|  |
| --- |
| *QXPhotoViewer* |



Rossier, Quentin

Les Autins 14

1584, Villars-Le-Grand

quentin.rossier@cpnv.ch

Xavier Schwab

Route d’Oron 40

1010, Lausanne

Xavier.Schwab@cpnv.ch

SI-T1a

19.12.2017



Table des matières

[Analyse préliminaire 5](#_Toc501466464)

[Introduction 5](#_Toc501466465)

[Organisation 5](#_Toc501466466)

[Objectifs 5](#_Toc501466467)

[Planification initiale 5](#_Toc501466468)

[Analyse 7](#_Toc501466470)

[Cahier des charges détaillé 7](#_Toc501466471)

[Définition de l’audience 8](#_Toc501466472)

[Analyse concurrentielle 8](#_Toc501466473)

[Cas d’utilisation 8](#_Toc501466474)

[Maquettes fonctionnelles 12](#_Toc501466475)

[Etude de faisabilité 13](#_Toc501466476)

[Nom de l’application 13](#_Toc501466477)

[Stratégie de test 14](#_Toc501466478)

[Budget initial 14](#_Toc501466479)

[Planification détaillée 15](#_Toc501466480)

[Conception 17](#_Toc501466481)

[Analyse de l’environnement 17](#_Toc501466482)

[Architecture du programme 17](#_Toc501466483)

[Définition de la charte graphique 18](#_Toc501466484)

[Maquette graphique 18](#_Toc501466485)

[Conception du code 19](#_Toc501466486)

[Réalisation 20](#_Toc501466487)

[Dossier de réalisation 20](#_Toc501466488)

[Description de tests effectués 21](#_Toc501466489)

[Erreurs restantes 21](#_Toc501466490)

[Dossier d’archivage 22](#_Toc501466491)

[Problèmes rencontrés 22](#_Toc501466492)

[Mise en service 22](#_Toc501466493)

[Conclusions 24](#_Toc501466494)

[Objectifs 24](#_Toc501466495)

[Points positifs et négatifs 24](#_Toc501466496)

[Difficultés particulières 24](#_Toc501466497)

[Suite et améliorations possible 24](#_Toc501466498)

[Annexes 26](#_Toc501466499)

[Sources – bibliographie 26](#_Toc501466500)

[Archives du projet 26](#_Toc501466501)

# Analyse préliminaire

## Introduction

Le client nous à demander à nous, TeknoQaX.SA, une entreprise constituée de deux jeunes technicien en développement, de réaliser une application de gestion de photo et de ses métadonnées. Elle devra pouvoir gérer une liste de tags personnalisable par l'utilisateur, pouvoir rechercher des photos précises. Classer et renommer les photos. Une démo du logiciel sera faite une fois le projet rendu. Ainsi qu'une présentation devant le client.

## Organisation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Chef de projet 1 :** | Nom : | **Schwab** | | Prénom : | Xavier |
| [Xavier.SCHWAB@cpnv.ch](mailto:Xavier.SCHWAB@cpnv.ch) | | | 079 382 22 86 | |
| **Chef de projet 2 :** | Nom : | **Rossier** | | Prénom : | Quentin |
| [Quentin.ROSSIER@cpnv.ch](mailto:Quentin.ROSSIER@cpnv.ch) | | | 079 860 95 90 | |
| **Mandataire :** | Nom : | CHEVILLAT | Prénom : | | Jerome |
| [Jerome.CHEVILLAT@cpnv.ch](mailto:Jerome.CHEVILLAT@cpnv.ch) | | 024/55 + 76079 | | |

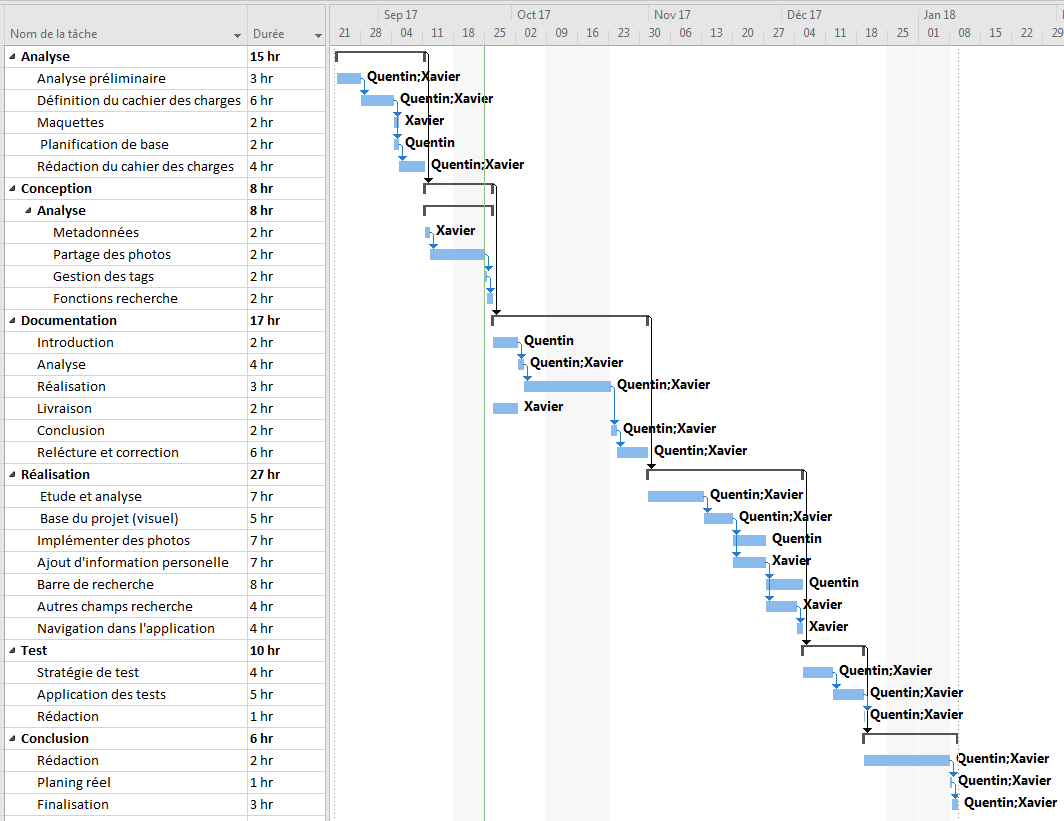
## Objectifs

Voici la liste des objectifs que le projet doit remplir afin de satisfaire notre client :

L’application dois pouvoir :

* Normaliser les noms de photos entre les différents appareilles
* Automatisée le nommage des photos
* Classer les photos par ordre chronologique
* Ajouter des informations personnelles aux photos et aux dossiers
* Visionner les photos sur différents appareils
* Faire des recherches pour trouver des photos
* Garder les informations ajoutées lors de changement d’application.
* L’accès au visionnage des photos dois pouvoir se faire depuis n’importe quel appareilles et système d’exploitation.

Planification initiale



**Grandes dates du projet**

* Début du projet 22.08.2017
* Présentation d’une partie fonctionnelle du projet 18.09.2017
* Remise de la réalisation et de la documentation 19.12.2017
* Fin du projet Semaine du 08.01.2017

# Analyse

## Cahier des charges détaillé

**Normaliser le nom des photos et les classer**

A chaque ajout de photo, l’application devras normaliser les noms des photos afin de s’y retrouver. L’application prendra la date de la prise de photo et un champ que l’utilisateur devras remplir, pour ensuite crée un nom global pour toutes les photos ajoutées pour qu’elles puissent être classées dans un ordre chronologique avec un nom clair.

**Informations personnelles :**

L’utilisateur doit pouvoir ajouter des informations personnelles sur les photos :

* Des lieux
* Des personnes
* Des tags
* Des favoris

Ainsi que d’autre informations sur les dossiers créent :

* Des commentaires

**Recherches :**

L’utilisateur dois pouvoir trouver une ou plusieurs photos et/ou un dossier précis à l’aide d’un champ de recherche et de plusieurs filtres :

* Date
* Favoris
* Personnes
* Tags
* Format de photo

**Stockage :**

Les données ajoutées par l’utilisateur doivent être stockées de tel manières à ce qu’on puisse les réutiliser dans un autre projet

**Visibilité sur d’autres appareils :**

Les dossiers et photos doivent être visible sur d’autre appareil connecté au même réseau local. Les informations ajoutées par l’utilisateur ne doivent pas forcément être visible, mais le classement des fichiers doit être respectés lors du visionnage des photos.

## Définition de l’audience

L’application servira au commanditaire de la solution, à sa famille et à ses proches qui pourront visionner les photos.

Il est possible que par la suite, le projet puisse être vendu comme application avec des modifications apportées pour le grand publique.

## Analyse concurrentielle

Picasa, dont le client à parler, n’est à présent plus mis à jour. Et a été remplacé par « Google photos ». Cette solution ne convient pas car, les photos sont stockées sur le cloud. Chose que le client ne souhaite pas.

SunlitGreen Photo Manager est un logiciel qui ne répond que partiellement au désire du client. Il est impossible de mettre les photos en favoris, de les rechercher autrement que par leurs titres ainsi que de les renommer automatiquement

## Cas d’utilisation

**Use cases**

On utilise ce programme de gestion de photos pour :

* Classer les photos et les dossiers par ordre chronologique
* Ajouter automatiquement la date au nom des photos
* Associer des informations personnelles à des dossiers et des photos
* Rechercher des photos à l’aide de recherche par mots-clefs et des filtres activable
* Visionner les photos sur un appareil externe
* Conserver les tags des photos dans le temps

**Scénarios**

|  |  |
| --- | --- |
| Identifiant | Scenario\_AffichePhoto |
| En tant que | Utilisateur |
| Je veux | Afficher les photos d’un dossier |
| Pour | Les voir dans l’ordre chronologique |
| Charge estimée | 4h |
| Priorité | Elevée |

|  |  |
| --- | --- |
| **Action** | **Réaction** |
| J'accède à l'application | La page d'accueil s'affiche |
| Je clic sur un dossier choisis | Les photos contenu dans le dossier sont affichées par ordre chronologique |

|  |  |
| --- | --- |
| Identifiant | Scenario\_AjoutPhoto |
| En tant que | Utilisateur |
| Je veux | Ajouter des photos |
| Pour | Les ajouter dans l’ordre chronologique |
| Charge estimée | 8h |
| Priorité | Elevée |

|  |  |
| --- | --- |
| **Action** | **Réaction** |
| J'accède à l'application | La page d'accueil s'affiche |
| Je double clic sur le dossier voulu | Les photos du dossier d’affichent |
| Je clic sur le bouton "ajouter des photos" | L'explorateur de fichier s'ouvre |
| Je sélectionne les photos voulues | Les photos sont ajoutés et s’affichent par ordre chronologique dans l'application |
| Je remplis le nom de base des photos | Les photos se renomment avec le nom de base ajouté et la date. |

|  |  |
| --- | --- |
| Identifiant | Scenario\_AficheTag |
| En tant que | Utilisateur |
| Je veux | Afficher les informations personnelles d’une photo ou d’un dossier |
| Pour | Consulter les tags |
| Charge estimée | 8h |
| Priorité | Elevée |

|  |  |
| --- | --- |
| **Action** | **Réaction** |
| J'accède à l'application | La page d'accueil s'affiche |
| Je clic sur une photo ou un dossier | Un menu contenant les informations personnelles s'affiche |

|  |  |
| --- | --- |
| Identifiant | Scenario\_ModifTag |
| En tant que | Utilisateur |
| Je veux | Modifier les informations personnelles d’une photo ou d’un dossier |
| Pour | Mettre à jour les tags |
| Charge estimée | 8h |
| Priorité | Elevée |

|  |  |
| --- | --- |
| **Action** | **Réaction** |
| J'accède à l'application | La page d'accueil s'affiche |
| Je clic sur une photo ou un dossier | Un menu contenant les informations personnelles s'affiche |
| Je remplie les tags et je clic sur le bouton "sauvegarder" situé en bas de la liste de tags | Les tags de la photo ou du dossier ont été enregistré |

|  |  |
| --- | --- |
| Identifiant | Scenario\_Triage |
| En tant que | Utilisateur |
| Je veux | Trier des photos |
| Pour | Afficher les photos voulue |
| Charge estimée | 8h |
| Priorité | Elevée |

|  |  |
| --- | --- |
| **Action** | **Réaction** |
| J'accède à l'application | La page d'accueil s'affiche |
| Je clic le bouton « filtre » | Une liste de filtre s’affiche |
| Je sélectionne les filtres souhaité | Un premier filtrage des photos est effectué |
| Je commence à taper la recherche voulue dans la barre de recherche | La barre de recherche propose des choix selon ce qui a été tapé |
| Une fois que j’ai fini de taper la recherche voulue, j’appuie sur la présélection du choix écrit | Les photos correspondant au filtres sélectionnées et à la recherche tapé s’affichent |

|  |  |
| --- | --- |
| Identifiant | Scenario\_AfficheExterne |
| En tant que | Utilisateur |
| Je veux | Voir des photos depuis un téléphone |
| Pour | Visionner des photos sur un appareil externe |
| Charge estimée | 10h |
| Priorité | élevée |

|  |  |
| --- | --- |
| **Action** | **Réaction** |
| J'accède à l'application avec une connexion au réseau de la maison | Les dossiers de photos s’affichent. |
| Je sélectionne un dossier pour voir ses photos | Les photos du dossier s’affichent par ordre chronologique. |

|  |  |
| --- | --- |
| Identifiant | Scenario\_Sauvegarde |
| En tant que | Utilisateur |
| Je veux | Changer d’application sans perdre tout le travail effectué |
| Pour | Eviter de devoir refaire toute les notations personnelles sur les photos. |
| Charge estimée | 10h |
| Priorité | Elevée |

Les informations personnelles des photos seront conservées dans les Meta données. De ce fait, n’importe quelle application capable de lire ses tags pourront récupérer et utiliser les informations de la photo.

|  |  |
| --- | --- |
| Identifiant | Scenario\_Recherche |
| En tant que | Utilisateur |
| Je veux | Faire une recherche via le champs de recherche avec un résultats correspondant |
| Pour | Trouver une photos, un dossier précis |
| Charge estimée | 6h |
| Priorité | Moyenne plus |

|  |  |
| --- | --- |
| **Action** | **Réaction** |
| J'accède à l'application | La page d'accueil s'affiche |
| Je place ma souris sur la barre de recherche et je clic | Le curseur clignote dans la barre de recherche |
| J’écris le nom de la photo recherchée | Propose des solutions au fur et à mesure de la frappe. |
| Clic ou presse « Entrée » sur le bon champ voulu ou dans la barre de recherche | Affiche le résultat de la recherche (photo ou dossier) |

|  |  |
| --- | --- |
| Identifiant | Scenario\_ErrorRecherche |
| En tant que | Utilisateur |
| Je veux | Faire une recherche via le champs de recherche sans résultats correspondant |
| Pour | Trouver une photos, un dossier précis sans trouver. |
| Charge estimée | Identique au Scénario\_001 |
| Priorité | Identique au Scénario\_001 |

|  |  |
| --- | --- |
| **Action** | **Réaction** |
| J'accède à l'application | La page d'accueil s'affiche |
| Je place ma souris sur la barre de recherche et je clic | Le curseur clignote dans la barre de recherche |
| J’écris le nom de la photo recherchée | Propose des solutions au fur et à mesure de la frappe. Puis plus aucune présélection |
| Presse « entrée » dans la barre de recherche | Affiche un message prévenant qu’il n’y a pas de résultats pour cette recherche. |

|  |  |
| --- | --- |
| Identifiant | Scenario\_Filtrage |
| En tant que | Utilisateur |
| Je veux | Faire une recherche via un ou des filtres |
| Pour | Trouver des photos, des dossiers, une personne précise, des favoris. |
| Charge estimée | 4h |
| Priorité | Moyenne |

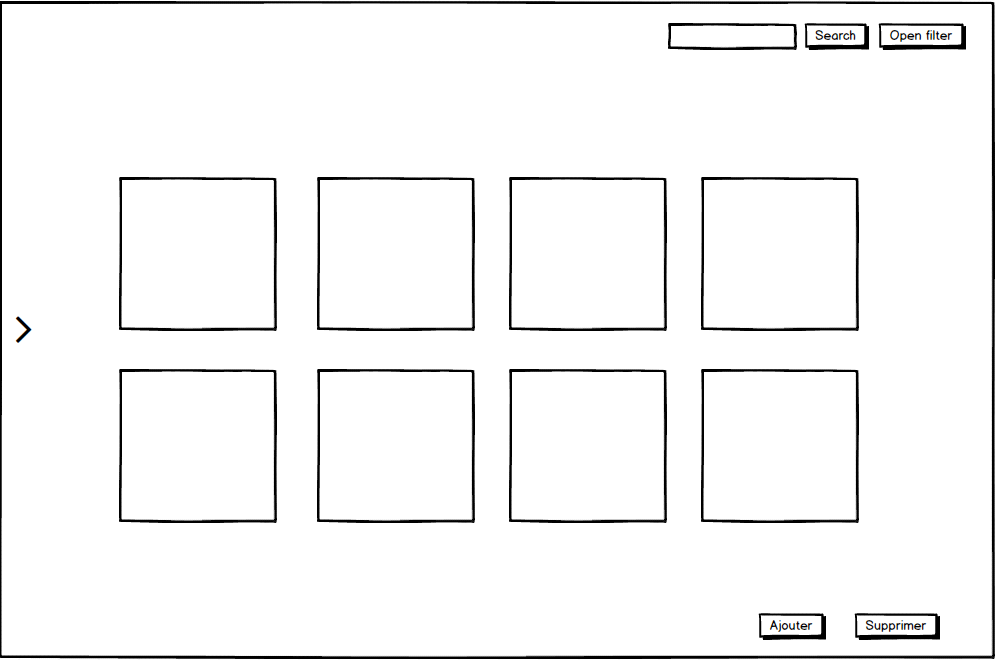
|  |  |
| --- | --- |
| **Action** | **Réaction** |
| J'accède à l'application | La page d'accueil s'affiche |
| Je clic le bouton « filtre » | Une liste de filtre s’affiche |
| Je sélectionne un premier filtre en cliquant sur le filtre « favoris » | Affiche uniquement les photos qui ont été mise en favoris. |
| Je choisis un deuxième filtre en ajoutant une plage de date. | Trie les photos favoris avec la plage de date demandée |
| Je choisis un troisième filtre en ajoutant la personne « Claude » à mes recherches | Affiche un message prévenant qu’il n’y a pas de résultats pour cette recherche. |

|  |  |
| --- | --- |
| Identifiant | Scenario\_Errorspace |
| En tant que | Utilisateur |
| Je veux | Sauvegarder les images et leurs informations |
| Pour | Garder la traces de mes modifications |
| Charge estimée | 10h |
| Priorité | Elevée |

|  |  |
| --- | --- |
| **Action** | **Réaction** |
| J'accède à l'application | La page d'accueil s'affiche |
| J’effectue une modification sur une photo en ajoutant des tags | La sauvegarde se fait automatiquement. |
| Je modifie le titre du dossier | Le dossier est sauvegardé automatiquement avec la modification du titre. |
| J’ajoute une centaine de photos d’un coup | Un message s’affiche pour prévenir l’utilisateur que le disque dur est saturé. La sauvegarde n’a donc pas eu lieu. |

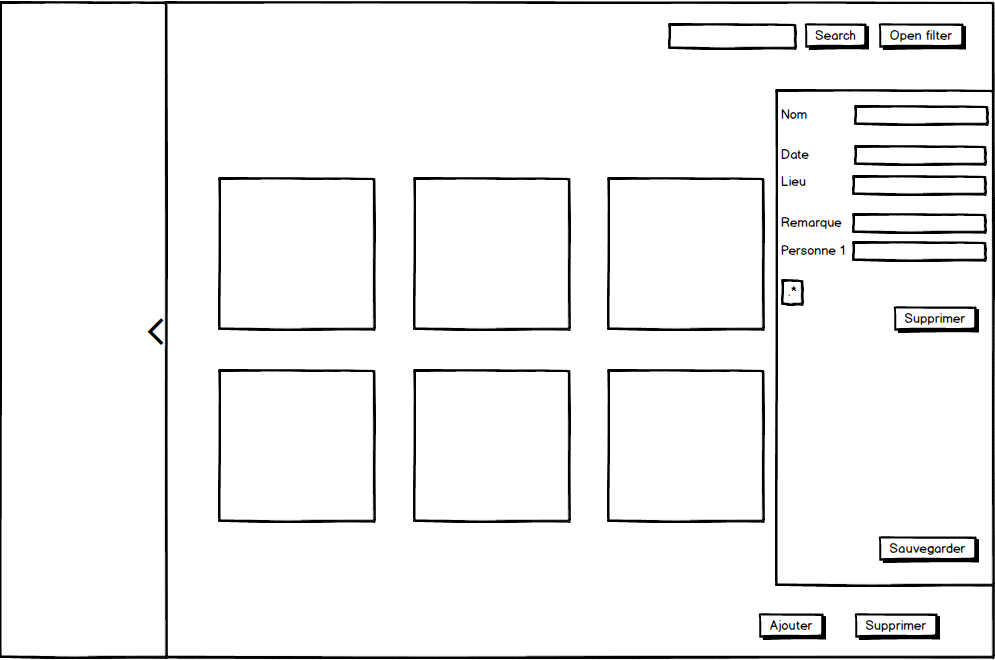
## Maquettes fonctionnelles

**Etat de base**



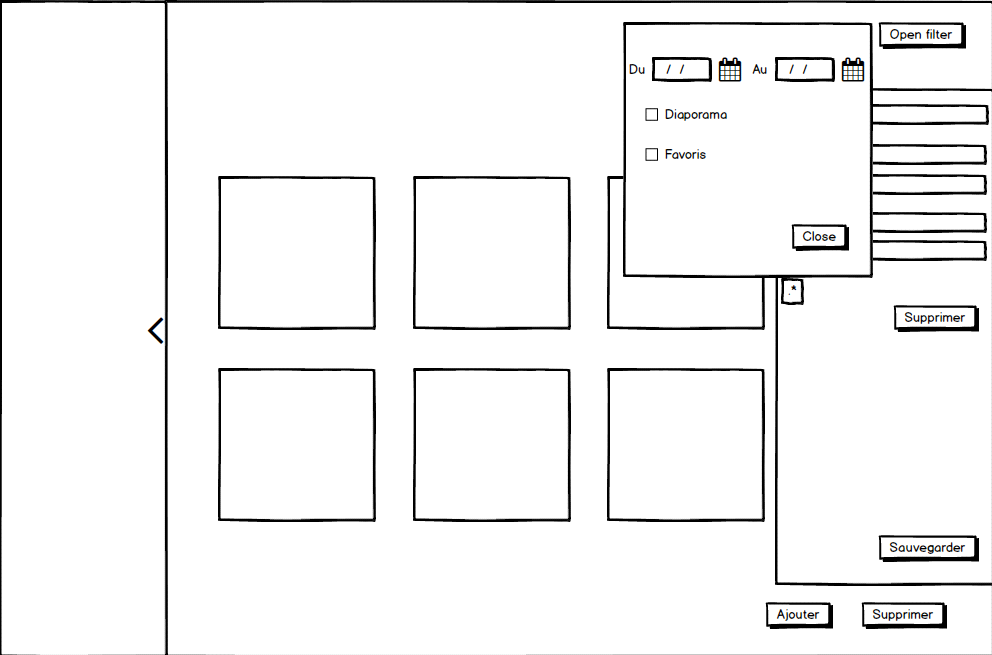
Tous les menus sont fermés et les photos du dernier dossier ouvert lors de la dernière fermeture du programme sont affichés.

**Menu principaux ouverts**



A gauche, l’arborescence de fichier est ouverte, tout comme le menu d’édition des informations personnelles à droite, qui s’affiche lors d’un simple clic sur une photo.

**Menu filtre**



Le menu des filtres sert à trier selon les contraintes choisies par l’utilisateur.

## Etude de faisabilité

Nous hésiter entre choisir le Framework ASP.NET ou WPF

Car ASP.NET est une solution web, et l’utilisateur souhaitait avoir une solution en local.

Nous avons choisi de faire ce projet en C# avec le Framework .Net car les deux programmeurs sont à l’aise avec le C# et l’un déjà utiliser se Framework. La phase d’apprentissage est donc réduite pour les deux développeurs.

Le Framework WPF est une Interface graphique utilisateur, utilisé pour faire des applications de bureau. Utilisant du XAML pour réaliser le design de l’application.

Au fur et à mesure de l’avancée dans le projet, nous avons rencontré aucun souci majeur qui nous a fait changer de solution. Nous n’avons pas encore trouvé de solutions permettant de crée de l’espace pour ajouter des métadonnées inexistantes sur une photo.

## Nom de l’application

Nous avons pensé à plusieurs noms pour cette application :

**QXPhotos**

Fait référence aux initiales des deux développeurs. Ce nom est libre mais on ne retrouve pas le coté gestion dans le titre.

**PhotoViewer**

Ce nom est simple, trop simple. Il décrit mieux l’application mais pas le coté de gestion.

**QXPhotosViewer**

Est donc le nom que nous avons choisi pour notre application. Un mélange des deux autres idées qui présente bien le projet en ajoutant une touche personnelle au nom.

## Stratégie de test

Pour tester l’application, nous allons tester tour à tour les différentes fonctionnalités. Les tests vont suivre les cas d’utilisations en premier lieux pour vérifier le bon fonctionnement des fonctions.

Les tests seront effectués au fil du développement de l’application avec des tests unitaires. Lorsque le projet sera terminé, nous testerons l’ensemble des fonctionnalités tour à tour dans l’application finale.

**Environnement de test :**

* Un ordinateur privé
* Un ordinateur inutilisé dans la salle de classe

**Données de test utilisés :**

Le client nous a fournis un échantillon de photos. Nous les utiliserons dans les tests de notre application.

**Participant aux tests :**

Les deux développeurs seront les deux testeurs de l’application.

**Liste des tests :**

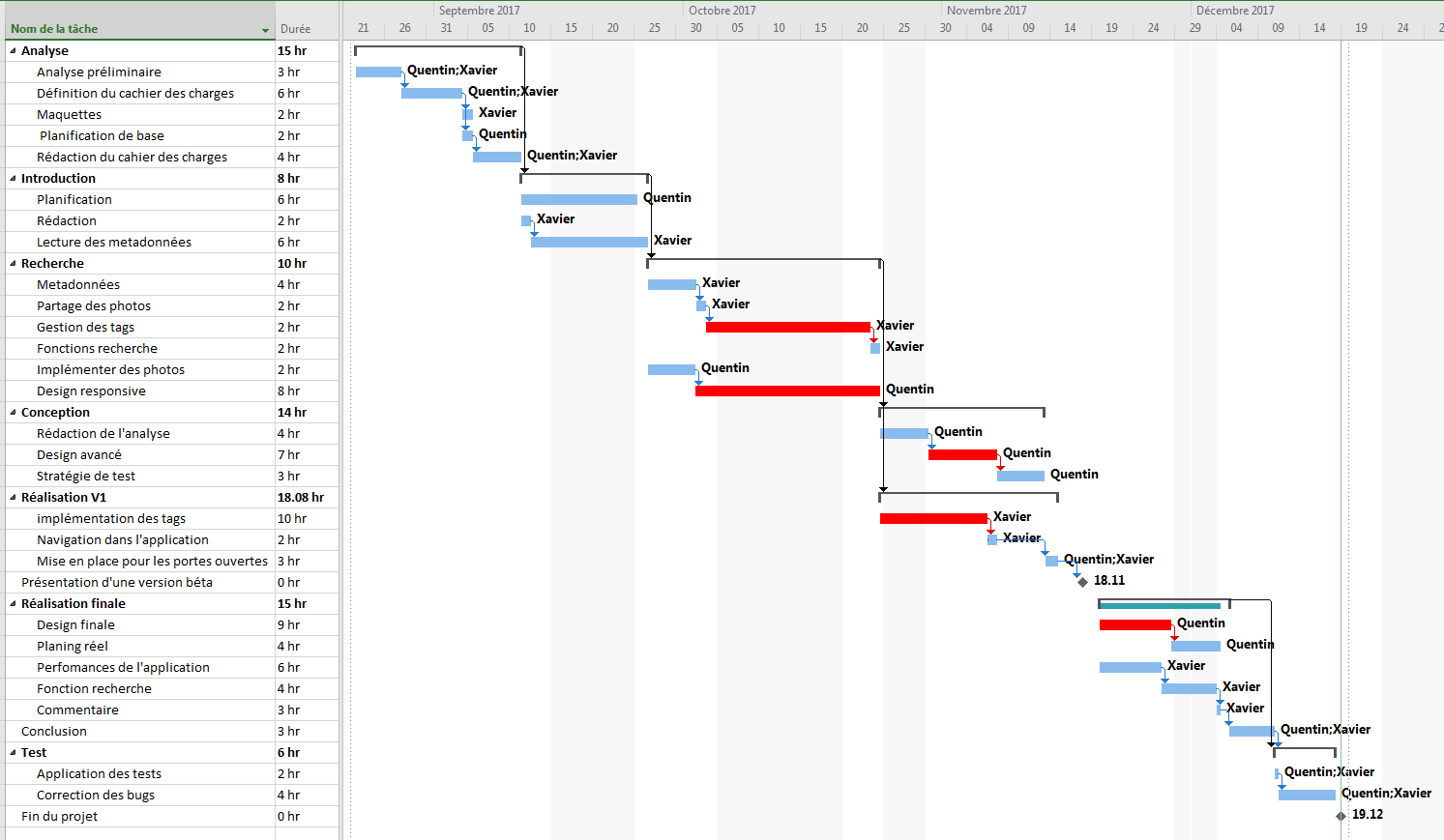
Ils seront vu dans la partie « descriptions des tests effectués », en même temps que leurs résultats.

## Budget initial

Les applications nécessaires pour ce projet sont fournies par le CPNV. Il n’y a donc aucun budget prévu à cet effet.

Pour le salaire, nous prenons une estimation de 30CHF / h pour un étudiant en technicien ES. Avec une estimation du temps de travail de 50h, la réalisation du projet couterait 3000 CHF.

## Planification détaillée



**Voici les grandes dates du projet :**

|  |  |  |
| --- | --- | --- |
| Analyse | 22.08.2017 | 11.09.2017 |
| Rendu du CDC | 15.09.2017 | |
| Introduction | 11.09.2017 | 26.09.2017 |
| Recherche | 26.09.2017 | 24.10.2017 |
| Conception / Réalisation V1 | 24.10.2017 | 13.11.2017 |
| Porte ouverte | 02.11.2017 | |
| Réalisation finale | 20.11.2017 | 05.10.2017 |
| Test | 11.12.2017 | 19.12.2017 |
| Fin du projet | 19.12.2017 | |

Le projet devra être rendu le 02.06.2017 à 8h50 ou avant. Avec une version papier et électronique pour chaque expert et responsable du projet.

La documentation sera écrite en même temps que les autres tâches. Ainsi que la planification et l’historique qui seront tenu à jour au fil du projet.

Code couleur :

De base : la tâche a été résolue dans les temps impartie.

Rouge : la tâche a été plus longue que prévue.

**Historique**

28.04.2017

Modification du canevas, ajout des cas d’utilisation. Changement de tâches et des heures attribué dans la partie Analyse

21.11.2017

Il commence à avoir du retard sur certaine partie bloquante du projet. Révisions des priorités.

30.11.2017

Date de fin annoncée au 19 décembre 2017. Plutôt que prévue, il va falloir trouver des solutions.

15.12.2017

Le projet est en majeur partie terminée malgré les retards. Certains objectifs jugés plus secondaires n’ont pas été programmé.

# Conception

## Analyse de l’environnement

**Liste du matériel HW :**

Le CPNV met à notre disposition un poste informatique complet ainsi qu’une connexion internet. Il n’y a pas d’équipement hardware spécifique requis pour la réalisation du projet.

**Liste des logiciels utilisés :**

* Visual studio 2017

Sera notre logiciel de développement

* Suite office 2016 (Word, Excel, Project, Outlook)

Nous servirons à documenter le projet et à nous organiser dans la réalisation de celui-ci

## Architecture du programme

Voici le principe de l’architecture du programme.

**MainWindow.xaml.cs**est la page ou la majeure partie des fonctionnalités sont codés. On y retrouve des fonctions redondantes :

GetTags() = récupère les metadata de l’images

AddTag() = ajouter des metadata aux images

Qui sont utilisés dans d’autres fonctions :

**TagButton\_Click**

Lors d’un clic sur le bouton d’ajout de tags, la méthode « TagButton\_Click() » est appelée. Celle-ci appelle la méthode « AddTags() » en lui fournissant les metadata voulue par l’utilisateur. Une fois ceci fait, elle appelle la méthode « GetTags() » Pour afficher les nouveaux tags.

**ImageButton\_Click**

Lors d’un clic sur une image, on appelle la méthode « GetTags () » et affiche les résultats ainsi que l’image en plus grand.

**txtNameToSearch\_TextChanged**

Lors d’une recherche, reprend la liste d’images affichée avant de trier avec les critères de l’utilisateur et ensuite, retourne la nouvelle liste.

**BrowseButton\_Click**

Sélectionne le dossier voulu, crée une liste d’image et renomme les photos avec la date de prise de vue récupéré avec la méthode « GetTags() » (ou de modification si l’autre date n’existe pas) et le nom du dossier

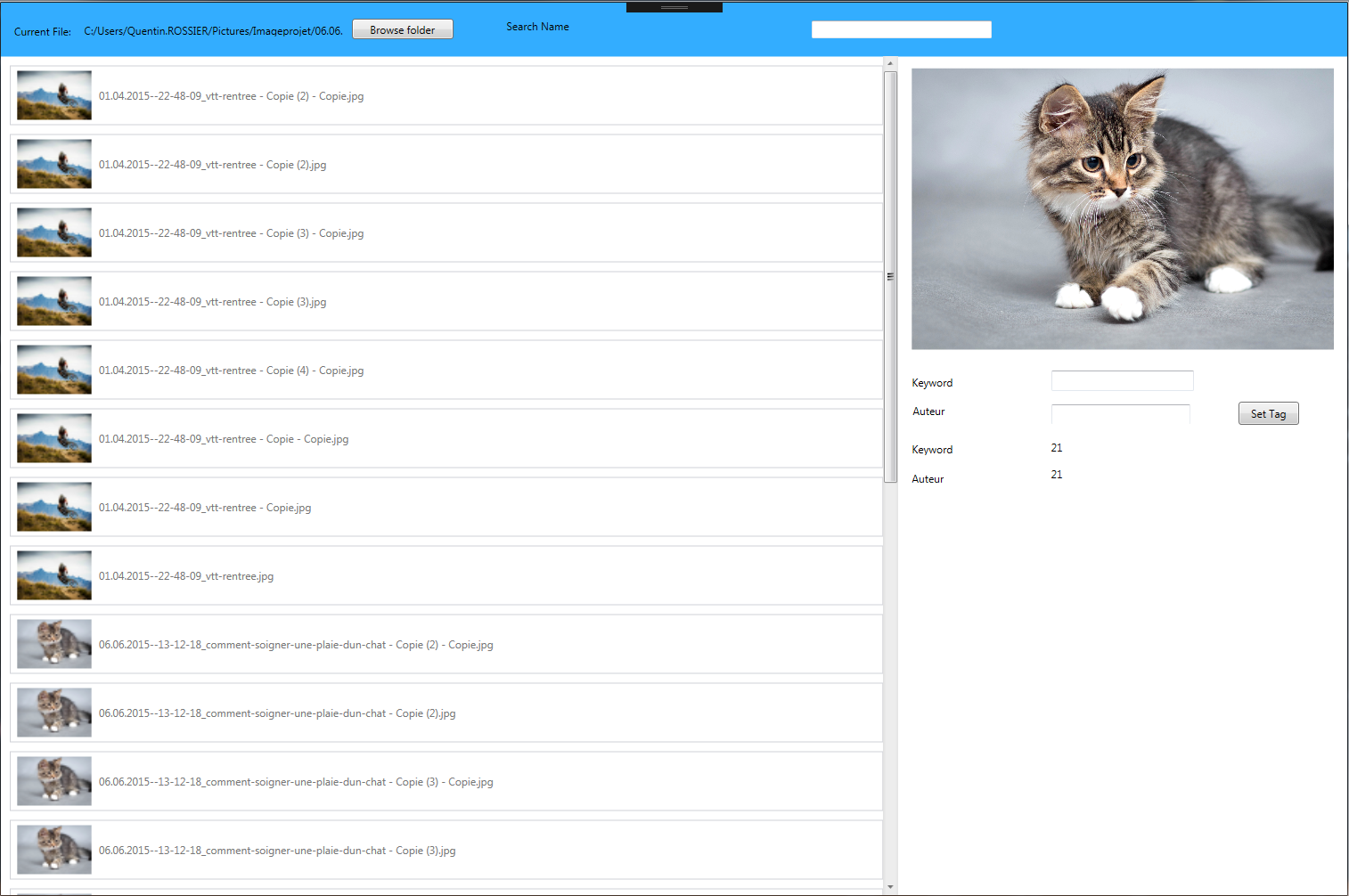
Les autres pages du document sont simples et expliquée dans la partie « dossier de réalisation »

## Définition de la charte graphique

Après avoir visualiser plusieurs applications du genre, Nous avons opté pour un visuel simple aux teinte épurée, vu que les images prendront la majeure partie du visuel de l’application.

Les couleurs du site seront du bleu « #33adff » et du blanc. Le bleu ne sera que peu présent pour équilibrer les zones de "non image". Et le blanc est choisis pour ne pas trop surcharger le visuel de l’application tout en la rendant lumineuse.

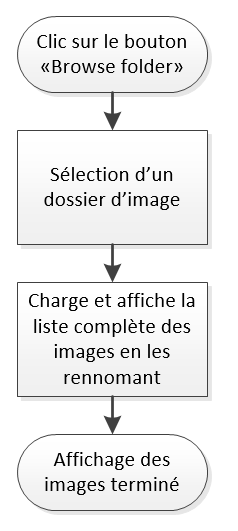
La police d’écriture choisis est « Segoe UI ». Qui est simple et lisible.

Maquette graphique

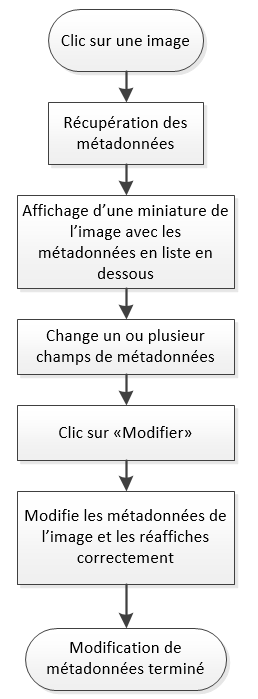
Voici un exemple de vue que l’application proposera. Cette page englobe la majeure partie du visuel de l’application. Il est possible que des changements soit effectué sur le design final du projet.

## Conception du code

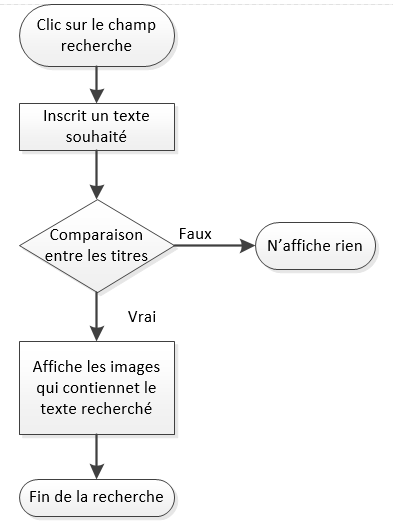
**Sélection d’un dossier pour l’affichage des photos**



**Sélection d’une image et modification des métadonnées**



**Recherche d’une photo selon un champ de texte**



# Réalisation

## Dossier de réalisation

Voici l’architecture des dossiers de l’application avec leur description :

Projet

├───Model

│ ├───ImageDetail.cs Classe pour la récupération des métadonnées

├───MainWindow.xaml Vue du projet

│ ├─── MainWindow.xaml.cs Contrôleur du projet

├───UriToBitmapConverter.cs Classe pour redimensionner les images

├───LinqExtensions.cs Classe pour convertir en collection observable

## Description de tests effectués

|  |  |  |  |
| --- | --- | --- | --- |
| Fonctionnalité | Manœuvre | Résultat attendu | Fonctionnelle ? |
| Affichage d’un dossier de photos | Cliquer sur le bouton « search folder » prendre un dossier de photos quelconque et en afficher la liste | Après un temps, affiche toute la liste de photo trié dans l’ordre alphabétique | Oui |
| Les photos doivent être renommées par la date de prise de la photo |  |
| Clic sur chaque photo tour à tour dans la liste | Doit afficher les photos en grand et leur liste de tags | Oui |
| Ajout de tags sur une photo | Cliquer sur une photo pour ensuite modifier ses tags | Les tags doivent être réafficher correctement dans l’application, et les métadonnées de la photos doivent être modifiées physiquement | Oui |
| Ajout de tags sur un dossier | Cliquer sur un dossier et modifier plusieurs tags | Les tags doivent être réafficher correctement dans l’application, et les métadonnées du dossier doivent être modifiées physiquement | Non |
| Recherche de photos | Inscrit une recherche dans le champ de recherche et la valide | Affiche les photos correspondantes au texte recherché | Oui |
| Si aucune photo ne correspond, n’affiche rien |
| Coche la recherche de favoris | Affiche toutes les photos mises en favoris | Non |
| Recherche une personne dans le champ de recherche | Affiche la liste des photos qui contient la personne dans les tags | Oui, recherche par « Auteur » |
| Stockage des informations ajoutées | Regarder physiquement sur la photo garde le changement des métadonnées | Les propriétés de la photo doivent changer pour garder les modifications de l’utilisateur. | Oui |

## Erreurs restantes

Voici la liste des erreurs restante dans le projet, par rapport à l’application prévue :

* Ajout de tags personnelles
* Gestions des erreurs

## Dossier d’archivage

Le projet a été créé sur GitHub afin que les deux participants puissent retrouver le projet et effectuer des modifications dessus sans se perdre. Tout en gardant un historique des modifications

Voici l’arborescence de dossier que nous avons créé pour ce projet :

───Canevas Liste de canevas utiles pour le projet

├───CDC

│ ├───CDC Cahier des charges et ses versions

│ │ └───Ancien Anciennes versions

│ ├───Maquette Fichier de maquettes

│ │ └───img Capture d’écran des maquettes

│ ├───Retour Retour du cahier des charges par le client

│ └───Uses-cases Liste des Uses-cases du projet

├───Documentation Liste des versions de la documentation

├───Planification Liste des versions de la planification

│ └───Image Capture d’écran des versions de la planification

├───Présentation

│ └───Portes-ouvertes Informations sur la présentation des portes ouvertes

└───PV Liste des PV du projet

## Problèmes rencontrés

|  |  |  |
| --- | --- | --- |
| **Problème** | **Analyse** | **Solution** |
| L’accès aux metadata des images était bloqué, en conséquence, on ne pouvait pas les modifier. | On à trouver après des recherches que c’est la liste des images affiché qui bloquait l’accès aux metadata. | La liste d’image est détruite avant la modification des metadata. Une fois ceci effectuée, elle est reconstruite. |
| Lors du chargement de la liste des images, la RAM était saturée par cette liste et causait l’arrêt prématuré de l’application. | Les miniatures gardaient le même poids que l’image originale. | Création d’une classe pour réduire la taille des miniatures. |
| Lors d’ajout d’une metadata qui n’a jamais été assignée, il est impossible de l’assigner. | L’image ne contient pas la place pour les metadata. Il faut donc crée cet espace. | Création d’une classe pour ajouter de la place pour les metadata dans l’image. |
| Lors d’une recherche, si un des metadata est vide, une erreur survient. | La recherche s’attend à un résultat non nul. | Ajout d’une condition LinQ « Where » pour filtrer les metadata vide. |
| Impossible d’écrire des metadata de lieux dans une image. | Il n’existe pas de metadata « lieux » dans les photos. | Utilisation des « mots-clefs » de la photo pour ajouter le lieux. |

## Mise en service

**Rapport de mise en service**

Pour installer notre application, il vous suffit de :

* Récupérer le fichier d’installation sur GitHub (<https://github.com/XSchwab/PremierProjetPhotoViewer>)
* Lancer le .exe
* Suivre l’installation
* Lancer l’application

**Liste des documents fournis**

* Page de présentation format A3
* Journaux de bord des deux participant
* Les fichiers d’installation du projet   
  ( <https://github.com/XSchwab/PremierProjetPhotoViewer> )

# Conclusions

## Objectifs

**Voici la liste des objectifs atteints :**

* Normaliser les noms de photos avec une date.
* Retirer le nommage « à la main » des photos importées.
* Classer les photos par ordre chronologique
* Ajouter des informations personnelles sur les photos tel que :
  + Des personnes (l’auteur de la photo est utilisé)
  + Des tags
  + Des lieux
* Être visible depuis plusieurs appareils comme :
  + Un téléphone
  + Une tablette
  + Un ordinateur
* Avoir des sauvegardes de données réutilisable en cas de changement de logiciel.
* Faires des recherches grâce à :
  + Une barre de recherche en auto complétion

**Voici la liste des objectifs non atteints :**

* Si la metadata n’ont jamais existé, on ne peut pas la rajouter.
* Ajouter des informations personnelles sur les dossiers tel que :
  + Des commentaires
  + Des filtres (favoris, date, format de photo, personne, tags)
* L’ajout d’information :
  + Des favoris

## Points positifs et négatifs

**Point positif :**

Ce projet nous a permis de voir de nouvelles technologie de traitement d'images. Ainsi que d’affirmer nos connaissances en C#.

**Point négatif :**

À cause de certaines parties du projet qui ont pris plus de temps que prévue, nous avons dû sacrifier d’autre partie moins importante, nous empêchant de terminer le projet à 100%.

## Difficultés particulières

La gestion des metadata et la création d'un design responsive fonctionnelle a pris plus de temps que prévu initialement. Nous avons donc perdu du temps sur ses deux parties respectives et avons alloué le reste du temps aux tâches importantes du projets, quitte à ne pas terminer complétement certains points du cahier des charges.

## Suite et améliorations possible

**Amélioration :**

* Terminer le projet en bonne et due forme
* Amélioration des performances globale

**Suite possible :**

* Faire une application pour mobile et tablette
* Ajout de fonctionnalité de traitement d’image supplémentaire par exemple des filtres graphique (sépia, noir et blanc, etc…)

# Annexes

## Sources – bibliographie

Nous avons consulté différents tutoriel et résolution de problème sur ses sites.

GitHub

Open classroom

Stackowerflow

code project

Gestion de tags: HTTP://blog.andreweichacker.com/2009/02/reading-and-writing-tags-for-photos-in-wpf

## Archives du projet

En annexe, sur GitHub.

Les journaux de bord, le manuel d’installation, la documentation du projet et le projet se trouves sur GitHub sous le lien suivant :

<https://github.com/XSchwab/PremierProjetPhotoViewer>