26/04/2021

Paola Costa

Sous la supervision de :  
C.Egger, chef de projet  
G.Gruaz, expert 1  
A.Roy, expert 2

Gestion de cave à vin

Application en C# et .Net

Chapitres sensés, selon thèmes et points principaux – structure claire – contenu des titres correct – ordre logique – conception des titres, textes et graphiques -> simplifie la lecture – informations sur toutes les pages, pas juste une ligne – saut de page utilisé à bon escient – numérotation appropriée – présentation propre

Table des matières

[1 Analyse préliminaire 3](#_Toc71644251)

[1.1 Introduction 3](#_Toc71644252)

[1.2 Objectifs 3](#_Toc71644253)

[1.3 Planification initiale 4](#_Toc71644254)

[2 Analyse / Conception 7](#_Toc71644255)

[2.1 Cadre du projet 7](#_Toc71644256)

[2.2 Concept 7](#_Toc71644257)

[2.2.1 Fonctionnalités 7](#_Toc71644258)

[2.2.2 Modèles de données 8](#_Toc71644259)

[2.2.2.1 Modèle de données conceptuel 8](#_Toc71644260)

[2.2.2.2 Modèle de données logique 9](#_Toc71644261)

[2.3 Maquettes 10](#_Toc71644262)

[2.4 Schéma de navigation 11](#_Toc71644263)

[2.5 Use Cases & Scénarii 12](#_Toc71644264)

[2.5.1 Use Cases 12](#_Toc71644265)

[2.5.2 Scénarios 13](#_Toc71644266)

[2.6 Diagrammes de classe 20](#_Toc71644267)

[2.7 Risques techniques 21](#_Toc71644268)

[2.8 Stratégie de test 21](#_Toc71644269)

[2.9 Infrastructure 22](#_Toc71644270)

[2.9.1 Matériel hardware et système d’exploitation 22](#_Toc71644271)

[2.9.2 Outils logiciels 22](#_Toc71644272)

[2.9.3 Architecture du projet 22](#_Toc71644273)

[2.10 Planification définitive 23](#_Toc71644274)

[3 Réalisation 26](#_Toc71644275)

[3.1 Dossier de réalisation 26](#_Toc71644276)

[3.1.1 Répertoires et fichiers du projet 26](#_Toc71644277)

[3.1.1.1 Répartition physique des fichiers 26](#_Toc71644278)

[3.1.1.2 Fichiers et description 26](#_Toc71644279)

[3.1.2 Produit fini 26](#_Toc71644280)

[3.2 Versions du projet 26](#_Toc71644281)

[3.3 Liste des documents fournis 26](#_Toc71644282)

[3.3.1 Programmation et scripts 26](#_Toc71644283)

[3.4 Description des tests effectués 27](#_Toc71644284)

[3.4.1 Tests unitaires 27](#_Toc71644285)

[3.4.2 Tests fonctionnels 27](#_Toc71644286)

[3.4.3 État des tests 28](#_Toc71644287)

[3.5 Problèmes rencontrés et résolution 29](#_Toc71644288)

[3.6 Erreurs restantes 29](#_Toc71644289)

[3.7 Comparaison des délais entre la planification et la réalisation 29](#_Toc71644290)

[4 Conclusions 30](#_Toc71644291)

[4.1 Atteinte des objectifs 30](#_Toc71644292)

[4.2 Maintien des délais 30](#_Toc71644293)

[4.3 Points positifs et négatifs 30](#_Toc71644294)

[4.4 Difficultés particulières 30](#_Toc71644295)

[4.5 Évolutions et améliorations 30](#_Toc71644296)

[5 Annexes 31](#_Toc71644297)

[5.1 Résumé du rapport du TPI 31](#_Toc71644298)

[5.2 Glossaire 32](#_Toc71644299)

[5.3 Sources – Bibliographie 33](#_Toc71644300)

[5.3.1 Pages internet consultées 33](#_Toc71644301)

[5.3.2 Personnes consultées 34](#_Toc71644302)

[5.4 Protocoles de discussion 35](#_Toc71644303)

[5.5 Journal de travail 35](#_Toc71644304)

[5.6 Manuel d’installation 36](#_Toc71644305)

[5.7 Archives du projet 36](#_Toc71644306)

Table des illustrations

Complète le texte – choix approprié – images visuellement lisibles, compréhensibles – illustrations pertinentes, légendées et appropriés

[Figure 1 : planification initiale partie 1 4](file:///C:\Users\paola.costa\Documents\GitHub\TPI_gestionCaveAVin\docs_avec_modification\rapport_COSTA.docx#_Toc71644307)

[Figure 2 : planification initiale partie 2 5](file:///C:\Users\paola.costa\Documents\GitHub\TPI_gestionCaveAVin\docs_avec_modification\rapport_COSTA.docx#_Toc71644308)

[Figure 3 : planification initiale partie 3 6](file:///C:\Users\paola.costa\Documents\GitHub\TPI_gestionCaveAVin\docs_avec_modification\rapport_COSTA.docx#_Toc71644309)

[Figure 4 : modèle conceptuel de données 8](file:///C:\Users\paola.costa\Documents\GitHub\TPI_gestionCaveAVin\docs_avec_modification\rapport_COSTA.docx#_Toc71644310)

[Figure 5 : modèle logique de données 9](file:///C:\Users\paola.costa\Documents\GitHub\TPI_gestionCaveAVin\docs_avec_modification\rapport_COSTA.docx#_Toc71644311)

[Figure 6 : maquette -> page d'accueil de l'application 10](file:///C:\Users\paola.costa\Documents\GitHub\TPI_gestionCaveAVin\docs_avec_modification\rapport_COSTA.docx#_Toc71644312)

[Figure 7 : maquette -> page d'ajout de bouteille(s) 10](file:///C:\Users\paola.costa\Documents\GitHub\TPI_gestionCaveAVin\docs_avec_modification\rapport_COSTA.docx#_Toc71644313)

[Figure 8 : schéma de navigation 11](file:///C:\Users\paola.costa\Documents\GitHub\TPI_gestionCaveAVin\docs_avec_modification\rapport_COSTA.docx#_Toc71644314)

[Figure 9 : diagramme des Use Case 12](file:///C:\Users\paola.costa\Documents\GitHub\TPI_gestionCaveAVin\docs_avec_modification\rapport_COSTA.docx#_Toc71644315)

[Figure 10 : diagramme de classe 20](file:///C:\Users\paola.costa\Documents\GitHub\TPI_gestionCaveAVin\docs_avec_modification\rapport_COSTA.docx#_Toc71644316)

[Figure 11 : planification définitive partie 1 23](file:///C:\Users\paola.costa\Documents\GitHub\TPI_gestionCaveAVin\docs_avec_modification\rapport_COSTA.docx#_Toc71644317)

[Figure 12 : planification définitive partie 2 24](file:///C:\Users\paola.costa\Documents\GitHub\TPI_gestionCaveAVin\docs_avec_modification\rapport_COSTA.docx#_Toc71644318)

[Figure 13 : planification définitive partie 3 25](file:///C:\Users\paola.costa\Documents\GitHub\TPI_gestionCaveAVin\docs_avec_modification\rapport_COSTA.docx#_Toc71644319)

[Figure 14 : problème rencontré -> référence de projet 29](file:///C:\Users\paola.costa\Documents\GitHub\TPI_gestionCaveAVin\docs_avec_modification\rapport_COSTA.docx#_Toc71644320)

# Analyse préliminaire

## Introduction

Lors d’un repas de famille ou avec des amis, il est toujours plaisant d’avoir une bonne bouteille de vin de sorti. 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.

## 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.

## 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.

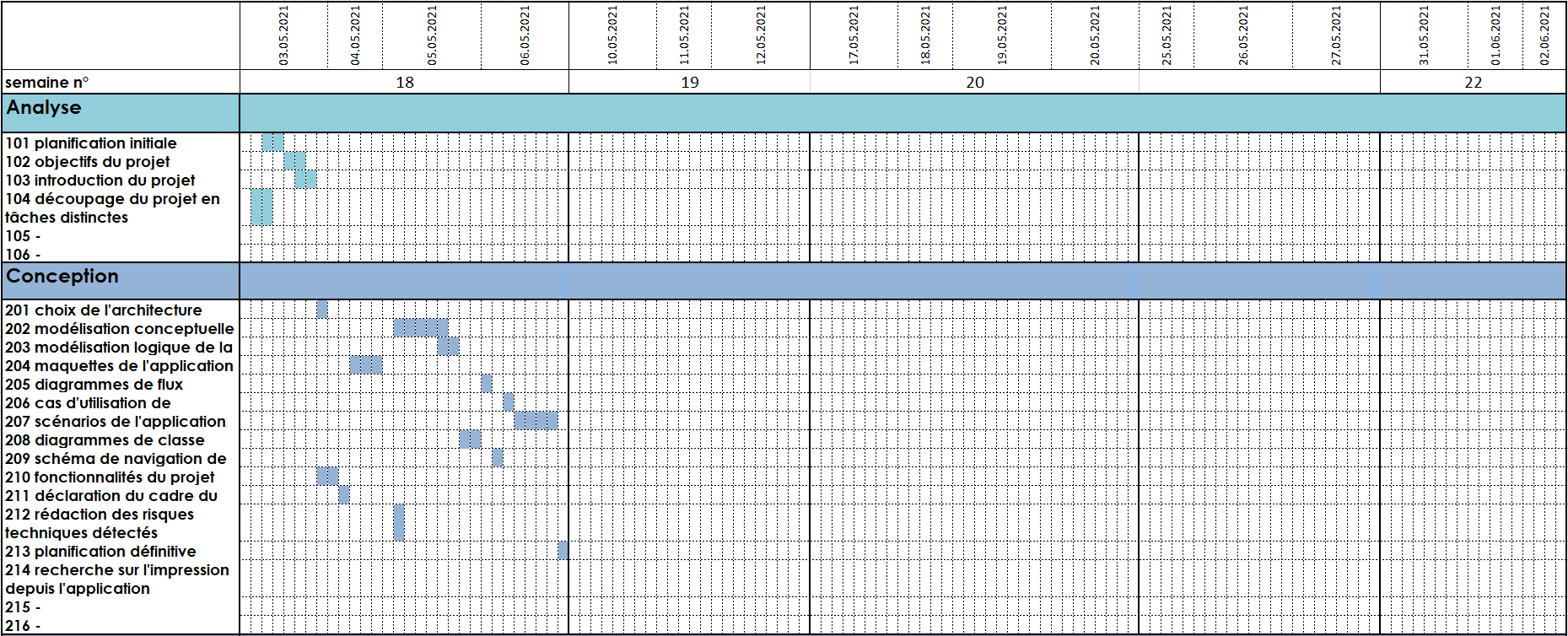


Figure 1 : planification initiale partie 1

Sur cette figure ci-dessous, 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 pour pouvoir tester intégralement l’application et dépanner le moindre souci par la suite. Prévoir l’implémentation avec autant de délai à la fin permet également d’avoir un petit peu de marge si un problème venait à arriver et à repousser le développement de l’application de quelques heures/jours. Le reste du temps sera consacré à la rédaction de la documentation demandée.

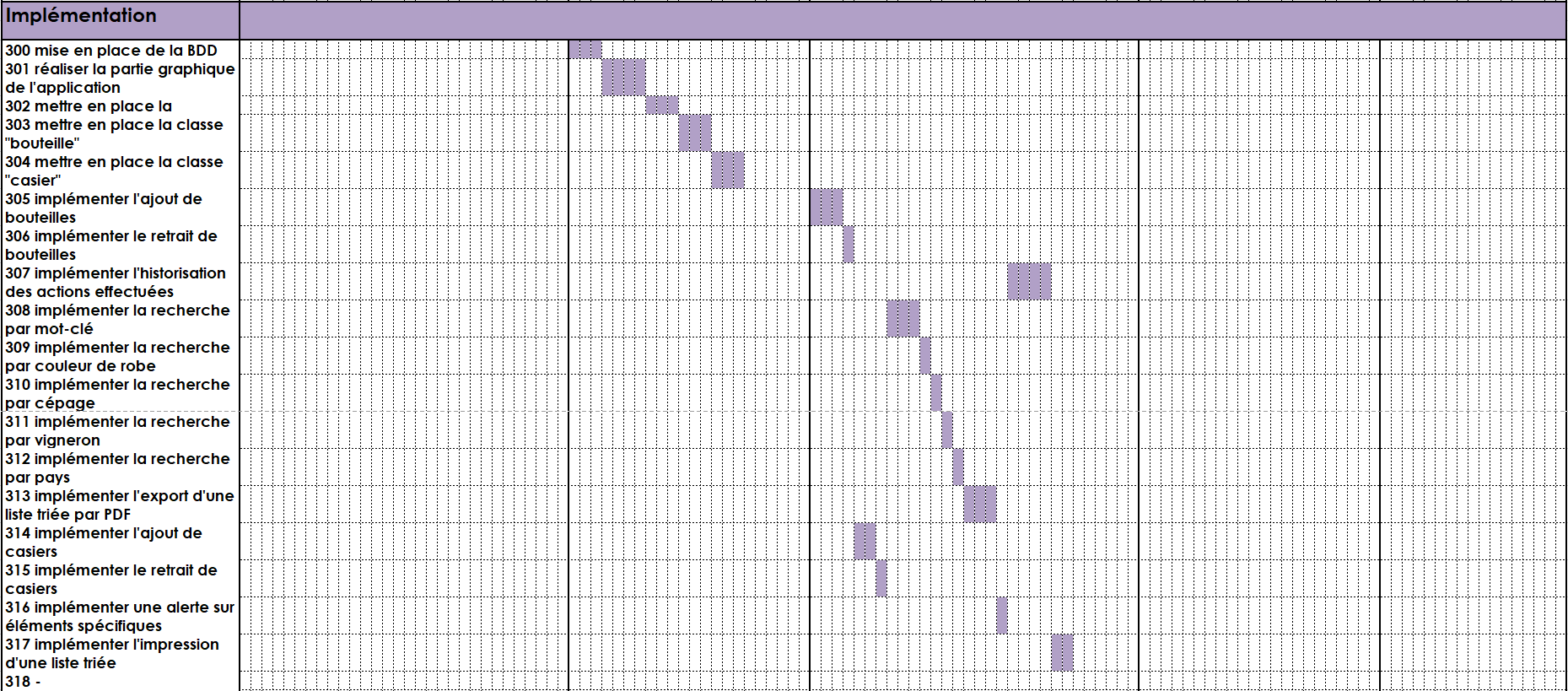


Figure 2 : planification initiale partie 2

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.

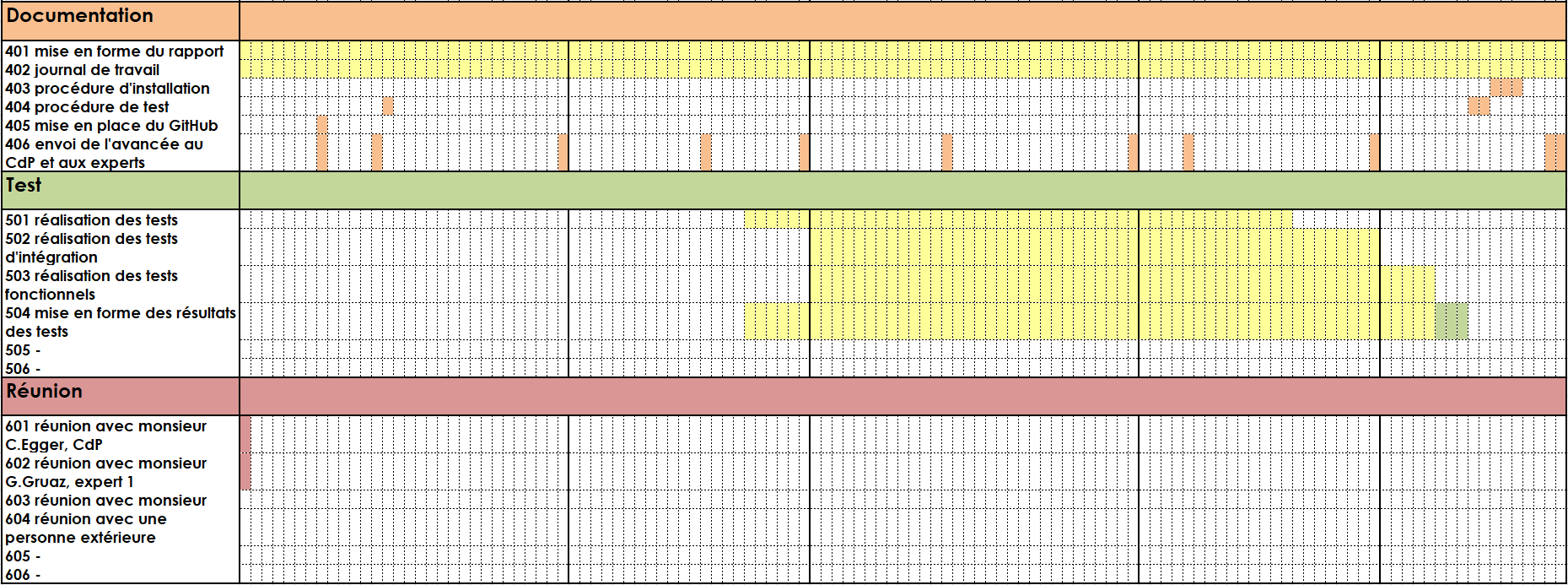


Figure 3 : planification initiale partie 3

# Analyse / Conception

## 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](#_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.

## Concept

### Fonctionnalités

En plus des différents éléments abordés dans le point [Objectifs](#_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](#_Use_Cases_&). Le fonctionnement exact de chaque point abordé ci-dessus y est développé.

### Modèles de données

#### 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.

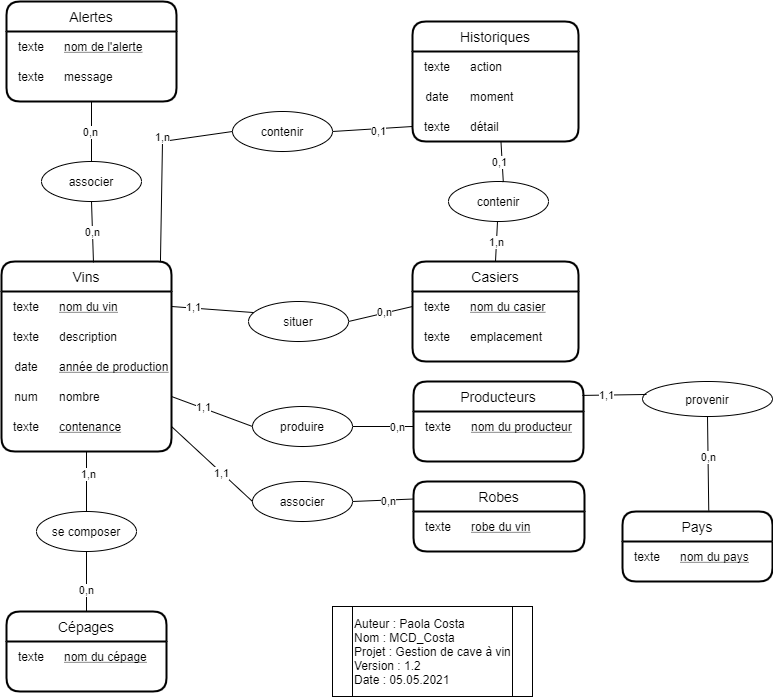


Figure 4 : modèle conceptuel de données

#### Modèle de données logique

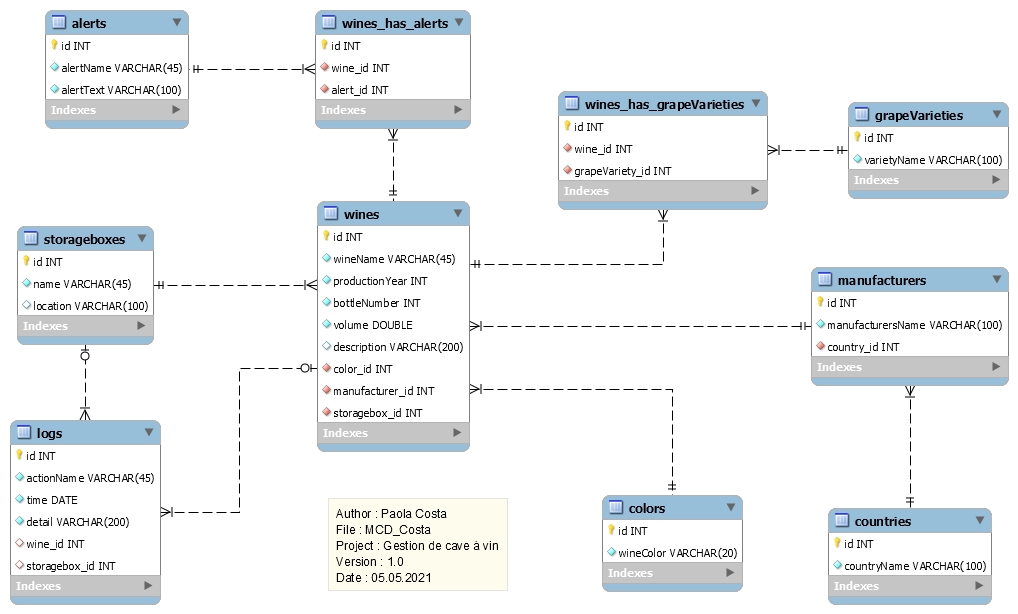


Figure 5 : modèle logique de données

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.
* 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 dans l’application. Cela peut concerner les bouteilles, les casiers ou les alertes.
* storageboxes : il s’agit des casiers dans lesquels sont stockés les bouteilles.
* alerts : il s’agit des alertes, associées aux bouteilles de vin
* 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, producteur, volume.

## 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.

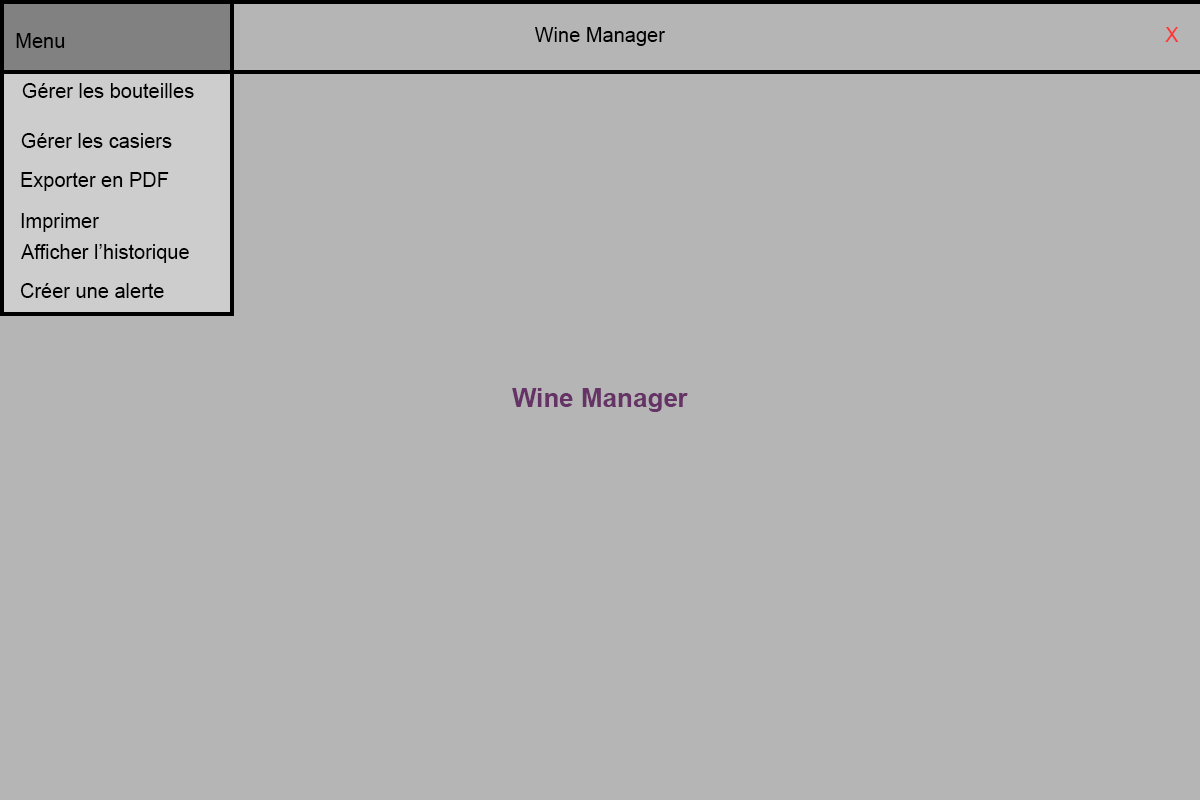


Figure 6 : maquette -> page d'accueil de l'application

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.

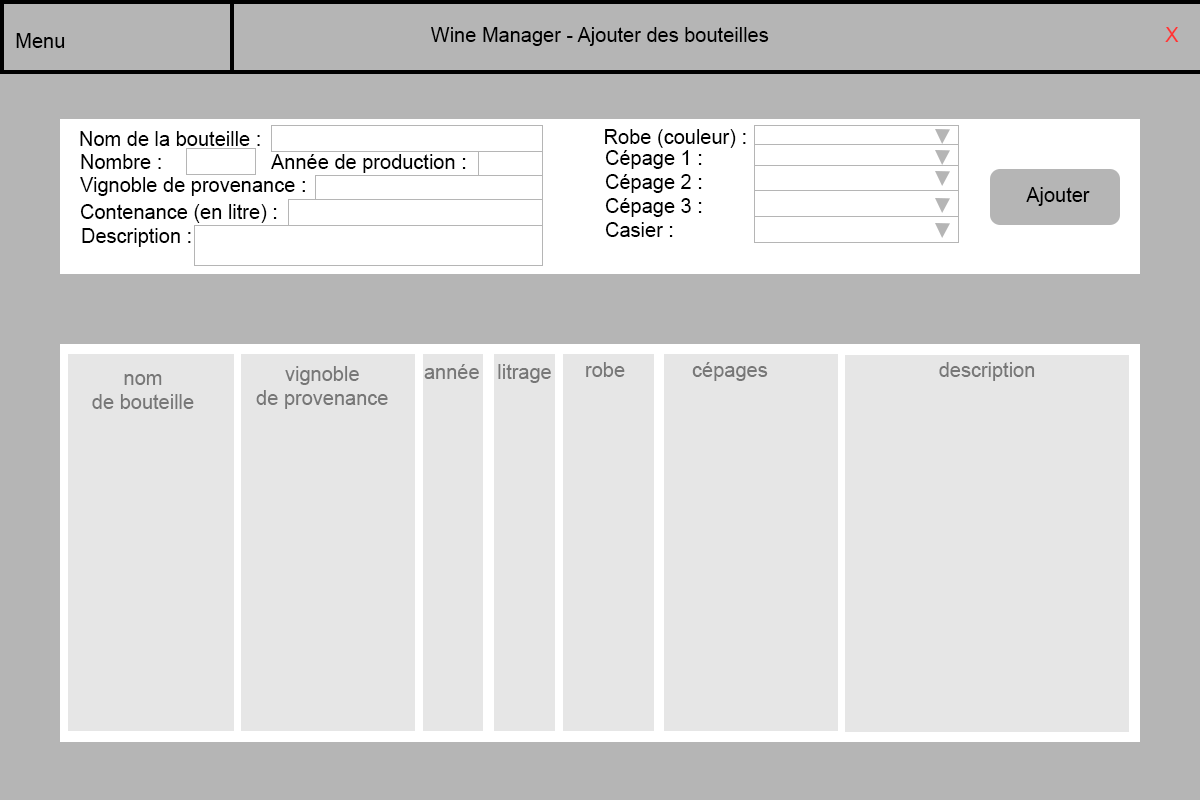


Figure 7 : maquette -> page d'ajout de bouteille(s)

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.

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.

## 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](#_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.

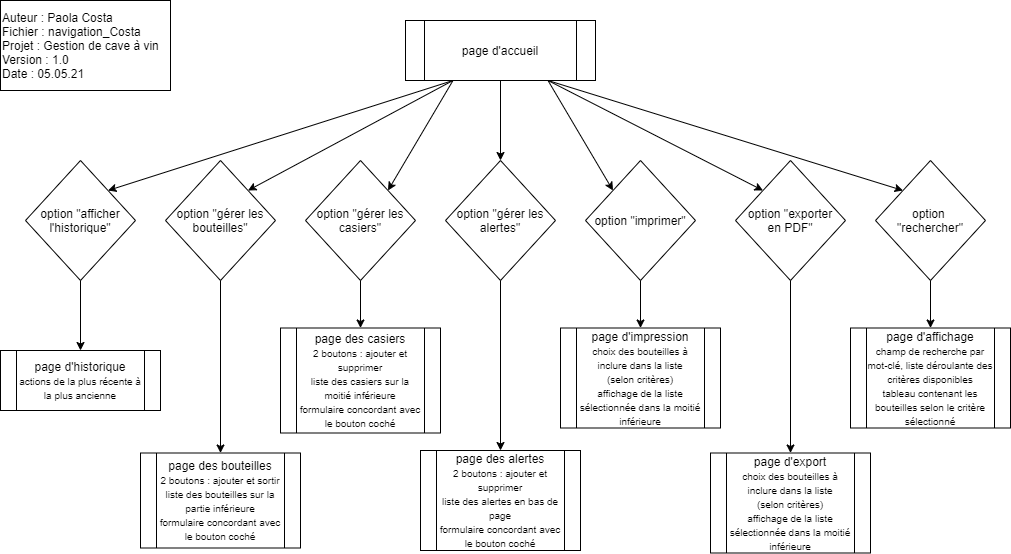


Figure 8 : schéma de navigation

## Use Cases & Scénarii

### Use Cases

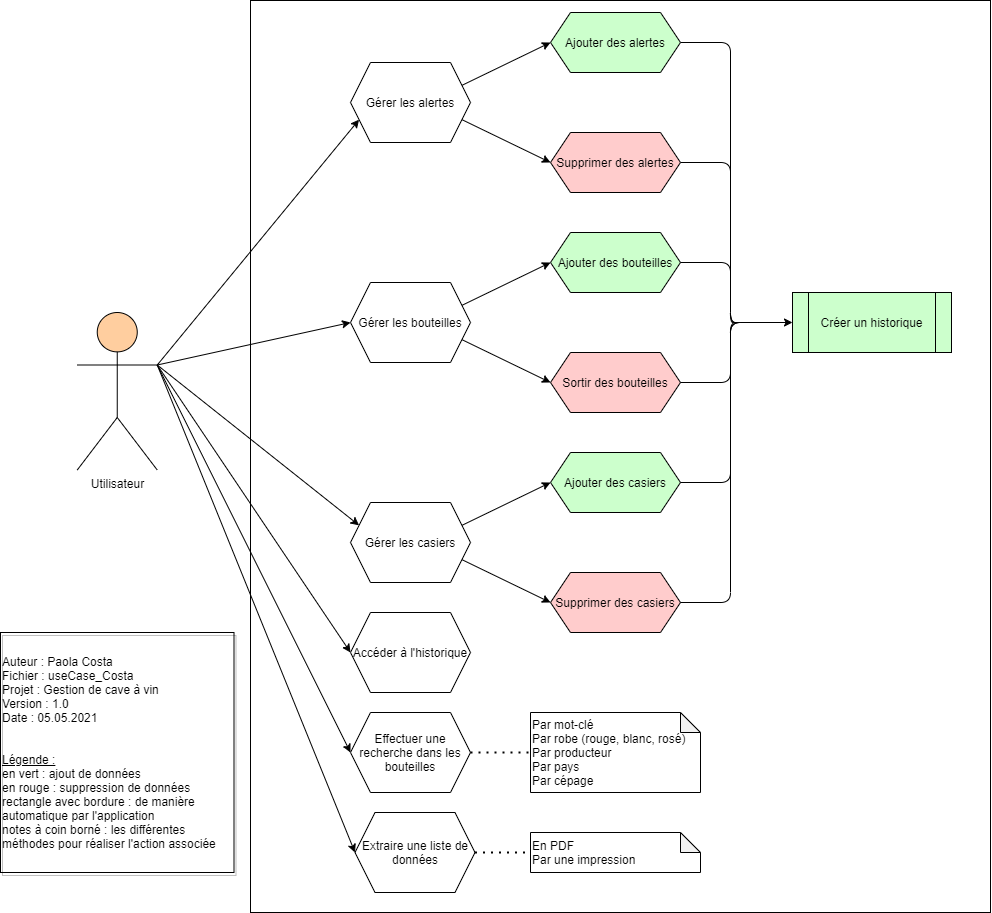


Figure 9 : diagramme des Use Case

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.

### 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 |
|  | Les champs nécessaires sont remplis correctement | Un message de confirmation s’affiche, avec les paramètres entrés notés |
| L’utilisateur annule l’ajout de bouteille(s) |  | Retour à la page des bouteilles, le formulaire est vide. |
| L’utilisateur confirme l’ajout de bouteille(s) |  | Retour à la page des bouteilles, avec une actualisation de la liste des bouteilles, en-dessous du formulaire vidé. |

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 | Un message de confirmation s’affiche, avec les paramètres entrés notés |
| L’utilisateur annule la sortie de bouteilles |  | Retour à la page des bouteilles, le formulaire est vide. |
| L’utilisateur confirme la sortie de bouteilles |  | Retour à la page des bouteilles, avec une actualisation de la liste des bouteilles, en-dessous du formulaire vidé. |

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 | Un message de confirmation s’affiche, avec les paramètres entrés notés |
| L’utilisateur annule l’ajout de casier |  | Retour à la page des casiers, le formulaire est vide. |
| L’utilisateur confirme l’ajout de casier |  | Retour à la page des casiers, avec une actualisation de la liste des casiers, en-dessous du formulaire vidé. |

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 | Un message de confirmation de suppression s’affiche, avec le nom et l’emplacement |
| L’utilisateur annule la suppression du casier |  | Retour à la page des casiers, le formulaire est vide. |
| L’utilisateur confirme la suppression du casier |  | Retour à la page des casiers, avec une actualisation de la liste des casiers, en-dessous du formulaire vidé. |

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 | Un message de confirmation s’affiche, avec les paramètres entrés notés |
| L’utilisateur annule l’ajout d’alerte |  | Retour à la page des alertes, le formulaire est vide. |
| L’utilisateur confirme l’ajout d’alerte |  | Retour à la page des alertes, avec une actualisation de la liste des alertes, en-dessous du formulaire vidé. |

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 |  | Un message de confirmation de suppression s’affiche, avec le nom et l’emplacement |
| L’utilisateur annule la suppression de l’alerte |  | Retour à la page des alertes, le formulaire est vide. |
| L’utilisateur confirme la suppression de l’alerte |  | Retour à la page des alertes, avec une actualisation de la liste des alertes, en-dessous du formulaire vidé. |

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 ou la description d’au moins 1 bouteille | Le tableau s’actualise et affiche les bouteilles correspondant à la recherche. Le champ texte de recherche est vidé. |

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

|  |  |  |
| --- | --- | --- |
| Action | Condition | Réaction |
| La personne sélectionne l’option « Exporter au format PDF » dans le menu. |  | La page qui s’affiche est la page des recherches. Un pop-up avec le critère à sélectionner s’affiche . |
| La personne sélectionne le critère de tri souhaité et valide. |  | La liste correspondante au critère s’affiche à l’écran. |
| La personne clique sur « valider le PDF ». |  | Un fichier PDF se crée sur le bureau de l’ordinateur. |

Scénario 7 : Imprimer

|  |  |  |
| --- | --- | --- |
| Action | Condition | Réaction |
| La personne sélectionne l’option « Imprimer » dans le menu. |  | La page qui s’affiche est la page des alertes. |
| La personne sélectionne le critère de tri souhaité et valide. |  | La liste correspondante au critère s’affiche à l’écran. |
| La personne clique sur « valider l’impression ». |  | Une fenêtre s’ouvre, pour effectuer la sélection d’imprimante. |
| La personne sélectionne les paramètres voulus et accepte l’impression. |  | L’impression a lieu, retour à la page d’accueil de l’application. |

## Diagrammes de classe

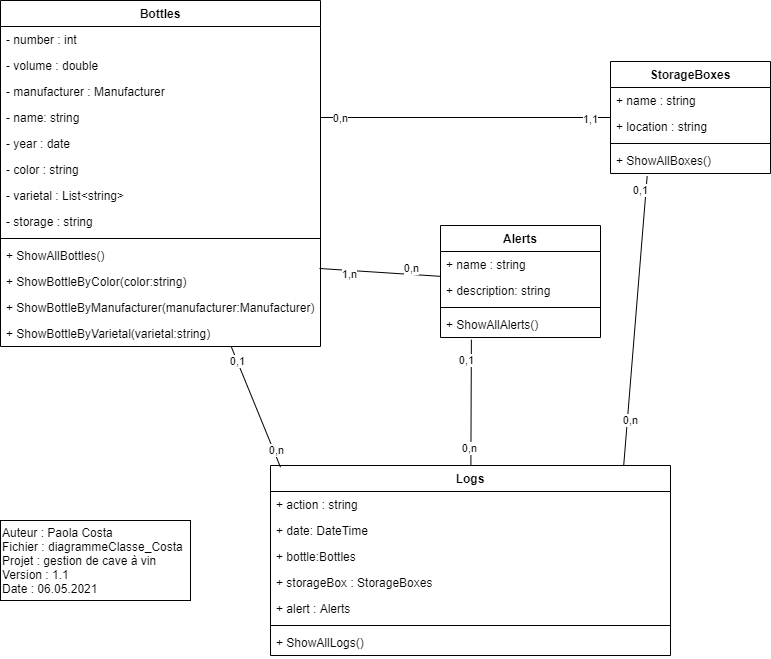


Figure 10 : diagramme de classe

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.

## 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.

## 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](#_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.

## Infrastructure

### 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.

### 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)

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.

### 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.

## Planification définitive

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*

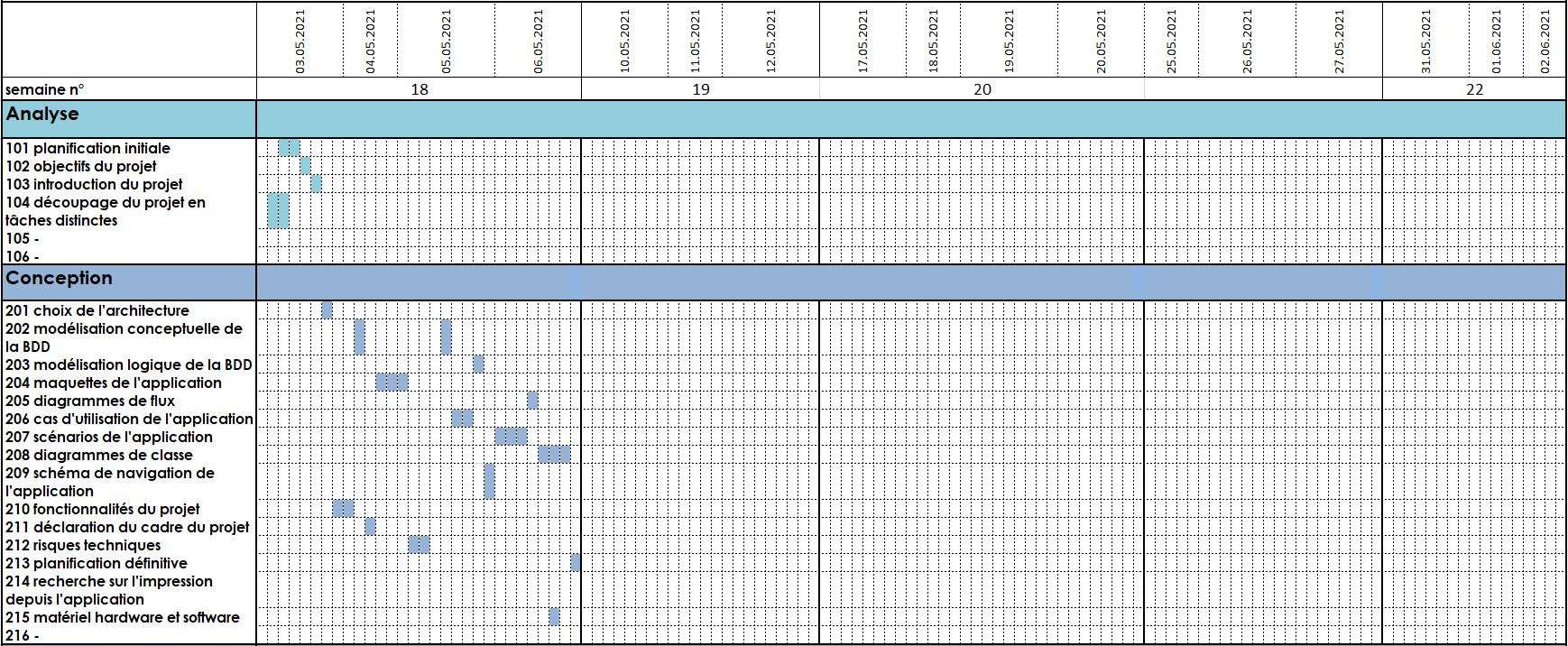


Figure 11 : planification définitive partie 1

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



Figure 12 : planification définitive partie 2

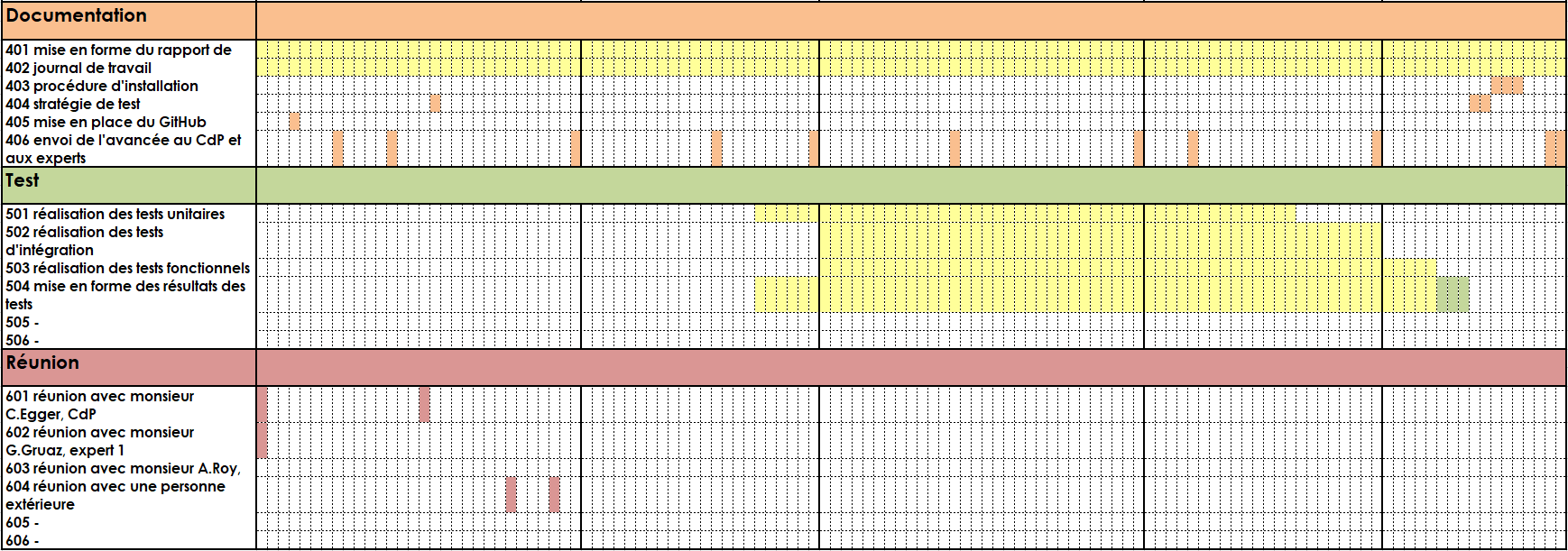


Figure 13 : planification définitive partie 3

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

# Réalisation

## Dossier de réalisation

### Répertoires et fichiers du projet

#### Répartition physique des fichiers

Explication globale sur la répartition des fichiers de projet – potentiellement inclure une explication sur la répartition des fichiers dans GitHub

#### Fichiers et description

Description des différents fichiers, avec leurs rôles

### Produit fini

Version du produit fini – état de complétion – logiciel + version nécessaire à l’utilisation

## Versions du projet

Date – version – contenu (avancée du développement) – emplacement -> github, en non-modifiable – garder le numéro de commit pour pouvoir revenir à cette version si souhaitée

## Liste des documents fournis

Liste exhaustive des documents pour le client, avec lien sur l’annexe + version

### 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

## Description des tests effectués

Exhaustif – au fur et à mesure – même si logique, le noter, avec capture d’écran pour justifier – légender les captures d’écran

### Tests unitaires

### 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.

### 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](#_Scénarios) et [Tests](#_Tests_unitaires).

|  |  |
| --- | --- |
| Nom du test | État |
| Scénario 1a : ajout de bouteilles |  |
| Scénario 1b : retrait de bouteilles |  |
| Scénario 2a : ajout de casiers |  |
| Scénario 2b : suppression de casiers |  |
| Scénario 3a : ajout d’alerte |  |
| Scénario 3b : suppression d’alerte |  |
| Scénario 4a : recherche par mot-clé |  |
| Scénario 4b : filtre par critère |  |
| Scénario 5 : afficher l’historique |  |
| Scénario 6 : imprimer |  |
| Scénario 7 : exporter en PDF |  |
| Test 1 : insertion de données par script dans la BDD |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## Problèmes rencontrés et résolution

Quand – quoi – pourquoi – comment le résoudre – aide ?

[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.

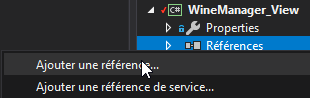


Figure 14 : problème rencontré -> référence de projet

[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.

## Erreurs restantes

Exhaustif – quoi – pourquoi – résultante – solution possible

## Comparaison des délais entre la planification et la réalisation

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

# Conclusions

## Atteinte des objectifs

Validation (ou non) de l’atteinte des objectifs

## Maintien des délais

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

## 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

## Difficultés particulières

Eléments qui ont particulièrement embêté (retard, …)

## 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

Page d’historique – afficher l’historique, à choix : des bouteilles, des casiers ou des alertes

Imprimer/exporter les logs

…

# Annexes

## 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

## Glossaire

Alphabétiquement trié – contient les termes spécifiques au TPI

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

---

C

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.

M

MCD

…

MLD

…

T

TPI

…

## Sources – Bibliographie

IMPORTANT – exhaustif – mieux vaut trop que pas assez – date, page, raison (référencé point positif/négatif ?) – choix judicieux, réfléchi (pas n’importe quel site)

### Pages internet consultées

<https://www.nuget.org/packages/itext7/7.1.15>

<https://www.codeguru.com/csharp/.net/net_general/generating-a-pdf-document-using-c-.net-and-itext-7.html>

Page des TPI vaudois : <http://www.tpivd.ch/>

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) : <https://www.bordeaux.com/fr/Notre-savoir-faire/La-naissance-du-vin/3-Assemblage> [04.05.2021]

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 :

<http://www.vin-vigne.com/cepage/#cepages-celebres> [11.05.2021]

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.

### Personnes consultées

Qui – pourquoi – quand

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.

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 a 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.

X.Carrel, professeur du CPNV

* [10.04.2021] 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.

## Protocoles de discussion

Retour rapide sur les discussions – date – qui – ce qui a été dit – améliorations à apporter – modifications à faire – éléments qui correspondent à ce qui est demandé – court et concis, mais complet

[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.

[06.05.2021] Discussion sur la gestion des fenêtres multiples en C#, avec J.Ithurbide

\*\*fenêtres imbriquées : trop compliqué, boutons partout : vieillot -> solution suggérée : masquer la fenêtre de base à l’ouverture d’une autre fenêtre, refermer la fenêtre lorsque l’action est complétée\*\*

## 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

## Manuel d’installation

Avoir un document complet et correct

## Archives du projet

Contenu – format de fichier