26/04/2021

Gestion de cave à vin

Application en C# et .Net

Paola Costa
Sous la supervision de :
C.Egger, chef de projet
G.Gruaz, expert 1
O.Rutz, expert 2

Table des matières

Table des illustrations

1 Analyse préliminaire

1.1 Introduction

Lors d'un repas de famille ou avec des amis, il est toujours plaisant d'avoir une bonne bouteille de vin sortie. Cependant, la nécessité de fouiller pendant plusieurs minutes la cave à vin pour trouver une bouteille adéquate n'est jamais agréable. Cette application a donc pour but de vous laisser choisir la bonne bouteille directement depuis votre ordinateur. Afin de valider mon CFC et dans le cadre de mon TPI, je vais réaliser une application de gestion de cave à vin. Il s'agit d'une application prévue pour un privé. Elle permettra à une personne, même novice en informatique, de gérer des casiers à bouteilles, d'y ajouter ou enlever des bouteilles, d'effectuer une recherche selon des critères particuliers, de consulter l'historique des actions effectuées, ainsi que d'associer une alerte à une bouteille particulière. L'intégralité des données propres à l'application seront stockées dans une base de données.
Ce projet a comme date de début le lundi 03 mai 2021, 08h50. Sa date de rendu finale est le mercredi 02 juin 2021, 10h35. Cela donne un total de 90h pour le réaliser. L'application est réalisée en C#, à l'aide de base de données MySQL.
Enfin, le travail de pré-TPI a été réalisé en amont. Son but a été de revoir les différentes technologies qui seront abordées dans ce projet. Le canevas du rapport et du journal de travail a été récupéré depuis ce précédent travail et adapté. L'intégralité du développement se fera lors du module et du temps mis à disposition.

1.2 Objectifs

Afin de mener à bien ce projet, de nombreux objectifs sont à compléter. La validation de ceux-ci permettra de déterminer le degré de complétion du projet. L'élément principal de ce projet consiste à créer une application « clé en main ». Cela signifie que l'application sera fonctionnelle sans investissements ultérieurs. Ensuite, l'application doit être accessible à des personnes ayant très peu de notions d'informatique. Son fonctionnement sera donc intuitif.

De plus, l'application contiendra plusieurs fonctionnalités. Il s'agit de celles qui sont citées ci-dessous :

- L'application doit permettre d'ajouter des bouteilles à la cave.
- L'application doit permettre de retirer des bouteilles de la cave.
- Il est possible d'effectuer une recherche selon des critères spécifiques.
- Les données de l'application sont stockées dans une base de données.
- L'application doit permettre d'exporter un PDF, contenant une liste de bouteilles.
- L'application doit permettre d'imprimer une liste de bouteilles spécifiques.
- L'application doit permettre d'afficher l'historique des actions effectuées dans l'application.
- L'application doit permettre d'organiser la case en casiers à bouteilles.
- L'application doit permettre d'ajouter des casiers à bouteilles.
- L'application doit permettre de supprimer des casiers à bouteilles.
- L'application doit permettre d'ajouter une alerte spécifique à une/des bouteille(s).

Enfin, afin de faciliter la mise en place de l'application, une procédure d'installation sera également fournie.

1.3 Planification initiale

Comme le CdC fourni début mai est complet et contient l'intégralité des informations nécessaires pour pouvoir mener à terme le projet, celui-ci se déroulera en mode cascade. La répartition des tâches est effectuée dès le début. Comme discuté avec monsieur C.Egger, les différentes planifications seront réalisées sur Excel. Un onglet « avancées » permet de suivre l'état des différentes tâches, afin de savoir en permanence où en est la réalisation du projet.

Sur la figure ci-dessous, on peut voir la répartition des tâches liées à l'analyse et à la conception. L'analyse préliminaire est réalisée entièrement le lundi 03 mai 2021, afin de permettre l'envoi de la planification initiale ce même jour. L'intégralité de ces tâches mènent jusqu'à la planification définitive, qui sera envoyée, comme convenu avec monsieur G.Gruaz, au plus tard jeudi 06 mai 2021. Chaque colonne représente un bloc d'une période scolaire, soit 45 minutes.

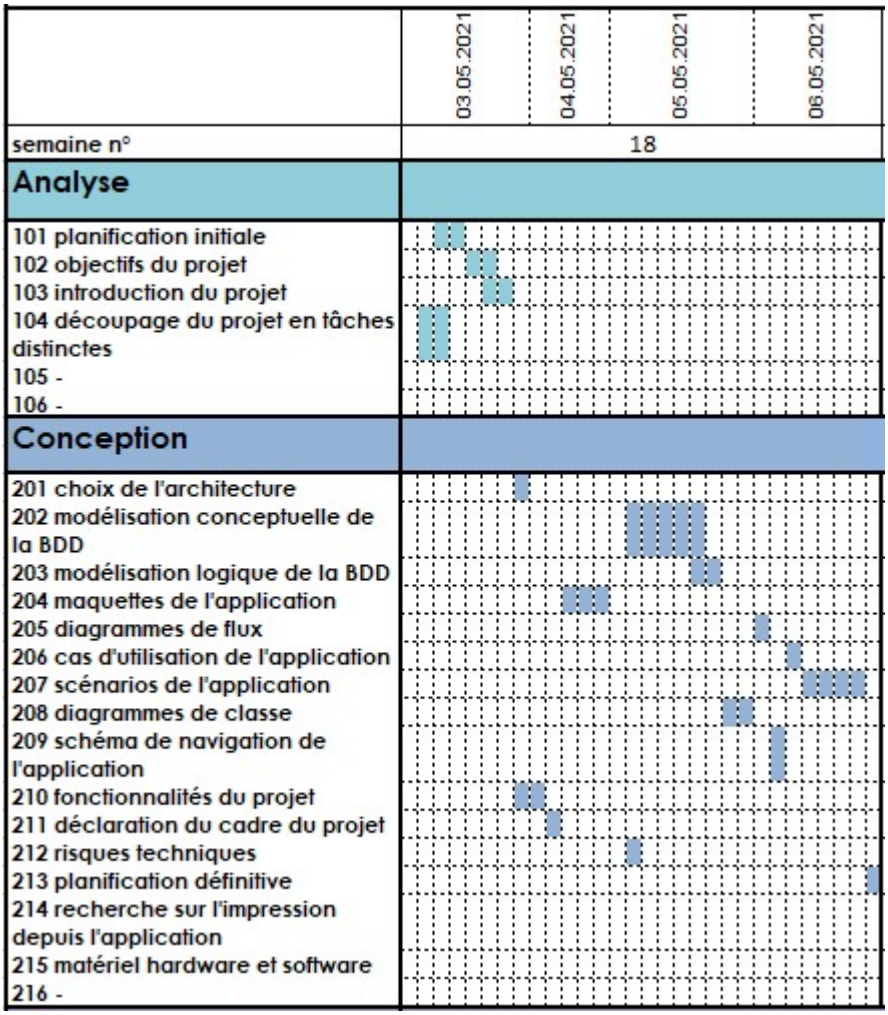


Figure 1 : planification initiale partie 1

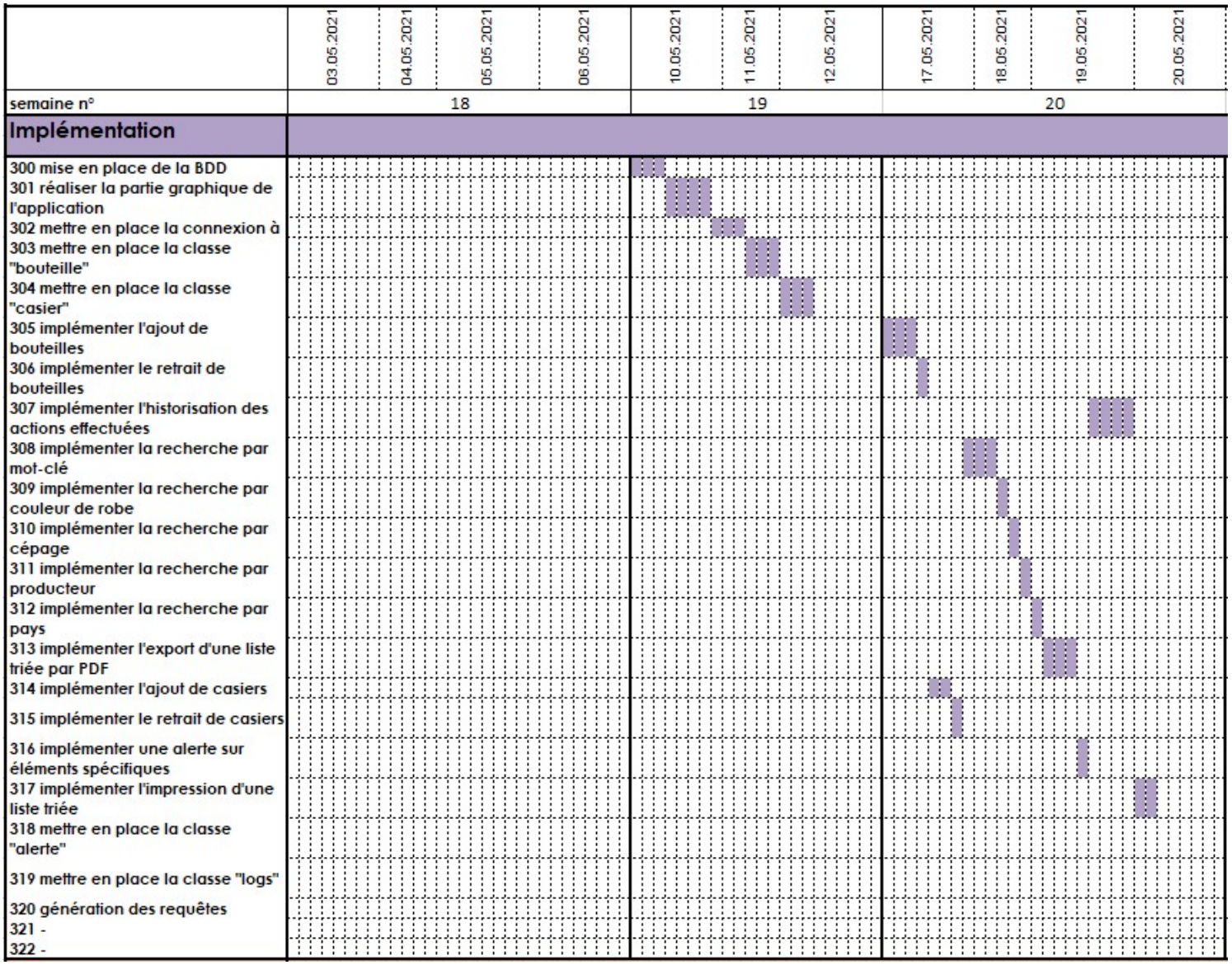


Figure 2 : planification initiale partie 2

Sur la figure ci-contre, il est possible de voir la répartition des tâches concernant l'implémentation.

L'entièreté du développement devrait pouvoir se réaliser sur deux semaines, afin de laisser suffisamment de temps à la fin pour pouvoir tester. Cela permet également de déborder un peu en cas de souci de développement.

Prévoir l'implémentation avec autant de délai à la fin donne ainsi un petit peu de marge.

Le reste du temps sera consacré à la rédaction du rapport de projet ainsi qu'à la procédure liée à l'installation.

Comme mentionné plus haut, les tests seront réalisés principalement à la fin du développement. Une partie sera malgré tout faite pendant la réalisation, afin de pouvoir s'assurer du bon fonctionnement du programme. Cependant, les tests des limites de l'application seront réalisés après que l'application soit fonctionnelle.

Les lignes avec un fond jaune représentent des tâches récurrentes, qui n'ont pas de durée définie. Il s'agit de tâches qui sont réalisées en continu, selon les besoins. C'est le cas pour le remplissage du journal de travail, de la mise en forme du rapport de projet et des différents tests.

De plus, les lignes concernant les réunions avec le CdP et les experts ne sont pas remplies, car les créneaux associés ne sont pas définis. Il s'agit donc de temps utilisé qui ne peut pas être prévu lors de la planification initiale.

Enfin, une comparaison des variations entre la planification initiale, la planification définitive (après l'analyse/conception) et la planification finale (après la réalisation) sera effectuée à la fin du projet. Le but de cette analyse sera d'étudier les différences et d'en déterminer leurs raisons.

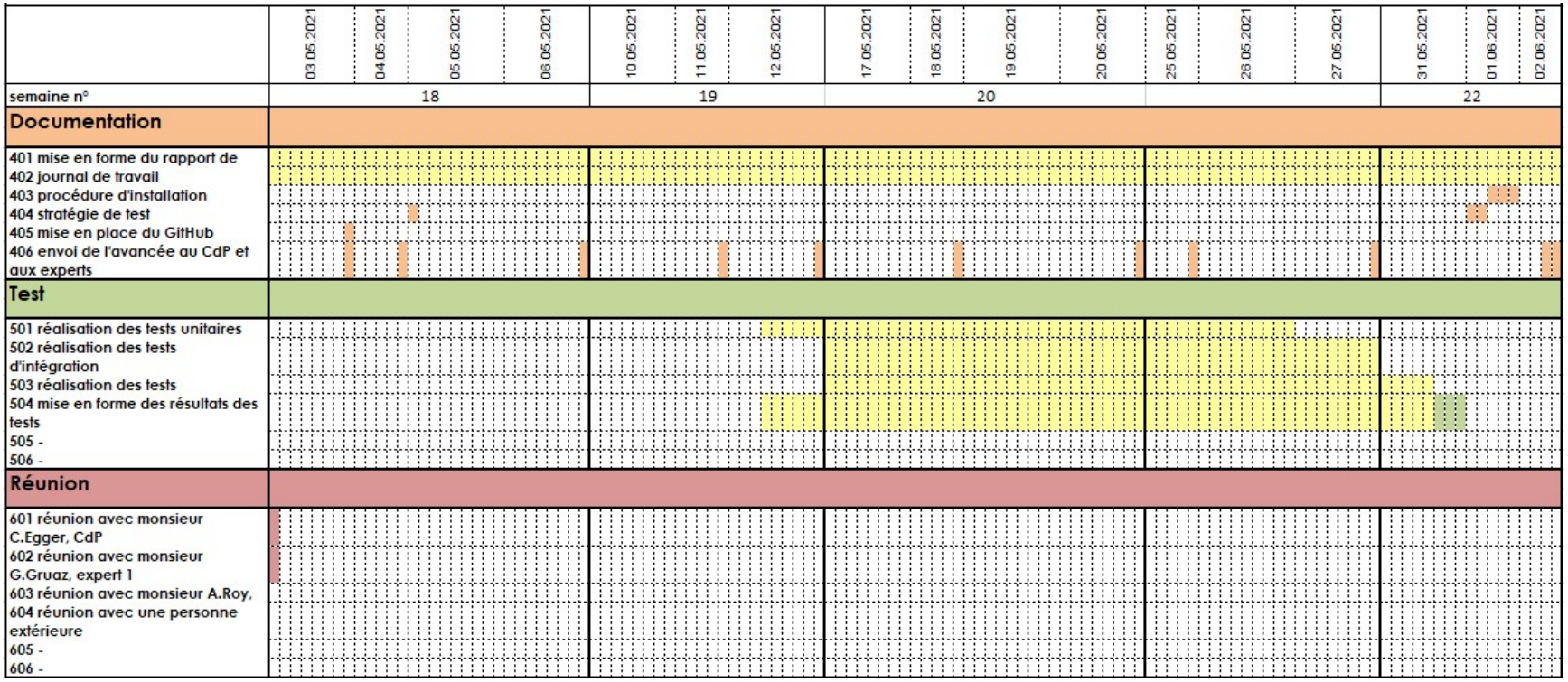


Figure 3 : planification initiale partie 3

2 Analyse / Conception

2.1 Cadre du projet

Comme cela a été mentionné précédemment, le temps à disposition pour réaliser le projet est fixe. 90 heures sont allouées pour le faire. Ces 90 heures commencent le lundi 03 mai 2021 à 8h50 et se terminent le mercredi 02 juin 2021 à 10h35. Aucun délai supplémentaire ne sera accordé.

Ensuite, seules les fonctionnalités mentionnées ci-après vont être implémentées. Les éléments d'amélioration ou d'évolution du projet, imaginés pendant la réalisation, seront mentionnées dans la conclusion, au point Évolutions et améliorations. Aucun élément complémentaire ne sera développé.

Enfin, toutes les technologies utilisées lors de ce projet ont été vues en cours. Même si certaines spécificités techniques ne sont pas connues, le langage en lui-même est un sujet étudié et maîtrisé.

2.2 Concept

2.2.1 Fonctionnalités

En plus des différents éléments abordés dans le point Objectifs de l'analyse préliminaire, plusieurs points techniques spécifiques doivent être respectés lors de la réalisation de ce projet. Ils sont explicités ci-dessous :

- Les données internes à l'application sont stockées dans une base de données MySQL.
- Au lancement de l'application, une connexion à la base de données est établie. Cette connexion est fermée lorsque l'application est quittée.
- Il est possible de gérer l'agencement de la cave en casiers à bouteille, dans lesquels les bouteilles sont placés.
- Il est possible d'ajouter des casiers à bouteilles supplémentaires.
- Il est possible de retirer des casiers à bouteilles.
- Il est possible d'ajouter des bouteilles.
- Il est possible de sortir (supprimer) des bouteilles.
- Il est possible d'ajouter une alerte, liée à certaines bouteilles spécifiques. Accompagnée d'un commentaire, cette alerte permet de « réserver » une bouteille pour une occasion spéciale.
- Il est possible d'effectuer une recherche par mot-clé sur l'application.
- Il est possible de trier les bouteilles présentes selon plusieurs critères spécifiques. Ces critères sont les suivants : le cépage, le producteur, le pays ou la robe du vin.
- Il est possible d'exporter, au format PDF, une liste des bouteilles respectant un critère particulier.
- Une historisation des actions effectuées sur l'application est disponible.

Des détails supplémentaires concernant ces fonctionnalités peuvent être trouvés dans le sous-chapitre Use Cases & Scénarios. Le fonctionnement exact de chaque point abordé ci-dessus y est développé.

2.2.2 Modèles de données

2.2.2.1 Modèle de données conceptuel

Le modèle conceptuel de données (MCD) ci-dessous représente les relations entre l'intégralité des données de l'application.

Parmi les éléments spécifiques à ce modèle, voici les points particuliers :

- L'entité « Robes » représente la couleur du vin. La liste de valeurs possibles sera restreinte aux valeurs les plus courantes : blanc, rosé et rouge.
- L'entité « Producteurs » comportera les différents producteurs à partir duquel provient le vin. Aucune méthode d'ajout de producteur n'est prévue pour le moment.
- L'entité « Casiers » gère des zones de stockage. Le texte « emplacement » est à titre indicatif pour l'utilisateur, il n'a aucun impact sur la réalité.
- L'entité « Historiques » sert à garder un historique des actions. Il comporte la date et l'heure exacte, l'action associée et le détail de ce qu'il s'est passé. Il contient soit une relation avec un vin, soit une relation avec un casier.

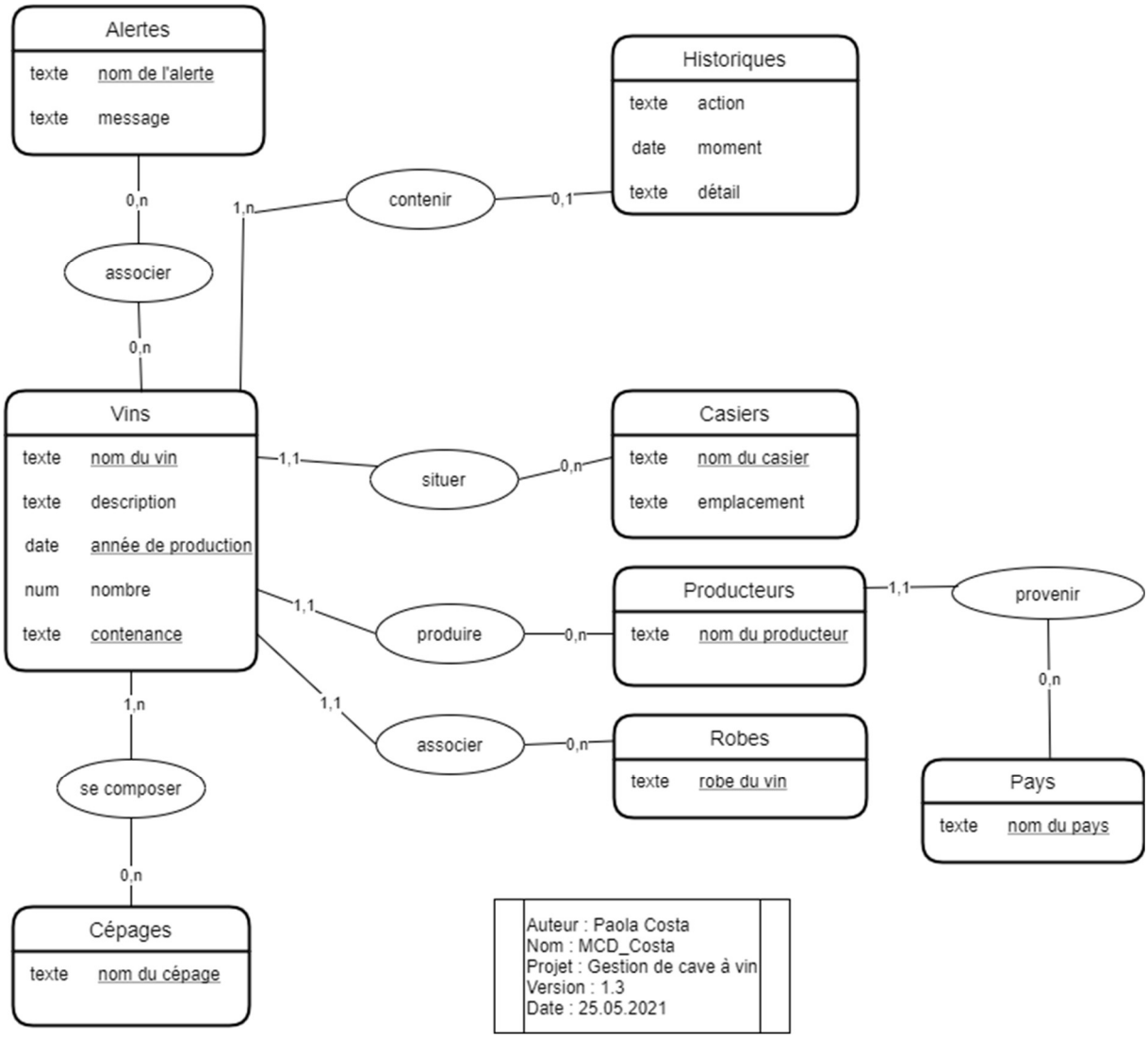


Figure 4 : modèle conceptuel de données

2.2.2.2 Modèle de données logique

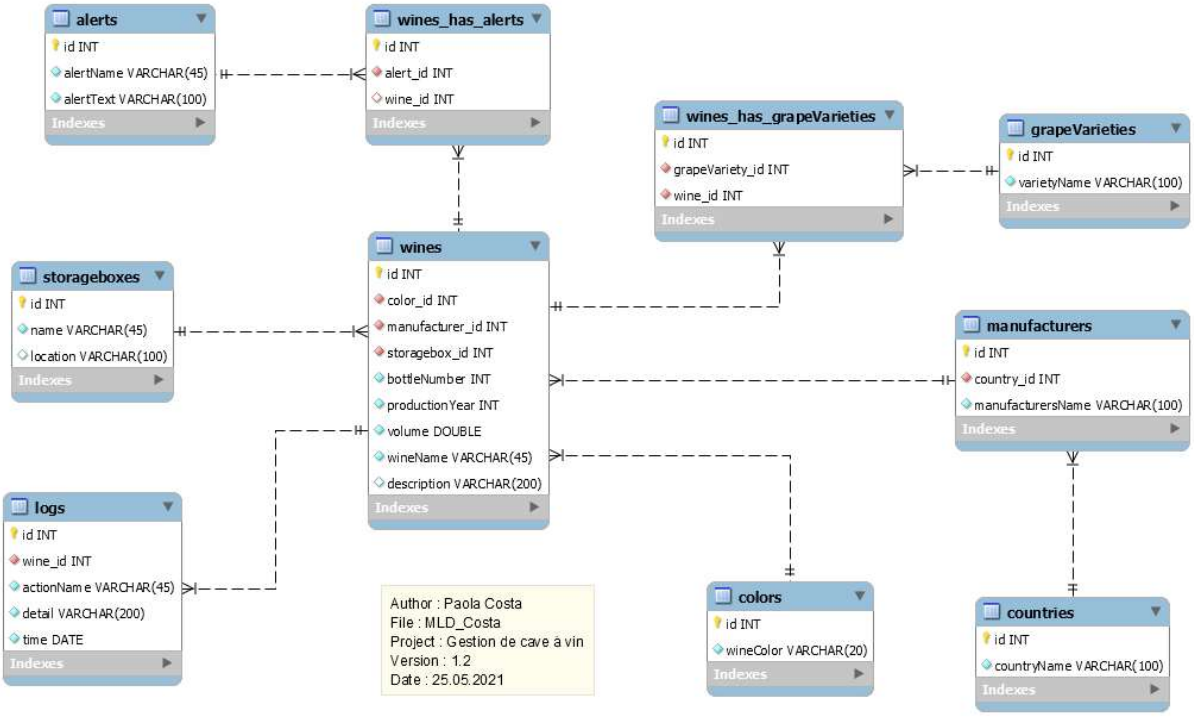


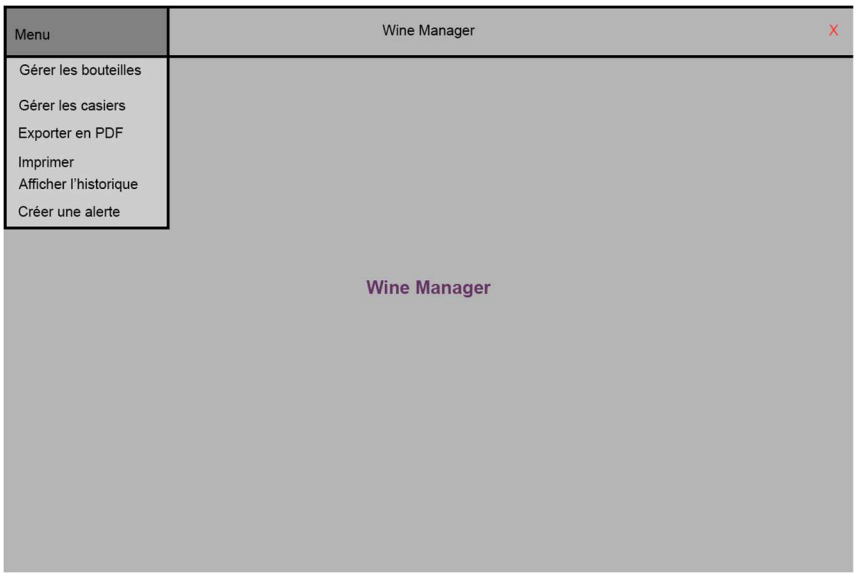
Figure 5 : modèle logique de données

Le modèle logique de données montre l'intégralité des tables présentes pour stocker les données. Ci-dessous se trouve une explication des différentes tables présentes :

- manufacturers : il s'agit du producteur du vin. Chaque producteur est associé à un seul pays.
- countries : il s'agit du pays dans lequel se trouve le producteur.
- colors : il s'agit de la couleur du vin, avec trois valeurs possibles : rouge, rosé et blanc.
- logs : il s'agit de l'historique des actions effectuées sur les bouteilles. Chaque action a un nom, une bouteille associée, une date de réalisation et le détail de ce qui est fait.
- storageboxes : il s'agit des casiers dans lesquels sont stockés les bouteilles.
- alerts : regroupe toutes les alertes. Celles-ci sont associées aux bouteilles.
- wines_has_alerts : il s'agit du lien entre les alertes et les bouteilles. La même alerte peut être attribuée à plusieurs vins différents et un vin peut avoir plusieurs alertes.
- grapeVarieties : il s'agit des cépages utilisés pour effectuer les assemblages. Comme le nombre de cépages existants est très important, une liste réduite sera utilisée ici.
- wines_has_grapevarieties : il s'agit du lien entre les vins et les cépages. Un vin peut être composé de plusieurs cépages et un cépage peut faire partie de plusieurs vins.
- wines : il s'agit des différents vins présents dans la cave. Un vin est défini par la combinaison nom, année, volume.

2.3 Maquettes

Afin de se représenter un minimum l'application, plusieurs maquettes vont être réalisées dans le cadre de ce TPI. Celles-ci vont être ajoutées ci-dessous et expliquées. Leur but principal est de donner une idée du visuel qui sera obtenu à la fin du projet.



Cette première maquette représente la vision que l'utilisateur a lors de l'ouverture de l'application.

Il s'agit d'une page d'accueil, sur laquelle on peut trouver le nom de l'application.

Les diverses fonctionnalités sont utilisables par un menu déroulant, situé en haut à gauche.

Figure 6 : maquette -> page d'accueil de l'application

Sur la maquette ci-contre, on peut voir la page sur laquelle il est possible d'ajouter des bouteilles à la cave.

La zone de formulaire en haut de l'écran permet d'entrer les données à ajouter, la zone inférieure permet de voir ce qui se trouve déjà dans la cave.

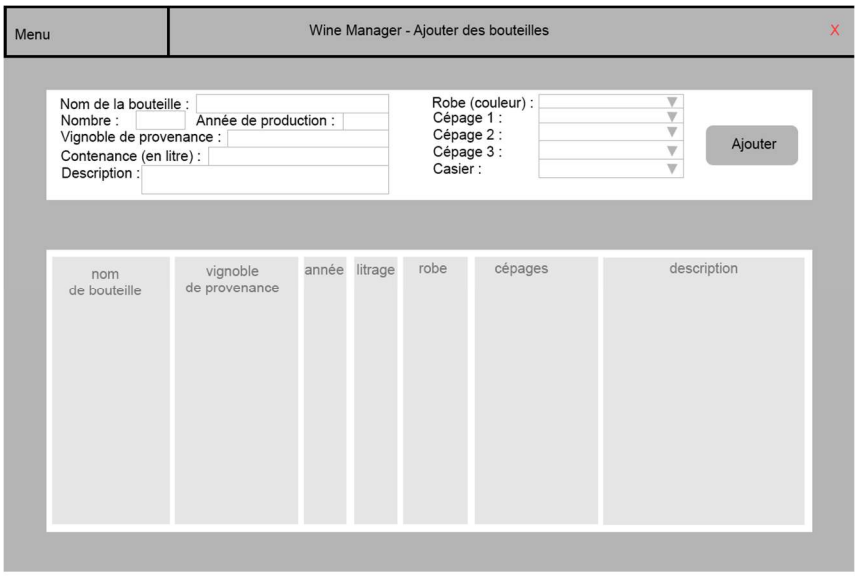


Figure 7 : maquette -> page d'ajout de bouteille(s)

Le retrait de bouteilles se fait sur une page au visuel semblable à celui de la figure 7. Les modifications sont que la zone de formulaire ne contient pas les mêmes champs. Il contient plusieurs champs de sélection, tels que : nom de bouteille, année, producteur, contenance et nombre.

Pour la partie de l'application permettant de gérer les casiers, le visuel est sur la même base que ci-dessus. Seulement, au lieu d'afficher les bouteilles en-dessous, on trouve l'affichage des différents casiers.

2.4 Schéma de navigation

Le schéma de navigation ci-dessous représente l'organisation de l'application. Chaque losange représente une des options du menu (voir figure 6, dans la partie Maquettes). Les rectangles sont les différentes pages de l'application. Le texte écrit en dessous de cela apporte des précisions sur le contenu de la page.

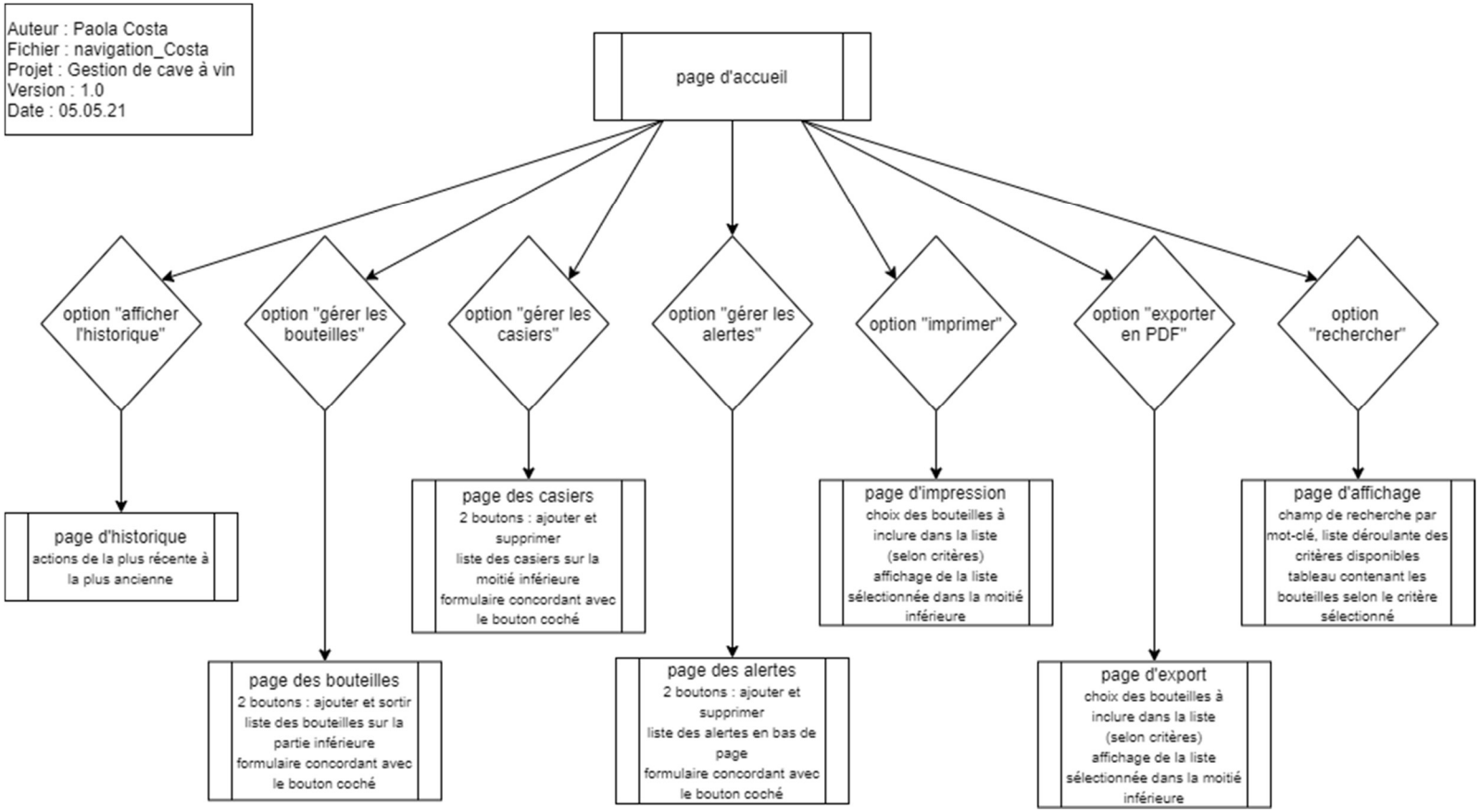


Figure 8 : schéma de navigation

2.5 Use Cases & Scénarii

2.5.1 Use Cases

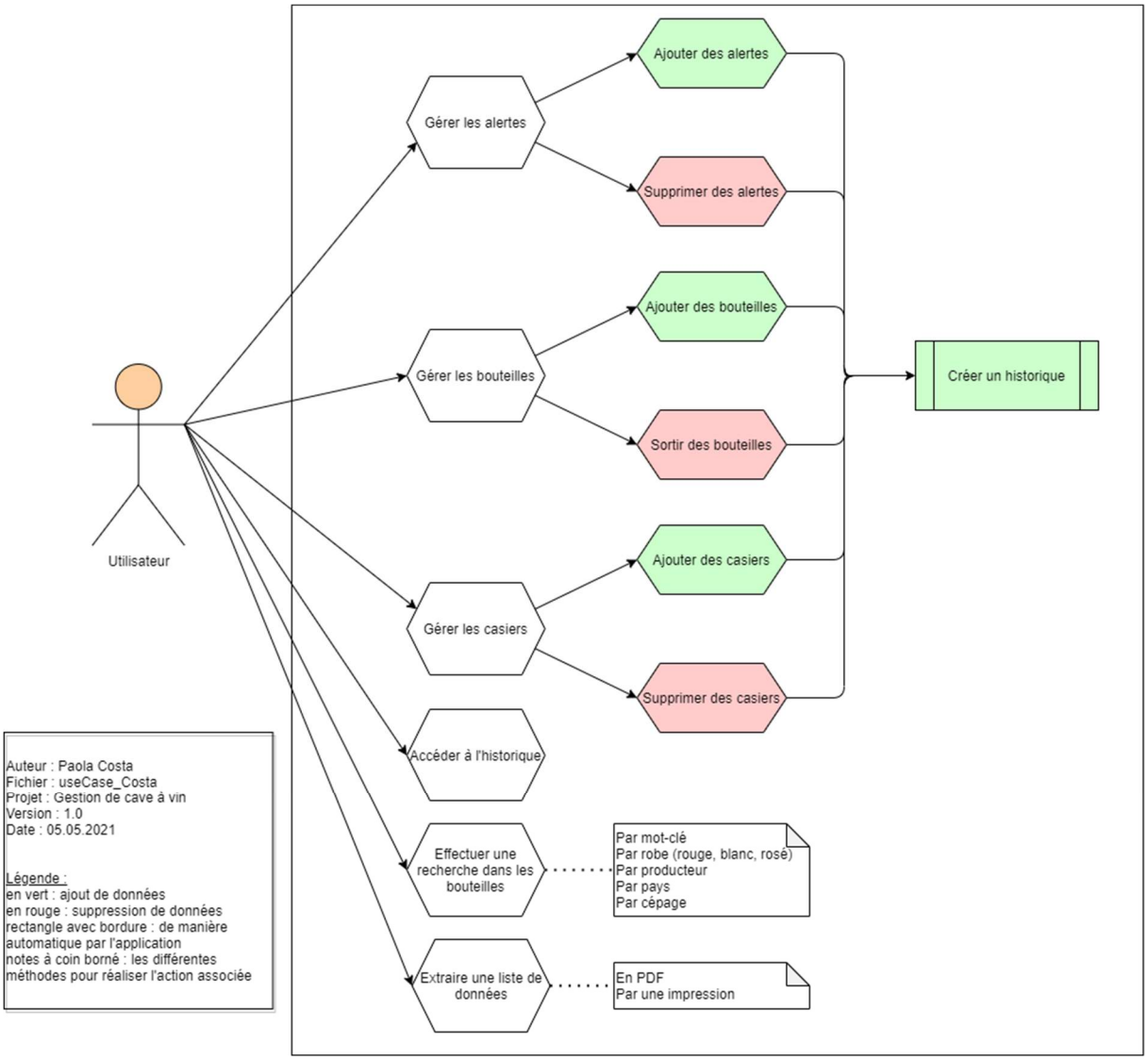


Figure 9 : diagramme des Use Case

Le diagramme ci-dessus représente les cas d'utilisation de l'application. Chaque hexagone au fond blanc représente une action, réalisable par l'utilisateur. Les hexagones verts et rouges représentent des actions plus spécifiques, respectivement des ajouts de données et des suppressions de données.

Le rectangle vert représente une action réalisée automatiquement par l'application, lorsqu'une des actions connectées est effectuée.

Les rectangles au coin corné représentent les différentes possibilités d'effectuer l'action à laquelle ils sont rattachés.

2.5.2 Scénarios

Les scénarios représentent les différentes actions réalisables dans l'application. Il s'agit d'une démarche détaillée du fonctionnement d'un élément précis. Ces scénarios vont être utilisés pour réaliser les tests fonctionnels par la suite. Ils sont donc tous associés à un numéro et à un nom, afin de pouvoir retrouver le test correspondant le moment venu.

Situation générique

Personne : utilisateur non authentifié
Emplacement dans l'application : page d'accueil

Scénario 1 : Gérer les bouteilles

Action	Condition	Réaction
La personne sélectionne l'option « gérer les bouteilles » dans le menu.		La page qui s'affiche est la page des bouteilles de la cave.

Scénario 1a : Ajouter une bouteille

Situation : l'utilisateur non authentifié est déjà sur la page des bouteilles (voir scénario 1)

Action	Condition	Réaction
La personne sélectionne le bouton « ajouter une bouteille »	L'option est déjà sélectionnée	L'affichage ne change pas, le formulaire affiché est celui d'ajout de bouteille(s)
	L'option sélectionnée a changé	Le formulaire d'ajout de bouteille(s) s'affiche
L'utilisateur remplit les champs du formulaire (nom, nombre de bouteilles, contenance, année de production) et sélectionne des valeurs dans les listes déroulantes (producteur, robe, cépage)	L'un des champs obligatoires n'est pas rempli	Un message d'erreur s'affiche
	Un champ est rempli avec une valeur non supportée	Un message d'erreur s'affiche
	Tous les champs sont remplis correctement	Retour à la page des bouteilles, avec une actualisation de la liste des bouteilles

Scénario 1b : Sortir une bouteille

Situation : l'utilisateur non authentifié est déjà sur la page des bouteilles (voir scénario 1)

Action	Condition	Réaction
La personne sélectionne le bouton « sortir une bouteille »	L'option est déjà sélectionnée	L'affichage ne change pas, le formulaire affiché est celui de retrait de bouteille(s)
	L'option sélectionnée a changé	Le formulaire de retrait de bouteille(s) s'affiche
L'utilisateur sélectionne des valeurs dans les listes déroulantes (nom, année, producteur, contenance) et remplit le champ spécifiant le nombre de bouteilles à sortir	L'un des champs requis n'est pas rempli	Un message d'erreur s'affiche
	Un champ est rempli avec une valeur non supportée	Un message d'erreur s'affiche
	La combinaison des valeurs sélectionnées n'existe pas dans les bouteilles existantes	Un message d'erreur s'affiche
	Les champs nécessaires sont remplis correctement	Retour à la page des bouteilles, avec une actualisation de la liste des bouteilles

Scénario 2 : Gérer les casiers

Action	Condition	Réaction
La personne sélectionne l'option « gérer les casiers » dans le menu.		La page qui s'affiche est la page des casiers de la cave.

Scénario 2a : Ajouter un casier

Situation : l'utilisateur non authentifié est déjà sur la page des casiers (voir scénario 2)

Action	Condition	Réaction
La personne sélectionne le bouton « ajouter une casier »	L'option est déjà sélectionnée	L'affichage ne change pas, le formulaire affiché est celui d'ajout de casier
	L'option sélectionnée a changé	Le formulaire d'ajout de casier(s) s'affiche
L'utilisateur remplit les champs du formulaire (nom, emplacement)	L'un des champs obligatoires n'est pas rempli	Un message d'erreur s'affiche
	Un champ est rempli avec une valeur non supportée	Un message d'erreur s'affiche
	Les champs nécessaires sont remplis correctement	Retour à la page des casiers, avec une actualisation de la liste des casiers

Scénario 2b : Supprimer un casier

Situation : l'utilisateur non authentifié est déjà sur la page des casiers (voir scénario 2)

Action	Condition	Réaction
La personne sélectionne le bouton « supprimer un casier »	L'option est déjà sélectionnée	L'affichage ne change pas, le formulaire affiché est celui de suppression de casier
	L'option sélectionnée a changé	Le formulaire de suppression de casier s'affiche
L'utilisateur sélectionne le casier à supprimer dans la liste déroulante	Le casier contient encore des bouteilles	Un message d'erreur s'affiche
	Le casier est vide	Retour à la page des casiers, avec une actualisation de la liste des casiers

Scénario 3 : Gérer les alertes

Action	Condition	Réaction
La personne sélectionne l'option « gérer les alertes » dans le menu.		La page qui s'affiche est la page des alertes.

Scénario 3a : Ajouter une alerte

Situation : l'utilisateur non authentifié est déjà sur la page des alertes (voir scénario 3)

Action	Condition	Réaction
La personne sélectionne le bouton « ajouter une alerte »	L'option est déjà sélectionnée	L'affichage ne change pas, le formulaire affiché est celui d'ajout d'alerte
	L'option sélectionnée a changé	Le formulaire d'ajout d'alerte s'affiche
L'utilisateur remplit les champs du formulaire (nom, description) et sélectionne dans la liste déroulante la ou les bouteilles à associer (nom, producteur, année de production, contenance)	L'un des champs obligatoires n'est pas rempli	Un message d'erreur s'affiche
	Un champ est rempli avec une valeur non supportée	Un message d'erreur s'affiche
	Les champs nécessaires sont remplis correctement	Retour à la page des alertes, avec une actualisation de la liste des alertes

Scénario 3b : Supprimer une alerte

Situation : l'utilisateur non authentifié est déjà sur la page des alertes (voir scénario 3)

Action	Condition	Réaction
La personne sélectionne le bouton « supprimer une alerte»	L'option est déjà sélectionnée	L'affichage ne change pas, le formulaire affiché est celui de suppression d'alerte

	L'option sélectionnée a changé	Le formulaire de suppression d'alerte s'affiche
L'utilisateur sélectionne l'alerte à supprimer dans la liste déroulante		
L'utilisateur clique sur le bouton « supprimer l'alerte »		Retour à la page des alertes, avec une actualisation de la liste des alertes

Scénario 4a : Effectuer une recherche par mot-clé

Action	Condition	Réaction
L'utilisateur sélectionne l'option « Rechercher » dans le menu		Affichage d'une page avec toutes les bouteilles dans un tableau. Une liste déroulante se trouve à côté d'un champ texte en haut à droite, à côté d'un bouton « Rechercher »
L'utilisateur entre du texte dans le champ texte à côté du bouton « Rechercher » L'utilisateur clique sur le bouton « Rechercher »	Le texte spécifié n'a aucune correspondance dans les noms et descriptions des bouteilles	Le tableau s'actualise et s'affiche vide.
	Correspondance entre le texte entré et le nom et la description d'au moins 1 bouteille	Le tableau s'actualise et affiche les bouteilles correspondant à la recherche.

Scénario 4b : Filtrer selon un critère

Action	Condition	Réaction
L'utilisateur sélectionne l'option « Rechercher » dans le menu		Affichage d'une page avec toutes les bouteilles dans un tableau. Des boutons se situent à côté d'un champ texte en haut à droite, à côté d'un bouton « Rechercher »

L'utilisateur sélectionne un des filtres présents en tant que bouton	Aucune bouteille ne correspond au critère sélectionné	Le tableau s'actualise et s'affiche vide.
	Le critère sélectionné est le même que la liste déjà affichée	Rien ne se passe.
	Au moins 1 bouteille correspond au critère sélectionné	Le tableau s'actualise et affiche les bouteilles correspondant au critère.

Scénario 5 : Afficher l'historique

Action	Condition	Réaction
La personne sélectionne l'option « Afficher l'historique » dans le menu.		La page qui s'affiche est la page de l'historique.

Scénario 6 : Exporter au format PDF

Situation : l'utilisateur non authentifié est déjà sur la page des recherches.

Action	Condition	Réaction
La personne clique sur « exporter en PDF		Un fichier PDF se crée sur le disque C:

Scénario 7 : Imprimer

Situation : l'utilisateur non authentifié est déjà sur la page des recherches.

Action	Condition	Réaction
La personne sélectionner « imprimer »		L'impression du fichier se lance

2.6 Diagrammes de classe

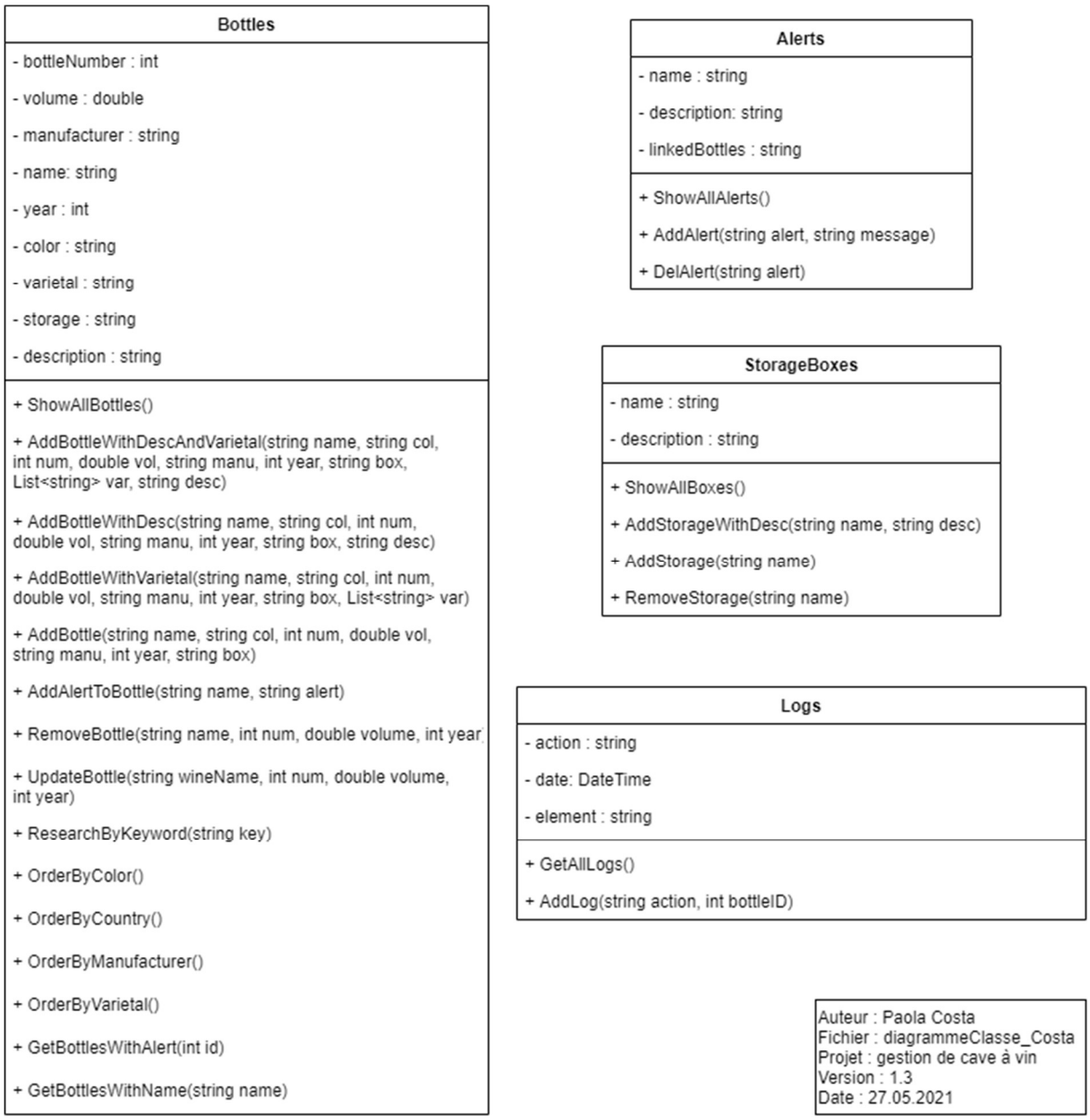


Figure 10 : diagramme de classe

Le diagramme de classe reprend les différentes classes nécessaires pour la partie graphique de l'application. Afin de pouvoir afficher des données dans l'application, il est nécessaire de stocker temporairement les données récupérées depuis la base de données, afin de pouvoir les afficher par la suite.
Les différentes classes ci-dessus servent exactement à ça :

- La classe « Bottles » permet de stocker les données liées à l'affichage des bouteilles.
- La classe « Alerts » permettra d'afficher les données liées aux alertes.
- La classe « StorageBoxes » permettra d'afficher les données liées aux casiers.
- La classe « Logs » permettra d'afficher l'historique des actions de l'application.

2.7 Risques techniques

Même si idéalement, il faudrait qu'aucun risque ne soit présent, il n'en est rien. De nombreux problèmes peuvent se poser lors du déroulement du projet.

Ainsi, un des éléments qui peut venir impacter l'avancée est un souci technique. Il est possible que, lors de la réalisation, un point spécifique du développement embête. Cela entraînerait des retards. La conséquence finale serait que le projet ne pourrait pas être complété dans les délais, laissant l'implémentation inachevée.

Afin d'éviter ce problème, le rythme d'avancée du projet proposée est soutenu. Cela laisse de la marge à la fin, au cas où un retard arriverait, afin que la réalisation puisse quand même être complétée. La demande d'aide à un professionnel, avant que cela ne s'éternise trop, est également une solution.

Un autre élément qui peut être problématique sont les absences. En effet, particulièrement avec le COVID-19 actuellement, il n'est pas inenvisageable qu'une quarantaine ou un confinement soit nécessaire alors que le développement devrait se faire. Une absence plus courte, due à la maladie, est également possible.

Si la situation d'une quarantaine devait se présenter, une prise de contact avec le CdP et les experts aurait lieu dès que possible. Cette discussion aurait pour but de déterminer des actions à suivre. Les deux solutions possibles sont : repousser la date de rendu, selon le temps d'inactivité ou autoriser l'avancée du projet depuis un lieu autre que le CPNV.

Enfin, il est possible que la motivation ne soit pas là 100% du temps. Cela aurait pour conséquence, encore une fois, d'entraîner des retards. Afin d'éviter ces baisses de motivation, la mise en place d'un fichier d'avancée des tâches, qui permet de voir où le projet en est, a été mis en place. Le fait de voir des tâches passer de « en cours » à « terminé » chaque jour motive et permet d'encourager la personne à continuer de travailler sur le projet.

2.8 Stratégie de test

Afin de limiter au maximum les problèmes techniques, des tests réguliers sont prévus, une fois la réalisation commencée. Ces tests ont pour but de détecter les erreurs avant que le développement n'ait trop avancé, pour que la perte de temps soit minimale. Ceux-ci ne seront pas validés, il s'agit juste de repérer les erreurs de compilation du programme.

Plusieurs types de tests vont donc être réalisés lors de ce projet. Il s'agit de tests unitaires et de tests fonctionnels. Les tests unitaires servent à tester une unité, un bloc de code ayant une fonction bien spécifique, alors que les tests fonctionnels permettent de vérifier que l'entièreté des unités interagissent correctement et retourne le résultat attendu.

Pour que le test unitaire ait un sens, il faut qu'un élément logique entre en compte dans l'application. Ainsi, seules les fonctions incluant de la logique seront testées avec des tests unitaires. Ces tests seront réalisés grâce aux classes de tests en C#.

Par rapport aux tests fonctionnels, leur réalisation se fait sur la base des Scénarios. Chaque scénario représente la démarche d'un test spécifique. Si le résultat obtenu correspond au résultat attendu dans le scénario, le test est considéré réussi.

Les tests liés aux scripts de la base de données seront uniquement fonctionnels. Lors de la génération de la base de données avec MySQL Workbench, si aucune erreur ne s'affiche dans la console, le test des scripts sera considéré comme validé.

2.9 Infrastructure

2.9.1 Matériel hardware et système d'exploitation

L'infrastructure hardware utilisée pour réaliser ce projet consiste uniquement en un ordinateur du CPNV, installé avec un système d'exploitation Windows10 x64. Aucun matériel supplémentaire n'est nécessaire.

2.9.2 Outils logiciels

Afin de mener à bien ce projet, divers outils logiciels vont être nécessaires. Ces logiciels sont listés ci-dessous. L'utilisation d'autres versions des logiciels peut potentiellement entraîner une incompatibilité et rendre le projet inaccessible, ou avec des erreurs qui ne sont pas présentes autrement.

Développement en C# : Microsoft Visual Studio Enterprise 2019, Version 16.8.5
- Paquet : MySql.Data, version 8.0.24 (intégration du MySQL)
- Paquet : iText7, version 7.1.15 (création de PDF)
- Paquet : FreeSpire.PDF, version 7.2.0 (impression d'un PDF)

Base de données : MySQLWorkbench, version 6.3.6 build 511 CE (64bits)

Maquettes : Adobe Photoshop 2019, version 10.0.6
Diagramme de classe, MCD, schéma de navigation : draw.io, version 13.9.9

Rapport de projet : Microsoft Word 2016 MSO, version 16.0.4266.1001
Planifications, journal de travail, avancée des tâches : Microsoft Excel 2016 MSO, version 16.0.4266.1001

Historisation des données : L'intégralité du pré-TPI est disponible sur GitHub. Les données sont présentes à l'adresse suivante :

https://github.com/AhVen98/TPI_gestionCaveAVin

GitHub est la méthode de versioning utilisée dans ce projet. Il s'agit du moyen mis en place pour pouvoir récupérer les anciennes versions du projet en cas de soucis/besoins.

2.9.3 Architecture du projet

L'architecture qui sera utilisée dans ce projet sera une architecture MVVM. Il s'agit d'une architecture qui effectue une liaison entre les vues et les modèles, et ce dans les deux sens. Cela signifie que les vues interagissent avec les modèles et que les modèles interagissent avec les vues.

L'architecture MVC (modèle, vue, contrôleur) est une architecture qui aurait potentiellement pu être utilisé. Cependant, lors du pré-TPI, j'ai réalisé un projet de gestion en C# en essayant d'intégrer le MVC. Il m'a été nécessaire, moins d'une semaine avant la reddition du projet, de faire marche arrière et d'enlever ce contrôleur, car une fonctionnalité de base n'était pas compatible avec sa présence et bloquait toute la réalisation. Ce choix a donc été abandonné et ne sera pas réitéré lors du TPI.

2.10 Planification définitive (TODO)

A REDIGER AU PROPRE !!!

Modif dans l'analyse/conception par rapport à la planif initiale -> ordre des tâches, temps réduit pour le MCD, scénarios + long, diagramme de flux non réalisé, car inutile

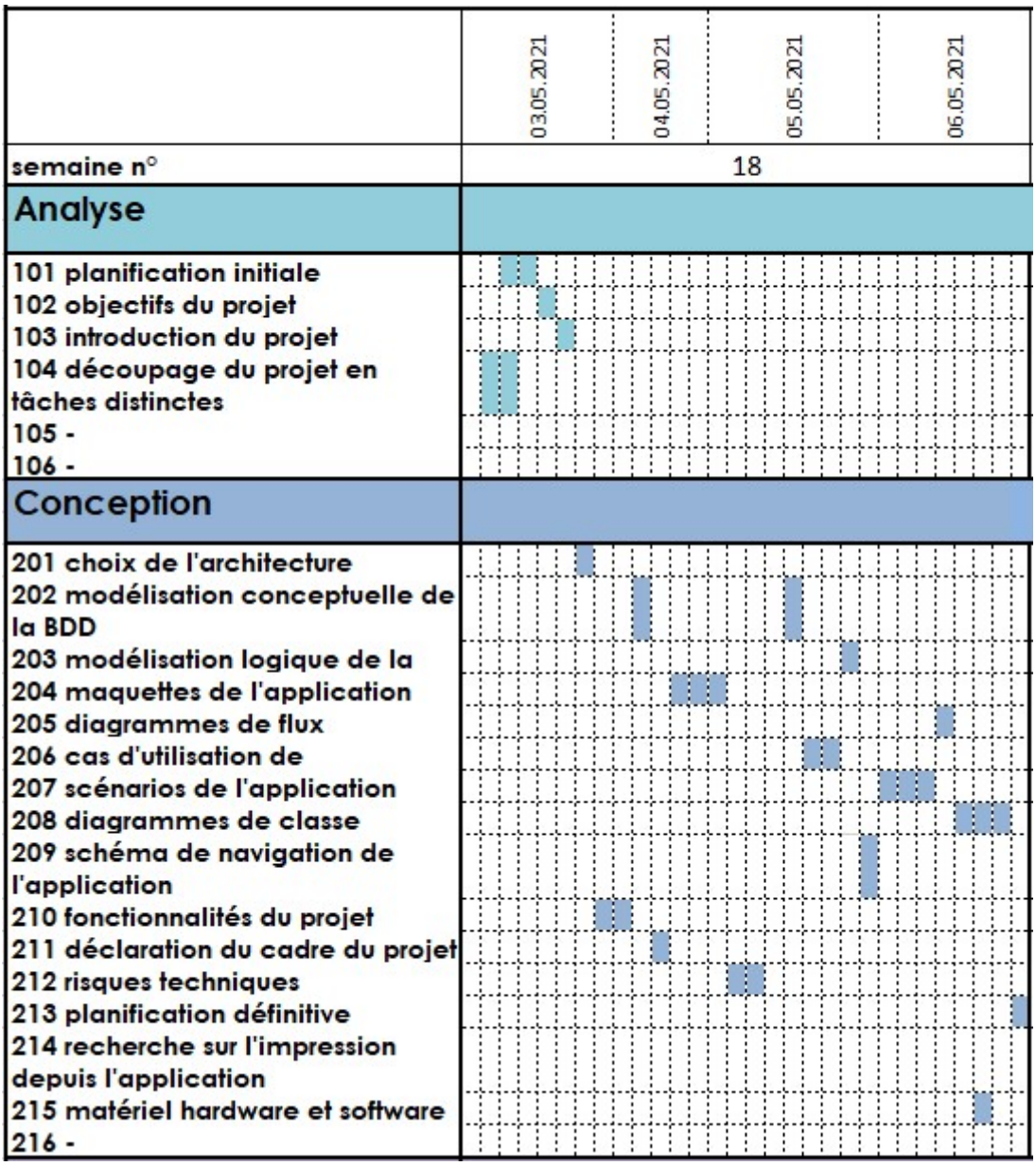


Figure 11 : planification définitive partie 1

Modif par rapport à l'initiale : classe supplémentaire à créer -> alerte et logs

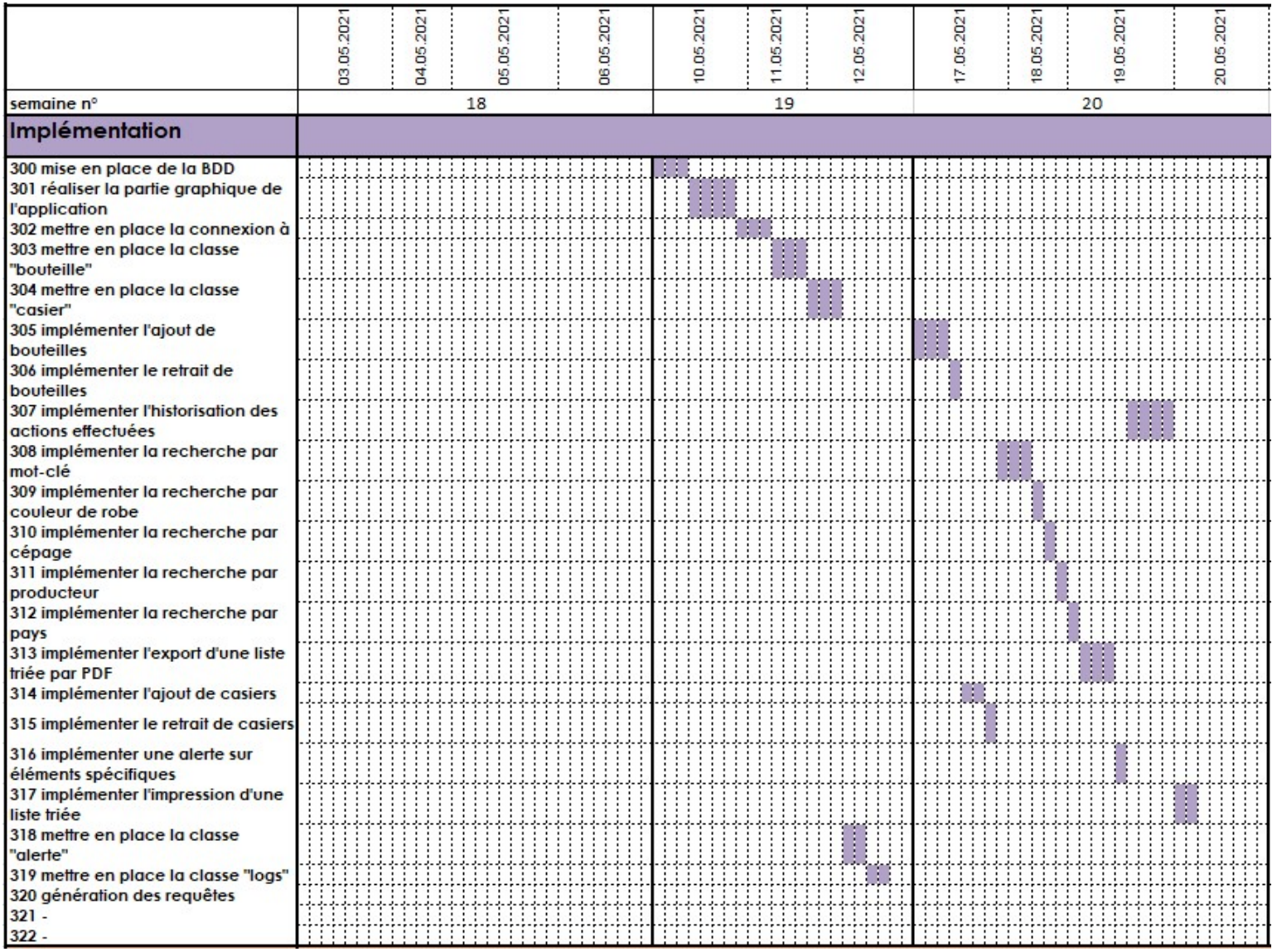


Figure 12 : planification définitive partie 2

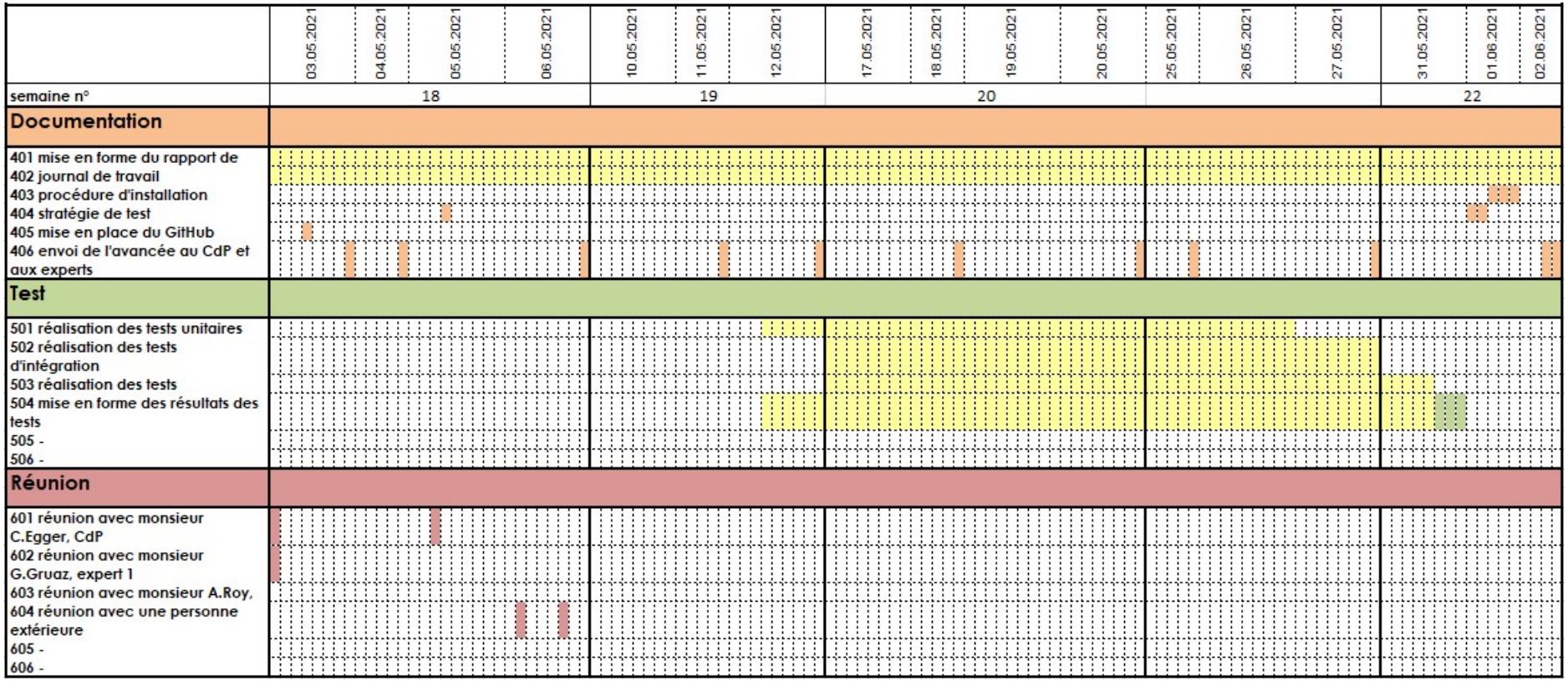


Figure 13 : planification définitive partie 3

Modif par rapport à l'initiale : réunion avec chef de projet + discussion avec personne extérieure -> imprévisible donc pas insérable dans la planif à l'avance.

3 Réalisation

3.1 Dossier de réalisation

3.1.1 Répertoires et fichiers du projet

3.1.1.1 Répartition physique des fichiers

Le projet C# consiste en une solution globale, « wineManager ». Celle-ci comporte 3 parties, appelés projets.

- La première, « wineManager_Library », contient l'intégralité des classes d'objets ainsi que les méthodes utilisées. Il s'agit du côté « données » de l'application.
- La seconde, « wineManager_View », regroupe l'intégralité des WindowsForms. C'est cette partie qui contient tout ce qui a trait au côté visuel de l'application.
- La troisième et dernière partie est « wineManager_Tests ». Il s'agit du projet qui contient tous les tests unitaires effectués lors du projet, afin de s'assurer du bon fonctionnement des différentes méthodes.

3.1.1.2 Fichiers et description

Partie 1 : wineManager_Library

- Alerts.cs : il s'agit d'une classe qui gère toutes les méthodes propres à l'affichage, l'ajout et la suppression d'alertes.
- Bottles.cs : il s'agit d'une classe qui gère toutes les méthodes permettant l'affichage, l'ajout et le retrait de bouteilles. L'affichage inclut les méthodes de tri des bouteilles selon différents critères.
- DBConnection.cs : il s'agit du fichier qui met en place la connexion à la base de données de l'application. Sans lui, il n'est pas possible d'établir la connexion.
- DBRequest.cs : il s'agit du fichier qui contient l'intégralité des requêtes SQL du projet. Toutes les classes qui ont besoin d'interagir avec la base de données passent par ce fichier pour le faire.
- Logs.cs : il s'agit de la classe qui gère l'historique des actions effectuées sur les bouteilles. Cette classe gère les méthodes d'affichage et d'ajout de logs. Il n'y a aucun moyen pour l'utilisateur d'appeler directement ces méthodes, elles sont appelées automatiquement sur des actions bien précises de l'utilisateur.
- StorageBoxes.cs : il s'agit de la classe qui gère les différents espaces de stockage de bouteilles. Les méthodes présentes gèrent l'affichage, l'ajout et la suppression de casiers.

Partie 2 : wineManager_View

Chaque fichier cité ci-dessous contient en réalité deux fichiers. Le premier, « .Designer.cs », comprend la partie graphique du fichier sous forme de code. Le second, « .cs », comprend les méthodes liées à cette partie graphique.

- AlertsManagement.cs : il s'agit du formulaire qui permet l'affichage des alertes. Il contient un récapitulatif des alertes présentes, ainsi qu'un espace de formulaire pour l'ajout ou la suppression d'alerte en haut de la page.
- BottlesManagement.cs : il s'agit du formulaire qui permet l'affichage des bouteilles. Il contient un récapitulatif des bouteilles présentes, ainsi qu'un espace de formulaire pour l'ajout ou le retrait de bouteilles en haut de la page.

- Logs.cs : il s'agit du formulaire qui permet l'affichage de l'historique des actions. Il contient uniquement un tableau qui affiche les différentes actions effectuées.
- MainPage.cs : il s'agit de la page d'accueil de l'application. Un menu en haut à gauche donne accès aux différentes pages.
- Research.cs : : il s'agit du formulaire qui permet d'effectuer des recherches dans les bouteilles. Les bouteilles présentes dans la base de données sont affichées et différentes options de tri sont disponibles. Il s'agit également de la page par laquelle l'impression et l'export au format PDF sont possibles.
- StorageBoxesManagement.cs : il s'agit du formulaire qui permet l'affichage des casiers. Il contient un récapitulatif des casiers déjà présents, ainsi qu'un espace de formulaire pour l'ajout ou la suppression de casiers en haut de la page.

Partie 3 : wineManager_Tests

Ce projet contient l'intégralité des classes de tests. Celles-ci permettent d'exécuter automatiquement les tests unitaires liées aux différentes méthodes du projet. Chaque fichier « .cs » du projet Test reprend la classe associée du projet Library.

3.1.2 Produit fini (TODO)

Version du produit fini – état de complétion – logiciel + version nécessaire à l'utilisation
Application wineManager, en .exe
Version 1.x, machine sous Windows10
Nécessaire : base de données locales (mysqlworkbench
Etat : fini, mais avec erreurs sur 1 fonctionnalité + améliorations possibles

3.2 Liste des documents fournis (TODO)

Liste exhaustive des documents pour le client, avec lien sur l'annexe + version
Planification initiale
Rapport de projet
Journal de travail
Procédure d'installation
Stratégie de test
Code source (archive.zip)
Script BDD (génération de la base, insertion des données)

3.2.1 Programmation et scripts

Interaction avec la base de données :
[insertion de données] :

Méthodologie : création d'un script SQL avec les données sur des lignes définies. Pour les données générées automatiquement en .xlsx -> génération des requêtes grâce à la fonction CONCATENER()

base de données -> génération – nom des fichiers + rôles – éléments nécessaires

3.3 Description des tests effectués (TODO)

3.3.1 Tests unitaires

Les tests unitaires sont des tests automatiques qui sont réalisables dans Visual Studio 2019 (projet wineManager_Tests de la solution wineManager). Afin de les lancer, il est nécessaire de changer la chaîne de connexion à la base de données.

Pour se faire, il faut commenter la ligne habituelle et enlever le commentaire de la ligne contenant la base de test. Il est également nécessaire de régénérer la base de données à chaque lancement des tests automatiques, sinon certains tests ne se valideront pas.

En regardant le tableau des tests unitaires à la page suivante, il est possible de voir que la majorité des tests unitaires échouent. Il ne s'agit pas d'un problème de développement des méthodes de l'application, comme l'atteste les tests fonctionnels réussis, mais d'un souci au niveau de l'implémentation des méthodes de tests.

La correction de ces tests pour les rendre fonctionnel est réalisable, mais cela nécessiterait un investissement en temps trop conséquent pour pouvoir l'inclure dans les délais restants.

3.3.2 Tests fonctionnels

Test 1 : Insertion de données dans la base de données

Méthodologie de test : vérifier qu'aucune erreur ne se produit lors de l'exécution du script d'insertion de données

Contrôle : la fenêtre de logs, présente dans MySQL Workbench, permet de s'assurer que toutes les requêtes ont été exécutées correctement.

Test 2 : Génération de la base de données

Méthodologie de test : vérifier qu'aucune erreur ne se produit lors de l'exécution du script d'insertion de données

Contrôle : la fenêtre de logs, présente dans MySQL Workbench, permet de s'assurer que toutes les requêtes ont été exécutées correctement.

Scénario 4a : Recherche par mot-clé

La recherche par mot-clé ne fonctionne pour le moment pas. Il reste une erreur au niveau de la requête SQL, qui rend l'utilisation de la recherche impossible et fait planter le programme.

3.3.3 État des tests

L'état des tests est un tableau qui reprend l'ensemble des tests qui ont été effectués, ainsi que leur état (validé / non validé). Cela permet de voir facilement qu'est-ce qui est fonctionnel et qu'est-ce qui ne l'est pas. Le détail des démarches de tests se trouvent dans Scénarios et Tests.

Tests unitaires :

Nom du test	État
AddAlerts_AllParams_OK	OK
DelAlerts_AllParams_OK	OK
ShowAllAlerts_NoSorting_OK	NO OK
AddAlertToBottle_Allparams_OK	OK
AddBottleWithDescAndVarietal_OK	NO OK
GetBottlesWithAlert_NoSorting_OK	NO OK
GetBottlesWithName_NoSorting_OK	NO OK
OrderByColor_alphabetical_OK	NO OK
OrderByCountry_alphabetical_OK	NO OK
OrderByManufacturer_alphabetial_OK	NO OK
RemoveBottle_Allparams_OK	OK
ResearchByKeyWord_existingword_OK	NO OK
ShowAllBottles_NoSorting_OK	NO OK
UpdateBottle_Allparams_OK	OK
AddLog_AllParams_OK	NO OK
GetAllLogs_NoSorting_OK	NO OK
AddStorageWDesc_AllParams_OK	OK
RemoveStorage__OK	OK

Tests fonctionnels :

Nom du test	État
Scénario 1a : ajout de bouteilles	OK
Scénario 1b : retrait de bouteilles	OK
Scénario 2a : ajout de casiers	OK
Scénario 2b : suppression de casiers	OK
Scénario 3a : ajout d'alerte	OK
Scénario 3b : suppression d'alerte	OK
Scénario 4a : recherche par mot-clé	NO OK
Scénario 4b : filtre par critère	OK
Scénario 5 : afficher l'historique	OK
Scénario 6 : imprimer	OK
Scénario 7 : exporter en PDF	OK
Test 1 : insertion de données par script dans la BDD	OK
Test 2 : génération de la BDD	OK

3.4 Problèmes rencontrés et résolution

[10.05.2021] Connexion à la base de données

Suite à l'expérience retirée du projet de pré-TPI, pour diminuer le nombres d'ouverture et fermeture à la base de données, j'ai décidé d'ouvrir la connexion une fois au lancement de l'application et de fermer cette connexion à la fermeture de l'application.

Le souci qui s'est présenté est le suivant : la vue principale (MainPage.cs) ne reconnaît pas la classe DBConnection, comme s'il manquait un référencement quelque part.

Afin de résoudre ce problème, je vais demander de l'aide pour obtenir une piste de résolution, afin de ne pas perdre trois heures à régler un petit problème, qui a des conséquences importantes sur le projet. L'aide sera demandée à F.Andolfatto.

L'aide apportée par F.Andolfatto le 11.05.21 a permis de corriger le problème : la référence manquante devait se faire dans le projet *WineManager_View*. Un clic droit sur Références -> Ajouter une référence (voir image ci-contre), avec une coche pour le projet *WineManager_Library* est l'élément qui manquait pour faire fonctionner le tout.

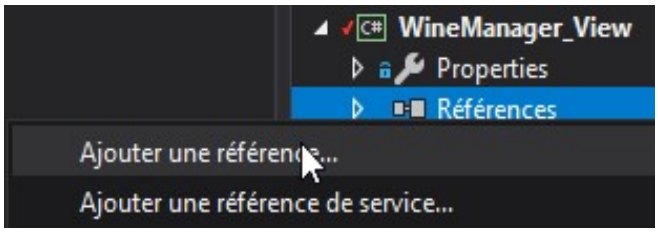


Figure 14 : problème rencontré -> référence de projet

[11.05.21] Récupération d'une liste dans une liste par requête SQL (classe « Bottles »)

Pour pouvoir afficher les différents cépages d'un vin, il m'est nécessaire de récupérer une liste de cépages, dans la liste représentant un vin spécifique.

Hors, il n'est pas possible de récupérer une liste de données d'une table intermédiaire si d'autres paramètres sont également nécessaires.

Ainsi, pour pouvoir insérer une liste dans la liste, il est nécessaire d'appeler une autre requête qui crée une liste. Celle-ci sera ensuite parcourue, pour intégrer élément par élément à la liste globale.

Cette méthode a été suggérée par X.Carrel, qui a pu m'expliquer le manquement dans ma logique de résolution et apporter des éléments théoriques complémentaires.

[12.05.2021] Bouton retour à l'accueil non fonctionnel

Le bouton permettant de retourner de la page sélectionnée à la page d'accueil ne fonctionne pour le moment pas comme il se doit.

La fenêtre active se ferme bien, mais la page d'accueil reste grisée et inactive, impossible d'interagir avec.

Plusieurs éléments ont été testés, des recherches supplémentaires sont encore nécessaires pour régler le souci.

Il s'agit d'un élément complémentaire pour fluidifier l'utilisation de l'application, ce n'est donc pas prioritaire pour le moment.

EDIT 25.05.2021 : une solution a été trouvée finalement. L'utilisation de la méthode « ShowDialog() » plutôt que « Show() » sur les formulaires permet d'obliger l'utilisateur à fermer le formulaire avant de pouvoir à nouveau interagir avec la page précédente.

[12.05.2021] récupération de la liste dans la liste (cf problème du 11 mai)

La requête SQL permettant de récupérer les données est désormais fonctionnelle. Cependant, seul le premier élément de la liste est récupéré, les suivants ne le sont pas. Du travail est encore nécessaire pour afficher la totalité des données.

EDIT 17.05.2021 : en recherchant la manière de mettre en place des listes déroulantes, la syntaxe nécessaire est apparue sur une des pages (voir source : récupération de données pour une liste déroulante depuis MySQL). En mettant en place les requêtes d'ajout de bouteilles à la BDD, les modifications ont été faites et cela a corrigé le problème rencontré.

[26.05.2021] pop-ups de confirmation

Lors de la conception, les scénarios intégraient des pop-ups de confirmation avant de valider la réalisation d'une action.

Au cours de la réalisation, j'ai pu m'apercevoir que mettre en place ces pop-up étaient tout sauf simples. De plus, cela avait pour conséquence de surcharger la page.

Le fait de valider les modifications de données par une MessageBox a donc été supprimé et les scénarios ont été modifiés en conséquence.

3.5 Erreurs restantes (TODO)

Recherche par mot-clé :

La fonctionnalité « Rechercher par mot-clé » ne fonctionne pas. Lorsque le bouton « Rechercher » est cliqué sur l'application, le programme plante et affiche un message d'erreur.

L'erreur restante à ce niveau-là est une erreur de syntaxe dans la requête SQL. Cette erreur se trouve dans la méthode « ResearchByKeyword ».

L'impact de cette erreur est minime. Seule la fonctionnalité de recherche par mot-clé est inutilisable, tout le reste est fonctionnel et n'est pas impacté.

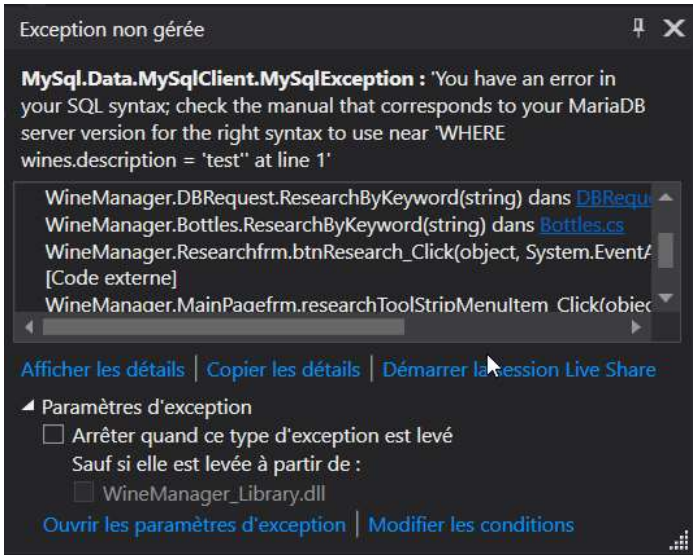


Figure 15 : erreur -> recherche par mot-clé

Afin de corriger cette erreur, il s'agit uniquement de consacrer du temps à trouver l'erreur de syntaxe SQL dans le code.

Tests unitaires :

Lors de l'exécution des tests unitaires, seuls 7 des 19 tests créés sont validés (36.8%). Cela provient d'une erreur de développement des tests et non pas de l'application, car les tests fonctionnels sont validés (voir État des tests pour plus de détails).

La raison pour laquelle les tests échouent semblent être une génération incorrecte de ma part des listes de contrôle de données. Les tests contiennent en effet un résultat fixe (resExpected) qui est comparée aux valeurs obtenus par l'application en exécutant la méthode à tester (resCalculated).

Dans le cas où le résultat fixe est mal généré, le test ne peut pas être validé car sa base de comparaison est incorrecte.

Techniquement parlant, les compétences nécessaires pour corriger ces erreurs dans les tests sont présentes. L'élément manquant dans ce projet est le temps.

Du côté fonctionnel, le fait que les tests échouent n'impactent en rien le fonctionnement de l'application. Tout fonctionne sans être dépendant de la réussite des tests.

3.6 Comparaison des délais entre la planification et la réalisation (TODO)

Chaque semaine – entre planif déf et réalité – avec image (gant) – retards ? avance ? -> raison

[05.05.2021] Retard de train
-> début du travail 25 minutes plus tard
[12.05.2021] Retard de train
-> début du travail 30 minutes plus tard
[17.05.2021] mise en place de l'ajout de bouteilles
-> beaucoup plus long que prévu : entre les tests de types, les requêtes et tout, cela a pris beaucoup plus de temps que prévu
[17.05.2021] implémentation des recherches
-> le temps envisagé pour la mise en place des différentes recherches est beaucoup plus élevé que le temps réel : la solution finale est beaucoup plus courte à mettre en place => équilibre avec l'ajout de bouteilles
[25.05.2021] Retard de train
-> début du travail 30 minutes plus tard
[26.05.2021] Débug des tests unitaires
3h de temps pour essayer de débugger les tests unitaires -> abandonné, car l'application est fonctionnelle derrière

4 Conclusions (TODO)

4.1 Atteinte des objectifs

Validation (ou non) de l'atteinte des objectifs

4.2 Maintien des délais

Comparaison de gant (définitif et final) – délai maintenu ? pourquoi ? - raison des retards / avances sur tâches individuelles

Ajout de bouteilles -> perte de temps : beaucoup de vérification de données à mettre en place, première méthode : temps important nécessaire
Recherche /tri -> temps très courts par rapport à l'estimation
Génération des pdfs -> débug à cause d'une erreur d'index : perte de temps à ce niveau-là
Mise en place des alertes -> relativement compliqué : perte de temps associée

4.3 Points positifs et négatifs

Bien passé ? – mal passé ? – raison – impact – connaissances acquises (comment, où) – réflexion critique sur approche + résultats obtenus – comparaison des variantes de solutions ou explications sur pourquoi il n'y en a pas – bilan personnel

++ beaucoup appris (débug, gestion de fenêtres, …)
++ envie de continuer à bosser dessus

-- sentiment de travail inachevé -> améliorations nécessaires
-- frustration de ne pas savoir résoudre les erreurs

-- code mal géré -> fonctionnel, mais retravaillable pour être plus léger et moins brouillon (passer des éléments en paramètres pour diminuer le nombre de méthodes …)

GLOBAL : bien passé, arrivé au bout, satisfaite, mais déçue d'un certain côté de la façon dont c'est codé. Une certitude : je ne suis pas faite pour développer 100% de ma journée et ça, je le sais désormais. Par contre, rédiger de la doc, c'est pas la même :3

4.4 Difficultés particulières

Eléments qui ont particulièrement embêté (retard, …)

Affichage de listes dans une liste
Gestion des fenêtres, pour empêcher d'avoir plusieurs fenêtres laissées actives
Utilisation du débug -> encore compliqué, pas instinctif -> perte de temps en essayant de trouver une erreur car l'outil est mal maîtrisé.
Impression de fichier -> trouver une solution viable a été compliqué + gestion de 2 paquets ayant des noms de classes identiques …
Motivation : certains jours -> compliqué d'être efficace : boulot moins « complet »

4.5 Évolutions et améliorations

Possibilité d'aller plus loin ? – modifications ? améliorations ? faire quelque chose autrement ? => projeter le projet dans le futur

Ajout de rôles – administrateurs et utilisateurs
Ajouter à l'historique les casiers et les alertes
Page d'historique – afficher l'historique, à choix : des bouteilles, des casiers ou des alertes
Imprimer/exporter les logs
Permettre de mettre la même bouteille dans plusieurs casiers différents
Suppression d'alertes/casiers en les sélectionnant dans le tableau de récap
Mettre en place un fichier de configuration pour pouvoir modifier facilement certains paramètres (nom de la base de données, nom de l'imprimante…)
Reprendre le code et le simplifier -> certaines méthodes doivent pouvoir être groupées
Gérer l'affichage des listes dans le tables -> peu lisibles si beaucoup d'infos, trouver une solution (retour à la ligne ?) …

5 Annexes

5.1 Résumé du rapport du TPI

3 § : situation de départ, mise en œuvre, résultats – destiné au grand public (termes accessibles) – aspects essentiels – page A4 max – pas de graphiques

5.2 Glossaire

A

Architecture MVC

L'architecture MVC est une architecture en trois blocs. D'une part se trouvent les modèles, avec toutes les données. D'une autre se trouvent les vues, avec le côté graphique de l'application. Le lien entre ces deux parties est réalisé à l'aide d'un contrôleur. Les vues n'ont aucun accès direct aux données.

Architecture MVVM

L'architecture MVVM est, contrairement à l'architecture MVC, une architecture en deux blocs, avec d'un côté les modèles et de l'autre les vues. Ces deux parties interagissent dans les deux sens, sans passer par un point central.

Assemblage

Dans le monde du vin, on parle d'assemblage dès le moment où plusieurs cépages sont utilisés afin d'obtenir un vin spécifique. Par exemple, un vin de Bordeaux est un assemblage de Cabernet-Sauvignon, de Cabernet-Franc et de Merlot, trois cépages connus.

Un vin peut donc être composé d'un ou plusieurs cépages.

B

BDD

BDD est un acronyme utilisé pour les termes « base de données ». Il s'agit d'une méthode de stockage d'informations dans plusieurs groupes (tables) interconnectés par des liens (relations).

C

CdC

Il s'agit d'un acronyme pour les termes « Cahier des charges ». C'est un document qui contient l'ensemble des informations mises à disposition pour la réalisation du projet d'A à Z.

CdP

Il s'agit d'un acronyme pour les termes « Chef de projet ». C'est la personne qui supervise le déroulement du projet et qui sera chargée à la fin de celui-ci d'évaluer le projet.

Cépage

Dans le monde du vin, on parle de cépage lorsque l'on parle d'un plant de vigne d'une variété spécifique, présent dans un endroit précis. Selon le soleil, l'humidité, le vent, …, le raisin donnera un goût spécifique et ne produira pas le même vin.

CFC

Il s'agit d'un acronyme pour les termes « Certificat fédéral de capacité ». Il s'agit du papier obtenu à la fin d'un apprentissage réussi en Suisse.

D

DataGridView

DataGridView est un contrôle de formulaire WindowsForms, au même titre qu'un bouton. Il sert à afficher des données sous forme de tableaux, où chaque ligne est une donnée différente.

Dropdown

Le dropdown est le terme anglais pour parler d'une liste déroulante. Il s'agit d'un champ contenant une liste prédéfinie dans laquelle on peut sélectionner une des valeurs, qui sera prise en compte, à la validation du formulaire par exemple.

M

MCD

Il s'agit d'un acronyme utilisé en lien avec les bases de données qui signifie modèle conceptuel de données. Son but est de représenter de manière graphique à l'aide d'entités les différents blocs d'informations et leurs liaisons.

MessageBox

Il s'agit d'une petite fenêtre pop-up contenant un message. Son utilité est souvent de faire passer un message à l'utilisateur ou de lui demander une confirmation/infirmation pour un élément spécifique.

MLD

Il s'agit d'un acronyme signifiant modèle logique de données. Son utilité est de pouvoir spécifier le type de données à fournir pour chaque élément ainsi que sa relation aux autres, sans pour autant spécifier le langage dans lequel ce sera implémenté.

P

Pop-up

Il s'agit d'une fenêtre qui apparaît sur l'écran lors de la réalisation d'une action et demande une validation avant de permettre la réalisation d'une autre action.

T

TPI

En informatique, dans le cadre d'un apprentissage, le TPI est le travail pratique individuel. Il s'agit d'un projet d'une durée définie qui permet d'évaluer les compétences professionnelles de l'étudiant.

5.3 Sources – Bibliographie

5.3.1 Pages internet consultées

Page des TPI vaudois : http://www.tpivd.ch/ [03.05.2021]
Page de référence afin de récupérer les documents d'évaluation (critères et grille)

Cépage (vin) : https://www.cavesa.ch/definition/cepage.html [03.05.2021]
Recherche sur la définition du cépage, pour commencer la mise en place du modèle de données

Assemblage (vin) : https://www.cavesa.ch/definition/assemblage.html [03.05.2021]
Recherche sur la définition de l'assemblage, dans le domaine du vin. Recherche effectuée pour savoir la cardinalité de la relation bouteilles ⇔ cépages

Assemblage (Bordeaux) : [04.05.2021]
https://www.bordeaux.com/fr/Notre-savoir-faire/La-naissance-du-vin/3-Assemblage
Recherche effectuée afin d'avoir un exemple parlant pour le glossaire, avec un vin d'assemblage connu et réputé, composé de cépages également connu

Génération de données : https://www.generatedata.com/#t1 [10.05.2021]
Afin de gagner un maximum de temps, plutôt que de générer les données à la main pour la base de données, utilisation du site pour générer des listes de données

Données pour les cépages : [11.05.2021]

http://www.vin-vigne.com/cepage/#cepages-celebres

Pour que les données utilisées fassent du sens, la liste de cépages a été prélevé de ce site pour le moment. Elle sera probablement complétée par des cépages d'autres pays par la suite.

Vérification du format de données : [12.05.2021]

https://docs.microsoft.com/en-us/dotnet/api/system.int32.tryparse?view=net-5.0

Pour ne pas faire d'erreurs, vérification dans la documentation officielle du fonctionnement du tryparse, afin de contrôler le type de la donnée saisie.

Récupération des données depuis SQL pour une liste déroulante : [17.05.2021]

https://stackoverflow.com/questions/12494634/fill-combobox-from-database

Afin de pouvoir récupérer les données depuis la base de données et donc, que les données soient toujours à jour et non pas figées, besoin de les récupérer au chargement de la page. Ce point n'a jamais été abordé en cours, donc nécessité de rechercher la méthode pour le faire.

La solution dont je me suis inspirée, parmi celles proposées, est celle de « shakur oussama », du 5 mai 2018.

La méthode s'intitule « void FillComboBox() ».

Contenance d'une bouteille de vin : [17.05.2021]

https://beaux-vins.com/quels-sont-les-noms-et-tailles-des-bouteilles-de-vin/

Pour connaître les formats classiques de bouteilles de vin, afin de ne pas mettre des valeurs aléatoires, cette recherche a été effectuée. Dans le cadre d'une cave d'un privé, il s'agit d'un choix arbitraire de limiter le volume à 3L.

Créer un exécutable à partir d'une application en C# : [17.05.2021]

https://www.campuslife.co.in/CSharp-WindowApplication/how-to-create-exe.php

Afin d'avoir une méthodologie pour créer l'exécutable de l'application, récupération d'un tutoriel expliquant la façon de faire point par point.

Ce n'est peut-être pas la meilleure source, mais les informations données sont suffisantes pour réaliser l'exécutable.

Vérification de la syntaxe exacte de requêtes SQL :[17.05.2021]

https://www.w3schools.com/sql/sql_distinct.asp#:~:text=The%20SQL%20SELECT%20DISTINCT%20Statement,the%20different%20(distinct)%20values.

Pour éviter les erreurs, consultation d'un site de référence régulièrement utilisé pour la syntaxe exacte d'une requête demandant des résultats distincts les uns des autres.

Affichage de groupbox superposées : [17.05.2021]

https://www.codeproject.com/Questions/236694/Problem-in-Controlling-Visibility-Properties-of-th

Afin de ne pas surcharger la page, les formulaires d'ajout et de retrait de bouteilles sont superposées. Hors, au moment d'afficher le formulaire de retrait, rien ne s'affiche, même avec un code fonctionnel.

Cette page a permis de se rendre compte qu'en glissant les éléments les uns sur les autres, j'ai intégré le groupe « retrait » au groupe « ajout ».

C'est la solution 2 qui a permis de résoudre l'affichage défectueux du formulaire.

Mise en place d'un tableau pour le fichier PDF à partir du dataGridView : [19.05.2021]
https://stackoverflow.com/questions/62003226/how-to-export-datagridview-as-pdf-using-itext7-note-not-using-itextsharp-in

Le code fourni donne une idée sur la mise en place du tableau. Recherche pour savoir comment faire la génération de tableaux sur PDF.

Faire fonctionner le bout de code fourni sur le site du dessus : [19.05.2021]
https://www.codeguru.com/csharp/.net/net_general/generating-a-pdf-document-using-c-.net-and-itext-7.html

Sur le site ci-dessus, il est possible de voir les différentes lignes « using », qui sont plus spécifiques que simplement « using iText ; ».
Permet de faire fonctionner les classes propres à iText7, qui est utilisé pour générer les PDF.

Choisir le paquet NuGet pour l'impression du tableau de données : [20.05.2021]
https://www.e-iceblue.com/Tutorials/Spire.PDF/Spire.PDF-Program-Guide/Print/How-to-print-PDF-document-in-C.html

e-iceblue propose un paquet NuGet gratuit, permettant d'imprimer facilement un PDF. Même si c'est limitant, cela sera considéré comme suffisant ici.

Explication sur le DataGridView : [26.05.2021]
https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.datagridview?view=net-5.0

Afin de pouvoir expliquer au mieux les termes spécifiques au C#, consultation de la documentation officielle pour utiliser le bon vocabulaire dans le glossaire.

Définition du MLD : [27.05.2021]
https://web.maths.unsw.edu.au/~lafaye/CCM/merise/mld.htm

Pour compléter le glossaire de manière aussi correcte que possible, recherche d'éléments d'explications claires et compréhensibles

5.3.2 Personnes consultées

J.Ithurbide, professeur du CPNV

[06.05.2021] Gestion des fenêtres de l'application

Discussion sur la façon d'implémenter la gestion des fenêtres en C#, lors de l'utilisation de WinForm.

[20.05.2021] Questions diverses

De nombreux problèmes restaient encore sans réponse, majoritairement à cause de ma difficulté à utiliser le débug. Afin de ne pas être coincé, ces diverses questions ont été posées :

- Lors de la génération du PDF, une erreur d'indexation a lieu : J.Ithurbide a pu me montrer comment cerner le problème pour trouver le bout de code faux. L'erreur était une erreur de copier-coller, une des variables « Cells » n'avaient pas été instanciées comme il faut.
- Discussion sur les méthodes possibles pour mettre en place une impression
- Discussion sur la meilleure façon de tester des méthodes qui sont des appels sur des requêtes SQL.
- Discussion sur différentes pistes pour vérifier la présence ou non d'un élément dans une base de données

F.Andolfatto, professeur du CPNV

[06.05.2021] Création d'un diagramme de classe

Afin d'éviter des erreurs de conception, une discussion sur la façon de réaliser un lien entre différentes classes dans le diagramme a eu lieu. La question principale était « Comment représenter l'expression 'soit a, soit b, soit c' dans un diagramme de classe ? », a, b et c, représentant des classes différentes.

[11.05.2021] Problème de référence dans le programme C#

Lors de la mise en place de la BDD, une erreur s'inscrivait :

> « Le nom de type ou d'espace de nom « DBConnection » est introuvable (vous manque-t-il une directive using ou une référence d'assembly ?) »

Après avoir passé l'après-midi à chercher une solution, un mail a été envoyé à F.Andolfatto afin de ne pas perdre trop de temps sur cette erreur.

La référence manquante m'a été indiqué et le souci a été résolu en quelques minutes le mardi matin 9h50.

[20.05.2021] Aide pour du débug

Lors de l'implémentation des alertes, il y avait des erreurs de partout et impossible de trouver leur provenance. L'aide de F.Andolfatto pour guider l'utilisation des outils de débug m'a permis de continuer à prendre en main cet outil et de corriger les erreurs.

Il s'agissait en fait d'une erreur de requête, où il manquait une condition.

[27.05.2021] Génération du .exe

N'ayant jamais eu l'occasion de générer une application en .exe à partir de Visual Studio et ayant déjà passé 50 minutes à chercher une manière de le faire le matin même, j'ai demandé à F.Andolfatto une manière simple de le faire. Elle m'a mis à disposition une marche à suivre pour générer le .exe de l'application, ce qui m'a permis de le faire par moi-même.

X.Carrel, professeur du CPNV

[10.05.2021] duplication de WinForm

Afin d'éviter de devoir reproduire depuis 0 le visuel d'une page de gestion, demande à X.Carrel sur la possibilité de le faire et encadrement pour le réaliser une première fois correctement. La méthode est désormais connue et peut être réutilisée par la suite, sans problèmes.

[17.05.2021] code en « chenis »

En regardant à nouveau le code une fois qu'il fonctionnait, l'impression que le code était « en bordel » ressortait. La demande à X.Carrel était donc, « est-ce normal d'avoir un code dans cet état » et « y a-t-il une solution pour ne pas avoir des fonctions aussi conséquentes avec les vérification de données (if, else if, else) ».

Sa réponse a été qu'il est normal d'avoir du code conséquent, qu'un point d'amélioration pourrait être d'avoir une seule méthode pour récupérer des données et passer en paramètres le type de données souhaitées.

En lançant le code pour tester, une erreur de vérification de champ apparaissait concernant les listes déroulantes. J'avais mal mis en place le test et au lieu de vérifier l'index de l'objet de la liste déroulante sélectionnée, j'étais sur l'objet sélectionné.

5.4 Protocoles de discussion

Les différentes sections ci-dessous sont des retours, avec une mise en avant des points abordés lors des rencontres avec les experts quand ils sont passés. Dans le cas où c'est pertinent, les discussions avec le chef de projet C.Egger seront également notées.

Les discussions avec des personnes autres que les experts ou le chef de projet ne seront pas répertoriées, il est possible d'avoir des détails à ce sujet au point 5.3.2 Personnes consultées.

[03.05.2021] Lancement du TPI, avec G.Gruaz et C.Egger

Discussion concernant le TPI avec monsieur Gruaz, sur le sujet du CdC. La discussion a inclus monsieur Egger, car cela concernait un des points du CdC.

Un des points du descriptif de projet a finalement été supprimé (création de rôles – administrateur et utilisateur), car son implémentation n'était pas nommée et dans le cadre d'une application pour un privé, son existence n'est pas obligatoire.

Il s'agit d'un point qui figurera cependant dans les améliorations/évolutions possibles du projet.

[05.05.2021] Retour sur le rendu du mardi 04 mai 2021, avec C.Egger

Discussion sur la planification initiale et le rendu du mardi 04.05.2021 avec monsieur Egger, CdP. Pour lui, l'avancée est correcte, la planification initiale est complète et cohérente. Il y a eu un échange d'idées sur le modèle conceptuel de données :

- C.Egger a soulevé le point de l'entité « vignobles », comme certains producteurs ont potentiellement plusieurs vignobles.
- Un deuxième point a été la raison de l'entité « robes » et sa signification dans ce contexte.
- Le dernier point discuté a été, dans l'entité « vins », la présence de l'alerte et du message d'alerte. Les arguments amenés ont été la possibilité qu'une alerte soit utilisée pour plusieurs vins différents lors de la même occasion ou, au contraire, qu'un même vin soit prévu pour plusieurs occasions séparées, nécessitant plusieurs alertes liées à ce vin. Sa proposition était d'ajouter une entité « évènement », afin d'associer par une alerte le(s) vin(s) concerné(s).

Globalement, tout se passe bien et il faut continuer comme cela, sans trop stresser.

[27.05.2021] Rencontre avec O.Rutz, expert 2

5.5 Journal de travail

Toutes les entrées du journal de travail – trouver une manière plus propre de le faire – voir clairement le statut des tâches – explication si nombre d'heures incohérent

5.6 Manuel d’installation

Avoir un document complet et correct

5.7 Archives du projet

Contenu – format de fichier
Code source .zip
Scripts bdd .sql
Rapport de projet .pdf
Journal de travail .pdf
Planification initiale .pdf
Procédure d'installation .pdf