

Tpi

« Caves »



Table des matières

1	Introduction	4
1.1	Cadre, description et motivation	4
1.2	Organisation.....	4
1.3	Objectifs	4
1.3.1	Gestion des publicités conviviale :.....	4
1.3.2	Gestion sécurisée et flexible des informations utilisateur :.....	5
1.3.3	Affichage et gestion détaillés des annonces :	5
1.3.4	Visualisation et filtrage anonymes des annonces :	5
1.3.5	Facilité de communication et de messagerie :	5
1.3.6	Sécurité et protection des données :	5
1.3.7	Panier d'achat	5
1.3.8	Messages	5
1.4	Planification initiale	6
2	Analyse.....	7
2.1	Cahier des charges détaille	7
2.1.1	Maquettes	8
2.1.2	Use Cases et Scénarios	23
2.1.3	Modèle conceptuel de données.....	27
2.2	Fonctions prévues	28
2.3	Méthode de projet	28
2.4	Planification	29
2.5	Stratégie de Test	32
3	Implémentation	32
3.1	Choix techniques.....	32
3.2	Architecture du Système.....	33
3.3	Système de base de données.....	35
3.4	Modèle Logique de données.....	38
3.5	Diagramme de Flux	40
3.6	Ergonomie de site (Bastien et Scapin)	45
3.7	Points techniques spécifiques.....	52
3.7.1	Enregistre les images	52
3.7.2	Utilisation et fonctionnalité de la carte.....	53
3.7.3	Fonctions de Filtrage et de Recherche	63
3.7.4	Fonctionnalité du Panier - Description Détailée.....	66
3.7.5	Fonction de messagerie dans l'application : Explication et Justification	70
3.8	Bugs	73
3.8.1	Problème de Réponse aux Messages dans le Système de Messagerie	73
4	Tests.....	75
5	Conclusions.....	78
5.1	Objectifs atteints	78

5.2 Objectifs non-atteints.....	79
5.3 Suites possibles au pour le projet	79
6 Annexes	81
6.1 Résume du Rapport de TPI.....	81
6.2 Glossaire.....	82
6.3 Sources	83
6.4 Journal de Travail	84

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 filtre.
- 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		Total											
Prévu	Akyüz	30.04.24	08.05.24	17.05.24	24.05.24	31.05.24	07.06.24	14.06.24	21.06.24				
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 Akyüz	1h 00 0h 30	1h 00 0h 30	0h 30 0h 30						2h 30 0h 30		04:00 04:00	
12 - Planification	Prévu Akyüz	1h 00 0h 15	0h 30 0h 15	0h 30 0h 15						2h 00 0h 15		06:20 04:45	SPRINT-1
13 - Créez le fichier	Prévu Akyüz	0h 30 0h 10								0h 30 0h 10		06:20 04:00	
14 - Icescrum & Github	Prévu Akyüz	3h 30 3h 15	1h 10 2h 00	1h 00 2h 00	2h 30 0h 30					8h 10 3h 15		04:00 06:20	
15 - Recherches	Prévu Akyüz	2h 00 3h 15	2h 00 3h 15	2h 00 3h 15	0h 30 0h 30					6h 30 0h 30		04:00 06:20	SPRINT-2
	Prévu Akyüz											04:45 04:00	
2 Implémentation										30 h 15	6 h 05		
21 - MCD-MLD	Prévu Akyüz	1h 00 0h 30	2h 00 1h 30	0h 45 0h 30	0h 20 0h 20					3h 05 1h 30		04:00 06:20	
22 - BDD-SQL	Prévu Akyüz	1h 45 1h 30	1h 00 1h 30	0h 45 0h 30	0h 20 0h 20					3h 50 2h 00		04:45 06:20	
23 - Balsamiq	Prévu Akyüz	2h 00 2h 00	0h 30 0h 30	0h 30 0h 30	0h 20 0h 20					3h 20 2h 35		04:00 04:45	SPRINT-3
24 - Backend	Prévu Akyüz	4h 00 2h 35	7h 00 6h 00	6h 00 5h 00	3h 00 3h 00					20h 00 7h 30		06:20 04:00	
25 - Frontend	Prévu Akyüz	1h 00 0h 45	3h 00 2h 35	1h 30 1h 30	2h 00 2h 00					0h 45 0h 45		01:45 01:45	SPRINT-4
26 - Map	Prévu Akyüz	4h 00 2h 00	1h 00 1h 00	0h 30 0h 30	1h 00 1h 00					6h 30 2h 00		Total 90:00 90 h 00	
3 Tests										10 h 35			
31 - Use-cases	Prévu Akyüz		2h 00 0h 30	1h 00 0h 30	0h 30 0h 30	1h 00 0h 30						4 h 30	
32 - Tests	Prévu Akyüz		0h 30 0h 30	0h 30 0h 30	0h 35 0h 35	0h 30 0h 30						2 h 05	
33 - Résultats des tests	Prévu Akyüz		0h 30 0h 30	0h 30 0h 30	1h 00 1h 00	2h 00 2h 00						4 h 00	
	Prévu Akyüz												
4 Documentation												14 h 30	3 h 55
41 - Journal de travail	Prévu Akyüz		0h 40 0h 20	0h 30 0h 20	0h 30 0h 20	0h 30 0h 20						2 h 10 0 h 20	
42 - Rédaction de la documentation	Prévu Akyüz		3h 00 1h 30	2h 30 2h 30	2h 00 2h 00	2h 20 2h 20						9 h 50 1 h 30	
43 - Contacts avec le chef de projet et les experts	Prévu Akyüz		1h 00 2h 05	0h 30 0h 30	0h 30 0h 30	0h 30 0h 30						2 h 30 2 h 05	

Figure 1: planification-1

3 Tests											10 h 35		
31 - Use-cases	Prévu Akyüz		2h 00 0h 30	1h 00 0h 30	0h 30 0h 30	1h 00 0h 30						4 h 30	
32 - Tests	Prévu Akyüz		0h 30 0h 30	0h 30 0h 30	0h 35 0h 35	0h 30 0h 30						2 h 05	
33 - Résultats des tests	Prévu Akyüz		0h 30 0h 30	0h 30 0h 30	1h 00 1h 00	2h 00 2h 00						4 h 00	
	Prévu Akyüz												
4 Documentation											14 h 30	3 h 55	
41 - Journal de travail	Prévu Akyüz		0h 40 0h 20	0h 30 0h 20	0h 30 0h 20	0h 30 0h 20						2 h 10 0 h 20	
42 - Rédaction de la documentation	Prévu Akyüz		3h 00 1h 30	2h 30 2h 30	2h 00 2h 00	2h 20 2h 20						9 h 50 1 h 30	
43 - Contacts avec le chef de projet et les experts	Prévu Akyüz		1h 00 2h 05	0h 30 0h 30	0h 30 0h 30	0h 30 0h 30						2 h 30 2 h 05	

Figure 2: planification-2

Grâce à l'autorisation reçue au début du Sprint-3, la planification a été réorganisée.

Le congé pris le premier jour du sprint-3 a été ajouté au dernier jour du sprint-4 et le planning a été réorganisé en conséquence.

CAVES - TPI

Projet		Planification																											
Caves		Total		30.04.24		08.05.24		17.05.24		24.05.24		31.05.24		07.06.24		14.06.24		21.06.24											
Prévu	90h00		29h25	24h40	16h45	19h10						31h24		07h24		14h24		21h24											
Akyüz	71h30		27h15	23h40	16h15	4h20																							
SEM	18	19	20	21	22	23	24	25																					
1 Analyse																													
																			16 h 10										
																			14 h 15										
																				04:00									
																				04:00									
																				06:20	SPRINT-1	29h25	Sem. 18						
																				04:45									
																				06:20									
																				04:00									
																				06:20									
																				04:00									
																				06:20									
																				04:00									
																				06:20									
																				02:25									
																				04:00									
																				06:20									
																				04:45									
																				06:20									
																				04:00									
																				04:00									
																				06:20									
																				04:00									
																				06:20									
																				04:05									
																				Total	90:00		90h00						
																				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							

Figure 3: Planification organisée après le statut d'autorisation

2 Analyse

2.1 Cahier des charges détaillé

- Période de réalisation

Dates: du 30 avril 2024 à 8h00 au 29 mai 2024 à 11h35 & Total des heures: 90 heures

- Exigences Techniques

1 ordinateur standard du CPNV avec Windows 10 professionnel

Suite Office

Visual Studio Enterprise 2022 / PHP Storm

Balsamiq

MySQL Workbench

Outils de gestion de projet : GitHub Project / Icescrum

- Livrables

Mardi et vendredi

Un commit significatif - Documentation mise à jour -Déploiement du site

À la fin du TPI

Sources et données sur le dépôt

Documentation sur le dépôt

Clé USB avec sources, documentation et journal à jour.

- Points techniques évalués spécifiques au projet:

- Ergonomie de site au niveau du guidage et de la charge de travail respectant les critères de Bastien et Scapin.
- Affichage de l'adresse du client sur la carte.
- Recherche filtrée par type de produit et prix.

4. Messagerie possible avec le producteur.
5. Réservation possible d'un produit choisi pendant une durée limitée.
6. Affichage des points de vente sur la carte.
7. Affichage des annonces sur la page d'accueil.

2.1.1 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.

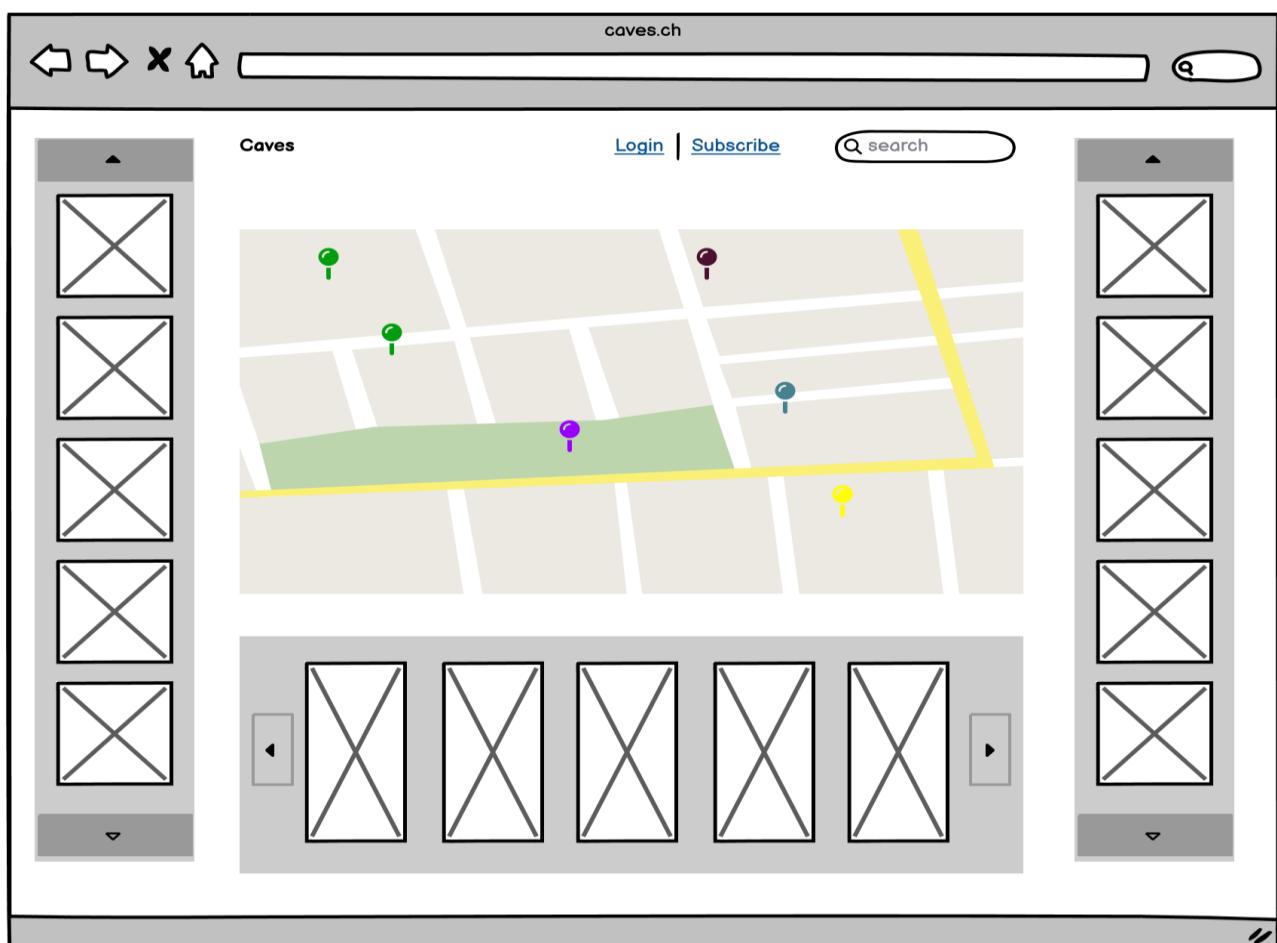


Figure 4: vue de la page d'accueil

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.

The screenshot shows a web browser window with the title 'A Web Page'. The address bar contains 'https://'. The main content area is a registration form for 'Caves'. It includes fields for 'Nom de l'entreprise (facultatif)', 'Nom', 'Prénom', 'e mail', 'Phone', 'Rue et N°', 'Code Postal', 'Ville', 'Canton', 'Password', and 'Confirm Password'. There is also a 'Subscribe' button at the bottom left. The browser interface includes standard navigation buttons (back, forward, stop, home) and a search bar.

Figure 5: formulaire d'inscription

The screenshot shows a map of Switzerland with major cities like Zurich, Lucerne, and Bern labeled. At the top left, there are 'Login' and 'Subscribe' buttons. The main area is a registration form titled 'Subscribe' with fields for 'Nom de l'entreprise (facultatif)', 'Name', 'Firstname', 'E mail', 'Numéro de téléphone', 'Adresse', 'Numéro du bâtiment', 'Code postal', 'Ville', 'Canton', 'Le mot de passe', and 'Confirmez le mot de passe'. A large green 'Subscribe' button is at the bottom right. The page has a watermark 'aaa' in two places.

Figure 6:Statut du formulaire d'inscription sur la page

- 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.

A screenshot of a web browser window titled "A Web Page". The address bar shows "https://". The main content area displays a login form for "Caves". The form has two input fields: "E-mail" and "Password", both represented by empty rectangular boxes. Below these fields is a "Login" button. In the top right corner of the form area, there is a "Subscribe" button. The browser interface includes standard navigation buttons (back, forward, stop, home) and a search bar at the top.

Figure 7: Formulaire de connexion

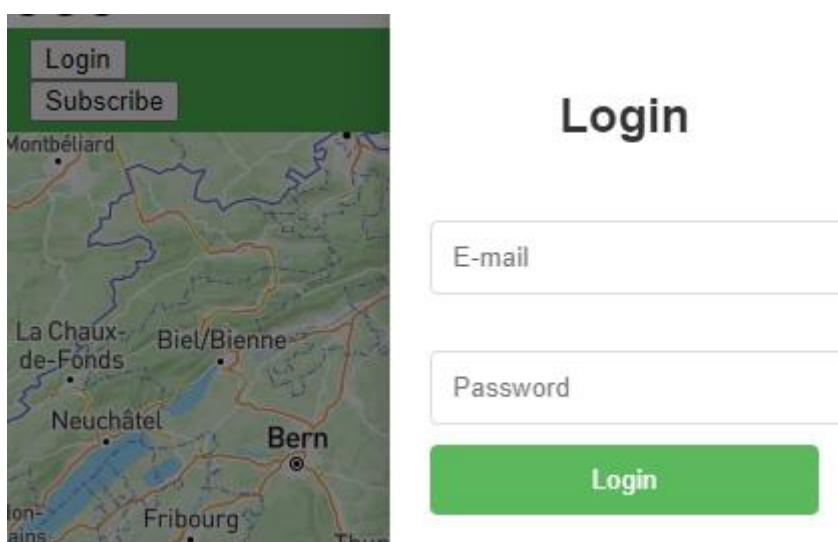


Figure 8: Statut du formulaire de login sur la page

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.



Figure 9: Page d'accueil après connexion

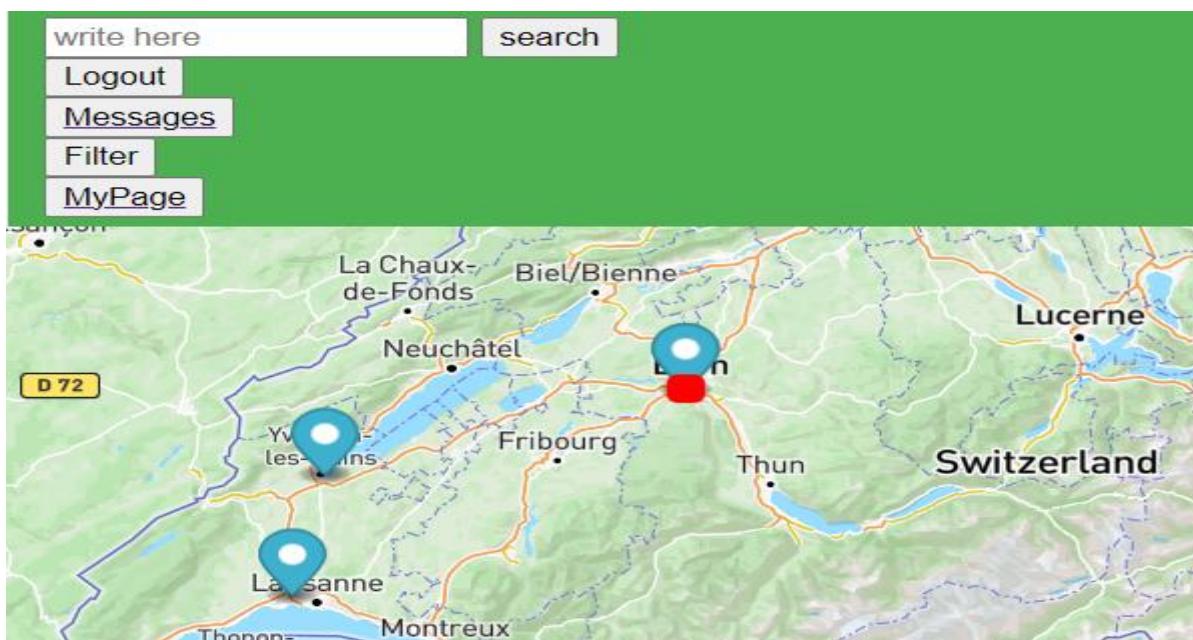


Figure 10: page d'accueil après connexion

- 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.

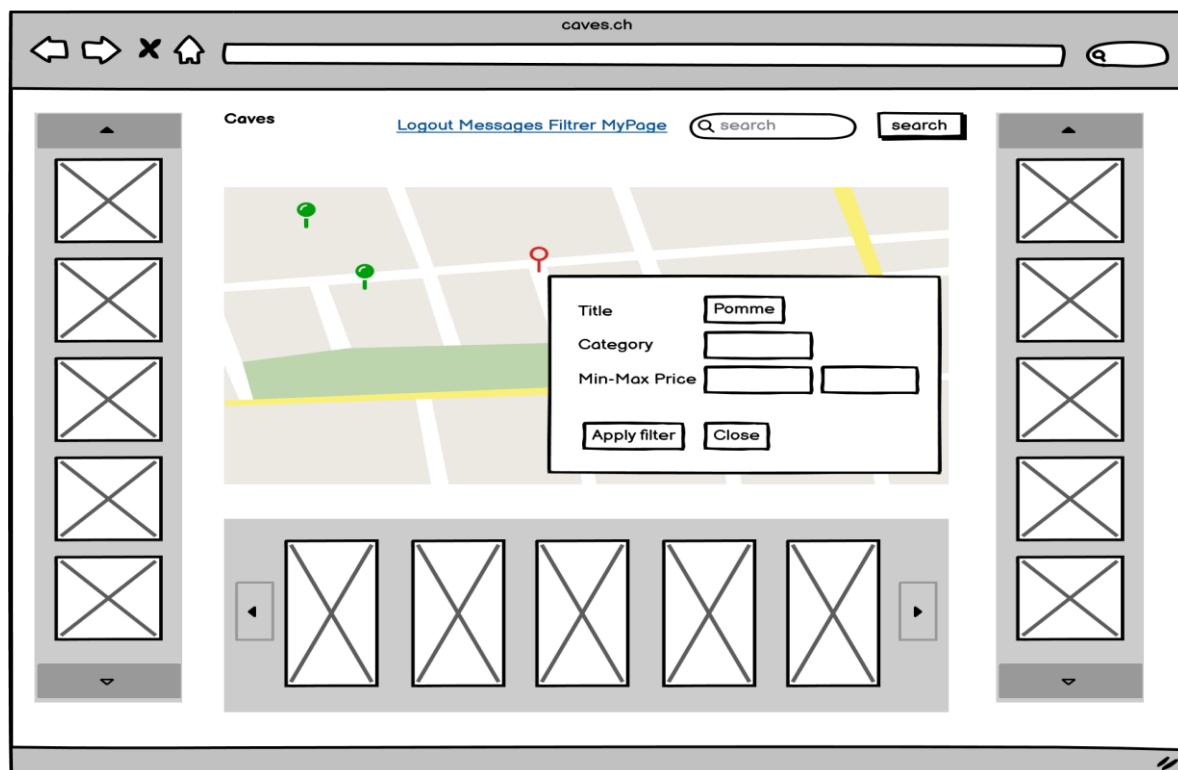


Figure 11: pop-up de la fonction de filtrage

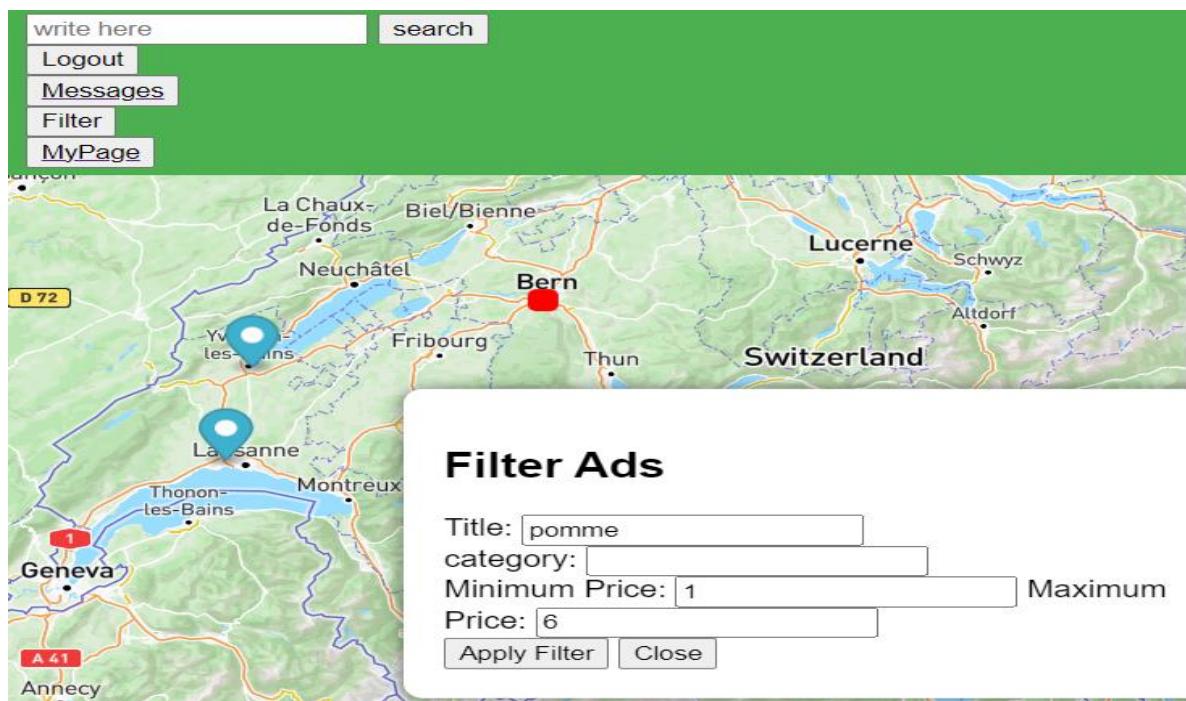


Figure 12: Vue de l'écran de filtrage sur la page d'accueil

CAVES - TPI

```

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'industrie", ...}
Ad Data:                                                                                   accueil.js:75
▶ {id: '77', title: 'pomme', situation: 'good', creation_date: '2024-05-07 11:07:15', street: 'Rue de la Mère', ...}
Adding marker for ad: pomme with query: Rue%20de%20l'industrie%208%2C%201400%20Yverdon-les-Bains%2C%20Vaud   accueil.js:92
Adding marker for ad: pomme with query: Rue%20de%20la%20Mère%2C%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

```

Figure 13: affichage des résultats du filtrage dans la console

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

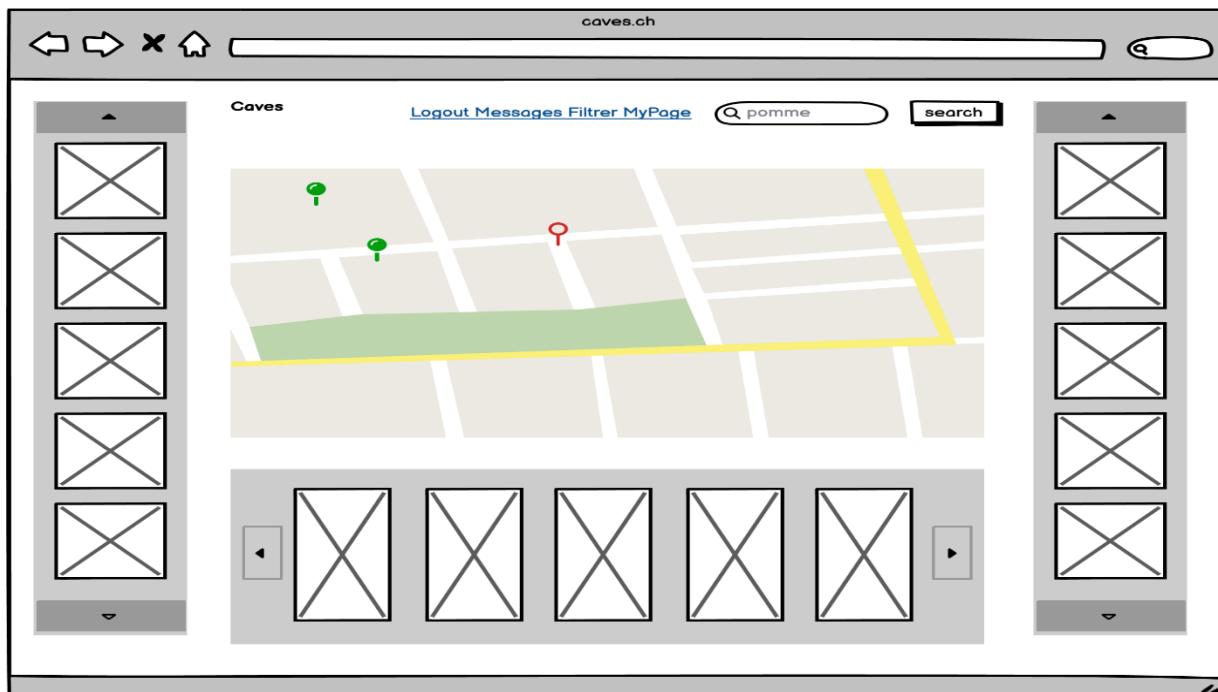


Figure 14: maquette de fonction de recherche

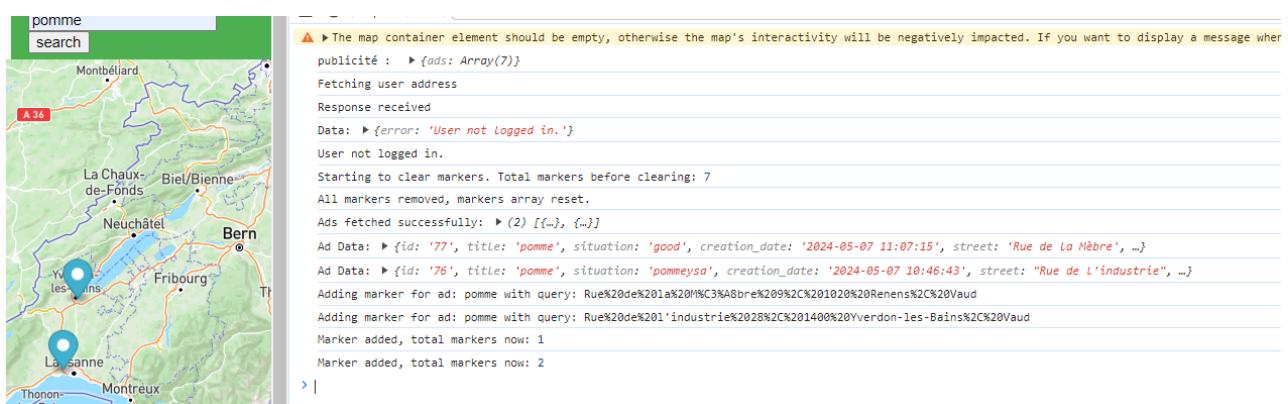


Figure 15: affichage des résultats de recherche dans la console

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 Phon

Post an Ad

Ad Title	<input type="text" value="situation"/>	fruit
product name:	<input type="text" value="Ürün Adı"/>	
price:	<input type="text"/>	
Stock:	<input type="text"/>	
images:	<input type="button" value="Sélect. fichiers"/> <input type="text" value="Aucun fichier choisi"/>	
<input type="button" value="Post Ad"/>		

Ads List

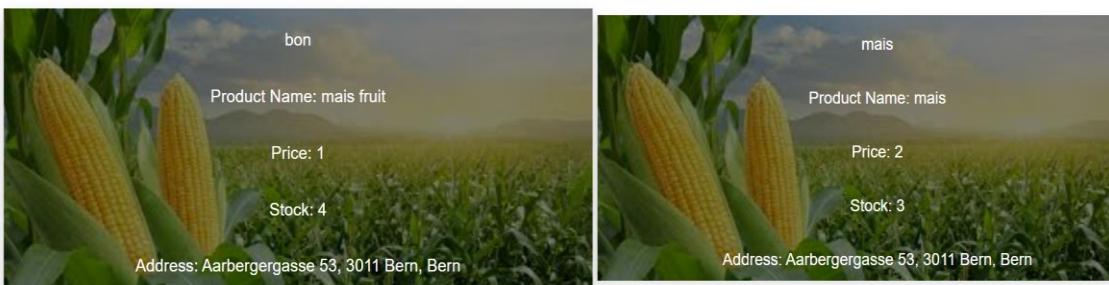


Figure 16: vue de la page personnelle

A maquette de page web intitulée "A Web Page" avec l'URL "https://mypage". La page est divisée en sections :

- Edit Personal Informations** : Contient des champs pour Nom, Prénom, e-mail, Phone, Rue et N°, Code Postal, Ville, Canton, et Password, avec un bouton "Save Changes".
- Post an Ad** : Contient des champs pour Title, Situation, Categories, product name, Price, Stock, et Images (avec un bouton "select fichier") et un bouton "PostAd".
- My Ads** : Affiche quatre encadrés vides nommés Item One, Item Two, Item Three et Item Four.

Figure 17: maquette de page personnelle

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.

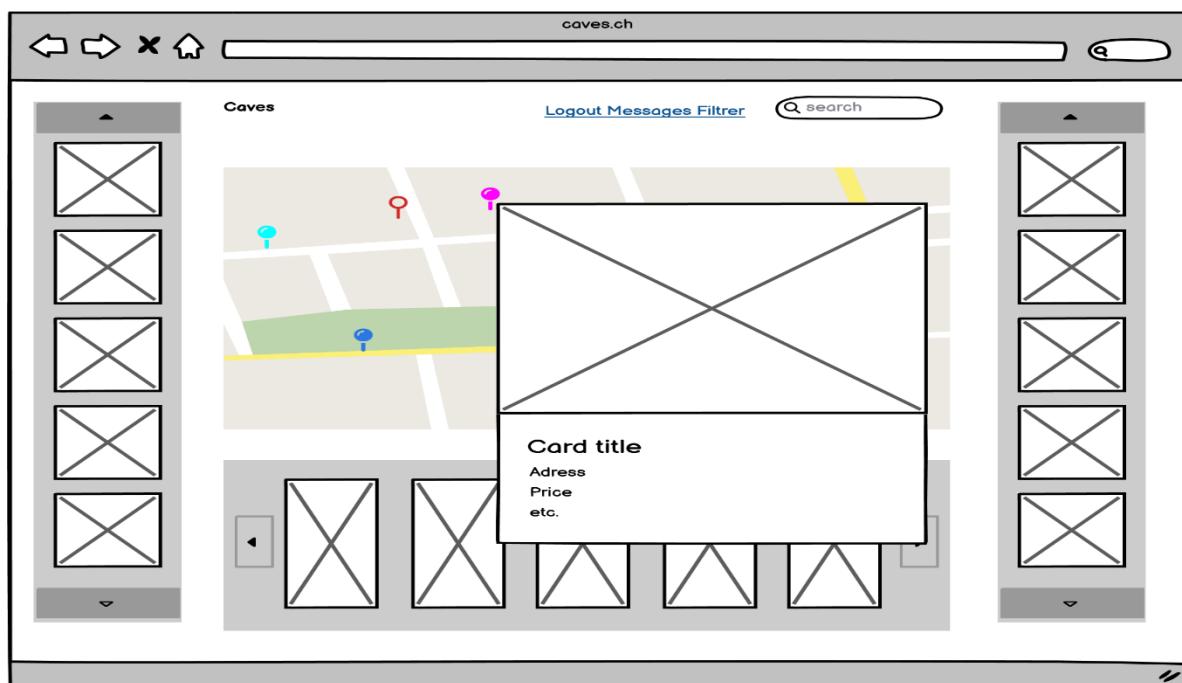


Figure 18: Maquette de clic sur les boutons de la carte



Figure 19: Afficher le détail de l'annonce sur la carte

- 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

Figure 21: Avant le processus de connexion

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)

Figure 20: Après le processus de connexion

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.

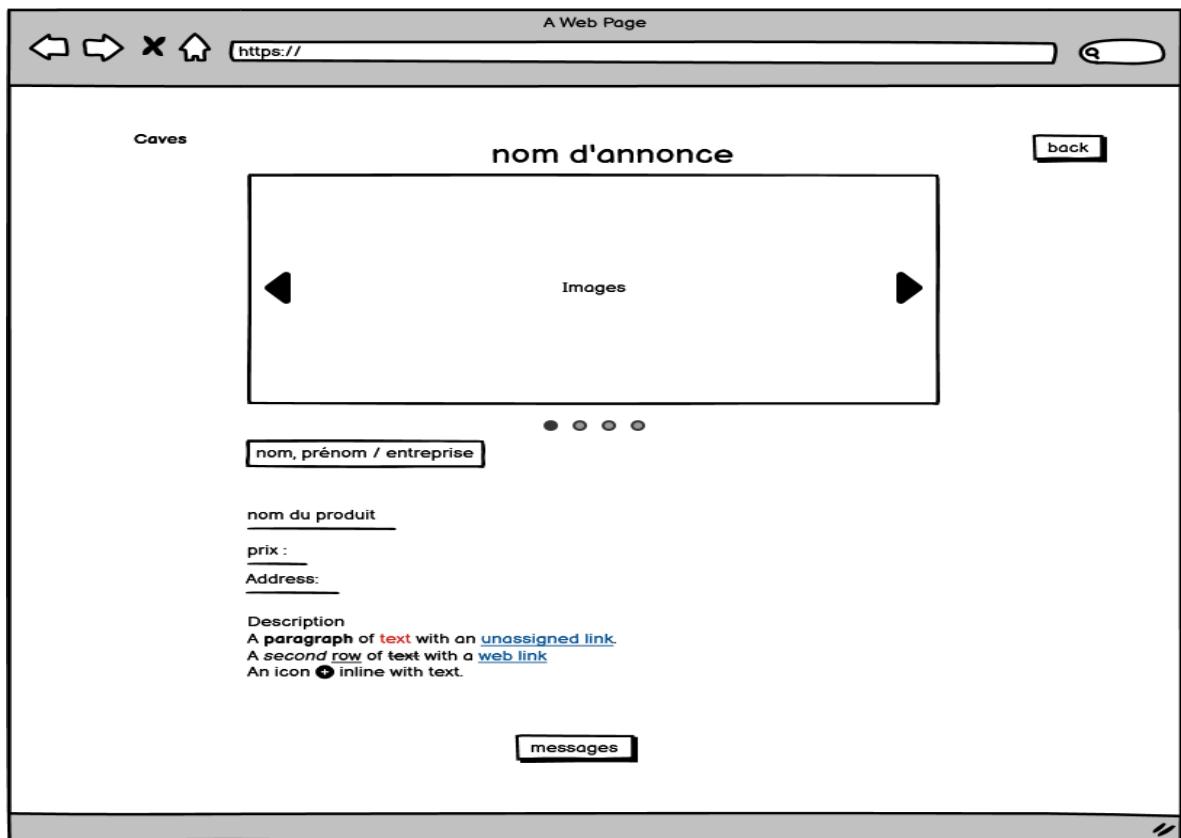
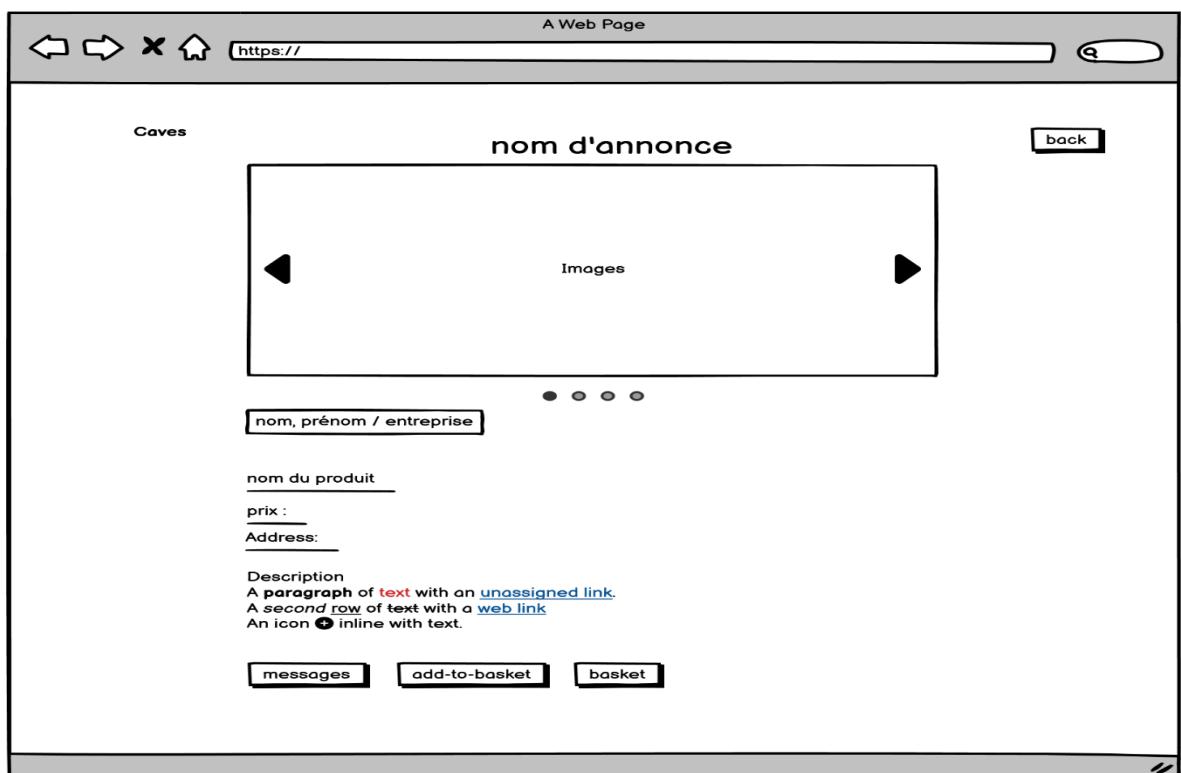


Figure 22: affichage de la page de détail de l'annonce



23: Affichage de la page de détail de la publicité après le processus de connexion

carrot



good

carrot_fruits

Price: 3

Stock: 4

Address: Rue de la Mèbre 9, 1020 Renens, Vaud

Creation Date: 07/05/2024

0
envoyer des messages
Ajouter au panier
Panier(0)

Figure 24: Page de détails de la publicité après le processus de connexion

- 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.

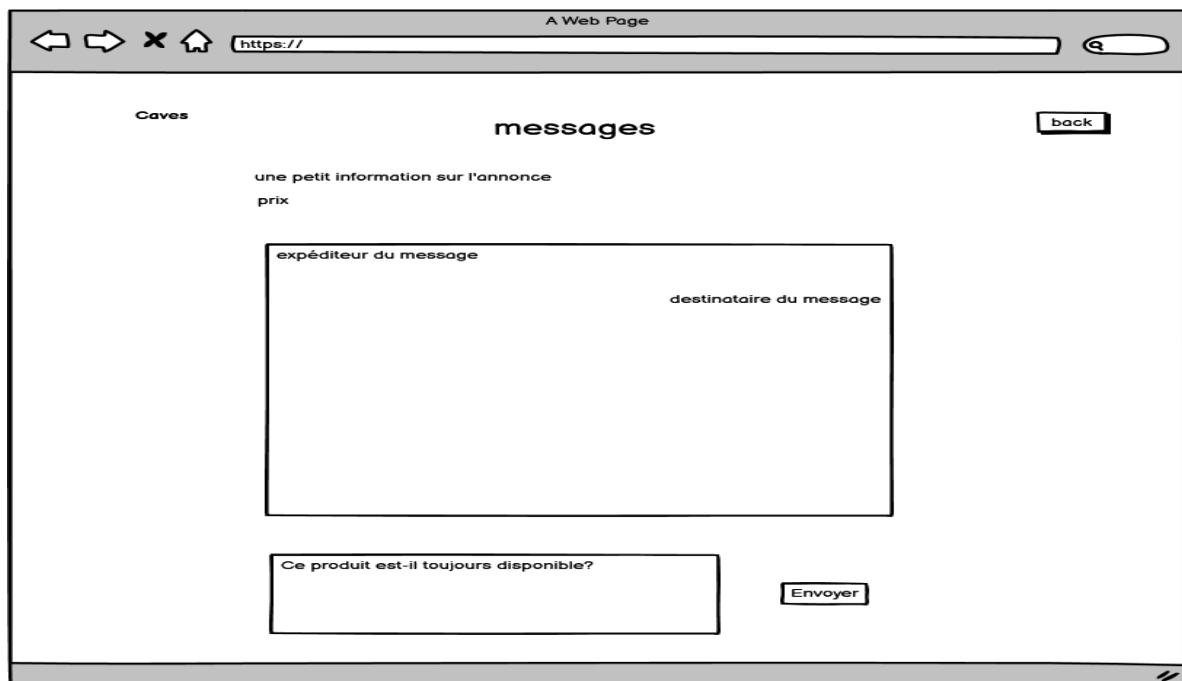


Figure 25: maquette de page d'envoi de message

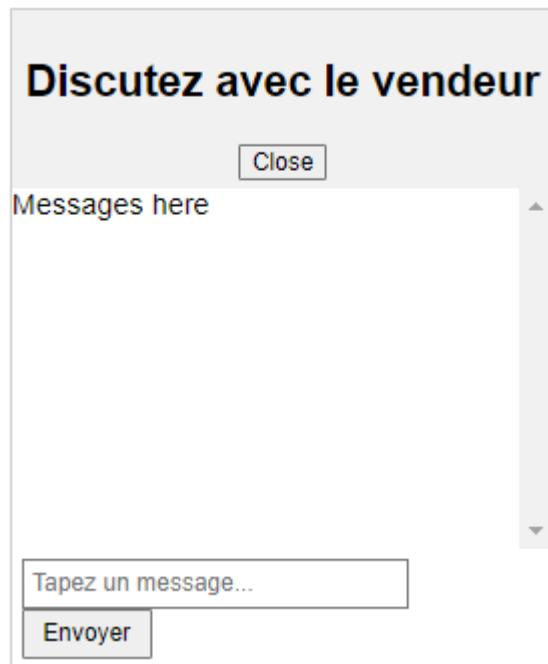


Figure 26: Écran d'envoi de message

- 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.

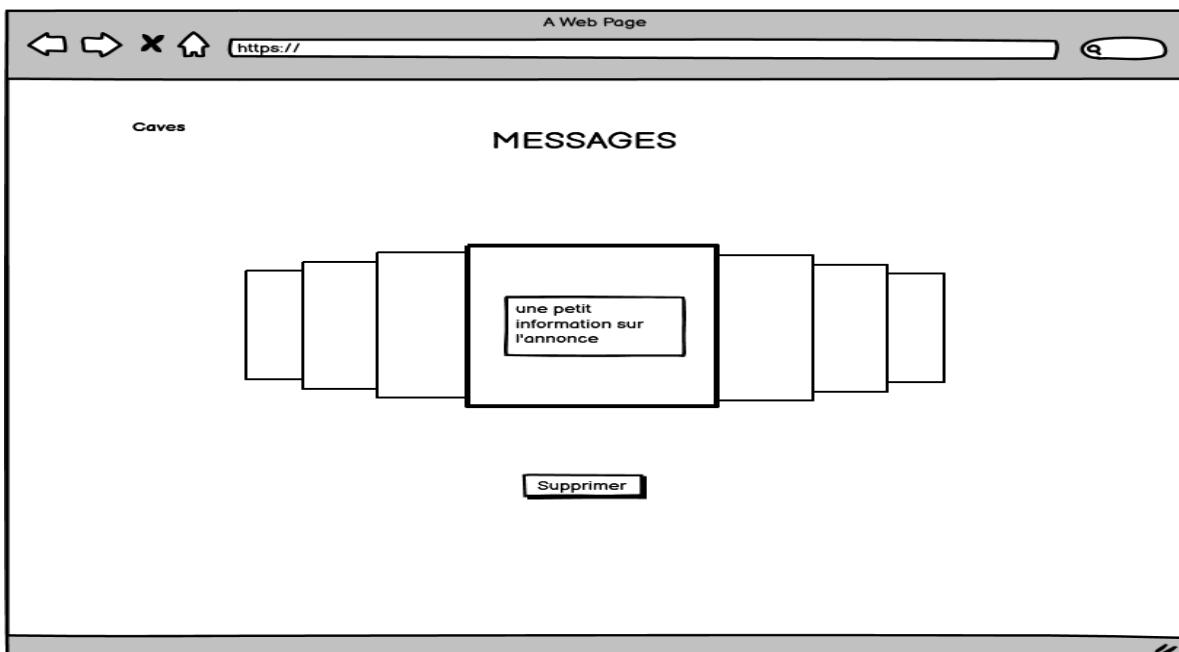


Figure 27: maquette d'affichage des messages

Your Messages

champignons - from coop coop

CV V

Sent: 28/05/2024 10:27:52 [Reply](#)

champignons - from coop coop

salu bern

Sent: 28/05/2024 10:27:44 [Reply](#)

champignons - from coop coop

hi migros

Sent: 24/05/2024 10:29:22 [Reply](#)

champignons - from migros migros

hi coop bern

Sent: 24/05/2024 10:28:32 [Reply](#)

Figure 28 : page de messages

- 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 <hr/>	Quantity <hr/>	Prix : <hr/>
 Image Name	Name Product <hr/>	Quantity <hr/>	Prix : <hr/>
		CheckOut	EmptyCart

Figure 29: Maquette de fonction de panier


Choose theme:

-  champignons - Quantity: 2 - Price: 8 CHF
-  pomme - Quantity: 1 - Price: 3 CHF

[VIDER LE PANIER](#)

[PAGE DE PAIMENT](#)

Figure 30: représentation du panier

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.

- L'en-tête "HEADER" est également édité et placé sur chaque page html. Le thème et le logo sont également visibles sur le dessus du modèle.

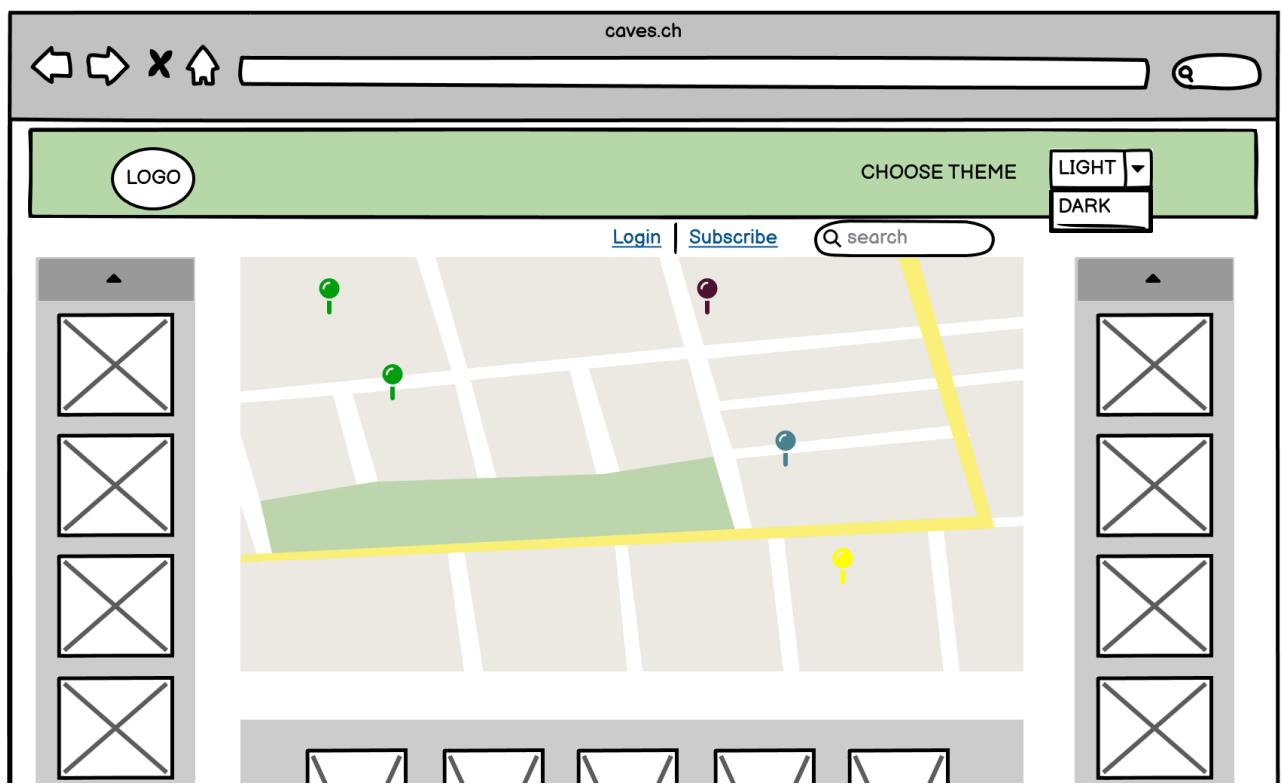


Figure 31: Titre d'en-tête et maquette de thème

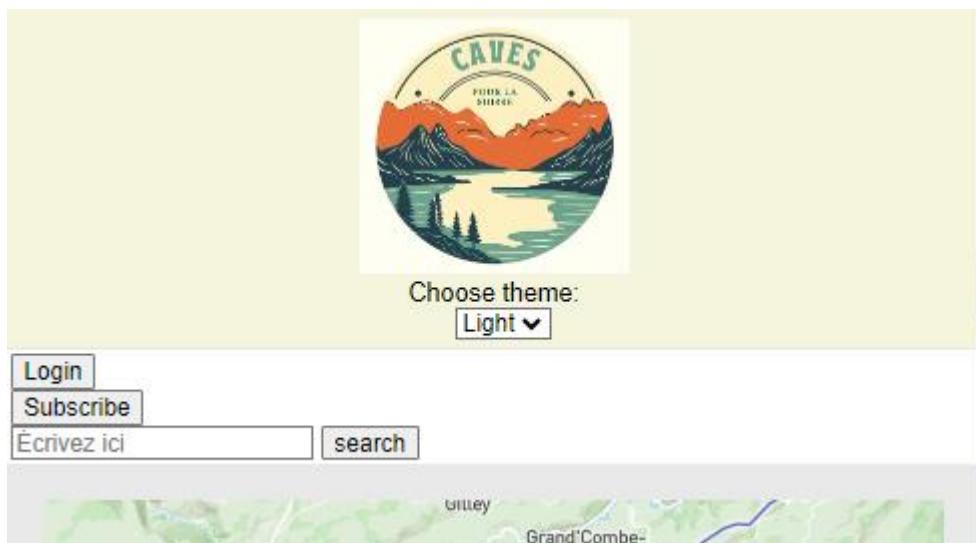


Figure 32: Affichage de l'en-tête et du thème sur le projet

- Le modèle utilisateur qui ouvre l'emplacement a également été préparé et l'utilisateur pourra accéder à son emplacement indépendamment du processus de connexion, mis à part son adresse. Le bouton violet dans le modèle représente l'utilisateur.

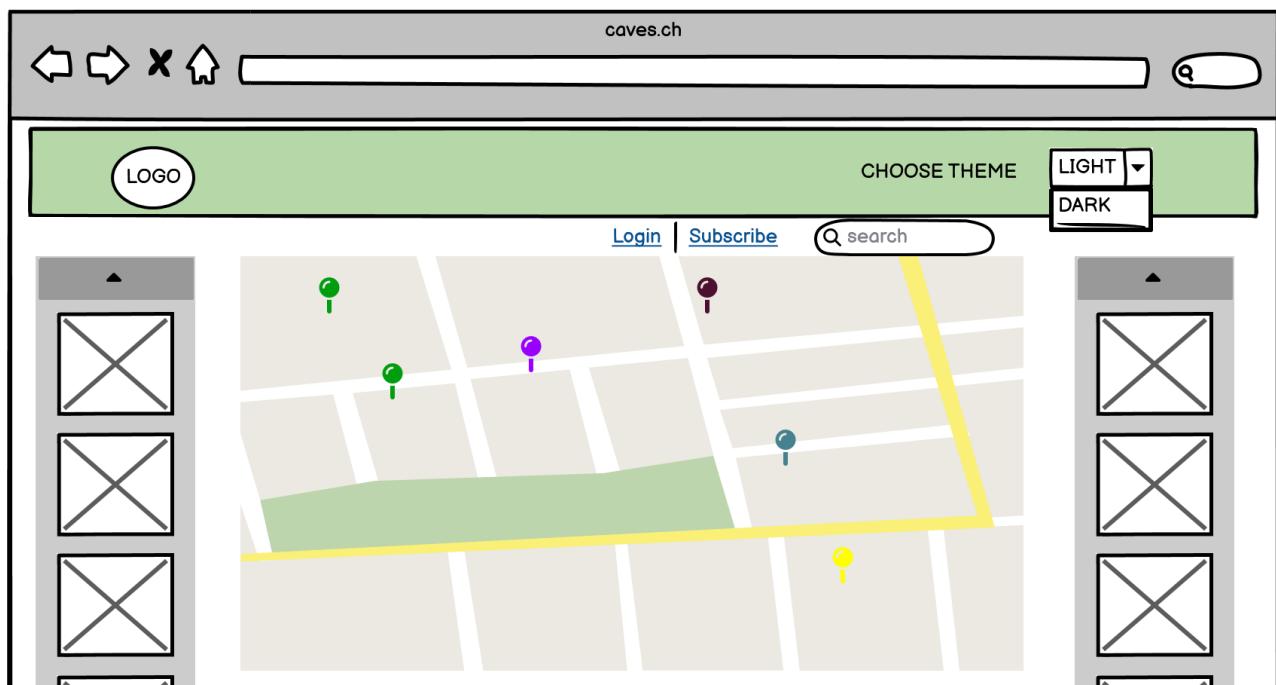


Figure 33: afficher l'emplacement de l'utilisateur sur le modèle

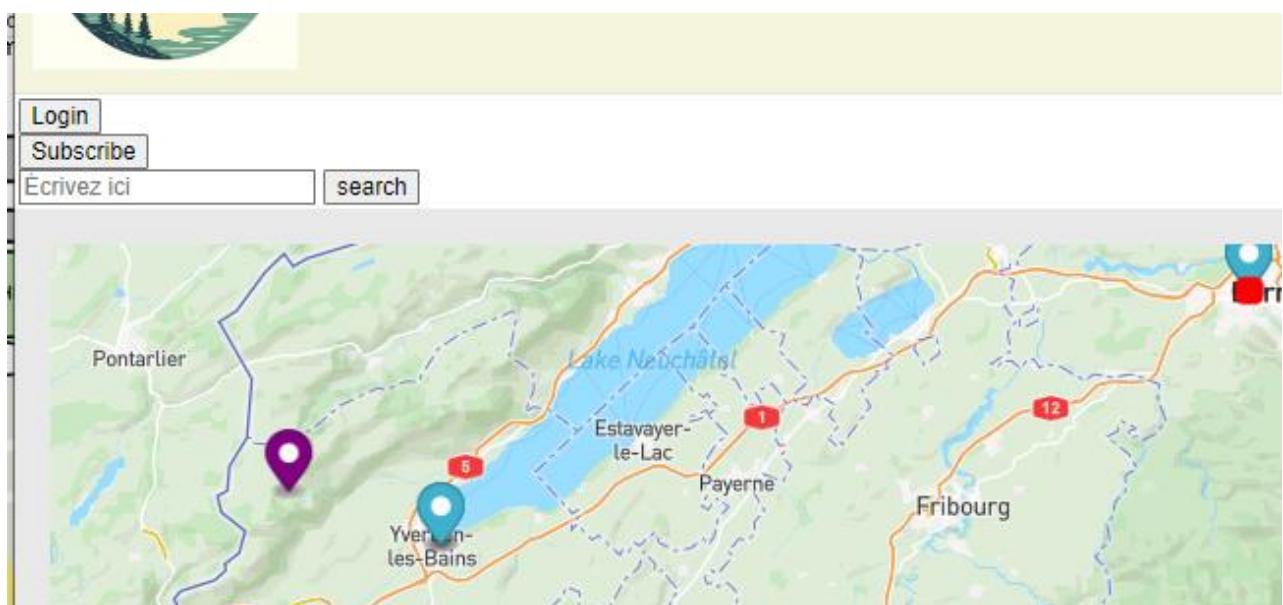


Figure 34: localisation active de l'utilisateur avec un bouton violet et de l'adresse de l'utilisateur avec un bouton rouge

2.1.2 Use Cases et Scénarios

« 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.

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	pas pouvoir contacter l'annonceur sans se connecter	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 utilisateur	Click sur bouton "Subscribe"	Ne pas avoir de compte avec la même adresse email	Inscription de l'utilisateur dans la base de données Session utilisateurs "ouverte", avec ses données chargées dans le sessionstorage	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. 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 valeurs email & password correctes Tout les champs du formulaires sont remplis	bouton "login" se transforme en bt "logout" un bouton "mypage" apparaît un bouton "messages" apparaît un bouton "filter" apparaît	Avertissement "@" si les informations de email sont mal saisies Avertissement donné si toutes les cases ne sont pas remplies Un message d'erreur s'affiche : "vos identifiants sont incorrects"
Fermer le formulaire de login	Cliquez n'importe où sur l'écran		Fermeture du formulaire de login	

Figure 35: Cette section contient des photos de cas d'utilisation.

CAVES - TPI

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	<p>La publicité est enregistrée dans la base de données.</p> <p>Les photos sont enregistrées dans la base de données sous forme d'URL.</p>	<p>Si les informations sur les prix et les stocks sont incomplètes, l'annonce ne sera pas enregistrée dans la base de données.</p>
				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	<p>Le bouton "LOGOUT" se transforme en bouton "LOGIN"</p> <p>Le bouton "MyPage" disparaît</p> <p>Le bouton "Messages" disparaît</p> <p>Le bouton "Filter" disparaît</p> <p>Les produits dans le panier sont supprimés.</p> <p>Le bouton d'adresse sur la carte de l'utilisateur disparaît.</p> <p>Sur la page de détails de l'annonce deviennent</p>	Le bouton est invisible
----------------	-------------------	---	--	-------------------------

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

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 filtrer en fonction des données du tableau dans la base de données.	L'utilisateur peut voir les annonces filtrées sur la carte et sur la page d'accueil en fonction des options de filtrage.	
	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.		

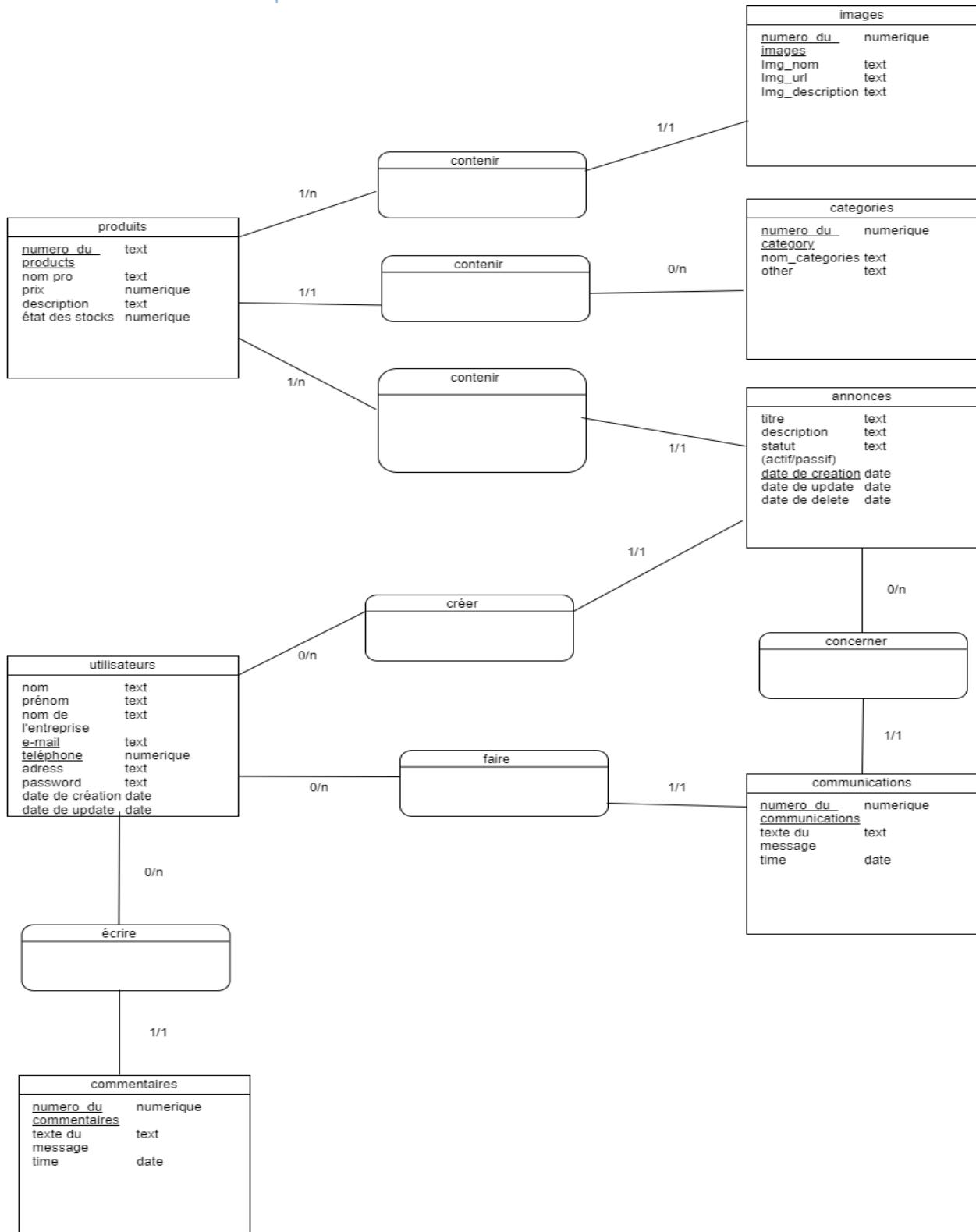
Option de search	Click bt "Search"	Chaque utilisateur peut effectuer une recherche même s'il ne se connecte pas.	Les produits recherchés sont recherchés via le titre et amenés à la page d'accueil. De plus, les publicités sur Ahrita sont concernées par cette recherche.
------------------	-------------------	---	---

CAVES - TPI

Option de Carte	Ce qui est affiché sur la carte	Toutes les publicités peuvent être consultées.	Lorsque vous cliquez sur les boutons publicitaires, une pop-up s'ouvre et lorsque vous cliquez sur la pop-up, vous accédez à la page de détail de l'annonce. Les utilisateurs peuvent afficher l'emplacement et l'adresse avec différents boutons.
		L'adresse de l'utilisateur connecté peut être affichée.	
		Si l'emplacement est activé, l'emplacement actuel peut être vu.	

Option de Panier	En cliquant sur le bouton Ajouter au panier	Le montant ajouté par l'utilisateur est ajouté au panier.	Le montant ajouté est visible dans le panier. Si la quantité ajoutée est supérieure à l'inventaire, aucun ajout au panier ne sera effectué. Les utilisateurs peuvent afficher l'emplacement et l'adresse avec différents boutons.
		L'adresse de l'utilisateur connecté peut être affichée.	
		Si l'emplacement est activé, l'emplacement actuel peut être vu.	

2.1.3 Modèle conceptuel de données



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

Figure 36 : Graphique MCD

2.2 Fonctions prévues

Les principales fonctions prévues dans ce projet sont décrites ci-dessous :

Gestion des utilisateurs :

Les utilisateurs peuvent s'inscrire et se connecter au système en créant un compte. Chaque utilisateur se voit attribuer un nom d'utilisateur et un mot de passe. Les utilisateurs peuvent modifier leurs comptes, les supprimer et gérer leurs produits, le cas échéant.

De plus, si l'utilisateur est connecté, il pourra se voir sur la carte, utiliser la page de détails de l'annonce et les options de filtrage.

Gestion des produits :

Les utilisateurs peuvent répertorier, modifier et supprimer leurs propres produits. Pour chaque produit, ils peuvent ajouter un titre, une description, un prix et des images si disponibles.

Options de filtrage et de recherche

Chaque annonce que les utilisateurs se connectent et ajoute à la base de données peut être filtrée en fonction de leurs préférences personnelles après avoir été affichée sur la carte et la page. Bien que l'option de recherche soit disponible pour chaque utilisateur, l'option de filtrage requise pour la recherche détaillée sera proposée comme fonctionnalité supplémentaire pour l'utilisateur enregistré et connecté.

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

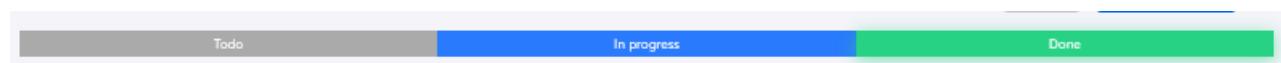
Les produits ajoutés par les utilisateurs sont répertoriés sur la page d'accueil. De plus, grâce à l'intégration cartographique, les utilisateurs peuvent voir les produits à proximité sur la carte en fonction de leur emplacement.

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.

2.3 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.

2.4 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.

CAVES - TPI

cave_release • In progress



Sprint 1 • In progress 30/04 | 07/05 Velocity 0 / 48

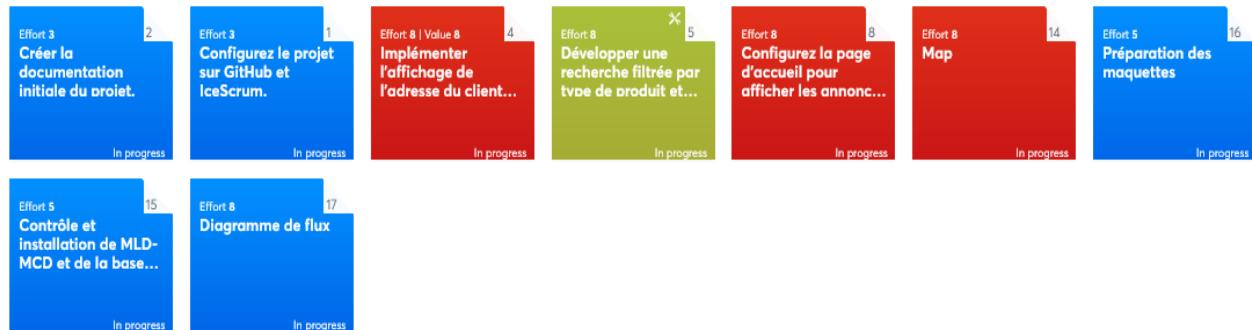


Figure 37: histoires de sprint-1

cave_release - Sprint 1

Stop the sprint

Todo In progress Done

Details Sprint notes

Days

Sprint goal

Établir l'infrastructure de base de notre projet, préparer les documents nécessaires et développer les fonctionnalités initiales.

Figure 38: objectif de sprint-1

Card Type	Value	Status	Progress
Preparation of prototypes	8	YA	0/1 In progress
Control and installation of MLD-MCD and database	10	BCNF	1/2 In progress
Preparation of prototypes	9	YA	0/1 In progress

Figure 39: système Kanban

- 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.



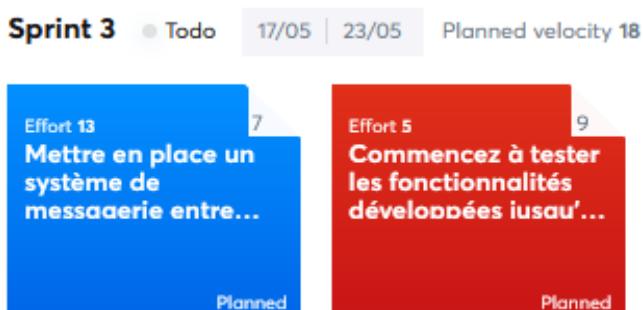
Figure 40: histoires de sprint-2

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.



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.

Figure 41: histoires de sprint-3

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



Figure 42: sprint-3 après sprint-2

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

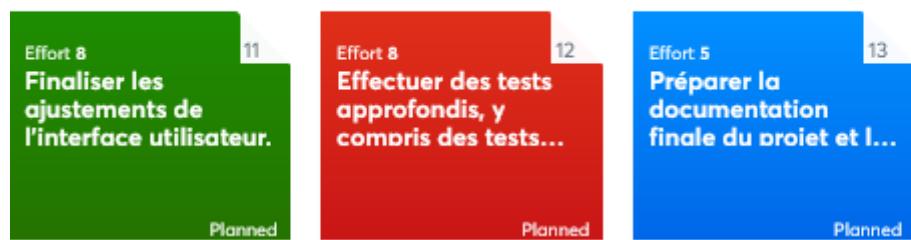


Figure 43: histoires de sprint-4

2.5 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.

3 Implémentation

3.1 Choix techniques

Dans mon projet, j'utilise une combinaison de technologies front-end et back-end, ainsi que divers outils de développement et de gestion de base de données pour offrir une expérience utilisateur optimale et gérer efficacement les données.

Technologies Employées

- PHP (avec PhpStorm): J'ai choisi PHP pour gérer les interactions avec la base de données et le traitement des données côté serveur. J'utilise PhpStorm comme IDE, car il enrichit mon environnement de développement avec d'excellents outils de débogage et de gestion de code.
- JavaScript (Fetch API): J'emploie JavaScript pour dynamiser les interactions client, en utilisant la Fetch

API pour des requêtes asynchrones, permettant ainsi de communiquer avec le backend sans recharger la page.

J'ai opté pour la Fetch API car elle offre une interface plus moderne pour les requêtes asynchrones, supporte nativement les promesses et rend mon code plus propre et plus facile à maintenir.

- MySQL (via HeidiSQL et MySQL Workbench): J'utilise MySQL pour stocker et récupérer les données nécessaires à l'application. HeidiSQL et MySQL Workbench me permettent de gérer visuellement les bases de données, effectuer des requêtes et optimiser les schémas.

Outils de Conception et de Modélisation

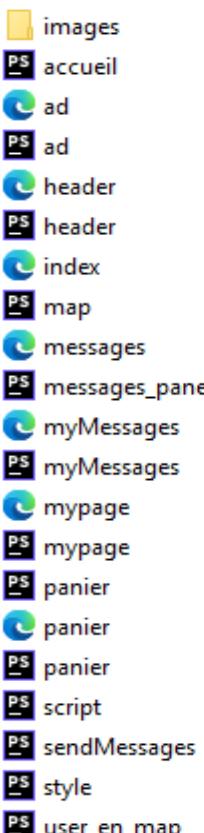
- Balsamiq: Pour la création rapide de maquettes initiales, je me sers de Balsamiq, qui aide à visualiser les interfaces utilisateur et à définir l'expérience utilisateur de manière intuitive.
- MySQL Workbench: Je l'utilise pour la modélisation logique des données (MLD), ce qui facilite grandement la conception et la visualisation des relations entre les données.
- Draw.io: Pour la modélisation conceptuelle des données (MCD), Draw.io est mon outil de choix, permettant de créer des diagrammes clairs et détaillés.

Outils de Test et de Documentation

- Excel: J'emploie Excel pour créer et gérer des tests, documenter des cas d'utilisation et suivre les erreurs et les fonctionnalités tout au long du développement.
- Word : Documentation projet

3.2 Architecture du Système

- Frontend



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.

- accueil.js : Ce fichier contient les fonctions JavaScript nécessaires pour la page d'accueil. Il gère le chargement des annonces, les opérations de filtrage et la gestion des marqueurs sur la carte.
- ad.html : C'est la page HTML où sont affichés les détails des annonces. Cette page présente les informations et les images des annonces aux utilisateurs.
- ad.js : Ce fichier fonctionne avec ad.html pour récupérer, filtrer et afficher les détails des annonces. Il gère également les interactions utilisateur.
- index.html : La page de connexion principale de l'application. Elle fournit une interface de base pour que les utilisateurs puissent se connecter ou s'inscrire.
- myMessages.html : La page HTML où les utilisateurs peuvent voir leurs messages. Elle fournit une interface de messagerie.
- myMessages.js : Fonctionne avec myMessages.html pour récupérer, envoyer et gérer les messages des utilisateurs.
- mypage.html : La page de profil utilisateur. Elle affiche les informations de l'utilisateur et le contenu personnalisé pour

Figure 44: Fichiers frontend

lui.

- mypage.js : Fonctionne avec mypage.html pour fournir les fonctionnalités dynamiques de la page de profil.
 - panier.html : La page HTML où les utilisateurs peuvent voir leur panier d'achat.
 - panier.js : Fonctionne avec panier.html pour gérer les opérations du panier (ajout, suppression, mise à jour des produits).
 - script.js
- Fichier JavaScript général contenant plusieurs fonctions :
- o Gestion des connexions et déconnexions des utilisateurs
 - o Validation des formulaires
 - o Chargement dynamique du contenu
 - o Mise à jour de l'interface utilisateur
 - o Récupération et envoi des données via Fetch
 - o Gestion des événements pour les éléments interactifs
 - o Gestion des messages d'erreur
- sendMessages.js : Gère l'envoi des messages par les utilisateurs.
 - user_en_map.js : Gère l'affichage des emplacements des utilisateurs sur la carte et les interactions avec celle-ci.
 - Images / (Dossier) : Contient toutes les images utilisées dans l'application, par exemple les images des produits, les images de fond, etc.

- 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.

- accueil.php : Utilisé pour récupérer les annonces de la base de données et effectuer des opérations de filtrage pour la page d'accueil. Crée des requêtes SQL en fonction des critères spécifiés par l'utilisateur et renvoie les résultats en format JSON.
- ad.php : Fichier PHP qui récupère les détails d'une annonce spécifique de la base de données et renvoie les informations en format JSON.
- checkStock.php : Vérifie l'état du stock d'un produit spécifique et renvoie les informations de stock en format JSON.
- dbConnector.php : Gère les connexions à la base de données. Utilise PDO pour se connecter à la base de données

Figure 45: Fichiers backend

MySQL et facilite les opérations de connexion.

- getMessages.php : Récupère les messages des utilisateurs de la base de données et les renvoie en format JSON.
- login.php : Gère les opérations de connexion des utilisateurs. Authentifie les utilisateurs et initialise les sessions.
- logout.php : Gère les opérations de déconnexion des utilisateurs. Termine la session de l'utilisateur.
- mypage.php : Gère et met à jour les informations du profil utilisateur. Récupère les informations de profil de l'utilisateur de la base de données et effectue les opérations de mise à jour nécessaires.
- restoreStock.php : Restaure la quantité de stock lorsque des produits sont retirés du panier. Effectue les mises à jour de stock nécessaires.
- sendMessages.php : Permet aux utilisateurs d'envoyer des messages. Enregistre les messages dans

la base de données.

- subscribe.php : Gère les opérations d'inscription des nouveaux utilisateurs. Récupère les informations des utilisateurs et les enregistre dans la base de données.
- updateStock.php : Met à jour la quantité de stock des produits. Gère les mises à jour de stock lors des ventes ou des réductions de stock.
- user_en_map.php : Récupère les informations d'adresse des utilisateurs de la base de données et les renvoie en format JSON.

3.3 Système de base de données

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

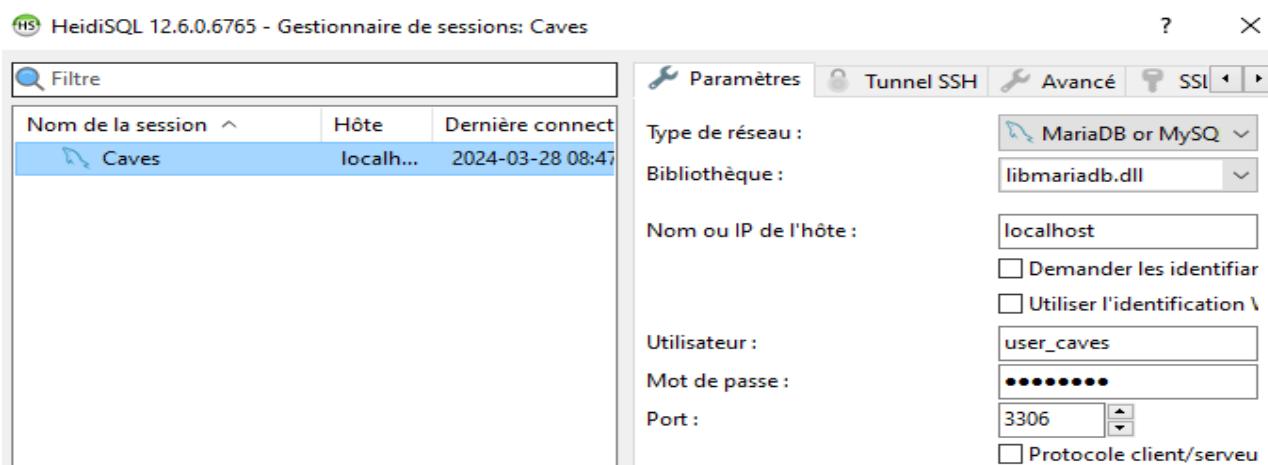


Figure 46 : Configuration de la base de données

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

Figure 47 : tables dans la base de données

Modèle de Données : Le modèle relationnel permet de définir les relations entre les différentes tables. Chaque table est structurée de manière à optimiser la récupération et la manipulation des données tout en garantissant la normalisation pour éviter les redondances.

Intégrité des Données : Des clés primaires et étrangères sont utilisées pour maintenir l'intégrité référentielle entre les tables. Les contraintes de clé étrangère garantissent que les relations entre les tables restent cohérentes lors des opérations de mise à jour et de suppression.

Détails de la Table des Messages

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'

Figure 48 : tableau des messages

```

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;

```

Explication de la Table messages :

- id: C'est la clé primaire de la table, utilisée pour identifier de manière unique chaque message. Elle est de type INT UNSIGNED et auto-incrémentée.
- buyer_id : Représente l'identifiant de l'acheteur qui envoie le message. Il s'agit d'une clé étrangère référencée à l'identifiant (id) de la table users.
- seller_id : Représente l'identifiant du vendeur qui reçoit le message. Il s'agit également d'une clé étrangère référencée à l'identifiant (id) de la table users.
- announcement_id : Représente l'identifiant de l'annonce concernée par le message. Cette colonne est une clé étrangère référencée à l'identifiant (id) de la table announcements.
- message_text : Contient le texte du message envoyé par l'acheteur. La longueur maximale est de 255 caractères.

- **sent_time:** Enregistre l'heure et la date d'envoi du message. Il est de type TIMESTAMP et prend par défaut la valeur de l'heure actuelle au moment de l'insertion.

Index et Contraintes:

- Des index sont créés sur les colonnes `buyer_id`, `seller_id` et `announcement_id` pour optimiser les performances des requêtes.
- Les contraintes de clé étrangère (`fk_messages_sender`, `fk_messages_receiver`, `fk_messages_announcement`) assurent que les identifiants de `buyer_id`, `seller_id` et `announcement_id` existent dans leurs tables respectives et garantissent l'intégrité référentielle.

Cette structure garantit que chaque message est correctement associé à un acheteur, un vendeur et une annonce spécifique, tout en maintenant l'intégrité et la cohérence des données.

3.4 Modèle Logique de données

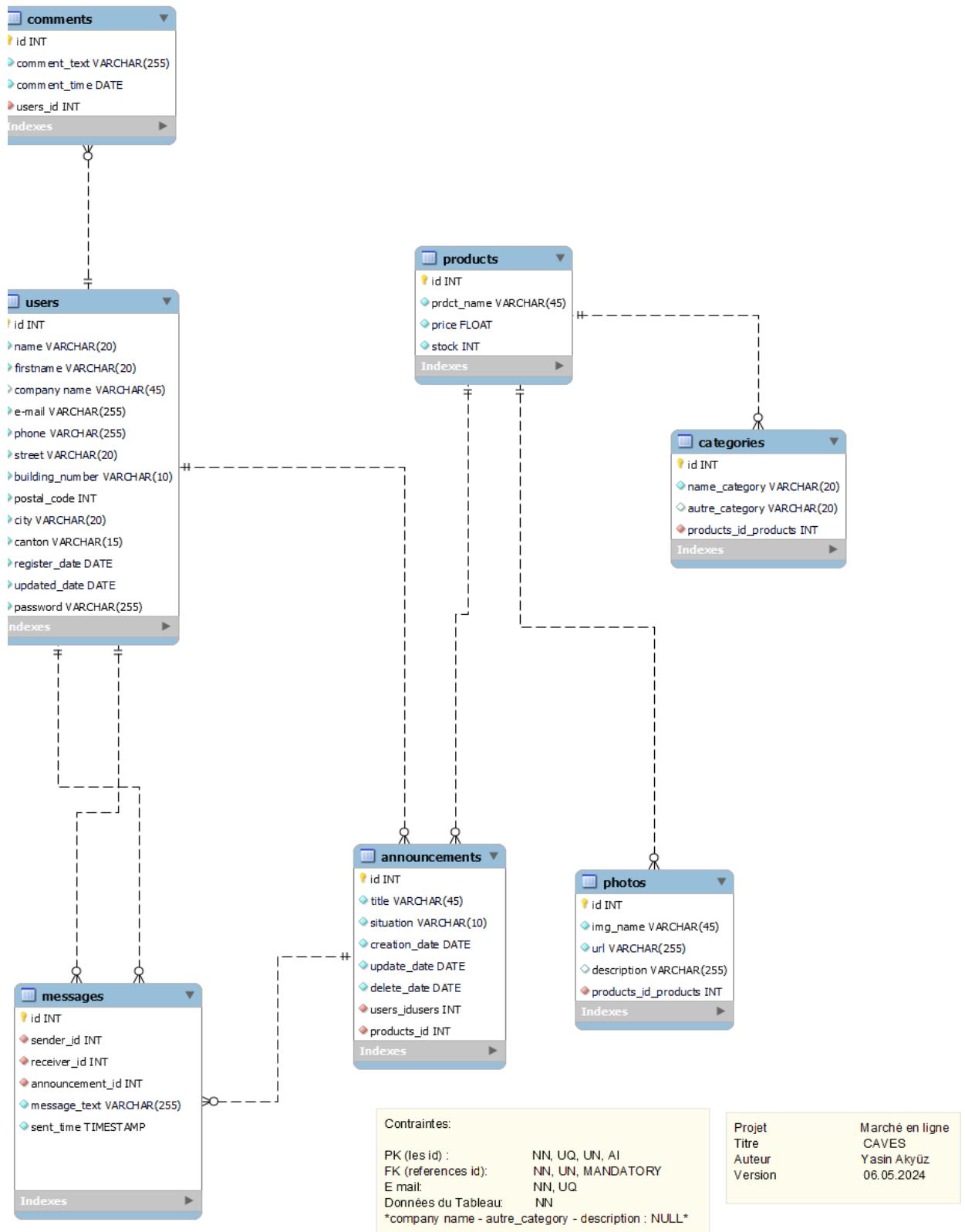


Figure 49 : Tableau MLD

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.

- 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.

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"/>	<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"/>	<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"/>	<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"/>	<input type="checkbox"/>
e-mail	VARCHAR(255)	<input type="checkbox"/>	<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"/>
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"/>	<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"/>	<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"/>	<input type="checkbox"/>
postal_code	INT	<input type="checkbox"/>	<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 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 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 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 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 type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure 50 : Tableau des utilisateurs

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"/>	<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"/>	<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"/>	<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"/>	<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"/>	<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"/>	<input type="checkbox"/>
users_idusers	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input 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 checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure 51 : Affichage du tableau d'annonce et des références

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.

3.5 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.

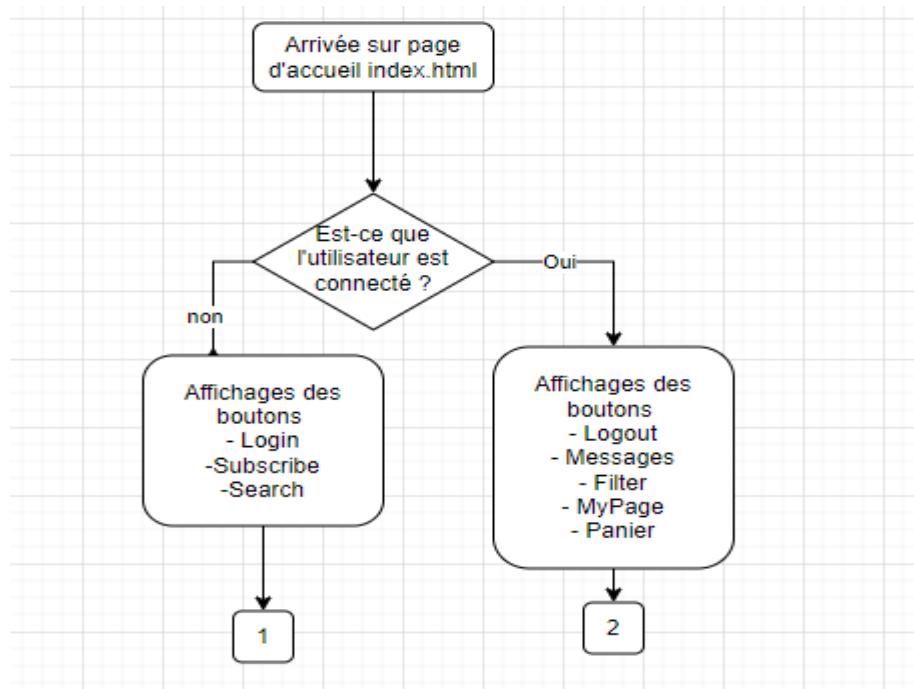


Figure 52: Début du flux dans le diagramme

CAVES - TPI

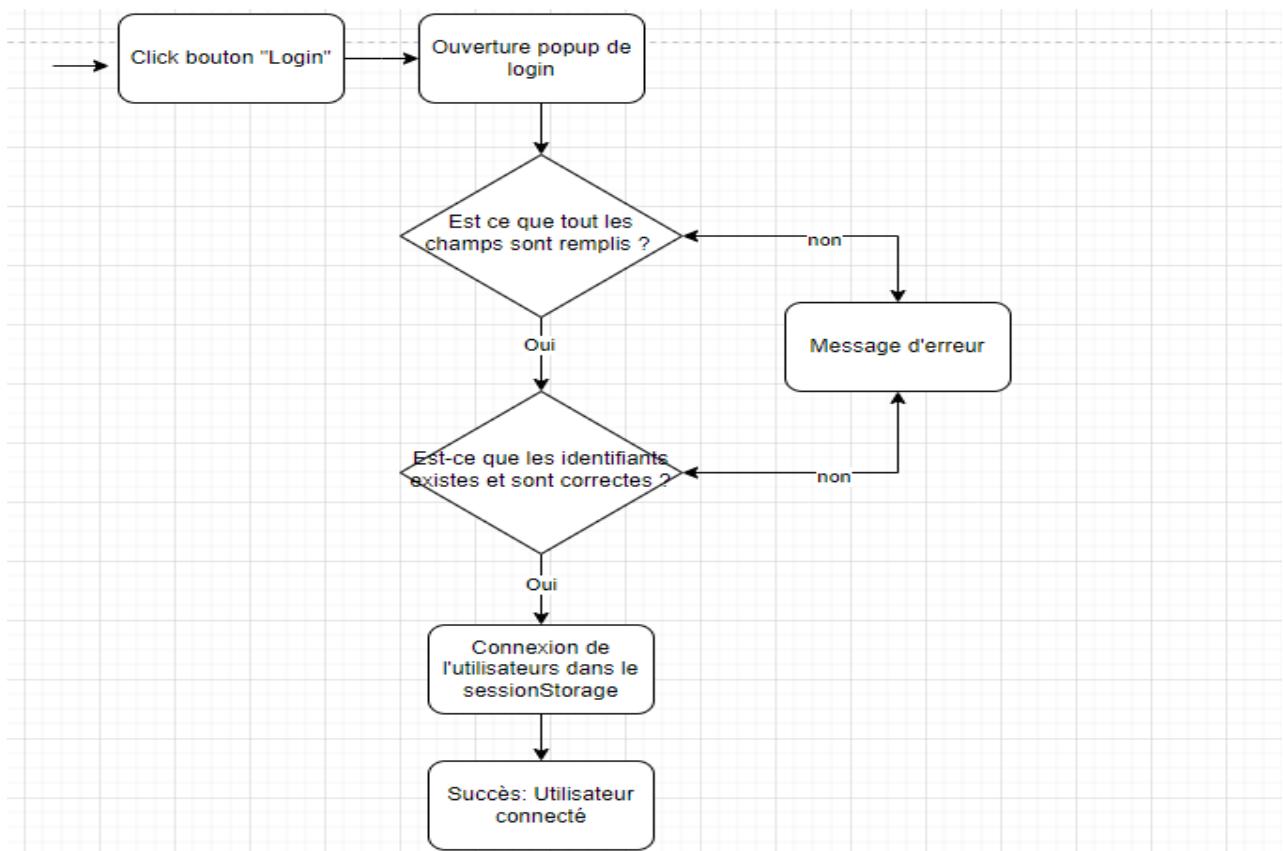


Figure 53 : schéma de connexion

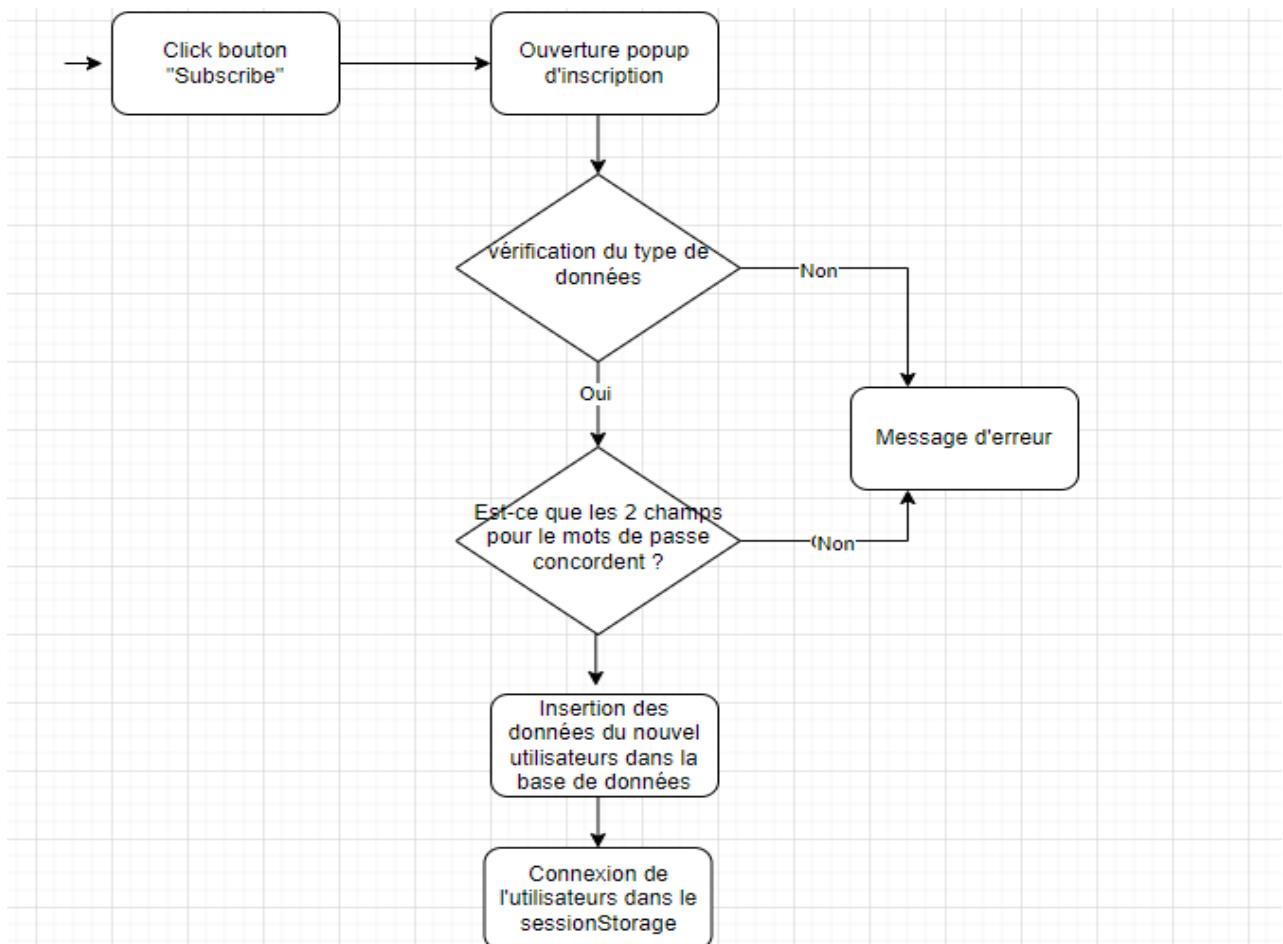


Figure 54 : schéma d'inscription

CAVES - TPI

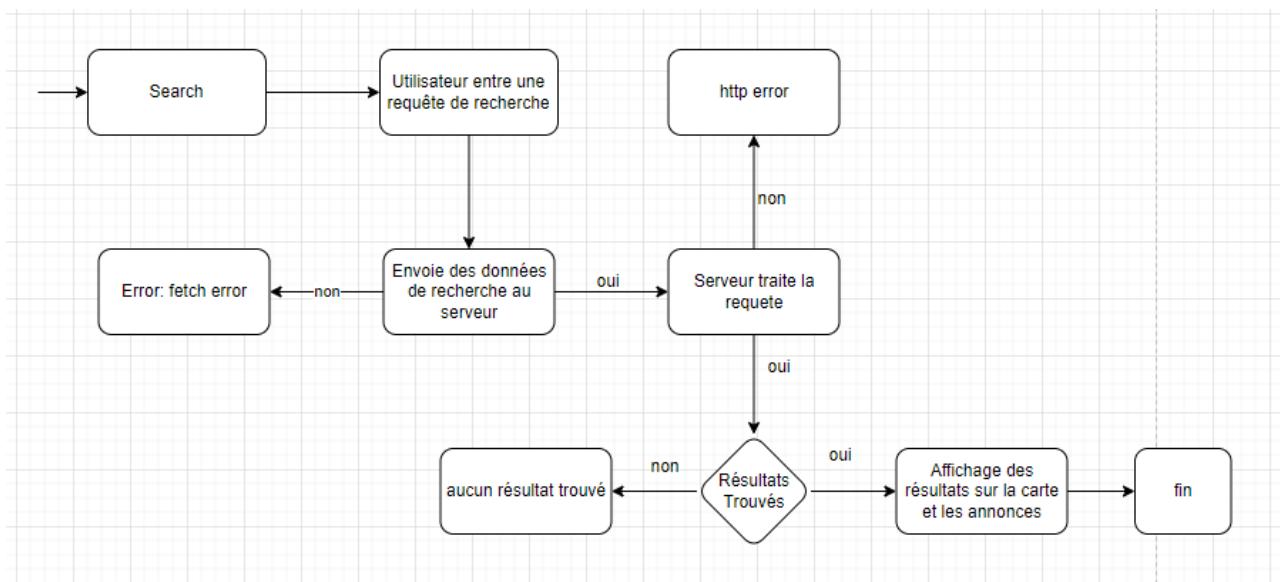


Figure 55 : diagramme de recherche

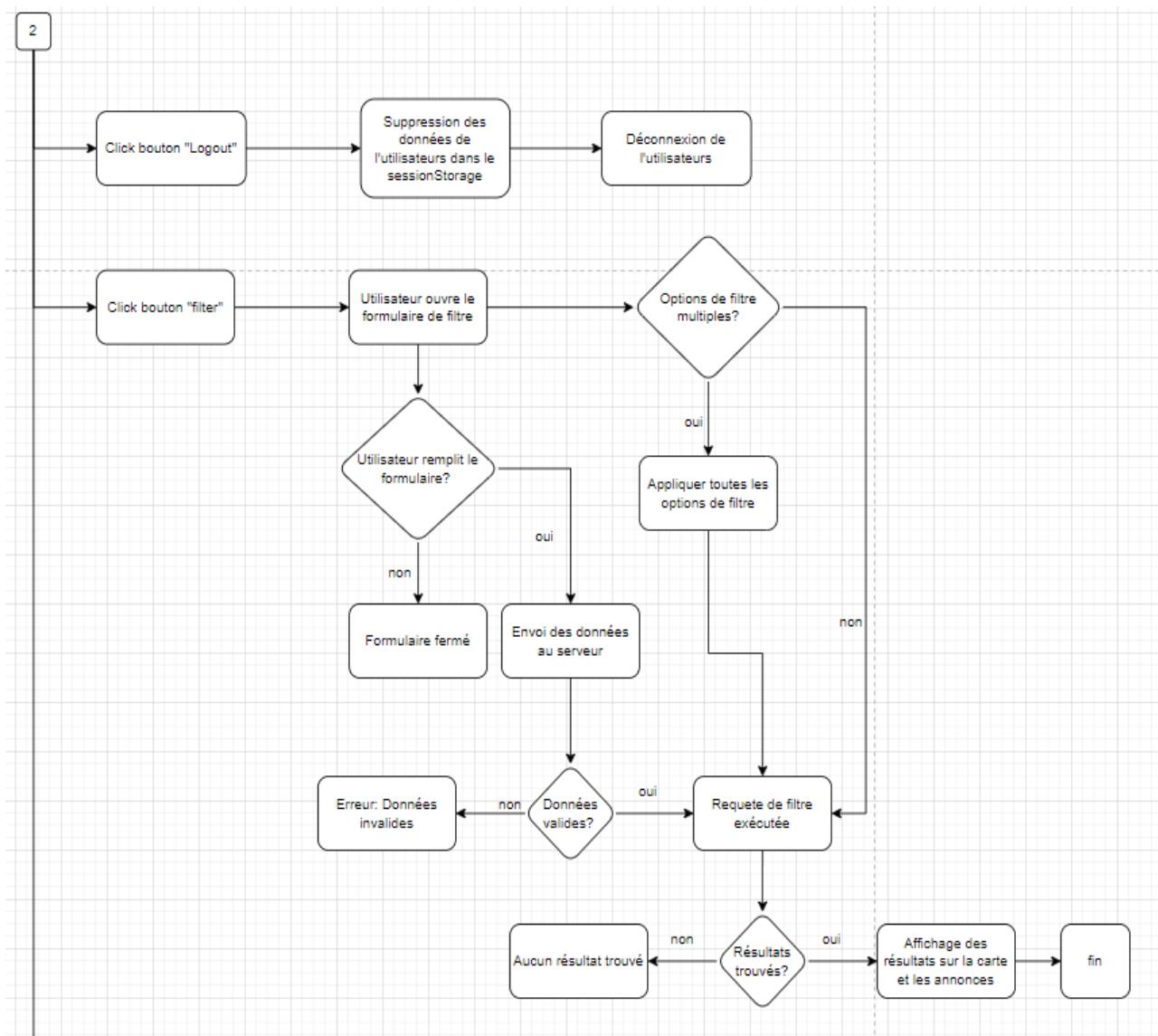


Figure 56 : schéma de déconnexion et de filtrage

CAVES - TPI

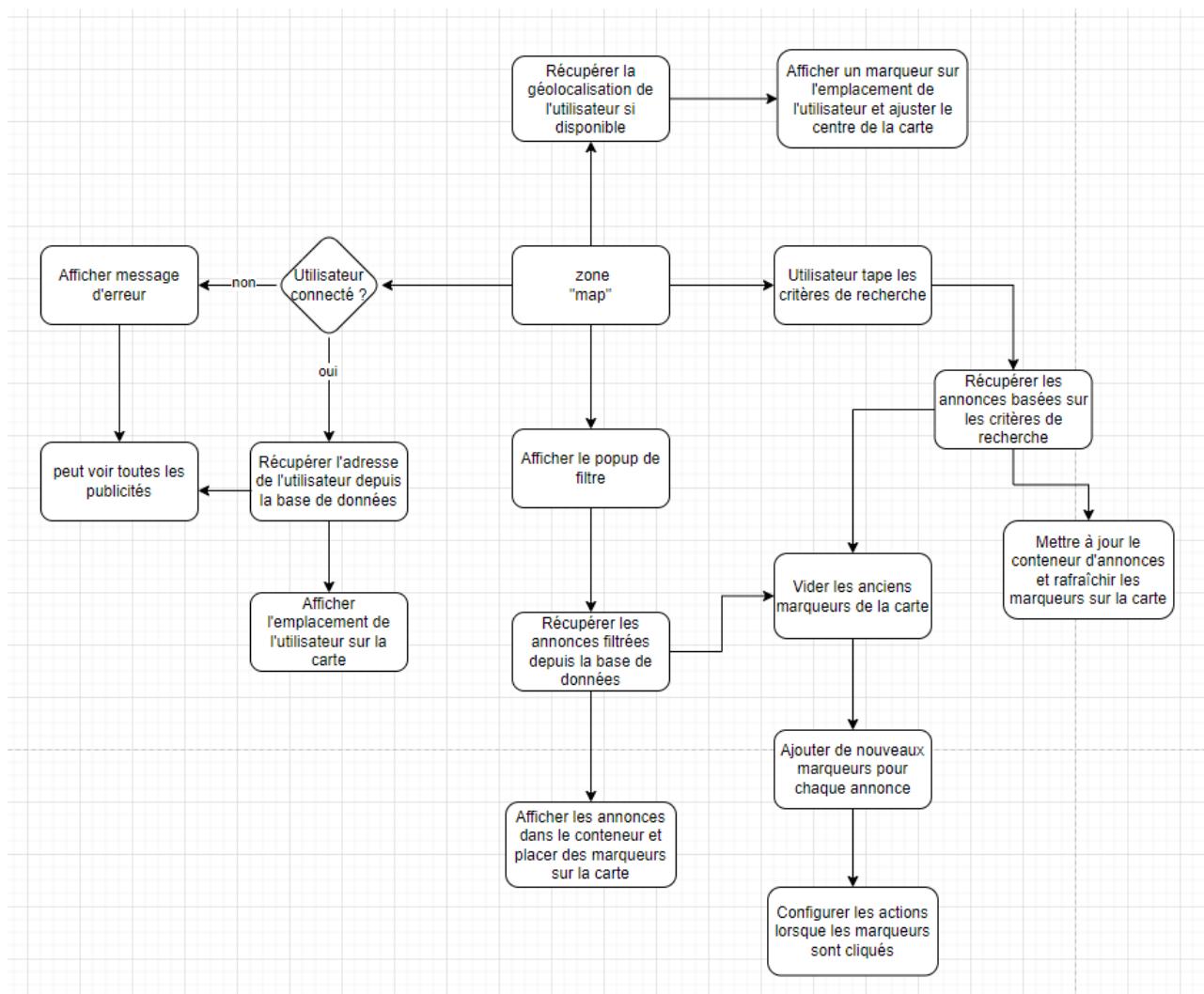


Figure 57 : diagramme de carte

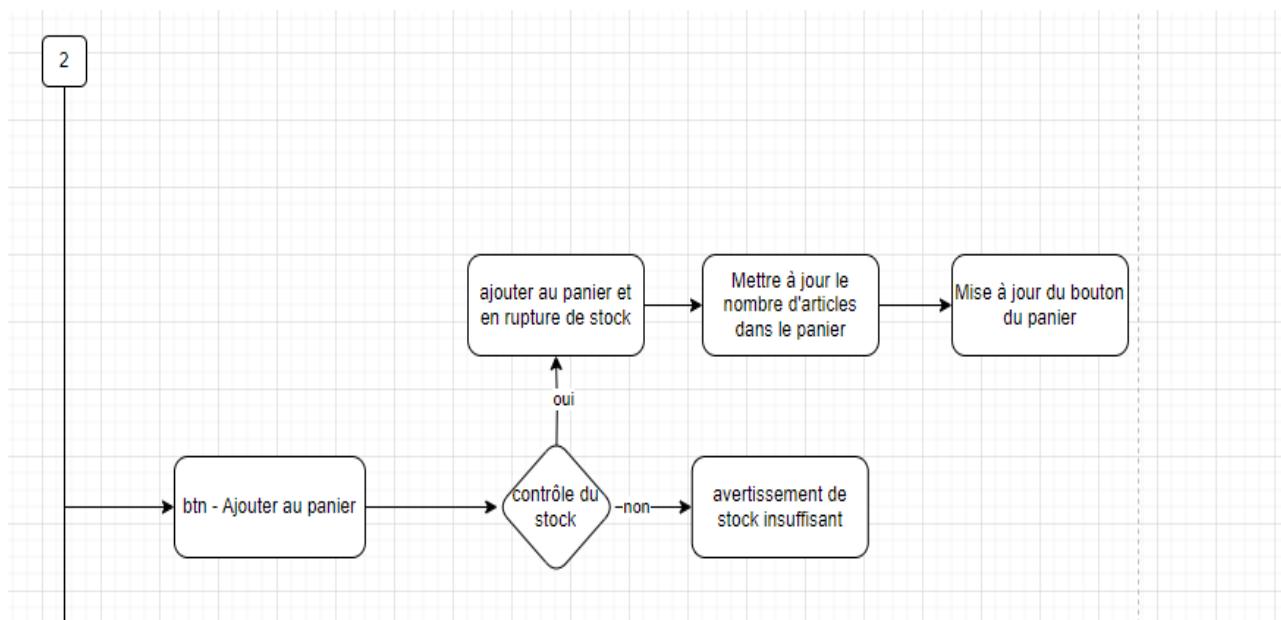


Figure 58 : ajout d'articles au diagramme du panier

CAVES - TPI

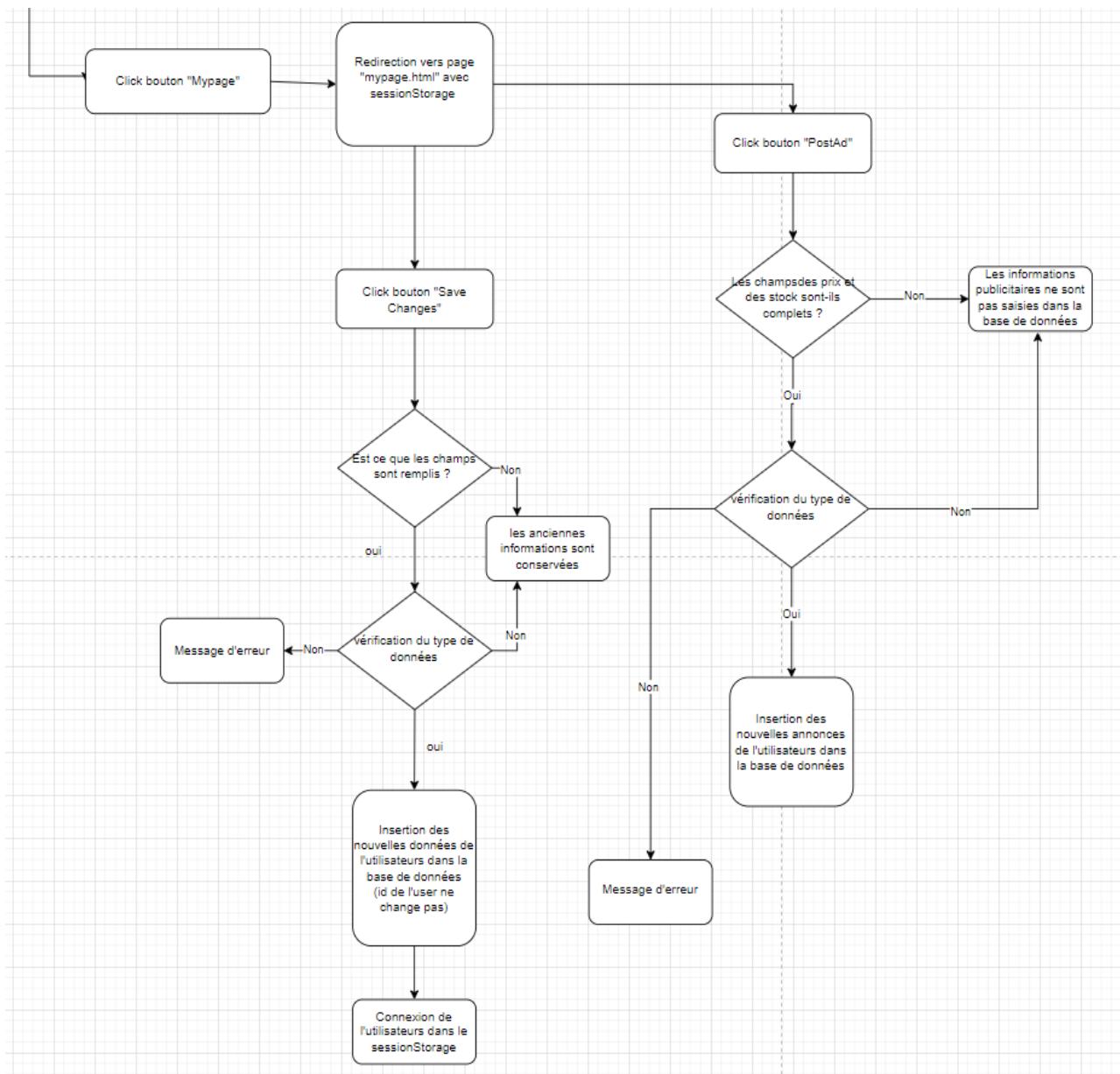


Figure 59 : diagramme de page personnelle

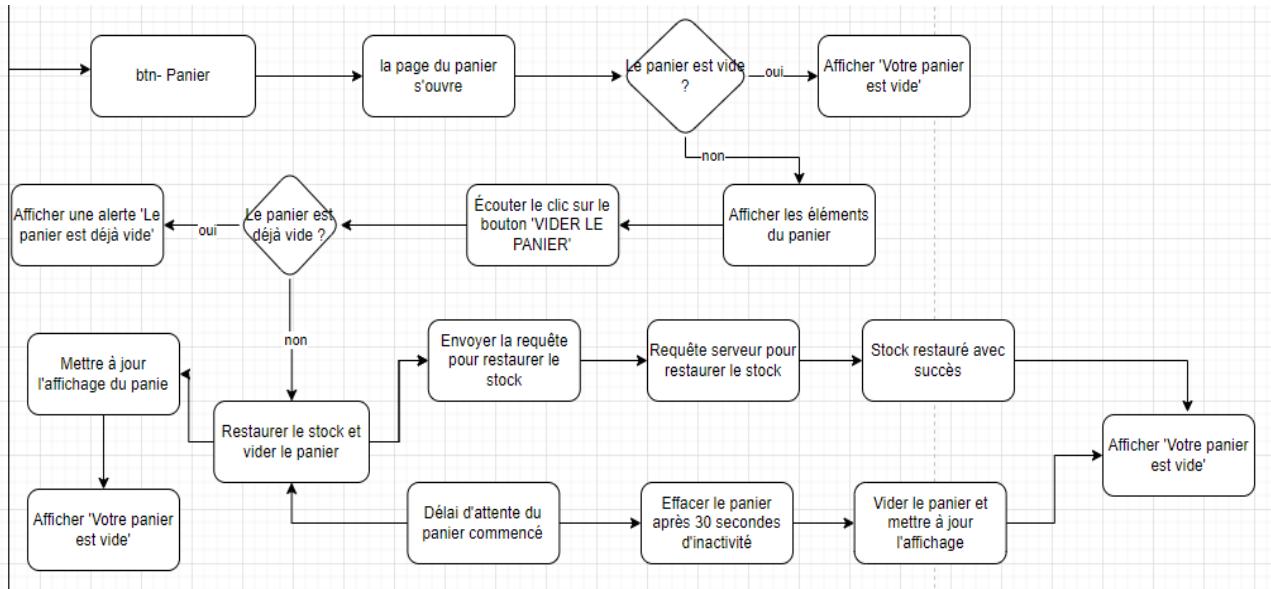


Figure 60 : diagramme du panier

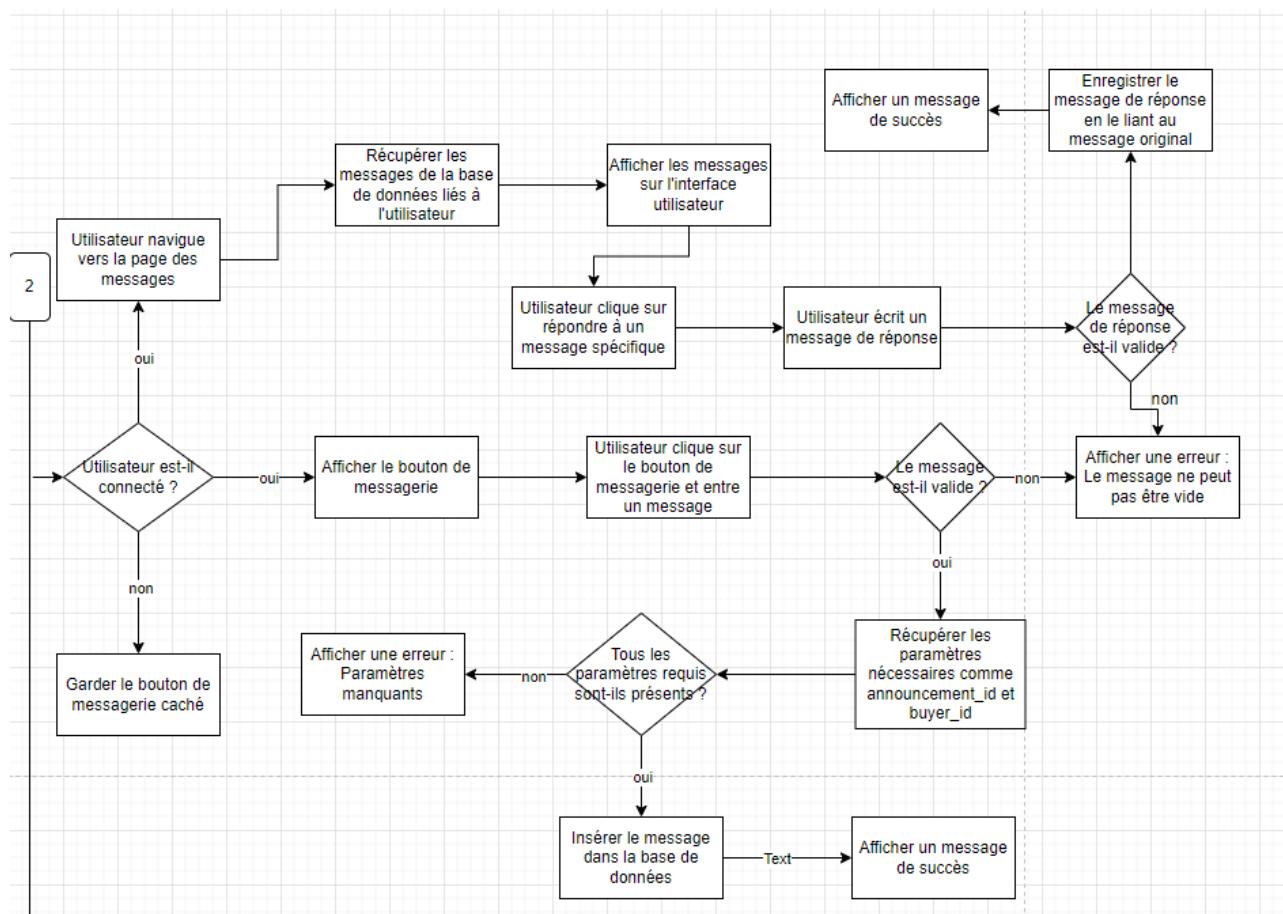


Figure 61 : schéma du système de messagerie

3.6 Ergonomie de site (Bastien et Scapin)

Ce document décrit les améliorations apportées à l'interface utilisateur du projet Caves pour répondre aux critères de Bastien et Scapin en matière de conception de l'interface utilisateur.

Les critères de Bastien et Scapin sont des recommandations de conception visant à améliorer l'utilisabilité des interfaces utilisateurs. Ils couvrent divers aspects tels que l'ergonomie, l'efficacité, la satisfaction de l'utilisateur, et la facilité d'apprentissage. Dans ce projet, nous avons appliqué plusieurs de ces critères pour optimiser l'expérience utilisateur.

- Critères de Bastien et Scapin utilisés :

Guidage : Les utilisateurs doivent être guidés dans leurs actions pour éviter les erreurs et maximiser l'efficacité.

Groupement / Distinction par la mise en forme : Les éléments similaires doivent être regroupés et ceux qui sont différents doivent être distingués par leur mise en forme.

Contrôle immédiat: Les utilisateurs reçoivent des retours immédiats sur leurs actions.

Contrôle explicite : L'utilisateur doit toujours être informé des actions qu'il peut effectuer et de l'état du système.

Retour immédiat : Fournir des retours immédiats aux actions des utilisateurs.

Flexibilité et efficacité : Le système doit permettre à l'utilisateur d'être plus efficace et de personnaliser son expérience.

Saisie et formatage automatique (Automatic Data Entry and Formatting) : Faciliter la saisie des données et leur formatage pour réduire les erreurs et améliorer l'efficacité.

Gestion des erreurs : Aider les utilisateurs à corriger les erreurs et à éviter les erreurs potentielles.

Champs de Formulaire avec Validation et Suggestions

Critère utilisé : Guidage - Contrôle explicite

Afin de garantir que tous les champs nécessaires sont correctement remplis par l'utilisateur, nous avons intégré des mécanismes de validation. Par exemple, les champs de saisie de l'adresse e-mail et du mot de passe lors de l'inscription et de la connexion doivent être remplis avant de soumettre le formulaire.

```
<label for="login-email">E-mail</label>
<input type="email" id="login-email" name="email" placeholder="example@example.com" required>
<div id="email-suggestions" class="suggestions"></div>
<label for="login-password">Password</label>
<input type="password" id="login-password" name="password" placeholder="Password" required>
<input type="submit" value="Login">
```

Suggestions de Saisie pour l'E-mail

Pour aider l'utilisateur à remplir rapidement et correctement son adresse e-mail, j'ai ajouté une fonctionnalité de suggestion de domaines courants (comme gmail.com, yahoo.com, etc.). Lorsque l'utilisateur commence à saisir son adresse e-mail et tape le caractère '@', une liste de suggestions s'affiche automatiquement, facilitant ainsi la complétion de l'adresse.

```
const emailInput : HTMLElement = document.getElementById( elementId: 'login-email' );
const suggestionsBox : HTMLElement = document.getElementById( elementId: 'email-suggestions' );
const domains : string[] = ['gmail.com', 'yahoo.com', 'hotmail.com', 'outlook.com'];
new *
emailInput.addEventListener( type: 'input', listener: function() : void {
    const value = emailInput.value;
```

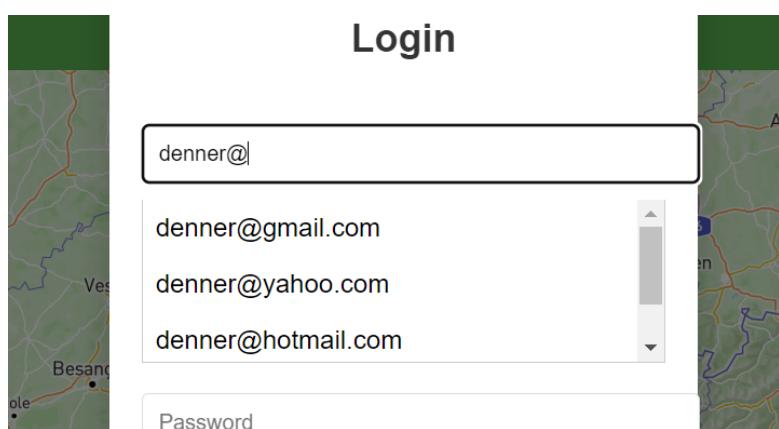


Figure 62 : Suggestions pour l'E-mail

Les utilisateurs reçoivent des retours immédiats sur leurs actions.

Critère utilisé : Contrôle immédiat

Lorsqu'un utilisateur commence à saisir son mot de passe, un retour immédiat est fourni sur la force du mot de passe. Si le mot de passe contient au moins 8 caractères, un message "Mot de passe valide" s'affiche en vert. Sinon, un message "Le mot de passe doit contenir au moins 8 caractères" s'affiche en rouge.

```
if (this.value.length >= 8) {
    passwordFeedback.textContent = 'Mot de passe valide';
    passwordFeedback.style.color = 'green';
} else {
    passwordFeedback.textContent = 'Le mot de passe doit contenir au moins 8 caractères';
    passwordFeedback.style.color = 'red';
```

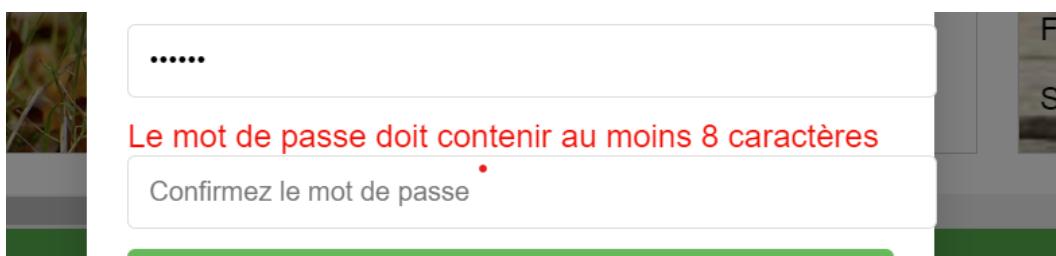


Figure 63 : exemple de mot de passe incompatible

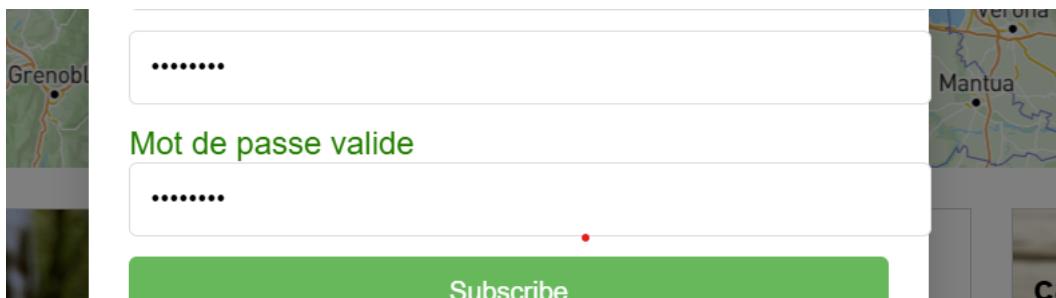


Figure 64 : exemple de mot de passe correspondant

Formatage automatique du numéro de téléphone

Critère utilisé :

Groupement / Distinction par la mise en forme - Flexibilité et efficacité - Saisie et formatage automatique

Pour améliorer l'ergonomie lors de la saisie du numéro de téléphone, j'ai ajouté une fonctionnalité de formatage automatique. Lorsque l'utilisateur saisit son numéro, celui-ci est automatiquement formaté en groupes de trois chiffres, ce qui améliore la lisibilité et réduit les erreurs de saisie.
(par exemple, 0123456789 devient 012 345 6789).

example@example.com

077 777 7777

Adresse

Figure 65: Formatage automatique

```
document.getElementById('phone').addEventListener('input', function() {
    let value = e.target.value.replace(/\D/g, '');
    if (value.length > 3 && value.length <= 6) {
        value = value.replace(/(\d{3})(\d+)/, '$1 $2');
    } else if (value.length > 6 && value.length <= 10) {
        value = value.replace(/(\d{3})(\d{3})(\d+)/, '$1 $2 $3');
    } else if (value.length > 10) {
        value = value.replace(/(\d{3})(\d{3})(\d{4})/, '$1 $2 $3');
    }
    e.target.value = value;
})
```

Figure 66: code d'exploitation

Confirmation du mot de passe

Critère utilisé : Gestion des erreurs

Lors de l'inscription, l'utilisateur doit confirmer son mot de passe. Si les mots de passe ne correspondent pas, un message d'erreur apparaît immédiatement, permettant à l'utilisateur de corriger l'erreur avant de soumettre le formulaire.

```
// Validate input
if (!filter_var($e_email, FILTER_VALIDATE_EMAIL)) {
    echo 'Invalid email format';
    exit;
}
// ...validate other fields
// Check if user already exists
$stmt = $conn->prepare("SELECT COUNT(*) FROM users WHERE e_email = ?");
$stmt->execute([$e_email]);
if ($stmt->fetchColumn() > 0) {
    echo 'User already exists';
}
```

Figure 67: format e-mail et code d'alerte pour les utilisateurs enregistrés

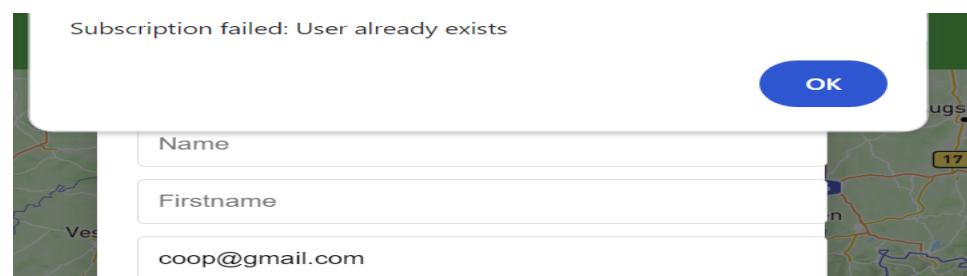


Figure 68: avertissement d'enregistrement d'utilisateur :

- Composants de l'En-tête et du Thème

En-tête :

L'en-tête offre une navigation aux utilisateurs sur le site web.

Les utilisateurs peuvent revenir à la page d'accueil à tout moment en cliquant sur le logo. Ceci est conforme au critère "Trouvabilité".

```
<header>
    <a href="index.html" id="home-link">
        
    </a>
    <label for="theme-selector">Choose theme:</label>
    <select id="theme-selector">
        <option value="theme-light">Light</option>
        <option value="theme-dark">Dark</option>
    </select>
```

Figure 69 : installation de l'en-tête et du thème

Le sélecteur de thème dans l'en-tête permet aux utilisateurs de personnaliser l'apparence de l'interface selon leurs préférences, soutenant le critère de "Contrôle utilisateur".



Figure 70 : vue d'en-tête sur la page du panier

Choix du Thème :

Deux options de thème sont proposées aux utilisateurs : clair et sombre, renforçant le critère de "Personnalisation".

La sélection de thème permet aux utilisateurs d'ajuster la lisibilité de l'écran selon leur confort visuel, servant ainsi le critère de "Visibilité".

```
const themeSelector : HTMLElement = document.getElementById( elementId: 'theme-selector' )
if (themeSelector) {
    const savedTheme : string = localStorage.getItem( key: 'theme' );
    if (savedTheme) {
        document.body.className = savedTheme;
        themeSelector.value = savedTheme;
    }
    themeSelector.addEventListener( type: 'change' , listener: function() : void {
        const selectedTheme : string = themeSelector.value;
```

Figure 71: code explicatif de la fonction de thème

Ce code JavaScript met à jour dynamiquement le style de la page selon le choix de thème de l'utilisateur, répondant ainsi au critère "Feedback immédiat".



Figure 72: apparition du thème sombre sur la page du panier

Retours sur l'Interaction Utilisateur:

Le thème de la page est mis à jour immédiatement lorsqu'un choix est fait, montrant directement les résultats des actions de l'utilisateur.

L'utilisateur vit une expérience de thème cohérente à travers le site, ce qui est conforme aux critères de "Cohérence" et "Standards".

- **Utilisation de la Carte selon les Critères de Bastien et Scapin !**

Contexte et Objectifs

L'utilisation efficace d'une carte dans une application web peut considérablement améliorer l'expérience utilisateur en facilitant la navigation et l'accès aux informations géographiques pertinentes.

Cette section explore l'intégration d'une carte interactive conformément aux critères ergonomiques de Bastien et Scapin, en mettant l'accent sur les fonctionnalités liées à la géolocalisation et à l'interaction avec des éléments cartographiques.

Fonctionnalités de la Carte

Visualisation de l'Adresse Personnelle

- Après connexion, l'utilisateur peut visualiser son adresse sur la carte marquée par un bouton de couleur distinctive, ce qui améliore la « guidance » en aidant l'utilisateur à se situer rapidement.

Interaction avec les Annonces

- Chaque annonce est accessible via un bouton sur la carte. Cliquer sur ce bouton redirige vers une page détaillée de l'annonce, facilitant ainsi « l'accès aux informations » pertinentes sans navigation supplémentaire.

Utilisation des Fonctions de Filtre et de Recherche sur la Carte

- Les boutons de filtre et de recherche permettent aux utilisateurs de « personnaliser l'affichage des annonces » sur la carte, répondant au critère de « contrôle utilisateur ».

Affichage de la Position Actuelle de l'Utilisateur !

- Si l'utilisateur active la géolocalisation, sa position est marquée par un bouton violet sur la carte, améliorant la « conformité » et la « flexibilité d'utilisation ».

- La carte se centre et zoome sur la position de l'utilisateur pour une meilleure visibilité.

- De code pour obtenir et montrer la position de l'utilisateur :

```

function getUserLocation() :void {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(
            successCallback: function(position :GeolocationPosition ) :void {
                console.log('User geolocation:', position);
                const userCoords :number[] = [position.coords.longitude, position.coords.latitude];

                const userLocationMarker = new mapboxgl.Marker({ color: 'purple' }) // purple marker
                    .setLngLat(userCoords)
                    .addTo(map);

                map.flyTo({ center: userCoords, zoom: 9 }); // Focus sur la carte sur l'emplacement de
            }
        );
    }
}

```

Figure 73: Code qui exécute la localisation de l'utilisateur

Sur cette photo, on peut voir la position actuelle avec le bouton violet ;

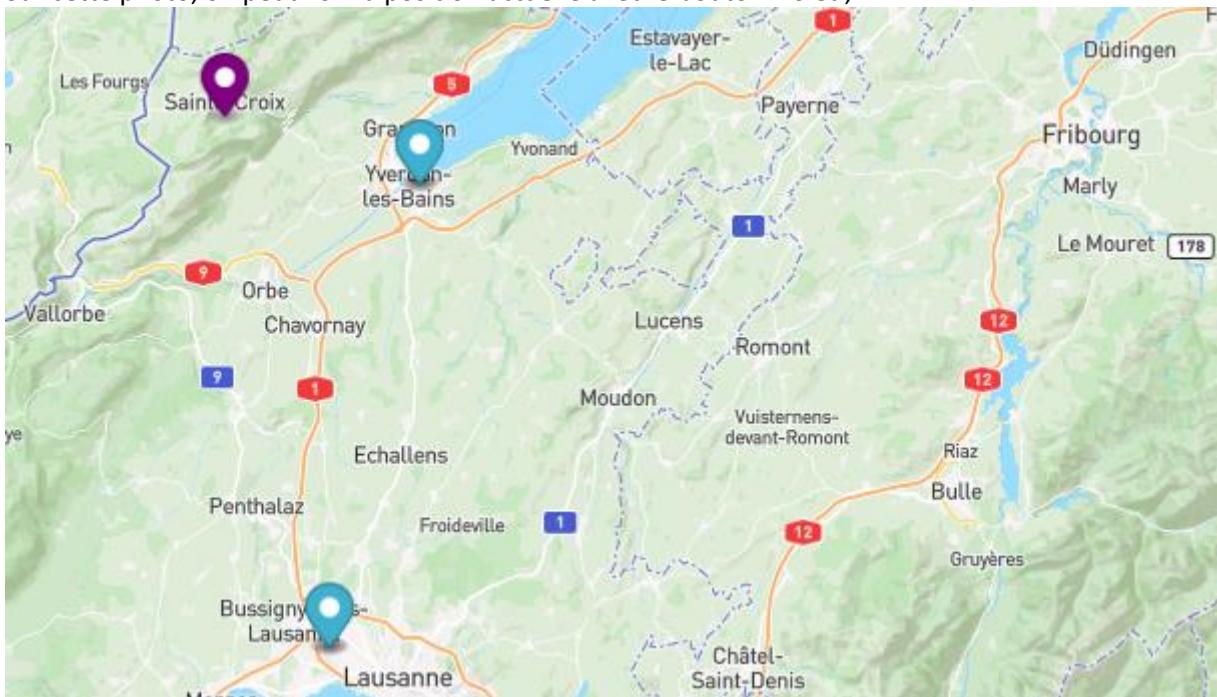


Figure 74 : Affichage de l'emplacement actuel et des publicités sur la page.

L'intégration de ces fonctionnalités cartographiques respecte les principes de Bastien et Scapin en offrant une interaction riche et intuitive, permettant aux utilisateurs de manipuler des données géospatiales de manière efficace. La combinaison de visibilité, de feedback immédiat, et de contrôle utilisateur améliore l'expérience globale et l'accessibilité de l'application.

Dans cette rubrique, j'ai essayé de montrer mes œuvres et leur apparition sur l'application selon le format de Bastien et Scapin. Étant donné que les détails d'utilisation de la carte sont inclus en détail dans les descriptions techniques de la carte, j'ai consacré cette section spécifiquement aux explications sur l'affichage de la position actuelle de l'utilisateur sur la carte.

Les critères de Bastien et Scapin comportent un large éventail d'options, j'en ai donc expliqué quelques-unes ci-dessus.

3.7 Points techniques spécifiques

3.7.1 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 - TPI

The screenshot shows the MySQL Workbench interface. On the left, there's a tree view of the database structure under 'Caves'. It includes tables like 'announcements', 'categories', 'comments', 'messages', 'photos', 'products', and 'users'. The 'photos' table is highlighted. On the right, a table named 'caves.photos' is displayed with 9 rows of data.

#	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

Figure 75 : Enregistrement des photos dans la base de données et dans la structure des fichiers.

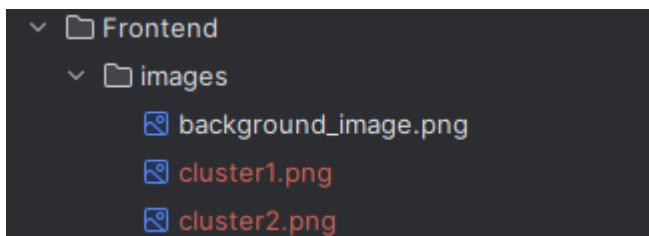


Figure 76:dossier d'images:

- 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.

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

Figure 77 : Modification de php.ini

3.7.2 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.

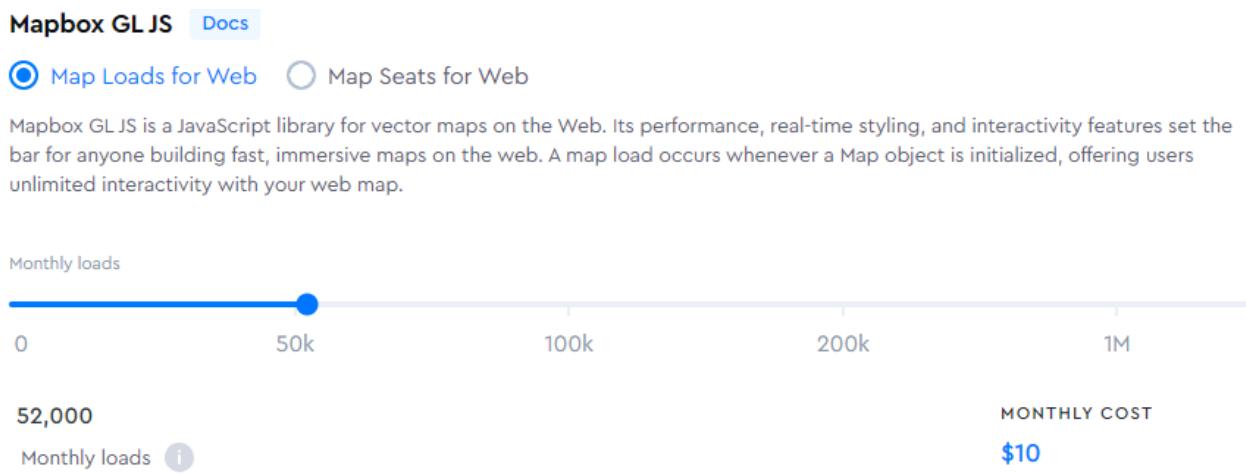


Figure 78 : informations les prix

- 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 GoggleMaps 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.eyJ1IjoieWFzaW5ha3l1eiIsImEi0iJjbHRjcHh4bXowMm40Mmtv0XJtdDRzMml1In0.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
});
```

Figure 79 : Démonstration d'installation de cartes

- 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.



Figure 80: Afficher des annonces sur la carte

```
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: "pommeysa"
      street: "Rue de l'industrie"
      title: "pomme"
    ▶ [[Prototype]]: Object
    ▶ 1: {id: '72', title: 'mais', situation: 'disponible', creation_date: '2024-05-07 09:22:46', street: 'Aarbergergasse', ...}
```

Figure 81 : Afficher les détails de l'annonce via la console

- 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('Backend/accueil.php') Promise<Response>
  .then(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}, ${ad.canton}`);
fetch(`https://api.mapbox.com/geocoding/v5/mapbox.places/${query}.json?access_token=${mapboxgl.accessToken}`)
  .then(response => response.json())
  .then(data => {
    if (data.features.length > 0) {
      const coordinates = data.features[0].center;
    }
  })
}

```

Figure 82: Conversion d'adresse en coordonnées

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: 'Aarbergergasse', building_number: '53', postal_code: '3011', city: 'Bern', canton: 'Bern'} ⓘ
  building_number: "53"
  canton: "Bern"
  city: "Bern"
  postal_code: "3011"
  street: "Aarbergergasse"
  ► [[Prototype]]: Object

Geocoding data:

```

Figure 83 : Obtenir l'adresse de l'utilisateur connecté

Sur cette photo, on peut voir l'adresse actuelle avec le bouton rouge.



```

Geocoding data:
▼ {type: 'FeatureCollection', query: Array(5), features: Array(1), attribution: 'NOTICE: © 2024 Mapbox and its suppliers. All rights reserved.'}
  attribution: "NOTICE: © 2024 Mapbox and its suppliers. All rights reserved. Use of this data is subject to the Mapbox Terms of Service."
  ▼ 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: 'dXJuOm1ieHBsYzpJaXc', 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: "Aarbergergasse 53, 3011 Bern, Switzerland"
      ▶ place_type: ['address']
      ▶ properties: {accuracy: 'rooftop', mapbox_id: 'dXJuOm1ieGFkcjpkYTgzYWIEwMS1iM2MxLTQxNGQtOGNjZC1jMjE2Y2I0NzNmOGI'}

```

Figure 84 : Le processus de conversion de l'adresse en coordonnées

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(elementId: '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);
```

Figure 85 : fonction d'affichage des annonces

```
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.json())
        .then(data => {
            if (data.features.length > 0) {
                const coordinates = data.features[0].center;
```

Figure 86: Ajout d'un marqueur à chaque annonce

```

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);

```

Figure 87: Afficher les marqueurs sur la carte

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.

Sur cette photo on peut voir les opérations sur la console ;

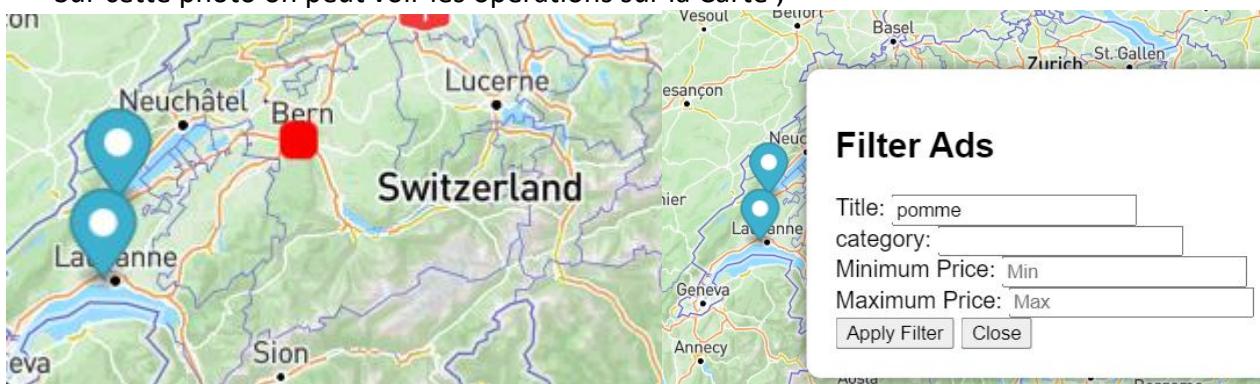
```

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) [{...}, {...}] ac
▶ 0: {id: '76', title: 'pomme', situation: 'pommeysa', creation_date: '2024-05-07 10:46:43', street: "Rue de l'industrie", ...} ac
▶ 1: {id: '77', title: 'pomme', situation: 'good', creation_date: '2024-05-07 11:07:15', street: 'Rue de la Mèbre', ...} ac
  length: 2 ac
▶ [[Prototype]]: Array(0) ac

Ad Data: ac
▶ {id: '76', title: 'pomme', situation: 'pommeysa', creation_date: '2024-05-07 10:46:43', street: "Rue de L'industrie", ...} ac
Ad Data: ▶ {id: '77', title: 'pomme', situation: 'good', creation_date: '2024-05-07 11:07:15', street: 'Rue de La Mèbre', ...} ac
Adding marker for ad: pomme with query: Rue%20de%20l'industrie%208%2C%201400%20Yverdon-les-Bains%2C%20Vaud ac
Adding marker for ad: pomme with query: Rue%20de%20la%20M%2C%208%201400%20Renens%2C%20Vaud ac
Marker added, total markers now: 1 acc
Marker added, total markers now: 2 acc

```

Sur cette photo on peut voir les opérations sur la Carte ;



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.

« La fonction de voir la position actuelle de l'utilisateur sur la carte » créée pour les critères Bastien et Scapin a été examinée dans la section Bastien et Scapin et n'a donc pas été ajoutée à cette section.

3.7.3 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('filter-btn').addEventListener('click', listener: function() : void {
    document.getElementById('filter-popup').style.display = 'block'; // Popup'ı 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('filter-form').addEventListener('submit', listener: async function(e) {
    e.preventDefault();
    // Obtenir des critères de filtrage
    const title = document.getElementById('title').value;
    const categoryName = document.getElementById('name_category').value;
    const minPrice = document.getElementById('min-price').value;
    const maxPrice = document.getElementById('max-price').value;
    await fetchAdsFiltered( params: {
        title: title,
        name_category: categoryName,
        min_price : minPrice,
        max_price : maxPrice
    });
});
```

Figure 88 : critères de filtrage

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 marques
    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('submit', listener, function(e : Event) {
    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.prdct_name as product_name, p.price as product_price,
        p.stock as product_stock, ph.url as photo_url, c.name_category
        FROM announcements a
        JOIN users u ON a.user_id = 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.

Les codes dans ces captures d'écran montrent le processus ;

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

    adsContainer.innerHTML = '';
}

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 ;

Filter Ads

Title:

category: fruit

Minimum Price: Min

Maximum Price: Max

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.

Nous pouvons voir que les résultats changent lorsque les critères de recherche changent ;

```
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: fruit

Minimum Price: Min

Maximum Price: 2



Figure 89: Ajouter le résultat du filtre sous forme de bouton

Résultat et détails des critères de filtre ;

```
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: "Aarbergergasse"
    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: 'Aarbergergasse', ...}
Adding marker for ad: mais2 with query: Aarbergergasse%2053%2C%203011%20Bern%2C%20Bern
Marker added, total markers now: 1
```

3.7.4 Fonctionnalité du Panier - Description Détailée

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);
}
```

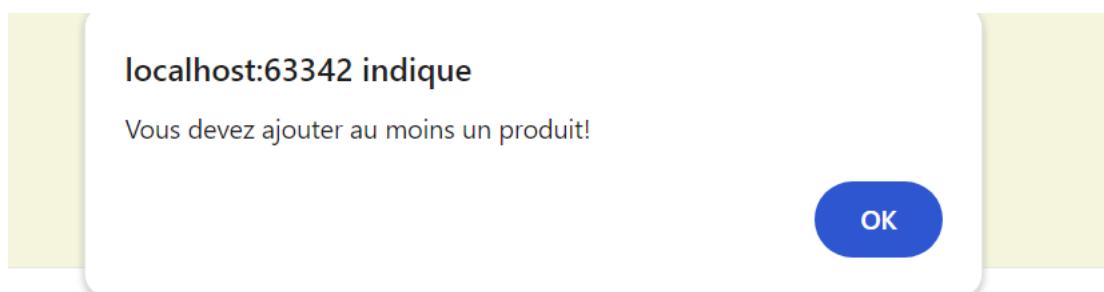


Figure 90 : Attention à aller au panier sans ajouter d'articles au panier

- 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>
                    

```

Figure 91: Affichage de l'annonce ajoutée au panier

- 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);
}
```

L'état du panier en fin de période ;

• pomme - Quantity: 1 -
Price: 3 CHF

VIDER LE PANIER PAGE DE PAIMENT

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...](#)

Figure 92 : Vidage et mise à jour du panier en fin de période

- 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(`Backend/checkStock.php?productId=${productId}&quantity=${quantity}`)
        .then(response : Response  => response.json())
        .then(data => {
            if (data.error) {
                alert(data.error);
            }
        })
}
```



pommeysa

pomme

Price: 3

Stock: 3

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

Creation Date: 07/05/2024

Figure 93: avertissement de stock insuffisant

Mise à jour du stock après confirmation

```
function updateStock(productId, quantity) : void {
    fetch(` /Backend/updateStock.php?productId=${productId}&decreaseAmount=${quantity}`)
        .then(response => response.json())
        .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.");
            }
        })
}
```



pommeysa

pomme

Price: 3

Stock: 3

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

Creation Date: 07/05/2024

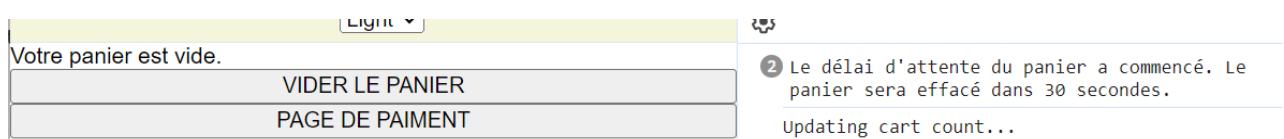
94: si le stock est suffisant

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(`http://Backend/updateStock.php?productId=${productId}&restoreAmount=${quantity}`)
        .then(response => response.json())
        .then(data => console.log(data))
}
```

```
$productId = isset($_GET['productId']) ? $_GET['productId'] : die(json_encode(['error' => 'Product ID required']));
$restoreAmount = isset($_GET['restoreAmount']) ? $_GET['restoreAmount'] : die(json_encode(['error' => 'Restore amount 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 rechargeé 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
				~	85		~	~

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
				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.

3.7.5 Fonction de messagerie dans l'application : Explication et Justification

La fonction de messagerie permet aux utilisateurs de communiquer directement avec les vendeurs pour obtenir des informations supplémentaires sur les annonces ou négocier les termes d'une transaction potentielle. Cette fonctionnalité renforce la transparence et l'interactivité de la plateforme.

Justification de la technologie

L'utilisation de JavaScript côté client (dans `sendMessages.js`) et de PHP côté serveur (`sendMessages.php`) offre une solution robuste et dynamique pour la gestion des messages. JavaScript gère les interactions utilisateur en temps réel sans recharger la page, tandis que PHP s'occupe de la communication avec la base de données pour stocker et récupérer les messages de manière sécurisée.

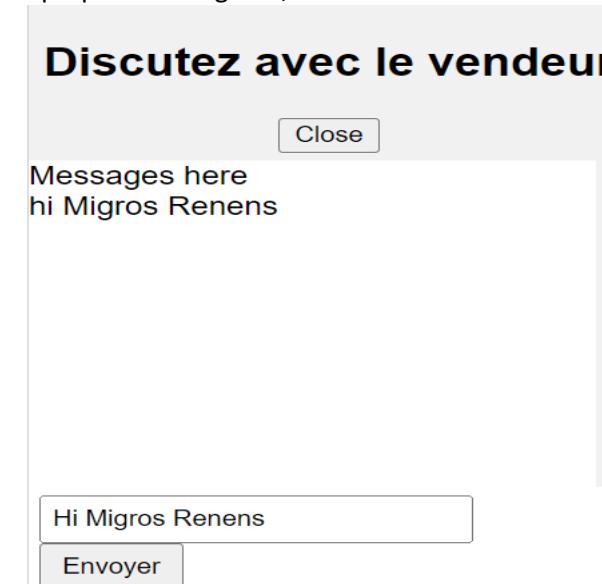
Envoi de messages :

Les utilisateurs peuvent envoyer des messages via une interface intégrée dans les pages de détail des annonces.

- Un contrôle est effectué pour s'assurer que le champ du message n'est pas vide avant l'envoi.
- Le message est envoyé au serveur où il est traité et stocké dans une base de données.
- Exemple de fonction sendMessage dans sendMessages.js :

```
function sendMessage(): void {
    const messageInput : HTMLElement = document.getElementById('message-input');
    if (!messageInput) {
        console.error('Message input element not found');
        return;
    }
    const message : string = messageInput.value;
    if (message.trim() === '') {
        alert('Message cannot be empty');
```

Pop-up de messagerie :



```
// Insérez le message dans le tableau des messages
$insert_query = "INSERT INTO messages (buyer_id, seller_id, announcement_id, message_text)
$stmt = $pdo->prepare($insert_query);
$stmt->bindParam(1, $buyer_id, PDO::PARAM_INT);
$stmt->bindParam(2, $seller_id, PDO::PARAM_INT);
```

Le message que j'ai envoyé a été enregistré dans la base de données et peut désormais être consulté par l'utilisateur.

Captures d'écran de la base de données :

#	id	buyer_id	seller_id	announcement_id	message_text
1	9	23	22	75	hi Migros Renens
2	10	23	22	75	Hi Migros Renens

	id	title	situation	creation_date	update_date	delete_date	users_idusers	products_id
6	74	carrot	disponible	2024-05-07 09:51:38	2024-05-07 09:51:38	(NULL)	17	82
7	75	carrot	good	2024-05-07 10:13:52	2024-05-07 10:13:52	(NULL)	22	83
8	76	pomme	pommeysa	2024-05-07 10:46:43	2024-05-07 10:46:43	(NULL)	17	84
9	77	pomme	good	2024-05-07 11:07:15	2024-05-07 11:07:15	(NULL)	22	85

Figure 95: Obtention de "l'identifiant utilisateur" avec "l'identifiant d'annonce"

On voit que les identifiants du message envoyé ici sont différents.

La personne qui envoie le message est spécifiée comme "buyer_id" et la personne qui reçoit le message est spécifiée comme "seller_id". Tandis que nous obtenons le "buyer_id" de l'utilisateur connecté, nous obtenons le "seller_id" de l'identifiant de l'utilisateur qui a placé l'annonce à partir du tableau des annonces.

Affichage des messages :

- Les messages sont affichés dans une zone de chat dédiée, améliorant l'expérience utilisateur par une interaction continue et instantanée.

Une fois que l'utilisateur a envoyé le message, lorsque l'autre utilisateur se connecte et clique sur le bouton Mes messages, il pourra voir les messages qu'il a reçus. En cliquant sur le bouton de réponse au message dans cette section, un message sera envoyé à l'utilisateur destinataire avec "buyer_id" et ce message sera enregistré dans la base de données.

Sélection de messages ;

```
if ($currentUserId) {
    $query = "SELECT m.message_text, m.sent_time, u.name, u.firstname, a.title, a.id as ad_id
    FROM messages m
    JOIN users u ON u.id = m.buyer_id
    JOIN announcements a ON a.id = m.announcement_id
    WHERE a.users_idusers = ?
    ORDER BY m.sent_time DESC";
```

Section Mes messages ;

Your Messages

carrot - from coop coop

Hi Migros Renens

Sent: 28/05/2024 22:31:42

hi coop Bern

carrot - from coop coop

hi Migros Renens

Sent: 28/05/2024 22:28:32

A ce stade, l'utilisateur peut envoyer le texte qu'il a écrit au destinataire en cliquant sur le bouton «

envoyer la réponse ».

Capture d'écran montrant que le message envoyé a été traité dans la base de données ;

#	id	buyer_id	seller_id	announcement_id	message_text
1	9	23	22	75	hi Migros Renens
2	10	23	22	75	Hi Migros Renens
3	11	22	22	75	hi coop Bern

La mise en place de la fonction de messagerie dans l'application non seulement augmente l'engagement des utilisateurs mais assure également que les interactions entre acheteurs et vendeurs sont gérées de manière efficace et sécurisée.

3.8 Bugs

3.8.1 Problème de Réponse aux Messages dans le Système de Messagerie

Description du problème

Lorsque les utilisateurs répondent à des messages dans le système de messagerie, les réponses sont incorrectement affichées dans leur propre interface de messagerie au lieu de celle du destinataire initial. Ce comportement entraîne une confusion, car les utilisateurs voient leurs propres réponses comme si elles étaient des messages reçus.

#	id	buyer_id	seller_id	announcement_id	message_text
1	9	23	22	75	hi Migros Renens
2	10	23	22	75	Hi Migros Renens
3	11	22	22	75	hi coop Bern

Your Messages

carrot - from migros migros

hi coop Bern

Sent: 28/05/2024 22:48:58 [Reply](#)

carrot - from coop coop

Hi Migros Renens

Sent: 28/05/2024 22:31:42 [Reply](#)

carrot - from coop coop

hi Migros Renens

Sent: 28/05/2024 22:28:32 [Reply](#)

Le message rédigé en réponse est enregistré dans la base de données. L'erreur que nous voyons lors de l'inscription est que l'utilisateur s'envoie des messages en raison du conflit entre "seller_id" et "buyer_id".

- Identification du problème:

Attribution des rôles dans les messages

Le système semble ne pas distinguer correctement entre l'expéditeur et le destinataire lors de l'envoi des réponses. Les messages de réponse utilisent peut-être constamment l'ID de l'expéditeur original comme destinataire pour toutes les interactions suivantes, ce qui fait que les réponses s'affichent dans la boîte de réception de l'expéditeur.

Gestion des identifiants dans la base de données

Comme observé dans les logs et la base de données, l'identifiant de l'acheteur (buyer_id) et celui du vendeur (seller_id) ne sont pas mis à jour ou inversés lors des réponses aux messages. Cela pourrait indiquer un

problème dans la logique de traitement des réponses sur le serveur backend.

Raisons potentielles:

Mauvaise gestion des identifiants lors des réponses

Le script côté serveur (sendMessages.php) peut mal gérer les ID lors de l'insertion de réponses dans la base de données, conservant l'expéditeur comme destinataire pour toutes les transactions subséquentes.

Solutions Possibles :

Révision de la logique côté serveur : Je vais examiner et potentiellement modifier le script « sendMessages.js » et « sendMessages.php » pour garantir que, lors de l'envoi d'une réponse, l'identifiant du destinataire soit correctement déterminé et utilisé. Cela pourrait nécessiter l'implémentation d'une logique conditionnelle qui permettrait d'intervertir les rôles d'expéditeur et de destinataire dans le cas des réponses aux messages. Cette approche est essentielle pour s'assurer que les réponses parviennent bien à la bonne personne et non pas à l'expéditeur initial.

De plus, il peut être envisagé de changer complètement la messagerie et de la rendre plus logique. J'envisage également la solution consistant à établir un système de threads dans lequel la messagerie entre l'acheteur et le vendeur peut continuer à tout moment, comme dans le système Facebook Messenger. J'envisage également d'utiliser node-js et websocket pour rendre le système de messagerie plus fonctionnel.

4 Tests

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.	

Figure 96: Des images de test sont incluses dans cette section.

CAVES - TPI

TEST		
Travaux à réaliser	Réalisé	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 reconexion 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.	

La page Messages est-elle active ?	Après s'être connecté, l'utilisateur peut se rendre sur la page où il peut consulter ses messages en cliquant sur le bouton messages.	
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.	
La position actuelle de l'utilisateur peut-elle être affichée sur la carte ?	Si l'utilisateur ouvre sa position, sa position actuelle est affichée sur la carte après l'actualisation de la page.	
L'utilisateur peut-il visualiser sa propre adresse sur la carte ? Une fois connecté, l'utilisateur peut visualiser sa position sur la carte.	Une fois connecté, l'utilisateur peut visualiser sa position sur la carte.	

CAVES - TPI

TEST		
Travaux à réaliser	Réalisé	Parties manquantes
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.	
Le produit peut-il être ajouté au panier ?	Il est ajouté et apparaît dans le panier.	
Le contrôle des stocks est-il effectué ?	Oui, s'il n'y a pas suffisamment de produits en stock, ils ne pourront pas être ajoutés au panier.	
Le processus de vidage du panier et la réservation chronométrée fonctionnent-ils ?	Lorsqu'un certain temps est attendu et que l'on clique sur le bouton Vider le panier, le panier est vidé.	
La base de données est-elle mise à jour tout au long de toutes les transactions du panier ?	Oui, la base de données est mise à jour à chaque fois que le panier est ajouté et que le panier est vidé.	
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.	
L'utilisateur peut-il envoyer des messages ?	Oui, chaque utilisateur peut envoyer un message au propriétaire de l'annonce à partir de la page de détail de l'annonce après le processus de connexion.	
Les utilisateurs peuvent-ils consulter les messages qu'ils reçoivent sur la page Mes messages ?	Oui, les utilisateurs peuvent consulter les messages qu'ils reçoivent.	
Ces transactions effectuées par les utilisateurs sont-elles enregistrées dans la base de données ?	Les messages reçus et envoyés sont affichés dans la base de données.	
Les réponses données par l'utilisateur peuvent-elles être reçues par l'autre utilisateur ?	non	Il y a une erreur à ce stade, l'expéditeur reçoit lui-même le message.
Le thème et le titre sont-ils accessibles depuis chaque page ?	Oui, cela améliore mon expérience utilisateur.	

5 Conclusions

Tout au long de ce projet, j'ai avancé avec l'objectif clair de réaliser les performances attendues dès le début. Travailler sur la fonctionnalité de la carte m'a particulièrement marqué; la diversité des actions possibles sur celle-ci a enrichi considérablement mon expérience et le projet lui-même.

Sur un autre volet, le point de blocage sur la fonction du panier a été un vrai défi. Cependant, concevoir une solution innovante et observer ses résultats positifs m'a considérablement motivé. Cette expérience a renforcé mon engagement envers le projet et a boosté mon énergie.

Explorer de nouvelles avenues pour les tâches de Bastien et Scapin, créer des thèmes, configurer des éléments divers et intégrer des positions en temps réel sur la carte ont été des aspects très excitants. Même si certains sprints ont été achevés rapidement, j'ai passé plus de temps que prévu sur les systèmes de messages et de panier. Néanmoins, je suis globalement satisfait des tâches que j'ai accomplies, en particulier la fonction de messages où, malgré quelques petits défis, j'ai réussi à répondre à mes attentes.

Travailler de manière indépendante sur un tel projet m'a appris que la planification et l'analyse sont encore plus cruciales qu'il n'y paraît, et que rester fidèle au calendrier est fondamental pour maintenir une discipline temporelle stricte. Les parties les plus difficiles pour moi ont été le suivi des documents en temps réel et l'ajout d'explications en français dans les codes et les documents, ce qui a été particulièrement exigeant en termes de temps.

En conclusion, le soutien non pressurisé du chef de projet et des experts a été déterminant. Il a transformé mes inquiétudes initiales en une satisfaction paisible et heureuse, concluant ainsi ce projet sur une note positive.

5.1 Objectifs atteints

Au terme de ce projet, je suis fier de présenter les réussites suivantes, qui démontrent une amélioration significative et une adaptation précise aux besoins définis initialement dans le cahier des charges :

- Intégration de la Gestion de Compte Agriculteur : J'ai réalisé l'intégration de la gestion de compte pour les agriculteurs, assurant une procédure d'inscription fluide et sécurisée.
- Dépôt Hebdomadaire de Travail : J'ai mis en place un système de dépôt de travail hebdomadaire sur un repository, ce qui a grandement amélioré la collaboration et le suivi continu du projet.
- Maquettes pour l'Interface Utilisateur : J'ai préparé des maquettes pour guider le design des pages web, assurant une expérience utilisateur visuelle cohérente et engageante.
- Développement du Diagramme de Flux : J'ai conçu le diagramme de flux qui a été essentiel pour comprendre et optimiser le parcours utilisateur à travers les fonctionnalités du site.
- Documentation des Cas d'Utilisation : J'ai élaboré le document des cas d'utilisation qui décrit les interactions entre l'utilisateur et le système, en détaillant chaque situation possible.
- Implémentation de la Publication d'Annonces : J'ai achevé la fonctionnalité permettant aux utilisateurs de poster des annonces, un aspect crucial pour la mise en marché de leurs produits.
- Mise à Jour des Informations Utilisateur : J'ai assuré que les utilisateurs puissent actualiser leurs informations de profil, ce qui a renforcé la précision des données et la confiance en la plateforme.

- Modélisation de la Base de Données : La modélisation de la base de données a été réalisée en respectant les normes BCNF, garantissant l'intégrité et la performance des données.
- Fonctionnalités pour Bastien et Scapin : Selon le cahier de charge, des fonctionnalités spécifiques pour Bastien et Scapin ont été définies et complétées, contribuant à une meilleure gestion des processus.
- Fonction de Filtrage : La fonction de filtrage fonctionne de manière indépendante selon différents critères, influençant activement la carte et les annonces, et affichant les résultats pertinents.
- Fonction de Recherche : La fonction de recherche reste accessible en continu pendant l'utilisation de l'application, affectant activement la carte et toutes les annonces sur la page.
- Affichage des Annonces sur la Carte : Les annonces et les utilisateurs sont visibles sur la carte avec des marqueurs distincts et il est possible de passer à la page de détail de l'annonce.
- Visualisation des Annonces : Les annonces peuvent être affichées avec leurs photos et détails, enrichissant l'expérience utilisateur.
- Fonctionnalité du Panier : La fonctionnalité du panier est activement utilisée, affectant la base de données avec des options telles qu'ajouter au panier, retirer du panier, et mettre à jour la base de données.
- Messagerie : Les utilisateurs peuvent accéder au bouton d'envoi de messages, écrire et envoyer des messages; les messages envoyés sont activement enregistrés dans la base de données et peuvent être visualisés dans la section "mes messages" par l'autre utilisateur.
- Conduite des Tests : J'ai mené des tests complets pour valider le bon fonctionnement de chaque fonctionnalité développée.

5.2 Objectifs non-atteints

Malgré les progrès significatifs réalisés, certains aspects du projet n'ont pas encore été accomplis :

Dans la fonction de messagerie, lorsqu'une personne ouvre une annonce et envoie un message, ce dernier est enregistré dans la base de données et envoyé au destinataire. Le destinataire peut alors voir ce message et ses détails dans la section "mes messages", qui sont extraits de la base de données. Si le destinataire souhaite répondre, il peut écrire et envoyer sa réponse, qui est correctement enregistrée dans la base de données.

Cependant, un problème persiste : en raison d'une erreur où les identifiants de l'acheteur et du vendeur sont les mêmes, le message est reçu par l'expéditeur lui-même au lieu du destinataire prévu. Ce problème, qui affecte la fonctionnalité d'échange entre les utilisateurs, reste à résoudre pour assurer une communication efficace et correcte sur la plateforme.

5.3 Suites possibles au pour le projet

Pour continuer à améliorer notre plateforme et assurer la sécurité et l'efficacité de notre service, je mettrai en place une page d'administration dédiée à la validation des annonces pour éviter les fraudes et les utilisations inappropriées. Cette fonctionnalité garantirait que seules les annonces authentiques et fiables soient visibles pour nos utilisateurs.

En outre, je reviserais entièrement notre système de messagerie en développant un système basé sur le threading, ce qui permettrait des conversations plus structurées et suivies entre les utilisateurs. Ce changement améliorerait significativement la communication sur notre site, rendant les échanges entre acheteurs et vendeurs plus clairs et plus efficaces.

En plus de cela, les autres fonctions que je souhaite ajouter à mon projet sont :

- Amélioration des Fonctionnalités Existantes : Je continuerais à peaufiner et optimiser les fonctionnalités actuelles, comme les systèmes de filtrage et de recherche, pour les rendre plus rapides et plus intuitifs. L'intégration d'une intelligence artificielle pour personnaliser les recommandations pourrait également être explorée.
- Extension des Fonctionnalités de Cartographie : J'intégrerais des options de cartographie plus avancées, telles que des vues en 3D ou des intégrations avec des plateformes de réalité augmentée pour enrichir l'expérience utilisateur lors de la visualisation des annonces sur la carte.
- Internationalisation : Je préparerais le projet pour une éventuelle expansion internationale en ajoutant des options multilingues et en adaptant le contenu aux marchés spécifiques.
- Développement Mobile : Je développerais une application mobile dédiée pour étendre la portée du projet et améliorer l'accessibilité pour les utilisateurs sur différentes plateformes. Cette application offrirait des notifications en temps réel et une meilleure géolocalisation.

6 Annexes

6.1 Résume du Rapport de TPI

Départ :

Ce projet a été conçu pour permettre aux utilisateurs de trouver rapidement et efficacement les produits dont ils ont besoin dans leur vie quotidienne, en utilisant des systèmes basés sur la géolocalisation via Internet. L'objectif principal était de faciliter l'accès aux annonces en fonction de la localisation des utilisateurs, tout en assurant une interface conviviale et efficace. Pour gérer le développement de ce projet complexe, la méthode Agile avec l'utilisation du système Kanban a été adoptée, permettant une réactivité et une adaptation constantes aux besoins du projet.

Mise en œuvre :

Le suivi du projet a été structuré en sprints, avec des intervalles de temps spécifiques pour chaque phase. Chaque sprint était dédié à des tâches précises, comprenant le développement, les tests et l'intégration continue des fonctionnalités telles que l'utilisation de cartes pour visualiser les annonces et les emplacements des utilisateurs, la communication directe avec les vendeurs, et la mise à jour en temps réel des stocks pour les réservations via le panier. Des tests rigoureux ont été effectués à chaque étape pour garantir la fonctionnalité et la fiabilité de chaque module. L'utilisation de GitHub a permis des mises à jour quotidiennes et une collaboration fluide entre l'équipe de développement, le chef de projet et les experts consultés régulièrement.

Résultats et Améliorations futures :

Le projet a réussi à créer une application facilitant la vie des utilisateurs grâce à l'implémentation des critères de Bastien et Scapin pour une ergonomie optimisée. Bien qu'une légère lacune ait été notée dans la fonctionnalité de messagerie, les fonctionnalités principales telles que la navigation et filtrage des annonces, les fonctionnalités de panier, et le processus d'inscription ont été pleinement réalisées. Un point notable pour les futures améliorations est l'introduction d'une page de contrôle administratif pour approuver les annonces avant leur publication, assurant ainsi la qualité et la pertinence des informations mises à la disposition des utilisateurs.

6.2 Glossaire

A

API :

Les interfaces par lesquelles l'application interagit avec le monde extérieur, les URL où les requêtes API sont envoyées et reçues.

C

Commit:

L'action d'enregistrer des modifications ou des mises à jour dans un système de contrôle de version, tel que GitHub

D

Diagramme de Flux:

Utilisé pour représenter visuellement le flux de données ou de processus dans le système.

F

Fetch API:

Utilisée pour les échanges de données asynchrones sur les pages web, permettant la communication avec le serveur sans recharger la page.

M

Mapbox :

Une bibliothèque JavaScript utilisée pour les fonctions de cartographie et basées sur la localisation, permettant aux utilisateurs d'interagir avec les données géographiques.

Marker (Marqueur):

Utilisé pour indiquer la position de l'utilisateur ou de l'annonce sur la carte.

S

Session:

Utilisée pour gérer les sessions des utilisateurs, représentant le temps pendant lequel l'utilisateur est connecté jusqu'à sa déconnexion.

Sprint:

Utilisé dans la méthodologie Agile pour planifier et suivre les différentes phases du projet, souvent avec des outils comme IceScrum.

U

User Story :

Sert à définir les fonctionnalités du point de vue de l'utilisateur et guide le processus de développement.

6.3 Sources

- ChatGPT

<https://chatgpt.com/auth/login>

- L' 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.burakalpkara.com/mapbox-ile-projemize-harita-eklemek>

<https://www.softkraft.co/mapbox-vs-google-maps/>

<https://www.mapbox.com/pricing>

<https://stackoverflow.com/questions/35069753/mapbox-gl-js-vs-mapbox-js>

- Panier

http://jmolline.free.fr/tutos/tuto_panier.html

https://mesutd.com/php-mysql-baglanma-mysqli-ekleme-silme-duzenleme-listeleme-dosya-yukleme#google_vignette

<https://grafikart.fr/tutoriels/panier-php-session-309>

- Bastien et Scapin

http://www.guillaumegronier.com/cv/resources/CriteresBastienScapin_v3.pdf

https://umtice.univlemans.fr/pluginfile.php/507354/mod_resource/content/1/Ergonomie%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=VJytpIDynQ4>

<https://www.php.net/manual/fr/indexes.functions.php>

<https://stackoverflow.com/questions/4232017/php-display-a-message-using-a-function-without-using-echo>

<https://stackoverflow.com/questions/72543587/how-to-display-message-in-html-tag-with-php-in-a-function>

<https://edutechwiki.unige.ch/fr/Accueil>

- Pour la traduction

ChatGPT, GoogleTraduction, Deppl

6.4 Journal de Travail

Nom	Akyüz Yasin Salih			29.05.2024
Journal				
Date	Sem	Activité	Heures	Remarques
	0			
30.04.2024	18	43 - Contacts avec le chef de projet et les experts	1:00	A 8 heures, l'accusation est reçue au cahier et le document est examiné. Les questions concernant les descriptions de poste sur le Cahier de Charge ont été répondues par le chef de projet.
30.04.2024	18	43 - Contacts avec le chef de projet et les experts	0:30	L'expert a été rencontré, des explications ont été entendues, des réponses aux questions ont été apportées et le document a été signé.
30.04.2024	18	11 - Analyse	0:30	Le cahier de charge a été réexaminé en détail et les mesures à prendre ont été notées.
30.04.2024	18	12 - Planification	0:15	Une planification a été faite pour les étapes à suivre tout au long du projet, et la création d'un fichier et l'installation d'icescrum et de github ont été déterminées comme premières étapes.
30.04.2024	18	13 - Créez le fichier	0:10	Un fichier a été créé sur votre lecteur C. Le document de projet a été créé.
30.04.2024	18	14 - Icescrum & Github	0:15	Le projet a été mis en place sur Github et Monsieur Benzonana a été ajouté au projet.
30.04.2024	18	14 - Icescrum & Github	0:55	Le projet a été créé sur IceScrum, des sprints ont été créés, Monsieur Benzonana a été ajouté au projet, Planning a été créé et des histoires ont été ajoutées aux sprints.
30.04.2024	18	41 - Journal de travail	0:20	Le journal de travail a été rempli pour la première journée, incluant les explications nécessaires.
01.05.2024	18	14 - Icescrum & Github	1:00	Des histoires ont continué à être créées. L'objectif de sprint a été spécifié pour le Sprint-1.
01.05.2024	18	42 - Rédaction de la documentation	1:00	Le document de projet a été travaillé et des sprints ont été ajoutés à la section planification.
01.05.2024	18	43 - Contacts avec le chef de projet et les experts	0:10	Les dossiers ont été vérifiés, un e-mail est envoyé au chef de projet et aux experts.
02.05.2024	18	23 - Balsamiq & Diagramme	2:00	Des Maquettes ont été préparés, des arrangements et des révisions ont été effectués.
02.05.2024	18	42 - Rédaction de la documentation	0:30	Des maquettes ont été ajoutées au dossier du projet.

CAVES - TPI

02.05.2024	18	22 - BDD-SQL	1:30	La base de données a été examinée et des recherches ont été menées sur d'éventuels ajouts.
02.05.2024	18	24 - Backend	1:00	Des efforts ont été faits pour enregistrer les publicités et les afficher ensuite avec des photographies.
02.05.2024	18	14 - Icescrum & Github	0:20	Les premiers téléchargements github ont été effectués vers le référentiel github créé.
02.05.2024	18	25 - Frontend	0:45	Des efforts ont été faits pour afficher les publicités sur l'écran principal avec des photographies.
03.05.2024	18	24 - Backend	0:45	Le problème de l'affichage de publicités avec photos sur la page d'accueil a été résolu.
03.05.2024	18	43 - Contacts avec le chef de projet et les experts	0:25	L'état actuel du projet a été discuté avec Monsieur Benzonana entre 10h45 et 11h10.
03.05.2024	18	24 - Backend	0:50	Une tentative a été faite pour fonctionnaliser le filtrage et la recherche. N'a pas pu être rendu fonctionnel à ce stade.
03.05.2024	18	26 - Map	1:00	En travaillant sur la carte, les publicités saisies par l'utilisateur ont été disposées pour apparaître sous forme de lieux sur la carte.
03.05.2024	18	26 - Map	1:00	La fonction indiquant l'adresse de l'utilisateur sur la carte avec son entrée a été complétée
03.05.2024	18	14 - Icescrum & Github	0:45	Les statuts des tests et des tâches ont été mis à jour dans Icescrum et le tableau des tâches a été vérifié. Le projet a été mis à jour sur github et un e-mail a été envoyé au chef de projet, qui devait être envoyé vendredi.
06.05.2024	18	12 - Planification	1:30	Dans le tableau de planification, les durées cibles ont été évaluées en fonction des titres et les intervalles de temps cibles ont été déterminés.
06.05.2024	18	42 - Rédaction de la documentation	2:00	La rédaction du rapport de projet continue, avec des ajouts faits aux sections introduction et analyse.
06.05.2024	18	15 - Recherches	1:00	Des recherches ont été effectuées sur la fonctionnalité de messagerie. Dans ce contexte, les tableaux MLD et MCD ont également été examinés.
06.05.2024	18	21 - MCD-MLD	1:00	La table des messages a été créée en détail, et les références entre cette table et les autres tables ont été organisées
06.05.2024	18	24 - Backend	0:30	Des codes ont été préparés pour la fonction de messagerie, et la page PHP a été modifiée.
06.05.2024	18	25 - Frontend	0:20	L'option de messagerie a été essayée d'être ajoutée à la page de détails de l'annonce

CAVES - TPI

07.05.2024	18	15 - Recherches	0:20	Une recherche de ressources a été effectuée pour le filtrage de plusieurs produits
07.05.2024	18	31 - Use-cases	0:20	Les -use-cases- ont été révisés et ajoutés au dossier du projet
07.05.2024	18	22 - BDD-SQL	0:15	Un dysfonctionnement a été constaté lors de la réinstallation de la table des messages dans la base de données. Elle a été restructurée et réinstallée.
07.05.2024	18	24 - Backend	2:00	Les codes nécessaires pour que la fonction de filtrage fonctionne sous plusieurs titres ont été écrits et des tests ont été effectués.
07.05.2024	18	32 - Tests	0:15	À la fin du sprint-1, les tests des processus ciblés ont été effectués sur l'application.
07.05.2024	18	41 - Journal de travail	0:10	Le journal de bord a été mis à jour, et le planning a été révisé.
07.05.2024	18	14 - Icescrum & Github	0:40	Comme c'était la fin du premier sprint, les états de sprint ont été révisés, les tests ont été vérifiés et le tableau des tâches a été mis à jour.
08.05.2024	19	14 - Icescrum & Github	0:45	Le Sprint-1 a été clôturé et le Sprint-2 a commencé. Les critères de test et les tâches ont été détaillés, et l'objectif du sprint a été défini.
08.05.2024	19	26 - Map	0:30	Des recherches ont été effectuées pour l'intégration de la carte et de nouveaux modes ont été examinés pour permettre la visualisation des annonces sur la carte.
08.05.2024	19	15 - Recherches	1:00	La création d'un panier et la réservation d'un produit dans le panier pour une durée déterminée ont été étudiées, et la meilleure et la plus pratique méthode a été recherchée.
08.05.2024	19	25 - Frontend	0:30	Des pages Panier ont été créées pour activer le panier. Le script et les codes HTML ont été écrits, actuellement une erreur est reçue.
08.05.2024	19	11 - Analyse	0:40	Le « diagramme de flux » a été examiné et des notes ont été prises pour d'éventuels ajouts.
08.05.2024	19	42 - Rédaction de la documentation	0:15	Le document a été révisé et un « diagramme de flux » a été ajouté.
08.05.2024	19	14 - Icescrum & Github	0:10	Les fichiers du projet ont été mis à jour, la version actuelle a été téléchargée sur github et un e-mail a été envoyé au chef de projet et aux experts.
13.05.2024	19	25 - Frontend	1:30	Des travaux ont été effectués pour améliorer la visibilité des boutons sur la page ad.html.
13.05.2024	19	15 - Recherches	0:30	Des recherches backend et frontend ont été réalisées sous le titre de Panier.
13.05.2024	19	25 - Frontend	1:00	Un lien a été établi entre l'apparence des boutons et le processus de connexion. Le stockage de la session a été effectué dans le fichier script.js et la connexion et les boutons de la page ad.html ont été activés.

CAVES - TPI

13.05.2024	19	42 - Rédaction de la documentation	1:30	Certaines mises à jour ont été effectuées en travaillant sur le document de projet.
13.05.2024	19	43 - Contacts avec le chef de projet et les experts	0:20	L'évaluation du Sprint-1 a été réalisée avec Monsieur Benzonana et les tests et tâches ont été vérifiés sur l'application.
13.05.2024	19	24 - Backend	1:00	Certains codes php liés au fonctionnement du Panier ont été écrits. Panier n'est pas actif à ce stade.
13.05.2024	19	41 - Journal de travail	0:05	Des études quotidiennes ont été ajoutées au journal.
13.05.2024	19	14 - Icescrum & Github	0:10	Les fichiers du projet ont été mis à jour, la version actuelle a été téléchargée sur github
14.05.2024	19	24 - Backend	2:30	Le fichier Panier.php a été créé pour l'activation de "Panier". Une tentative a été faite pour obtenir des informations d'annonce pour Panier via "l'announcements_id et user_id" entre les fichiers ad.php et panier.php.
14.05.2024	19	15 - Recherches	0:30	Des recherches ont été menées pour trouver des solutions aux problèmes rencontrés lors de l'intégration du panier dans l'application. Les problèmes ne sont pas résolus à ce stade
14.05.2024	19	23 - Balsamiq & Diagramme	0:30	Des ajouts ont été apportés aux modèles et leur apparence a été arrangée.
14.05.2024	19	42 - Rédaction de la documentation	0:20	Des modèles nouvellement préparés ont été ajoutés au dossier du projet.
14.05.2024	19	14 - Icescrum & Github	0:10	Les fichiers du projet ont été mis à jour, la version actuelle a été téléchargée sur github
15.05.2024	19	22 - BDD-SQL	0:30	Il a été vérifié s'il y avait un problème avec la base de données concernant les annonces apparaissant dans le panier et il a été décidé qu'il n'y avait pas de problème.
15.05.2024	19	25 - Frontend	1:00	Les fichiers panier.js et ad.js ont été édités pour le panier.
15.05.2024	19	24 - Backend	1:55	Travaillé avec panier.php et ad.php. Le problème de l'ajout de produits au panier et de leur visualisation a été résolu, mais le problème de quantité persiste. Le processus de réservation dans la base de données a commencé.
15.05.2024	19	41 - Journal de travail	0:10	Des études quotidiennes ont été ajoutées au journal.
15.05.2024	19	22 - BDD-SQL	0:10	Pour l'installation SQL, la base de données a été exportée et la version SQL a été ajoutée aux documents.
15.05.2024	19	14 - Icescrum & Github	0:10	Les fichiers du projet ont été mis à jour, la version actuelle a été téléchargée sur github

CAVES - TPI

16.05.2024	19	22 - BDD-SQL	0:30	La base de données a été vérifiée pour voir si le problème dans les transactions "panier" était causé par la base de données.
16.05.2024	19	25 - Frontend	1:15	Les codes nécessaires ont été écrits pour supprimer le produit ajouté au panier après un certain temps.
16.05.2024	19	32 - Tests	0:20	Des tests utilisateurs ont été effectués
16.05.2024	19	33 - Résultats des tests	0:40	Les résultats des tests ont été enregistrés sous forme écrite.
16.05.2024	19	31 - Use-cases	1:00	Des mises à jour et des ajouts ont été apportés au fichier « user cases ».
16.05.2024	19	43 - Contacts avec le chef de projet et les experts	0:25	Une réunion a eu lieu avec Monsieur Grégory Charmier au sujet du projet.
16.05.2024	19	24 - Backend	1:30	Un travail a été fait sur le fichier panier.php pour résoudre les problèmes, des fusées SQL ont été vérifiées, des erreurs ont été affichées sur la console et des tentatives ont été faites pour les résoudre.
16.05.2024	19	14 - Icescrum & Github	0:10	Les fichiers du projet ont été mis à jour, la version actuelle a été téléchargée sur github
16.05.2024	19			le sprint-2 est terminé.
17.05.2024	20	14 - Icescrum & Github	0:20	sprint-2 a été vérifié, le planning de travail a été planifié.
17.05.2024	20	43 - Contacts avec le chef de projet et les experts	0:30	L'évaluation Sprint-2 a été effectuée et terminée. Sprint-3 a été discuté.
17.05.2024	20	24 - Backend	1:00	Le fichier panier.php a été divisé en parties, les fichiers updateStock, checkStock, restaurerStock ont été créés et le contenu a été modifié.
17.05.2024	20	25 - Frontend	0:30	La section frontend a été réorganisée en fonction des nouveaux fichiers backend.
17.05.2024	20	14 - Icescrum & Github	0:05	Les fichiers du projet ont été mis à jour, la version actuelle a été téléchargée sur github
21.05.2024	20	24 - Backend	1:10	La dernière version de la fonction du panier a été mise à jour et vérifiée. L'impact des opérations effectuées dans le panier sur la base de données a été examiné. Les fonctions pour maintenir les stocks à jour dans la base de données en temps réel ont été complétées
21.05.2024	20	11 - Analyse	0:30	Le document a été révisé, les tâches dans iceScrum ont été vérifiées. décidé de procéder à une mise à jour du diagramme de flux.
21.05.2024	20	23 - Balsamiq & Diagramme	2:00	Des travaux ont été effectués sur le diagramme de flux. Les parties recherche et filtre ont été ajoutées.

CAVES - TPI

21.05.2024	20	42 - Rédaction de la documentation	0:05	Le diagramme de flux a été ajouté au fichier et sa disposition a été vérifiée.
21.05.2024	20	14 - Icescrum & Github	0:05	Les fichiers du projet ont été mis à jour, la version actuelle a été téléchargée sur github
22.05.2024	20	25 - Frontend	1:30	Les codes HTML et js pour la fonction de messagerie ont été réduits. de nouveaux fichiers ont été créés.
22.05.2024	20	24 - Backend	2:10	Les fichiers PHP ont été créés pour la communication avec la base de données. Les messages envoyés ont été enregistrés dans la base de données.
23.05.2024	20	25 - Frontend	0:45	Lors du travail sur l'application de messagerie, il a été observé que des enregistrements étaient enregistrés dans la base de données en utilisant un mauvais identifiant utilisateur. Ensuite, les codes javascript ont été vérifiés.
23.05.2024	20	24 - Backend	2:25	Des vérifications back-end ont été effectuées pour enregistrer la base de données avec le bon ID, et le problème a été résolu. Le message envoyé par l'acheteur a été enregistré dans la base de données avec les bons seller_id, buyer_id et ad_id.
23.05.2024	20	25 - Frontend	1:30	Des codes HTML ont été organisés pour le système de messagerie et les codes de script nécessaires ont été écrits pour être transférés vers la page via le script.
23.05.2024	20	24 - Backend	1:30	Le problème de l'envoi de messages à l'annonceur pendant que les utilisateurs écrivent des messages a été résolu, l'annonceur peut désormais recevoir des messages. Une fois le message reçu, une erreur provoquée par le backend persiste dans la section d'écriture de réponse.
23.05.2024	20	41 - Journal de travail	0:05	Des études quotidiennes ont été ajoutées au journal.
23.05.2024	20	14 - Icescrum & Github	0:05	Les fichiers du projet ont été mis à jour, la version actuelle a été téléchargée sur github
24.05.2024	21	14 - Icescrum & Github	0:35	Le Sprint-3 a été clôturé et le Sprint-4 a commencé. Les critères de test et les tâches ont été détaillés, et l'objectif du sprint a été défini.
24.05.2024	21	43 - Contacts avec le chef de projet et les experts	0:25	L'évaluation Sprint-3 a été effectuée et terminée. Sprint-4 a été discuté.
24.05.2024	21	15 - Recherches	0:30	Lors du démarrage de Sprint-4, l'état final du projet a été vérifié et ce qui pouvait être fait pour la dernière partie restante de l'application de messagerie a été vérifié.
24.05.2024	21	42 - Rédaction de la documentation	2:30	Travail sur le document de projet. Des explications sur la fonctionnalité de la carte ont été ajoutées au document.

CAVES - TPI

24.05.2024	21	41 - Journal de travail	0:10	Des études quotidiennes ont été ajoutées au journal.
24.05.2024	21	14 - Icescrum & Github	0:10	Les fichiers du projet ont été mis à jour, la version actuelle a été téléchargée sur github
27.05.2024	21	42 - Rédaction de la documentation	3:00	Le document de projet a été mis à jour, des descriptions de cartes et de paniers ont été ajoutées.
27.05.2024	21	15 - Recherches	1:30	Les descriptions dans les fichiers frontend et backend ont été traduites en français pour Bastien & Scapin. Google Translate a été utilisé.
27.05.2024	21	25 - Frontend	1:00	Le fichier de style a été modifié. Le fichier d'en-tête a été créé et un travail a été effectué sur bastien et scapin.
27.05.2024	21	13 - Créez le fichier	0:30	La mise en page du contenu du document a été vérifiée
28.05.2024	21	23 - Balsamiq & Diagramme	1:30	Un organigramme des fonctions « Panier », « messages », « Carte » a été préparé et ajouté au dossier du projet.
28.05.2024	21	25 - Frontend	1:00	Travail sur les codes de la mission de Basrien et Scapin, et des titres tels que thème et en-tête ont été ajoutés au projet. De plus, l'affichage de la localisation de l'utilisateur sur la carte a été réalisé sous le titre de Bastien&Scapin.
28.05.2024	21	42 - Rédaction de la documentation	1:30	Le document de projet est presque terminé. des scénarios de tests et de cas d'utilisation ont été rédigés.
29.05.2024	21	25 - Frontend	1:00	Les études de Bastien et Scapin ont été réalisées
29.05.2024	21	31 - Use-cases	0:20	nouvelles fonctions de cas d'utilisation ajoutées au document
29.05.2024	21	32 - Tests	0:30	Des tests ont été effectués sur la page et un journal de bord avec différents mots de passe a été trouvé dans la fiche utilisateur. La source de l'erreur était due à l'ajout accidentel d'une ligne de commentaire au code, et cette erreur a été corrigée.
29.05.2024	21	33 - Résultats des tests	0:15	Les résultats des tests ont été préparés et ajoutés au document
29.05.2024	21	42 - Rédaction de la documentation	1:00	Le document a été vérifié et les traductions nécessaires ont été effectuées. Des explications ont été ajoutées aux photographies jugées nécessaires.