Maxime DACISAC - Alban NIEMAZ
1er année DUT Informatique
IUT des Cézeaux, Clermont Ferrand
Professeur : Marc Chevaldonné

Projet WPF

Singleton

Sommaire

Table des Figures

Contexte

Singleton est une application d'information ludique réservée aux personnes majeures.

Vous êtes à la recherche d'une idée cadeau ou voulez simplement faire bonne impression lors d'un barbecue ? Singleton est faite pour vous !
Vous êtes collectionneur de belles et/ou bonnes bouteilles d'alcool ? Singleton est faite pour vous !
Vous voulez faire un retour sur un alcool que vous avez apprécié ou, au contraire, détesté ? Singleton est faite pour vous !
Vous voulez tester vos connaissances sur la proportion d'alcools avec vos amis ? Singleton est faite pour vous !
Vous voulez découvrir de nouveaux alcools de manière ludique ? Singleton est faite pour vous !

L'application Singleton est une application graphique interactive qui regroupe un large choix d'alcools régulièrement mis à jour.

Grace à Singleton, vous aurez la possibilité de trouver l'alcool qui vous convient grâce à ses nombreuses fonctionnalités telles que le filtre par type, le tri par propriété ainsi qu'une description détaillée de chaque alcool répertorié. Vous pourrez bénéficier des avis éclairés des autres utilisateurs (et donner le vôtre selon vos envies) afin de vous aider à faire le bon choix. Enfin, une fonction 'Surprise' vous permettra aussi de découvrir des alcools de manière ludique.

Personas

Nom : Paul

Age : 19 ans

Profession : Étudiant

Paul est célibataire et est en deuxième année de fac de droit. Tous les jeudis il sort en soirée étudiante et apporte une bouteille. Il compte impressionner ses amis en achetant des bouteilles originales, que personne ne connaît, à toutes les soirées pour se faire passer pour un expert. En utilisant l'application Singleton il pourra facilement trouver son bonheur et passer pour le boss de la soirée. Peut être trouvera t'il une copine en l'impressionnant ?

Nom : Manon

Age : 25 ans

Profession : Infirmière

Manon ne boit pas d'alcool et voudrait faire plaisir à son mari pour son anniversaire en lui offrant un très bon vin qu'il n'a encore jamais goûté. Grace à Singleton et ses fonctionnalités avancées de filtre et de recherche, ses descriptions détaillées et ses avis éclairés, elle va pouvoir trouver l'article qu'elle recherche et faire de son mari un homme heureux.

Nom : Michel

Age : 70 ans

Profession : Retraité

Michel a une passion pour les belles et bonnes bouteilles de whisky. Il souhaiterait collectionner les bouteilles mais sa femme n'est pas d'accord car pour elle il n'y connaît absolument rien et dépense son argent dans le vide. Mais grâce à l'application Singleton qu'il va montrer à sa femme il pourra enfin collectionner les bouteilles de Whisky car sa femme sera convaincue qu'il n'achète pas n'importe quoi. Michel passe aussi beaucoup de temps à donné son avis sur les Whisky qu'il à goûté pour aider les autres utilisateurs débutants.

Sketch

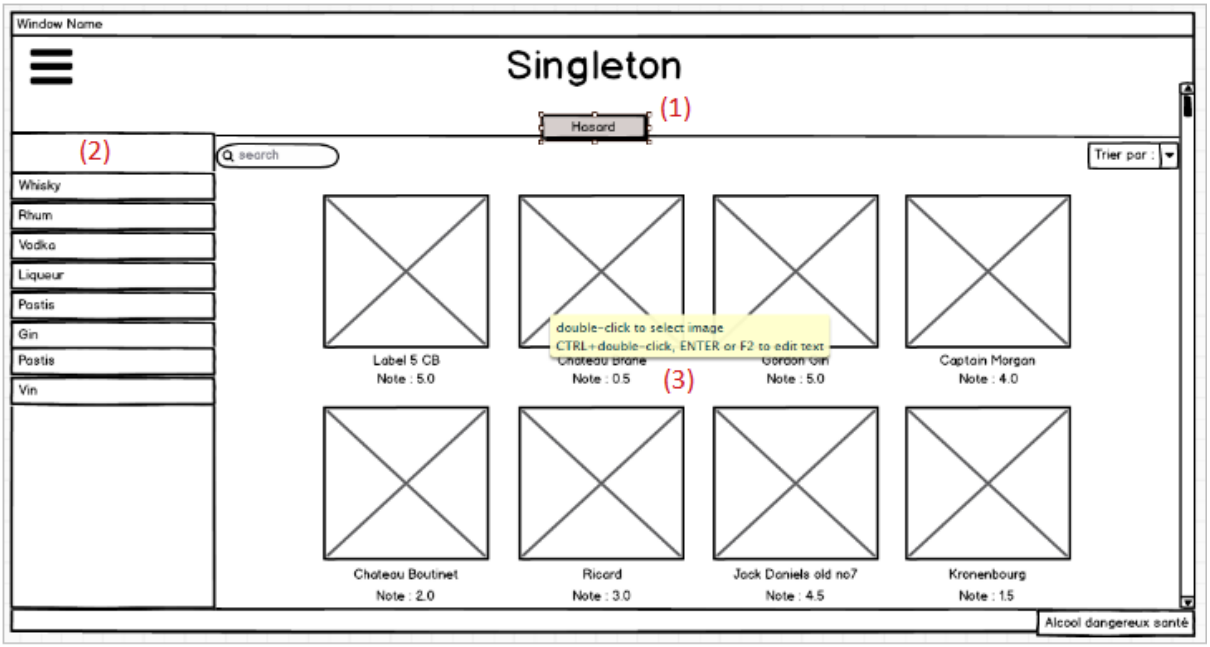
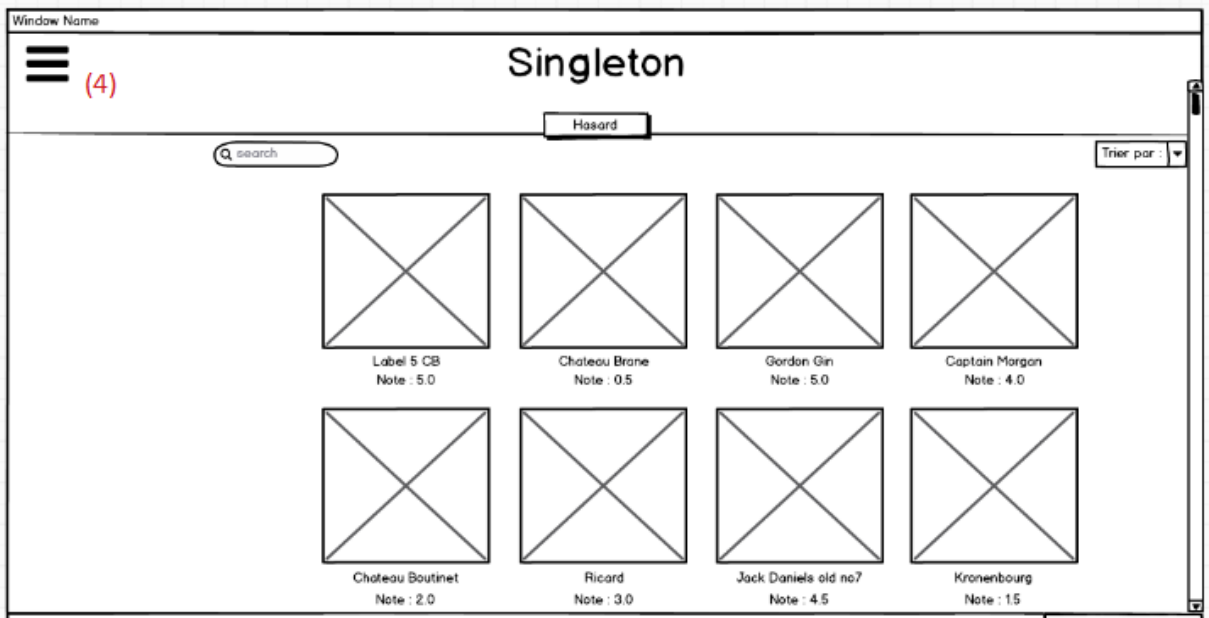


Figure 1: Sketch 1

Page principale à l'ouverture de l'application. En haut, il y a le bandeau qui est toujours présent. Il y a un bouton «Hasard»(1) qui permet d'afficher le descriptif d'un alcool choisi de manière aléatoire. A gauche se trouve le menu(2) avec le type des alcools présent dans l'application. On peut donc filtrer les alcools selon le(s) type(s) sélectionné(s). Au centre se trouve les alcools(3) qui sont représentés par leur image, leur nom, leur prix et leur note. Leur note est la moyenne des notes de leurs avis.



Le bouton[4] en haut à gauche dans le bandeau sert à cacher le menu pour trier par type s'il n'est pas utilisé. On peut le faire réapparaître en appuyant à nouveau sur ce même bouton.

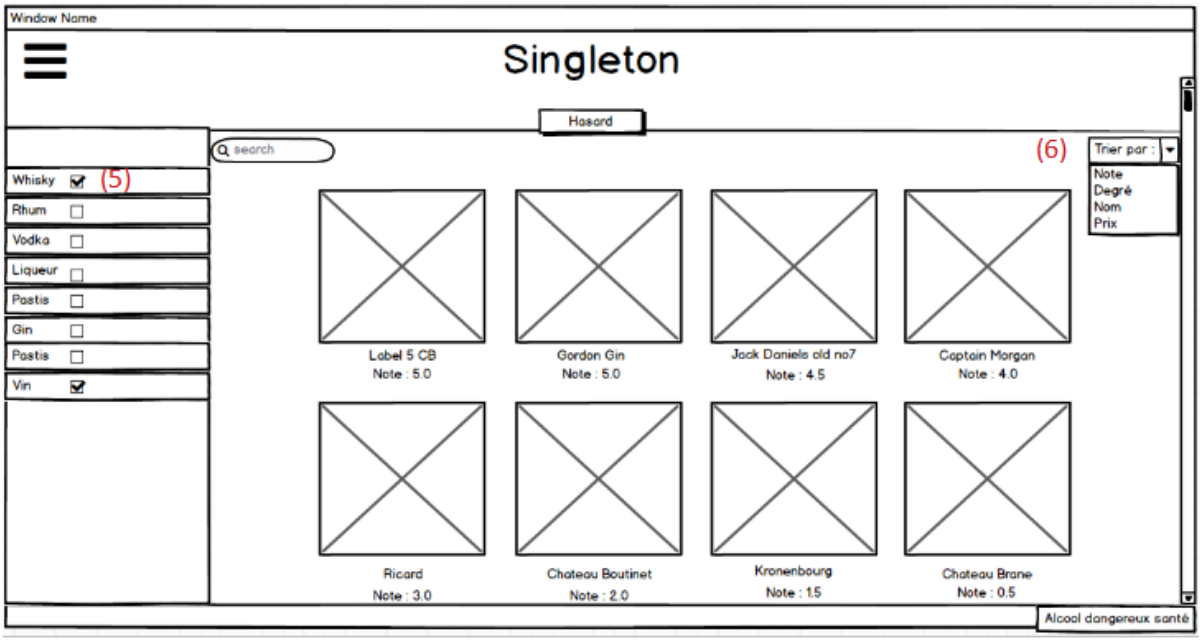


Figure 3: Sketch 3

Le menu de gauche[5] sert à filtrer les alcools affichés en fonction du type sélectionné. On peut sélectionner plusieurs types, par exemple si on cherche du Whisky et du Vin en même temps.

La liste de droite[6] sert à trier les alcools affichés par Note, Degré (d'alcool), Nom ou Prix. Le tri par défaut est 'Nom'. Les tris par 'Nom' et 'Prix' sont croissants alors que les tris par 'Degré' et 'Note' sont décroissants. Le tri se fait aussi en fonction du ou des filtres sélectionnés c'est à dire que le tri ne tri que les alcools affichés et donc filtrés (ou pas).

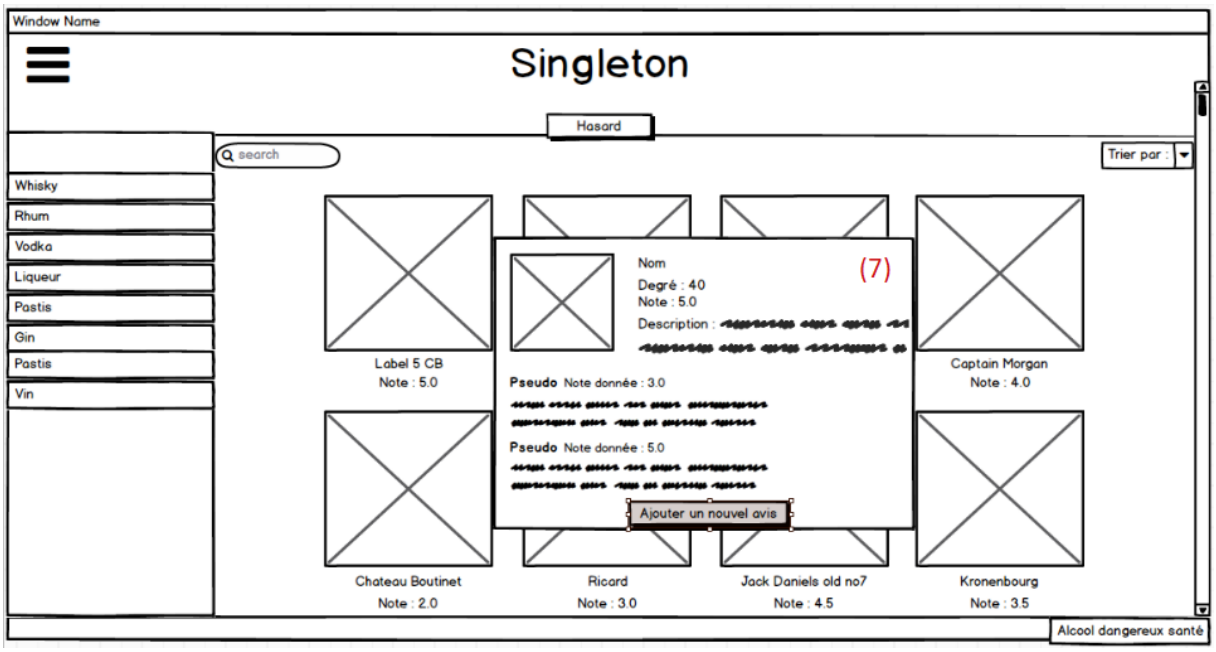


Figure 4: Sketch 4

Quand on clique sur un alcool une fenêtre[(7)] apparaît. Celle-ci contient plus d'information (description et degré d'alcool) ainsi qu'une liste d'avis d'utilisateurs. Un avis se compose du pseudo, de la note attribuée, d'une date et du commentaire de l'utilisateur.

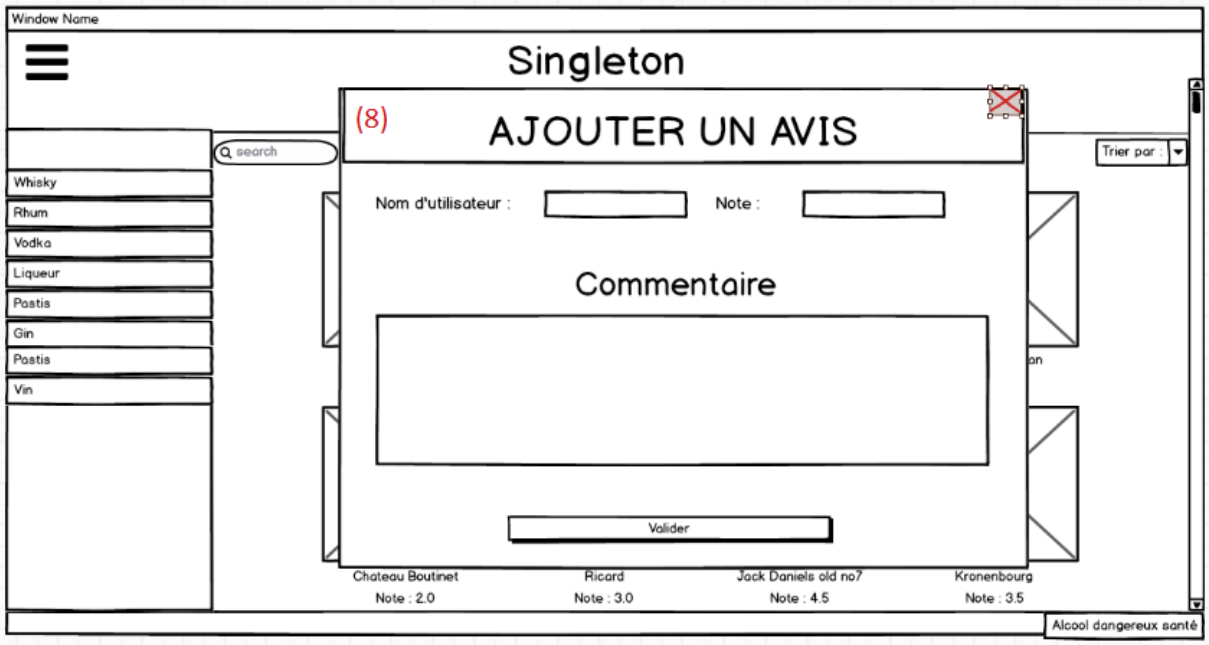


Figure 5: Sketch 5

Il est possible d'ajouter un nouvel avis en appuyant sur «Ajouter un nouvel avis». Une nouvelle fenêtre(8) s'ouvre permettant à l'utilisateur de spécifier son pseudo, sa note et son commentaire, qui sera automatiquement ajouté à la liste des avis existants une fois cliqué sur « Valider ». La date de l'avis est ajoutée de manière transparente et correspond au moment où l'utilisateur valide son avis.

Storyboard

Story 1 : filtrer et trier les alcools

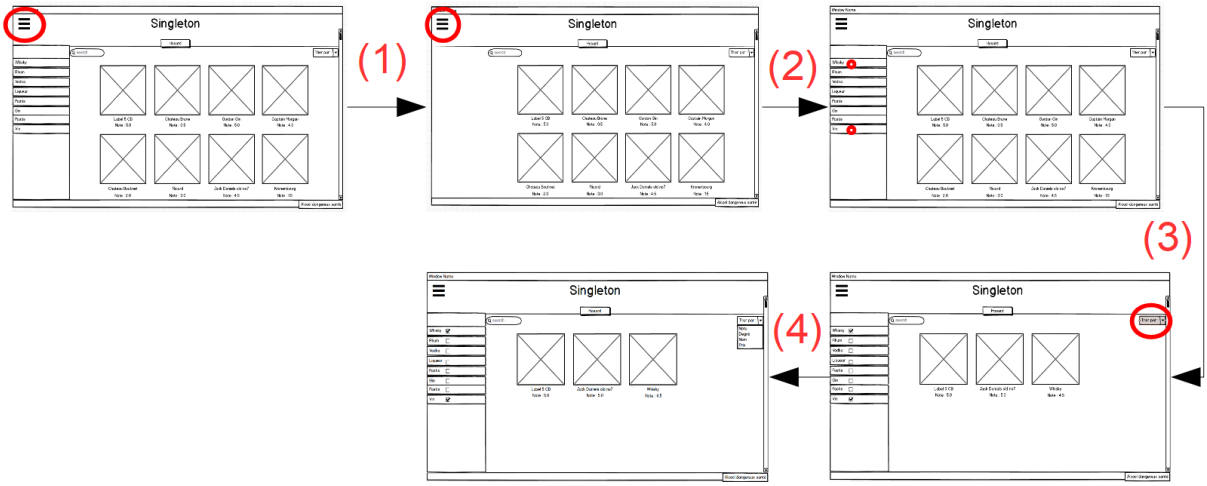


Figure 6: Story 1 "filtrer et trier"

1) Cliquer sur le bouton du menu pour cacher les filtres

2) Cliquer à nouveau pour afficher les filtres

3) Sélectionner le (ou les) filtres désirés pour filtrer les alcools

4) Choisir un type de tri pour trier les alcools affichés

Story 2 : découvrir un alcool et ajouter un avis

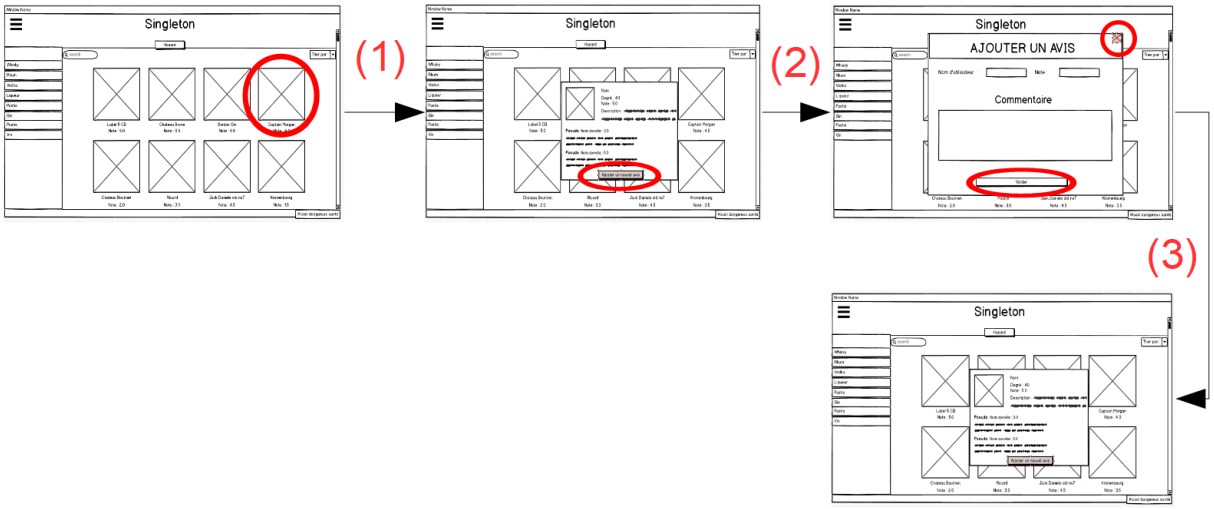


Figure 7: Story 2 "découvrir un alcool et ajouter un avis"

1) Cliquer sur un alcool pour accéder aux informations détaillées

2) Cliquer sur le bouton « Ajouter un avis »

3) Remplir les champs demandés puis valider son avis ou quitter la fenêtre

Story 3 : Bouton Hasard

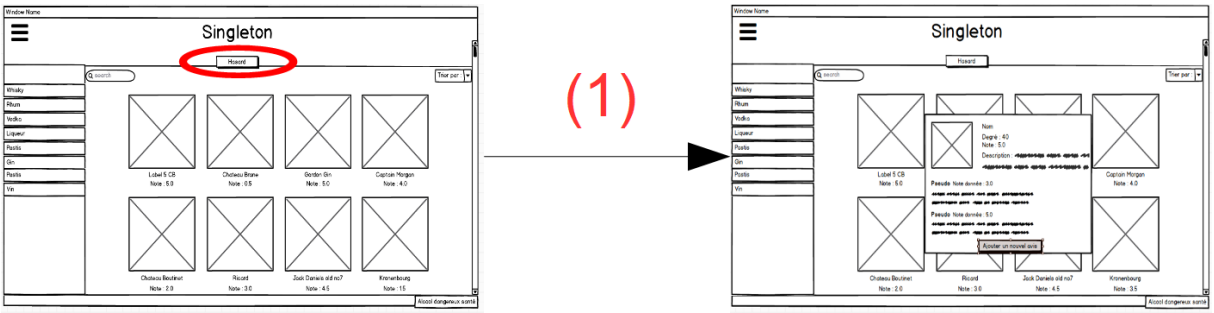


Figure 8: Story 3 « Bouton Hasard »

1) Appuie sur le bouton « Hasard ». Une description d'un alcool choisi de manière aléatoire s'ouvre.

Diagramme d'utilisation

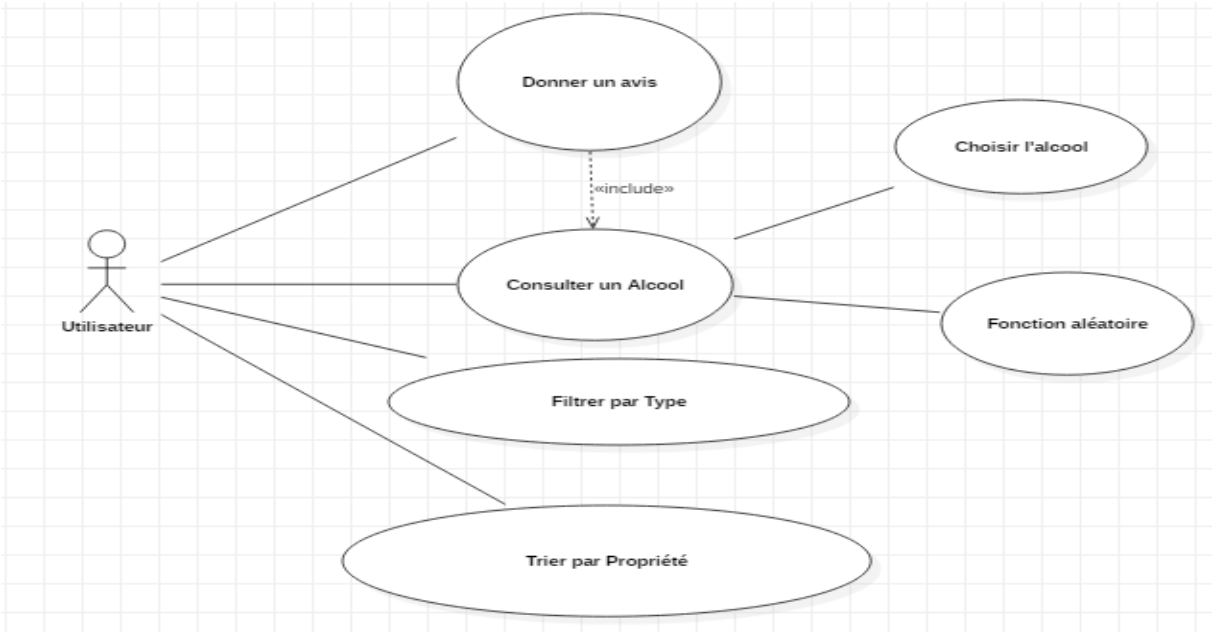


Figure 9: Diagramme d'utilisation

Filtrer par type

- Nom : Filtrer par type
- Objectif : Filtrer les alcools affichés en fonction des types cochés
- Acteur : L'utilisateur
- État en entrée : Page d'accueil avec ou sans filtre ou tri préalable.
- État en sortie : Page d'accueil contenant seulement les alcools dont le(s) type(s) à(ont) été coché(s), tout en gardant le tri en cours.
- Description : L'utilisateur filtre les alcools en choisissant un type. L'observable collection AlcoolsAffiches est mise à jour et donc la vue graphique (RightUC) également.

Trier par propriété

- Nom : Trier par propriété
- Objectif : Trier les alcools affichés en fonction du tri sélectionné correspondant à des propriétés d'un Alcool
- Acteur : L'utilisateur
- État en entrée : Page d'accueil avec ou sans filtre ou tri préalable.
- État en sortie : Page d'accueil trié en fonction du tri sélectionné, agissant sur le filtre en cours si besoin.
- Description : L'utilisateur trie les alcools en choisissant une catégorie. L'observable collection AlcoolsAffiches est mise à jour et donc la vue graphique (RightUC) également.

Consulter un Alcool

- Nom : Consulter un Alcool.
- Objectif : Avoir d'avantage d'information sur un alcool.
- Acteur : L'utilisateur
- État en entrée : Page d'accueil avec ou sans filtre/tri préalable.
- État en sortie : Page de l'alcool contenant toutes les informations disponibles sur l'alcool sélectionné.
- Description : L'utilisateur clique sur un alcool. La page de l'alcool en question (PopUpUC) s'affiche à l'écran.

Donner un avis

- Nom : Donner un avis
- Objectif : Ajouter un avis pour un alcool.
- Acteur : L'utilisateur
- État en entrée : L'utilisateur se trouve sur la page de l'alcool (PopUpUC) sur lequel il a cliqué. Il clique ensuite sur le bouton «Ajouter un avis » qui fait apparaître la fenêtre d'ajout d'avis (AjoutAvisWindow).
- État en sortie : L'utilisateur sors de la fenêtre d'ajout d'avis en cliquant sur « Valider ».
- Description : L'utilisateur ajoute un avis sur l'alcool de son choix. L'observable collection ListAvis est mise à jour ainsi que la note de l'alcool en question si l'utilisateur appuie sur le bouton « valider », et si les conditions sont remplies. Tout autre action entraîne simplement le retour à la page de l'alcool.

Utiliser la fonction « aléatoire »

- Nom : Utiliser la fonction « aléatoire ».
- Objectif : Afficher la page d'un alcool aléatoire.
- Acteur : L'utilisateur.
- État en entrée : Page d'accueil.
- État en sortie : Page de l'alcool contenant toutes les informations disponibles sur l'alcool sélectionné aléatoirement.
- Description : L'utilisateur clique sur le bouton « surprise » et la page d'un alcool aléatoire s'affiche à l'écran.

Rechercher

- Nom : Rechercher.
- Objectif : filtrer les alcools par leur nom en se basant sur le texte entré par l'utilisateur.
- Acteur : L'utilisateur
- État en entrée : Page d'accueil avec ou sans filtre/tri.
- État en sortie : Page d'accueil avec seulement les alcools ayant dans leur nom ce qu'aura tapé l'utilisateur dans la barre de recherche.
- Description : L'utilisateur tape ce qu'il désire dans la barre de recherche. A chaque nouvelle lettre, l'observable collection AlcoolsAffiches est mise à jour et filtrée pour ne contenir que les alcools dont le nom contient ce qu'aura entré l'utilisateur.

Cette fonction est prévue pour être implémentée s'il nous reste du temps, mais n'est pas présente dans le code.

Diagramme de paquetage UML

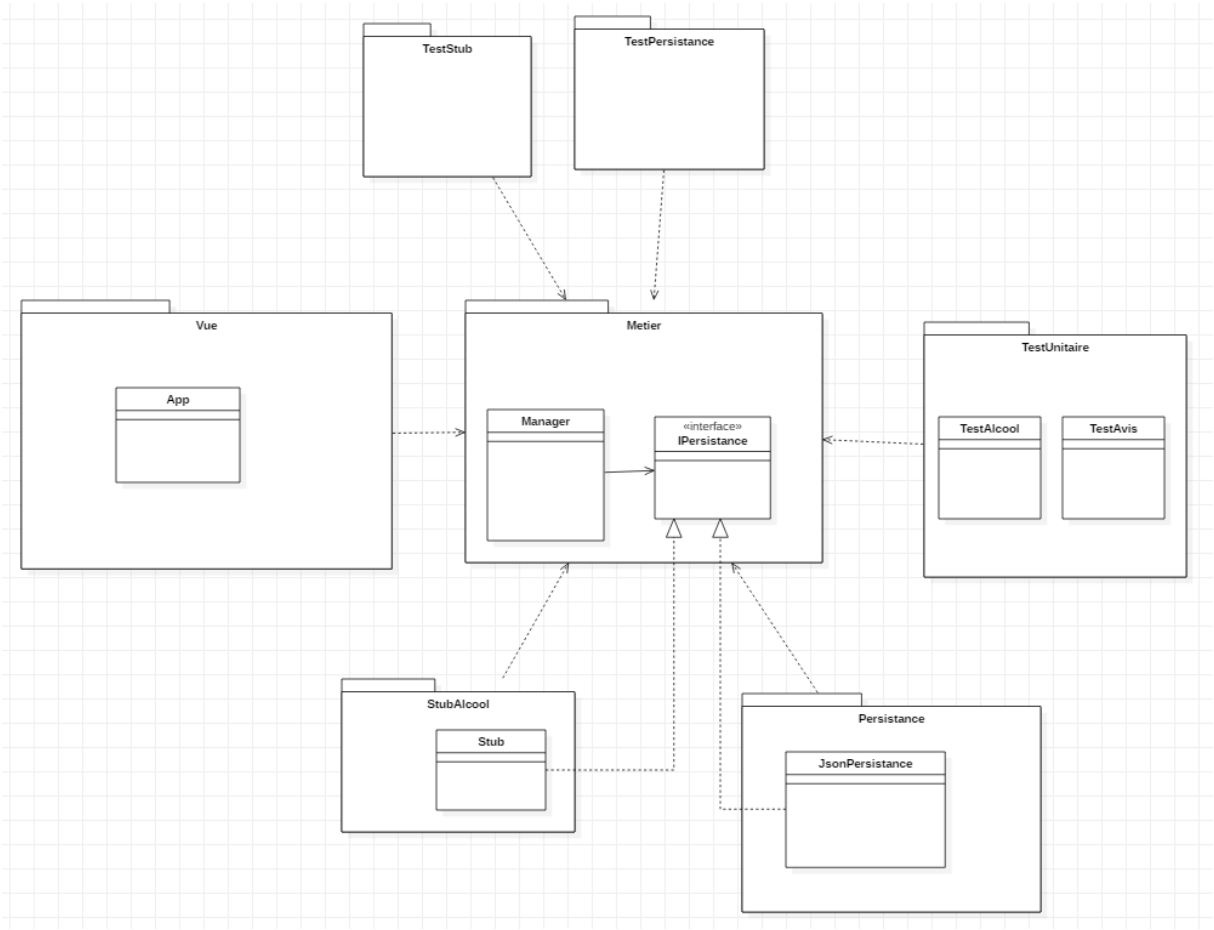


Figure 10: Diagramme de paquetage UML

Le package *Vue* référence *Métier*. La vue et *Métier* sont reliés par l'intermédiaire du Manager. Le manager est instancié une seule fois au chargement de l'application (dans App). L'utilisation d'un manager (en mode Façade) simplifie les appels depuis *Vue* vers *Métier*.

Le manager est construit avec une instance de IPersistance (patron Stratégie) qui dans notre cas charge et sauvegarde du Json. Ce patron permet de facilement remplacer le format de sauvegarde (ex : XML).

Les packages *Persistance* et *StubAlcool* référencent *Métier.* Il est chargé de sérialiser et dé-sérialiser les alcools. *StubAlcool* a été utilisé au début le chargement et la sauvegarde n'était pas encore crée.

Tests référence Métier, StubAlcool et Persistance pour pouvoir implémenter les différents tests.

Le package *Vue* s'occupe du visuel et le 'code behind' de la vue appel les fonctions du Manager.

Diagramme de classe UML

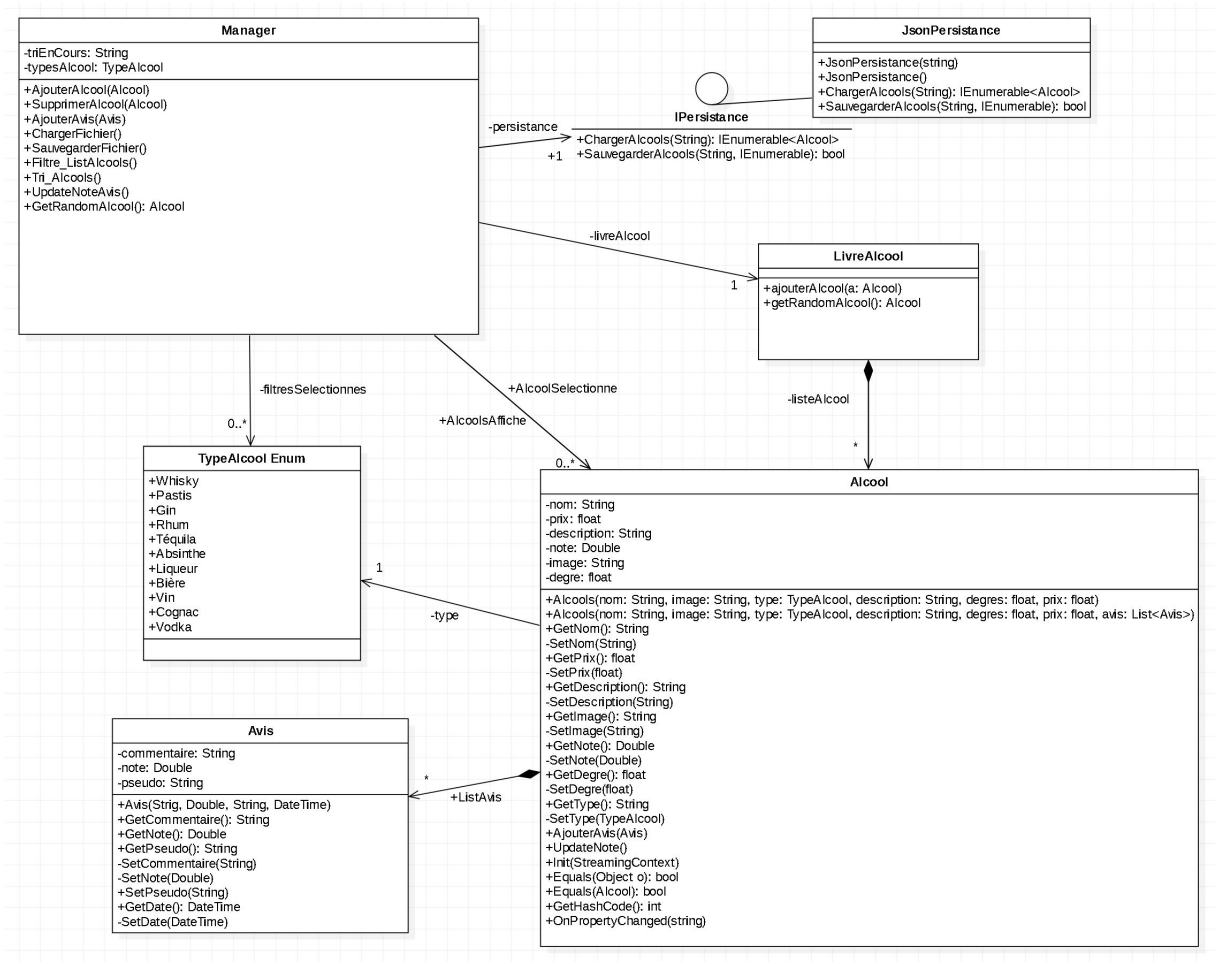


Figure 11: Diagramme de classe UML

Manager : Le Manager simplifie l'utilisation des classes métiers auxquelles en proposant des méthodes pour les accéder et leur déléguer (presque) toutes les tâches. Chaque classe a une responsabilité. Il gère par exemple l'ajout d'alcools en s'appuyant sur LivreAlcool. Idem pour la persistance qui est effectué par l'implémentation de IPersistance.

Alcool : Un Alcool est défini par un nom, un prix, une description, une note, un degré d'alcool, une image et un type enum d'alcool. Tous les alcools possèdent aussi une liste d'avis. A partir de la classe Alcool, on peut modifier un Alcool, mettre à jour sa note en fonction de la note de ses avis (UpdateNote) ou encore ajouter un avis.

LivreAlcool : LivreAlcool est la liste ou se trouve tous les Alcools. Elle possède les méthodes d'ajout et de suppression des Alcools.

IPersistance : c'est une interface simple pour charger et sauvegarder des fichiers. Elle est implémentée par JsonPersistance qui gère le format Json.

Diagramme de séquence

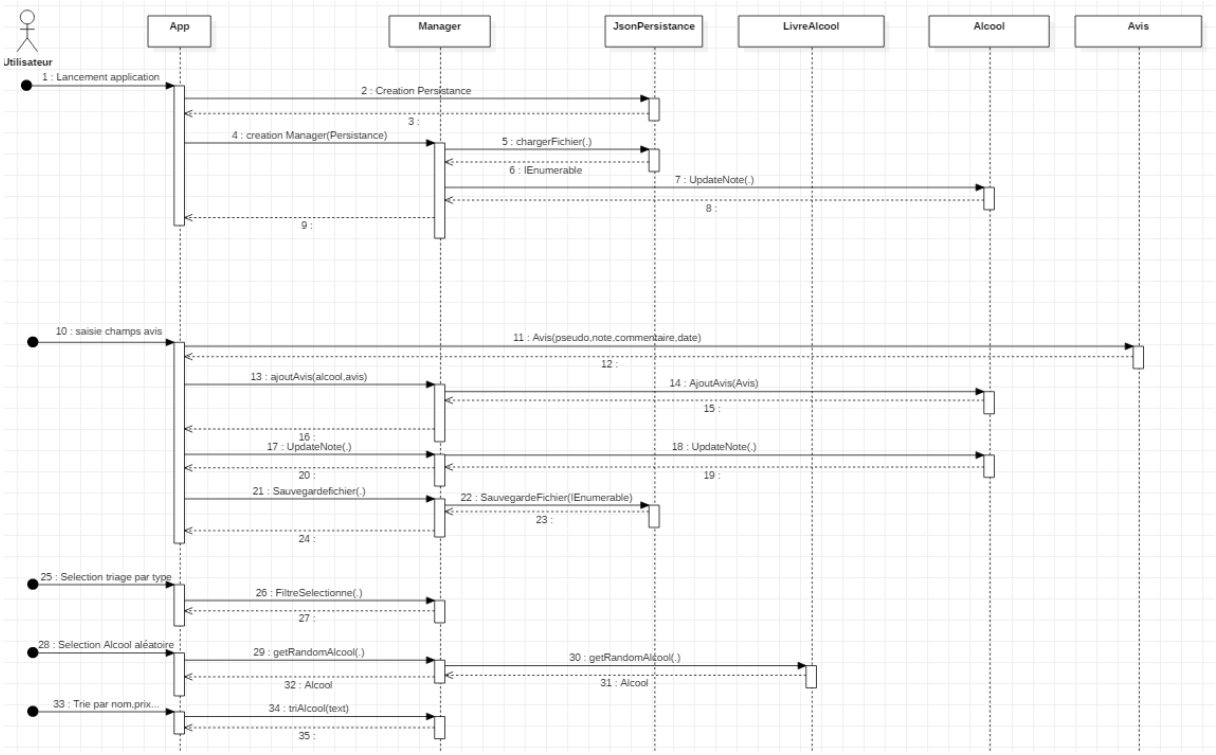


Figure 12: Diagramme de séquence

1- : Lors du lancement de l'application, App est appelé. Lorsqu'il est appelé il crée le Manager(4) et lui donnant comme paramètre la Persistance à utiliser. Donc la Persistance est aussi créée(2). Ensuite le programme entre dans le constructeur de Manager et charge le fichier Json(5) qui contient les alcools. Les notes de tous les alcools sont mises à jour(7) (moyenne des notes des avis) puis le programme sort du constructeur de Manager.

10- Dans la fenêtre d'ajout d'avis l'utilisateur saisi les champs demandés (pseudo, note et commentaire). Ensuite quand il clique sur valider il y a la création d'un nouvel avis(11). Ensuite cet avis est ajouté à la liste des avis de l'alcool sélectionné(14). Ensuite les notes des alcools sont remises à jour(18). Enfin la sauvegarde est appelée. La sauvegarde(22) à lieu à chaque ajout d'un avis.

Pour filtrer les alcools par type, *FiltreSelectionne*(26) est appelée puis le trie s'effectue.

Pour la sélection aléatoire d'un alcool, l'utilisateur clique sur le bouton « Surprise » et la fiche technique de l'alcool apparaît. L'alcool sélectionné prend donc la valeur que retourne GetRandomAlcool(29) qui est appelée dans le Manager puis dans LivreAlcool, qui contient la liste des alcools. Un alcool au hasard est retourné (un alcool qui se trouve dans la liste des alcools disponible). Ensuite cet alcool est affiché.

Architecture

L'application Singleton est une application WPF (Windows Presentation Foundation) qui utilise la méthodologie Modèle / Vue / Vue-Modèle, une variation du patron de conception MVC (Modele Vue Controleur). Cette approche permet entre autres une séparation claire entre le code métier et sa représentation graphique.

L'architecture de Singleton s'inscrit dans cette approche avec un Manager omniprésent, et met en avant plusieurs patrons.

Le patron de conception utilisé est le patron *Façade* à travers donc la classe *Manager*. Il a pour but de masquer la complexité des objets métiers et de proposer des méthodes simples pour rendre le code plus lisible. Il permet aussi de réduire les dépendances entre les différents packages puisque tout passe par lui. Dans le métier, le *Manager* à accès à toutes les classes et délègue le travail aux classes métiers suivant leur responsabilité.

Un autre patron utilisé est celui de *Stratégie* qui consiste à éviter une dépendance directe entre deux classes. C'est le cas entre Manager et *JsonPersistance* grâce à l'interface *IPersistance* (*Métier*). Cela a l'avantage de pouvoir changer facilement l'implémentation de la persistance sans toucher au Manager, par exemple si dans le futur on voulait stocker nos données en base on aurait juste à créer la classe *BddPersistance* qui implémente *IPersistance* puis à passer *BddPersistance* comme paramètre lors de la construction du Manager. Le code est donc plus réutilisable. Pour le moment, *JsonPersistance* se charge de charger et sauvegarder les données (alcools) depuis et vers un fichier Json. Il utilise pour cela le package *System.Runtime.Serialization.*

Singleton utilise aussi l'*Injection de dépendance*, un patron qui se rapproche du patron *Stratégie*. Le Manager est instancié une seule fois coté *Vues* (*App) qui* référencent le package Métier. Le DataContext de la vue principale *MainWindow* est le Manager lui-même ce qui permet un Binding efficace et complet sur toutes les fonctionnalités de l'application. L'injection de dépendance favorise aussi la ré-utilisabilité.

L'affichage des filtres par type d'alcool est géré par le user control *LeftUC* dont la *DataContext* est le *Manager*. Le *binding* des types à afficher est lié à une collection read-only du *Manager*. La gestion des filtres appliqués est effectué par les méthodes du *Manager AjouterFiltreSelectionne*, qui gère tous les types sélectionnés et *Filtre_ListAlcools* qui met à jour l'observable collection des alcools à afficher. Même fonctionnement pour le tri qui est lui aussi appelé après filtrage pour garder le tri en cours même après l'application d'un nouveau filtre.

L'affichage des alcools est géré par le user control *RightUC* via une *ObservableCollection* du *Manager*. Le Manager propose des méthodes pour modifier cette collection, ce qui a pour effet de mettre à jour automatiquement la vue.

L'affichage de la description d'un alcool est gérée par le user control *PopupUC* dont le DataContext est attaché à l'alcool sélectionné dans la vue RightUC (Binding *SelectedValue*), encore une fois via le *Manager* et la propriété *AlcoolSelectionne*.

L'affichage de l'ajout d'un avis se fait dans la fenêtre *AjoutAvisWindows*. Celle-ci se charge de récupérer les champs entrés par l'utilisateur, de créer un Avis à la volée, de mettre à jour la note moyenne de l'alcool en question et de lancer la sauvegarde. Le tout dans le *'code behind'* et encore une fois par l'intermédiaire du *Manager*. Si les champs de l'avis sont erronés, par exemple si la note dépasse 5.0, la propriété Note lève une exception, une popup avec le message d'erreur est affichée à l'écran.