail a déjà été utilisé, il affichera un avertissement utilisateur existe.	
Si le champ non obligatoire n'est pas renseigné, y aura-t-il une erreur dans la base de données ?	Si les champs qui peuvent être laissés vides ne sont pas remplis, ces données dans la base de données resteront vides.	
Avertissement si les informations saisies par l'utilisateur ne correspondent pas au type de données.	Si les informations appropriées ne sont pas saisies pour les types de données tels que les e-mails ou les numéros, un avertissement de type de données incorrect apparaît à l'écran.	
Les informations saisies lors de l'inscription ont-elles été placées dans les colonnes appropriées de la base de données ?	Les informations sont au bon endroit et la fonction Auto Incrimination est active.	
Le mot de passe de l'utilisateur est-il haché et enregistré dans la base de données ?	Le mot de passe a été enregistré sous forme hachée.	
Est-ce que cela donne un avertissement si le type d'e-mail n'est pas valide pendant le processus de connexion ?	Lors de la connexion par e-mail, un avertissement apparaît pour un type d'e-mail inapproprié.	
Y a-t-il un avertissement si le mot de passe saisi lors du processus de connexion ne correspond pas au mot de passe saisi au moment de l'inscription ?	L'utilisateur est averti de saisir le mot de passe correct et ne peut pas se connecter à moins que le mot de passe et les informations de courrier électronique corrects correspondent à la base de données.	
Une fois le processus de connexion terminé, y a-t-il un avertissement de connexion réussie à l'écran ?	L'utilisateur est informé par un écran d'avertissement qu'il s'est connecté.	
Les boutons changent-ils après le processus de connexion ?	Les boutons changent et de nouvelles fonctions apparaissent après le processus de connexion.	

TEST		
Travaux à réaliser	Réalise	Parties manquantes
Le bouton filtre est-il fonctionnel ?	Lorsque vous cliquez sur le bouton de filtre, une fenêtre contextuelle de filtre publicitaire avec le titre du titre apparaît, par exemple. Cette pop-up répertorie les annonces en fonction de leur titre et les affiche sur l'écran.	
Le bouton ma page est-il actif ?	Lorsqu'il clique sur le bouton, l'utilisateur accède à sa propre page et peut modifier ses informations personnelles, publier une annonce et consulter les annonces qu'il a placées.	
Peut-il modifier les informations utilisateur sur sa page personnelle ?	Vous pouvez modifier vos informations, les anciennes informations sont conservées dans les cases que vous laissez. Si un nouveau mot de passe est déterminé, le nouveau mot de passe est haché et enregistré.	
Après avoir modifié les informations utilisateur, la base de données sera-t-elle mise à jour instantanément sans erreur ? Y a-t-il un problème lors de la reconnexion avec de nouvelles informations ?	Les informations sont instantanément mises à jour dans la base de données et l'utilisateur ne rencontre aucun problème lors de sa connexion.	
Le bouton de l'annonce est-il fonctionnel ? Une fois l'annonce placée, les informations sur l'annonce sont-elles placées dans différents tableaux, en préservant les références ?	L'annonce est distribuée à toutes les tables requises.	
Plusieurs URL d'image sont-elles téléchargées dans la base de données avec l'annonce ? Les images sont-elles enregistrées dans le chemin de sauvegarde ?	Alors que les URL sont enregistrées dans la base de données pour éviter que la base de données ne devienne gonflée, les fichiers image sont enregistrés dans le chemin de sauvegarde.	
L'utilisateur peut-il voir uniquement les annonces qu'il a saisies avec son user id sur sa propre page ou toutes les annonces sont-elles visibles ?	L'utilisateur ne pourra voir que les publicités qu'il a placées sur la page.	
Les publications des utilisateurs peuvent-elles être modifiées ?		L'utilisateur ne dispose pas de ces fonctionnalités à ce stade. Cette fonction sera ajoutée dans le futur.

TEST		
Travaux à réaliser	Réalise	Parties manquantes
La page Messages est-elle active ?		Cette page sera modifiée et activée à l'avenir.
Le bouton de déconnexion est-il fonctionnel ?	Lorsque vous cliquez sur le bouton Déconnexion, le processus de déconnexion est effectué et les boutons reviennent à leur état précédent.	
Une carte a-t-elle été ajoutée à la page d'accueil ?	Il y a une carte sur la page d'accueil.	
Les publicités sont-elles visibles sur la carte et les informations publicitaires sont-elles accessibles ?	Les annonces peuvent être affichées sur la carte et, lorsqu'une annonce est cliquée, elle redirige vers la page de détails de l'annonce.	
Toutes les publicités saisies par tous les utilisateurs peuvent-elles être vues sur la page d'accueil ?	Toutes les annonces apparaissent sur la page d'accueil et les annonces sont renouvelées selon les périodes déterminées.	

5 Conclusions

5.1 Objectifs atteints

5.2 Objectifs non-atteints

5.3 Suites possibles au pour le projet

6 Annexes

6.1 Journal de travail

7 SOURCES

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/stringify

- Image

<http://bilgisayar-muhendisleri.blogspot.com/2014/01/php-mysql-image-upload-etme-ve-okuma.html>
<https://kodusta.com/blog/php-pdo-ckeditor-resim-yukleme-ve-veritabanina-kaydetme>
https://www.w3schools.com/php/php_mysql_insert_multiple.asp
<https://stackoverflow.com/questions/68038388/how-to-upload-image-on-mysql-database-using-php>

- Map

<https://docs.mapbox.com/mapbox-gl-js/guides/>
<https://www.burakalpikara.com/mapbox-ile-projemize-harita-eklemek>
<https://www.softkraft.co/mapbox-vs-google-maps/>
<https://www.mapbox.com/pricing>

- Panier

http://jmolline.free.fr/tutos/tuto_panier.html
https://mesutd.com/php-mysql-baglanma-mysqli-ekleme-silme-duzenleme-listeleme-dosya-yukleme#google_vignette

- Bastien et Scapin

http://www.guillaumegronier.com/cv/resources/CriteresBastienScapin_v3.pdf
https://umtice.univlemans.fr/pluginfile.php/507354/mod_resource/content/1/Ergonomie%2C%20Crit%C3%A8res%20de%20Scapin%20et%20Bastien.pdf

- Messages

<https://medium.com/@mkarabulut44/php-ve-javascript-ile-online-sohbet-sistemi-935bb8e33d20>
<https://www.youtube.com/watch?v=VJytplDynQ4>

