
AGENDA
Projet Structure de données

Introduction :

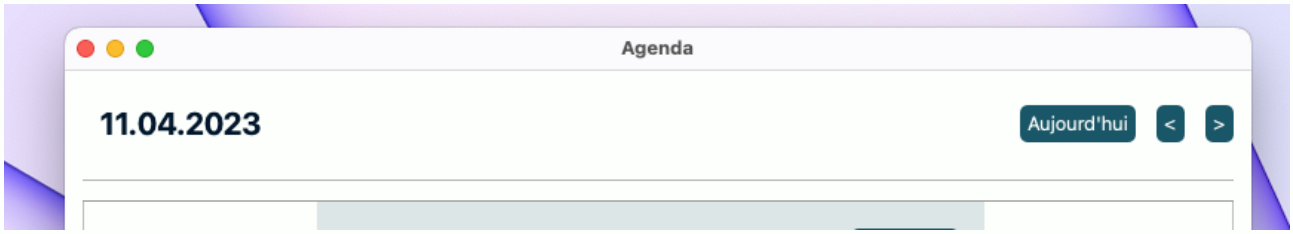
Dans le cadre de notre cursus universitaire, nous avons eu l'opportunité de travailler sur un projet de groupe portant sur le développement d'une application en C++ dédiée à l'organisation de rendez-vous et la gestion d'un calendrier numérique. Ce rapport a pour objectif de retracer le processus de création de cette application, en mettant en avant les différents aspects de son fonctionnement, de sa structure ainsi que de ses caractéristiques.

Tout d'abord, nous nous attacherons à décrire le fonctionnement de l'interface de l'application, ainsi que les fonctionnalités qu'elle offre à l'utilisateur. Nous aborderons également les choix esthétiques qui ont présidé à la création de cette interface, en mettant en avant la contribution de chacun des membres du groupe dans sa conception.
Nous continuerons en présentant les différentes étapes de l'organisation du programme et en expliquant les choix techniques qui ont été pris pour garantir une utilisation optimale de la mémoire et des ressources disponibles. Nous décrirons également les différents processus mis en place pour assurer le bon fonctionnement de l'application, ainsi que les méthodes de test et de validation mises en œuvre.
Nous nous pencherons par la suite sur le fonctionnement du programme lui-même, en expliquant les rôles et les interactions des différentes class qui ont été mises en place pour assurer son efficacité optimale. Nous aborderons notamment les aspects relatifs au back-end et au front-end de l'application, en mettant en avant les spécificités de chacune de ces composantes.
Nous évoquerons par la suite la contribution de chaque membre du groupe dans la réalisation de ce projet, en mettant en avant les compétences et les savoir-faire qui ont été mobilisés tout au long de la phase de développement.
En dernier lieu, nous procéderons à une évaluation critique du travail accompli, en évoquant les points forts et les points faibles de l'application. Nous proposerons également des pistes d'amélioration pour les développements futurs, en mettant en avant les axes de recherche qui pourraient permettre de renforcer les performances de l'application.

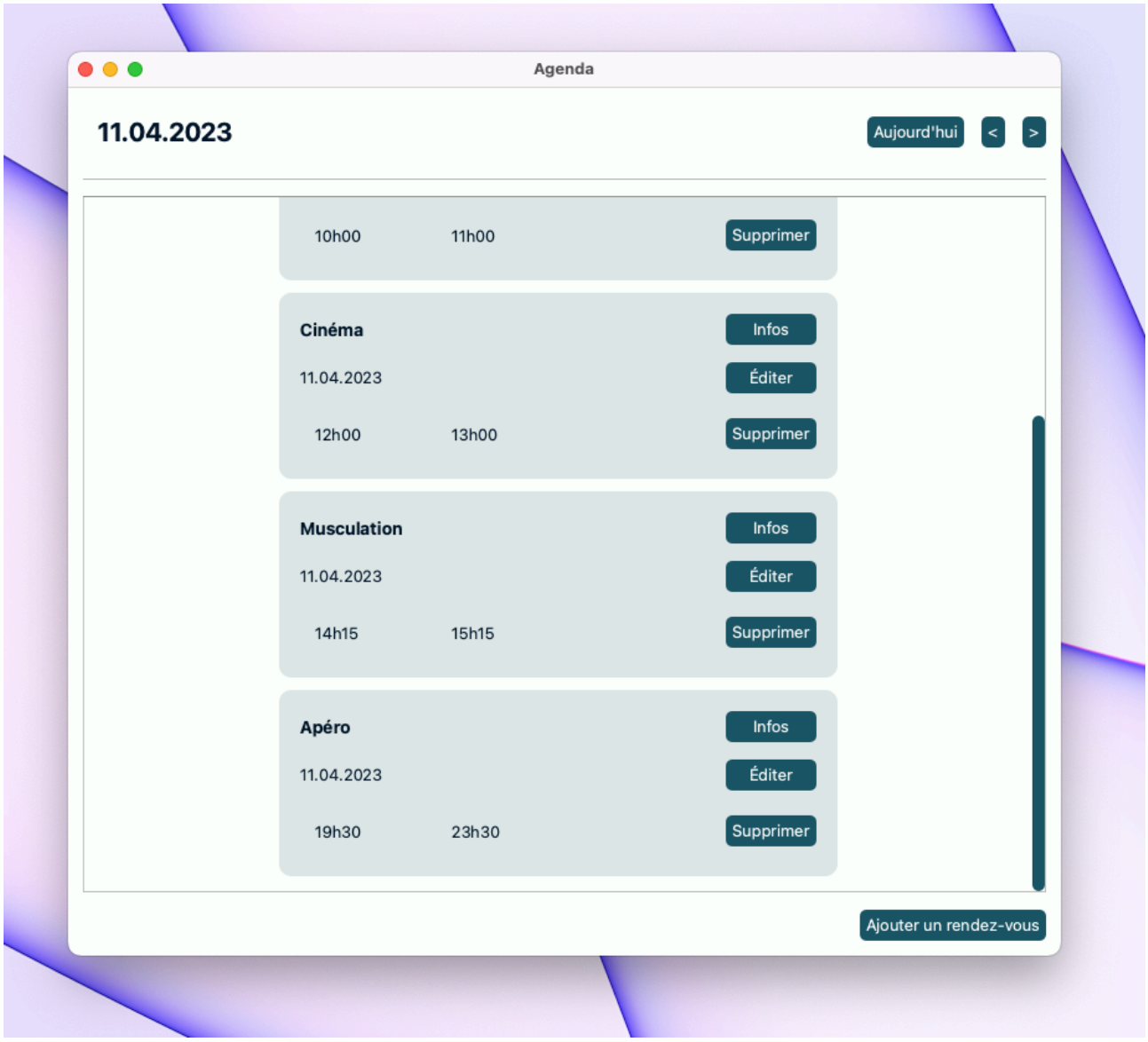
Conception et fonctionnalités de l’interface

Le but de notre projet était de développer un agenda numérique offrant un système de stockage et d'organisation de rendez-vous, ainsi que des informations concernant les personnes y participant. Il était primordial que le contenu soit facilement interrogeable et modifiable, en offrant la possibilité de supprimer ou d'éditer les données stockées. Dans cette première partie, nous allons décrire le fonctionnement de l'interface et les différentes fonctionnalités proposées à l'utilisateur, en les comparant aux objectifs initiaux de l’application.

La fenêtre graphique de l'application s'organise en deux grandes parties distinctes. Tout d'abord, un bandeau supérieur dans lequel s'affiche sur la gauche la date sélectionnée et à droite duquel se trouvent trois boutons offrant la possibilité de naviguer facilement entre les différents jours. Le premier permet d'accéder à la date du jour actuel, le second au jour précédant la date sélectionnée et le troisième au jour lui succédant. Cette première partie est ainsi dédiée à la navigation entre les différentes journées, avec une indication visuelle de la date et une facilité d'accès à l'ensemble des jours du calendrier.



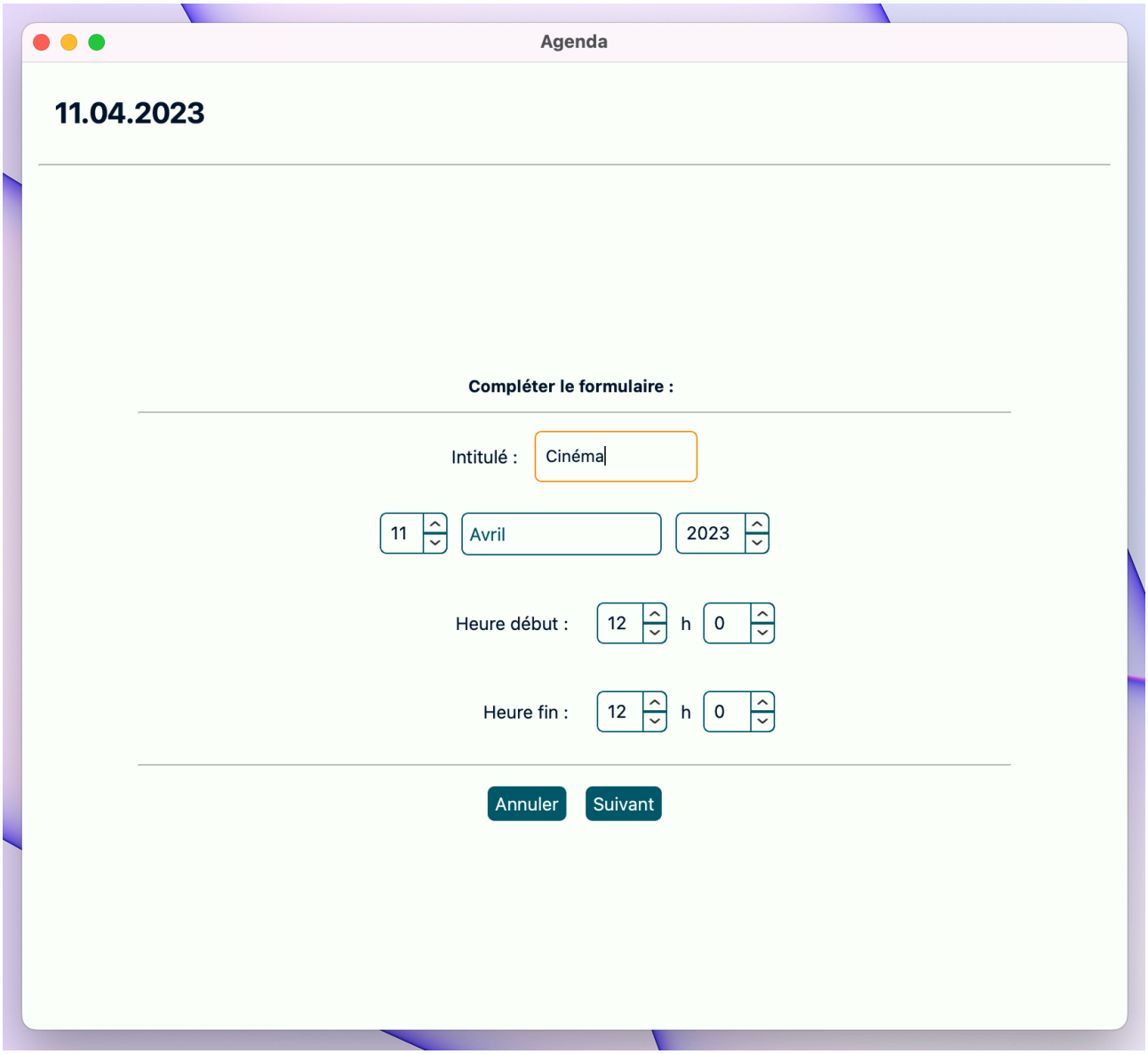
La partie principale de l'écran quant à elle, affiche les rendez-vous du jour sélectionné. Ils sont affichés dans un Widget personnalisé : "rendezVouxBox", qui affiche l'intitulé du rendez-vous ainsi que son horaire de début et de fin. De plus, chaque rendez-vous est muni de boutons (QPushButton) "infos", "éditer" et "supprimer", permettant à l'utilisateur d'interagir facilement avec le contenu du rendez-vous. La fenêtre principale est également munie d'un bouton en bas à droite, permettant l'ajout d'un nouveau rendez-vous. Cette partie de l'application permet donc à l'utilisateur de visualiser ses rendez-vous de manière claire et intuitive, tout en offrant la possibilité d'ajouter de nouveaux événements en un clin d’œil. On notera également que l’affichage des "rendezVousBox" se fait dans une zone déroulante permettant de conserver un affichage clair, lisible et compréhensible même lors des journées chargées.



Dans le but de permettre à l'utilisateur d'interagir facilement avec les rendez-vous stockés, les rendezVousBox sont munis d'un QPushButton "éditer" permettant d'afficher un formulaire de création de rendez-vous pré-remplis avec les informations existantes du rendez-vous sélectionné. Cette fonctionnalité permet de modifier rapidement et facilement les informations associées à un rendez-vous, telles que le titre, l'heure de début ou de fin, ou encore la liste des participants. Notons de plus que le formulaire en question est identique au formulaire de création d'un rendez-vous. De ce fait, l'utilisateur ne se perd pas dans une multitude d'affichages.
Chaque rendezVousBox est également équipé d'un bouton "supprimer", permettant de retirer facilement un rendez-vous de l'agenda. Cette fonctionnalité répond à l'un des objectifs initiaux du projet, qui était de permettre à l'utilisateur de modifier ou de supprimer les données stockées dans l'application en toute simplicité. En somme, l'interface développée permet une gestion intuitive et efficace des rendez-vous, offrant à l'utilisateur un accès rapide et simple à toutes les fonctionnalités nécessaires pour gérer son emploi du temps de manière optimale.

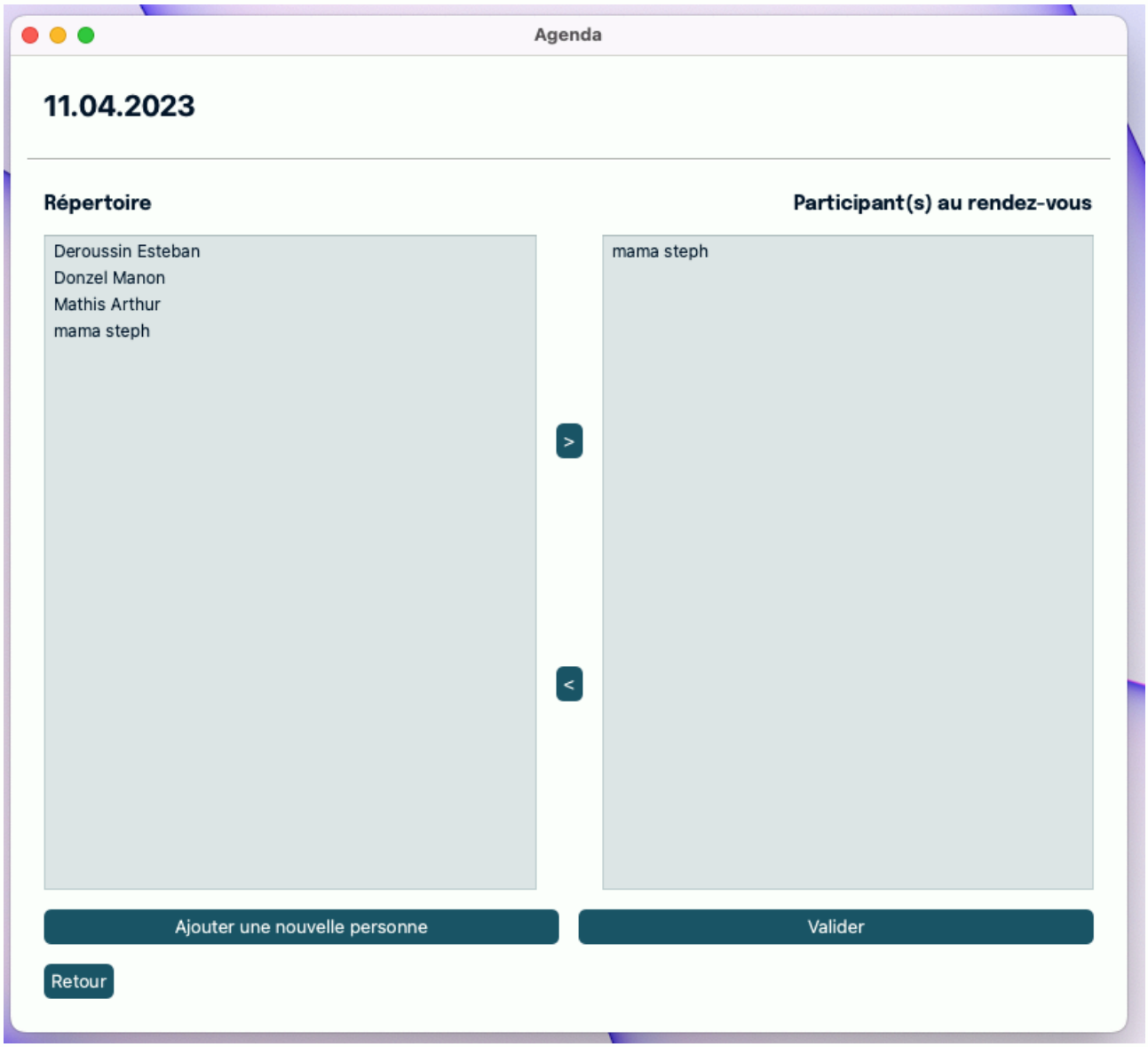
Pour finir le bouton "infos" affiche la liste des personnes présentes au rendez-vous ainsi que leur coordonnées.

Lorsque l'utilisateur souhaite ajouter un nouveau rendez-vous, il lui suffit de cliquer sur le bouton "Ajouter un rendez-vous" situé en bas à droite de la fenêtre principale. Alors, l'affichage des "rendezVousBox" est temporairement caché pour laisser place à un formulaire création de rendez-vous. Ce formulaire est composé d'une zone de texte modifiable (QLineEdit) permettant de saisir l'intitulé du rendez-vous. Il est également doté de trois QWidgets personnalisés pour la sélection de la date et de l'horaire de début et de fin du rendez-vous : "SelectionneurDate" et deux "SelectionneurHeure".



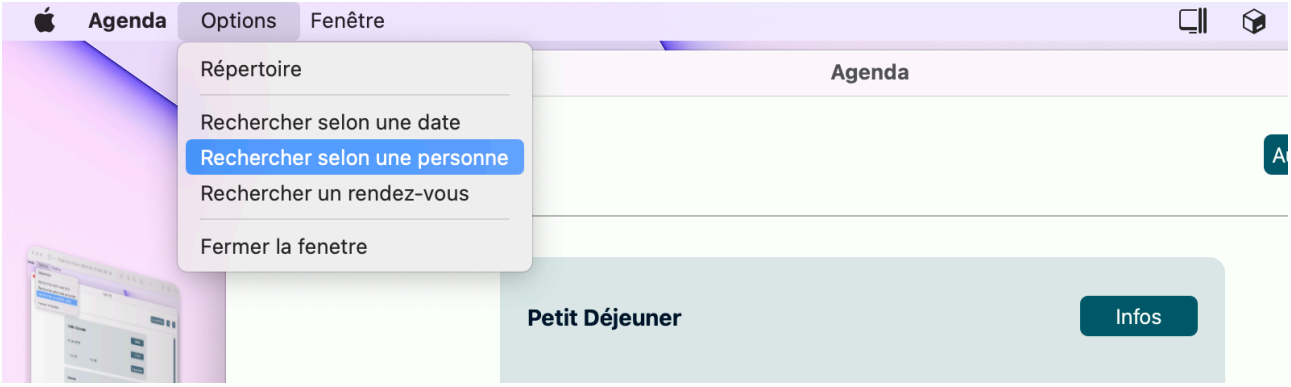
Une fois les informations de base saisies, l'utilisateur peut cliquer sur le bouton "Suivant" pour passer à la prochaine étape. Il est alors redirigé vers un nouvel affichage lui permettant d'ajouter les personnes participant au rendez-vous. Cette étape est gérée par un gestionnaire d'ajout de personne un QWidget contenant deux QList. La première

sur la gauche contient tous les contacts déjà implémentés, la seconde sur la droite uniquement ceux qui assisteront au rendez-vous. Pour simplifier l'ajout des personnes, nous avons mis en place un système simple, on clique sur la personne devant assister à l'événement, puis sur le boutons flèche en direction de la QList de droite. La suppression d'une personne présente se fait de façon analogue, on sélectionne la personne puis on clique sur la flèche dans le sens opposé.



Si l'utilisateur décide finalement de ne pas ajouter de rendez-vous, il peut annuler l'opération en cliquant sur le bouton "Annuler". Cette action permet de revenir à l'affichage des rendez-vous déjà existants dans l'agenda. On notera que le rendez-vous n'est créé et ajouté à la mémoire qu'après la seconde validation (lors de l'ajout des personnes présentes). En somme, l'ajout de rendez-vous est une opération simple et intuitive grâce au formulaire de saisie et au gestionnaire d'ajout de personne, offrant à l'utilisateur une grande flexibilité dans la création et la gestion de ses rendez-vous.

Maintenant que nous avons présenté les fonctionnalités de la fenêtre principale, intéressons-nous à la barre de menu située en haut de l'application. Celle-ci offre plusieurs options, notamment la possibilité de rechercher un rendez-vous selon la date, la personne présente ou l'intitulé. Il est également possible d'afficher les contacts ou de fermer l'application. Enfin, la barre de menu permet de recadrer rapidement la fenêtre en cas de besoin. Ces options offrent un moyen rapide et efficace de naviguer et de rechercher des informations spécifiques dans l'application.



La fonctionnalité de recherche est sans aucun doute l'un des éléments les plus utiles de notre application. Que vous cherchiez un rendez-vous selon une date, une personne ou un titre, nous avons tout prévu pour vous permettre de trouver rapidement ce que vous cherchez.

Si vous cherchez un rendez-vous en fonction d'une personne, il vous suffit d'ouvrir notre pop-up de recherche dédié. En entrant simplement le nom de la personne concernée, notre interface vous montrera instantanément tous les rendez-vous où elle doit être présente. Plus besoin de parcourir de longues listes ou de naviguer entre plusieurs interfaces, notre fonctionnalité de recherche est simple et intuitive.

De même, si vous cherchez un rendez-vous selon un intitulé spécifique, notre pop-up de recherche est là pour vous aider. En quelques clics, vous pouvez saisir le titre du rendez-vous et notre application affichera tous les rendez-vous correspondants, tout est à portée de main.

Enfin, si vous cherchez un rendez-vous selon une date précise, notre bouton de recherche dédié est là pour vous aider. En quelques clics, vous pouvez sélectionner visuellement la date que vous recherchez grâce à notre calendrier intégré, ou simplement saisir la date au format "xx-xx-xxxx". Une fois la date saisie ou sélectionnée, notre application vous montrera instantanément tous les rendez-vous correspondants.
Cette fonctionnalité est très utile pour gagner du temps dans la recherche de rendez-vous spécifiques, en évitant d'avoir à parcourir de nombreux éléments pour trouver ce que vous cherchez. Avec la barre de menu et ses différentes fonctionnalités, vous

pouvez naviguer facilement et rapidement dans l'application pour trouver les informations dont vous avez besoin.

Organisation du programme

Afin de concevoir un projet cohérent, nous avons passé un temps certain dans l'élaboration d'une architecture de fichiers permettant de relier les parties du code orientées back-end au partie en charge du rendu graphique. La problématique était de concevoir un sytème permettant de manipuler sans risque les données, tout en conservant une interface efficace et intuitive.

Pour se faire, nous avons orienté notre développement vers une programmation objet en générant quatre class gérant les entités de l'application : les dates, les horaires, les personnes et les rendez-vous. Par la suite, dans le but d'organiser les données et de simplifier leurs mises-jours, exploitations et sauvegardes, nous les avons stockées dans deux listes chainées : une pour les rendez-vous et une pour les personnes. Cette méthode permet de simplement sécuriser et lier les données. Par exemple un rendez-vous est composé d'une liste de personnes, d'une date, d'un horaire de début et de fin ainsi que d'un intitulé. Une fois les class "Date", "Horaire" et "Personnes" codées, il suffit des les inclure (#include) dans la class "RendezVous" et on peut alors utiliser leurs méthodes. Chaque class sécurisant ses données et garantissant leur intégrité, leurs exploitations devient plus simple puisque les erreurs logiques ne peuvent plus être faites.

De plus, la class MainWindow joue un rôle central dans l'application en regroupant toutes les autres class et en permettant leur interconnexion. Les class "Date", "Horaire", "Personne" et "RendezVous" sont des composantes essentielles du back-end, car elles permettent la gestion des différentes entités de l'agenda. Grâce à la programmation orientée objet, ces class sont en mesure d'être complètement autonomes. Elles gèrent leur création, mise-à-jour, utilisation et destruction en veillant à ne pas générer de fuite de mémoire et sans compromettre leur intégrité. Les class ListePersonnes et ListeRendezVous, quant à elles, constituent des listes chaînées qui stockent les informations des class du back-end afin de les organiser de manière optimale.
En analysant les choses de plus prés, on constate que les class de front-end communiquent avec "MainWindow" qui gère au travers des listes chainées l'ensemble des données de l'application. Les listes chainées ainsi que la class "MainWindow" sont alors le symbole de l'interconnexion profonde de cette architecture de fichiers, tout transite par eux.

On notera que pour simplifier l'étude des différentes class et de leurs liens nous avons généré une documentation Doxygen comprenant la déclaration et les commentaires de chaque variables, de chaque méthode, de chaque class ainsi que les diagrammes représentant ces connexions inter-class.

Fonctionnement des différentes class

Nous avons divisé notre projet en deux grandes parties, pour permettre une gestion plus efficace de le répartition et de sa gestion : le back-end et le front-end. Le back-end contient plusieurs class, notamment celles dédiées à la gestion des dates, horaires, personnes et rendez-vous, ainsi que les listes associées, qui ont été traitée par l'utilisation de listes doublement chainées. Le front-end, quant à lui, contient un ensemble de class, à savoir : AfficheurDate, AfficheurHoraire, AfficheurRdv, AjouteurRdv, AjoutPersonneRdv, ContactWidget, InfosRdv, RechercheParDate, RechercheParNom, RendezVousBox, SélectionneurDate, SélectionneurHeure, ainsi que la MainWindow. Nous avons choisi de diviser le front-end en plusieurs class, héritant de Widget, pour pouvoir en réutiliser certaines de manière simple et efficace, afin de nous en servir en tant que composant. Cela a grandement facilité le développement du projet.
Voyons à présent, de manière plus détaillée, chaque class du projet, en commençant par le back-end puis en abordant le front-end, en expliquant leurs rôles et fonctionnalités respectives.

Commençons par la class « date" qui est responsable de la gestion des dates, représentées par un jour, un mois et une année. Elle dispose de méthodes pour récupérer et modifier les valeurs de ces composants, ainsi que pour vérifier la validité de la date et pour incrémenter ou décrémenter le jour tout en maintenant la validité de la date. Pour améliorer les performances, notre code utilise des typedef pour définir un type entierc (un entier non signé de petite taille) et une fonction pour convertir une chaîne de caractères en entierc. Ces types permettent d'utiliser des entiers de taille réduite pour stocker les composants de la date. La class "date" offre également trois surcharges d'opérateurs de comparaison : "==" pour l'égalité, "<" pour l'infériorité stricte et "<=" pour l'infériorité ou l'égalité.

La class "horaire" est une autre composante importante de notre back-end. Elle permet de gérer les horaires dans notre programme, qui sont représentée par des entierc contenant les heures et les minutes. Des méthodes pour récupérer et modifier l'heure et les minutes, ainsi que des méthodes pour vérifier si l'heure et les minutes sont valides et si un horaire est compris entre deux autres horaires, ont été implémentés. Elle comprend également des opérateurs de comparaison pour les horaires, ceux-ci permettent de vérifier l'égalité, l'infériorité stricte, et l'infériorité ou l'égalité. En combinant cette class avec la class "date", nous pouvons gérer de manière efficace et précise les rendez-vous dans notre application.

Passons maintenant à la class "personne". Elle permet de stocker des informations sur une personne telles que son nom, prénom, adresse e-mail et numéro de téléphone. Elle dispose de méthodes pour mettre à jour et récupérer chaque information de la personne, ainsi que des fonctions pour vérifier la validité de chaque information, comme le format de l'adresse e-mail ou le nombre de chiffres dans le

numéro de téléphone. De plus, cette class possède un opérateur de comparaison qui permet de comparer un nom avec le nom d'une personne.

La class "listePersonne" est une implémentation d'une liste doublement chaînée de personnes, représentées par la class "personne" vue précédemment. Chaque maillon de la liste est représenté par la class "chainonPersonne", qui contient un pointeur vers la personne qu'il représente, ainsi que des pointeurs vers le maillon suivant et précédent de la liste. Cette class offre plusieurs méthodes pour insérer des personnes dans la liste en ordre alphabétique (par chainonPersonne ou par personne) et une pour supprimer des personnes de la liste chaînée. Il est également possible de chercher une personne dans la liste et la renvoyer elle ou son chainon. Elle dispose également d'une méthode pour obtenir le nombre de personnes dans la liste, savoir si une personne se trouve dans la liste, et pour renvoyer le pointeur du premier maillon de la liste et le changer.

L'objectif du projet étant avant tout la gestion d'un carnet de rendez-vous, il fallait de manière évidente implémenter une class rendez-vous pratique et efficace. De ce fait, notre projet dispose d'une class "rendezVous", qui hérite des class "date" et "horaire". Elle représente un rendez-vous et contient donc des attributs tels que le nom du rendez-vous, la date, l'heure de début et de fin du rendez-vous, ainsi qu'un vecteur de pointeurs vers des chainons de personnes participant au rendez-vous. La class a plusieurs méthodes pour accéder et modifier les attributs du rendez-vous, ainsi que pour ajouter un vecteur de participants au rendez-vous. Il y a également une méthode de tri par ordre alphabétique pour trier les noms des participants. En outre, la class "rendezVous" surcharge l'opérateur d'égalité pour permettre la comparaison entre un nom de référence et le nom d'un rendez-vous.

Pour regrouper les rendez-vous du carnet, nous avons, tout comme le répertoire, implémenté une liste doublement chainée. On dispose d'une part d'une class nommée "chainonRendezVous", qui est utilisée pour représenter chaque maillon de la liste doublement chaînée de rendez-vous. La class contient un pointeur vers le rendez-vous qu'il représente, ainsi que des pointeurs vers le maillon suivant et précédent de la liste. D'autre part, nous avons la class "listeRendezVous" qui elle, est définie pour contenir des rendez-vous sous forme de liste doublement chaînée. La class offre plusieurs méthodes pour insérer et supprimer des rendez-vous, dans l'ordre chronologique, ainsi que pour rechercher des rendez-vous selon leur nom, leur date ou les personnes impliquées. Elle permet également de vérifier la disponibilité d'une personne pour un rendez-vous à une date et une heure données. La class "listeRendezVous" contient un pointeur vers le maillon tête de la liste, ainsi que des méthodes pour modifier ou accéder à ce pointeur. On peut également retourner la taille de la liste.

Maintenant que nous avons terminé avec la partie back-end de notre projet, intéressons-nous au front-end. Nous allons présenter les différentes class qui ont été développées pour l'interface utilisateur.

Débutons par la class "AfficheurDate", présente dans le front-end du projet. Cette class permet d'afficher une date principalement mais aussi un texte dans notre interface utilisateur graphique. Une instance de la class "date" est passée en paramètre du constructeur pour représenter la date à afficher. La class contient également une méthode publique "afficherDate()" qui permet d'afficher la date dans l'afficheur. Les membres privés incluent un pointeur vers un objet "QLabel" pour contenir la date, un pointeur vers un objet pour gérer le layout du composant, ainsi qu'une méthode privée pour générer les différents composants de l'affichage. Cette class encapsule donc la logique nécessaire pour afficher une date dans une interface graphique.

En continuité avec la class "AfficheurDate", la class "AfficheurHoraire" permet d'afficher un horaire dans notre interface utilisateur graphique. Elle prend en paramètre une référence constante à un objet de la class "horaire" dans son constructeur. Cette ne contient que deux membres privés : une variable "d_heure" contenant l'horaire à afficher et une méthode privée qui crée l'affichage de l'horaire.

En lien avec les class précédentes, la class AfficheurRdv permet également d'effectuer un affichage, cette fois-ci, celui des rendez-vous dans une interface graphique. Contrairement aux class précédentes, cette class possède plusieurs méthodes pour gérer l'affichage des rendez-vous, notamment selon une date, une personne donnée ou un rendez-vous donné. On peut également réinitialiser l'affichage à l'aide de deux méthodes (effacerAffichage et viderTableau). La méthode creeTableauDate permet ainsi de créer un tableau de rendez-vous selon une date donnée et d'initialiser directement l'affichage. La méthode onSupprimerRdvId permet quant à elle de supprimer un rendez-vous sélectionné par l'utilisateur et onInfosRdvId permet de consulter les participants d'un rendez-vous sélectionné. La class contient également des tableaux de pointeurs vers des rendez-vous et leurs affichages, un pointeur vers le layout principal, la liste de rendez-vous, ainsi que des groupes de boutons permettant la modification, la consultation ou la suppression de rendez-vous.

Passons maintenant à la class AjouteurRdv, qui permet d'ajouter un rendez-vous à la liste de rendez-vous ou d'en modifier un. Cette class est constituée d'un ensemble d'éléments graphiques, tels qu'un champ de texte pour l'intitulé du rendez-vous et des composants de sélection de la date et des heures, que nous présenterons ultérieurement. Le constructeur de la class permet d'initialiser les différents éléments du composant graphique grâce à la méthode de création de l'affichage. De plus, les méthodes onHeureDebut et onHeureFin permettent de s'assurer que l'heure de fin est automatiquement supérieure à l'heure de début en effectuant une vérification et une mise à jour automatique des valeurs saisies. Les différents éléments du composant graphique sont rendus accessibles en tant qu'attributs publics de la class, ce qui permet de les manipuler depuis l'extérieur de la class et de récupérer les valeurs saisies par l'utilisateur pour l'ajout d'un rendez-vous. En somme, cette class regroupe les éléments nécessaire pour créer un composant graphique d'ajout et de modification de rendez-vous dans notre interface utilisateur.

La class AjoutPersonneRdv est une class qui permet d'ajouter des personnes à un rendez-vous en cours de création ou modification. Cette class est utilisée en complément de la class AjouteurRdv et permet à l'utilisateur de sélectionner les

participants pour un rendez-vous et de les ajouter au rendez-vous en cours de création ou modification. Elle contient des widgets tels que des boutons "ajouter" et "retirer" pour manipuler les listes de personnes disponibles et sélectionnées, ainsi qu'un bouton "valider" pour créer et ajouter le rendez-vous à une liste de rendez-vous. Un bouton retour permet également de retourner à la fenêtre correspondant à la class AjouteurRdv. La class utilise des pointeurs vers le répertoire et l'agenda, pour accéder à des données relatives aux personnes et aux rendez-vous, ainsi qu'un pointeur vers l'AjouteurRdv associé, pour afficher les informations du rendez-vous créé ou modifié. La class contient également des méthodes pour réagir aux interactions avec les boutons, telles que l'ajout et la suppression de personnes sélectionnées et la mise à jour de la liste des répertoire, car, il est en effet possible d'ajouter une nouvelle personne au répertoire et au rendez-vous à partir de cette fenêtre.

La class "ContactWidget" est une class qui gère l'affichage et la manipulation des contacts du répertoire. Elle est similaire à la class "AjoutPersonneRdv", car elle utilise également des pointeurs vers le répertoire et l'agenda pour accéder aux données relatives aux personnes et aux rendez-vous. Le constructeur de la class prend en paramètre deux pointeurs "listePersonne*" et "listeRendezVous*", qui indiquent l'adresse mémoire des listes chaînées gérant respectivement le répertoire et les rendez-vous. La class a trois slots privés qui réagissent aux événements des boutons pour ajouter, modifier ou supprimer un contact dans le répertoire. Le pointeur vers l'agenda est utile lors de la suppression. En effet, il permet d'assurer que la personne n'est présente dans aucun rendez-vous avant de la supprimer. Une méthode privée permet de créer l'interface graphique du widget, tandis qu'une seconde méthode permet de mettre à jour la liste des contacts à l'affichage. Un bouton permet également de revenir à la page principale.

La class "InfosRdv" permet de créer une interface graphique dédiée à l'affichage des participants d'un rendez-vous sélectionné dans l'interface principale de l'application. Cette interface affiche le nom, prénom et les coordonnées (email, numéro de téléphone) de chaque participant dans un widget de type QScrollArea avec un layout vertical. Chaque participant est affiché dans un bloc distinct comportant un label pour son nom complet et un label pour chacune de ses coordonnées. Une ligne de séparation est ajoutée entre chaque bloc. La taille de la fenêtre est fixée, pour garder une interface esthétique et cohérente. Cette class utilise un pointeur chainonRendezVous pour accéder aux informations relatives au rendez-vous à afficher.

La class RechercheParDate est en lien direct avec l'affichage et la manipulation des rendez-vous. Elle permet de créer une interface graphique pour sélectionner une date et rechercher les événements correspondants à cette date. Elle utilise un calendrier pour permettre la sélection de la date. Lorsque l'utilisateur appuie sur le bouton de recherche, la class déclenche un signal qui est capté par la class principale pour effectuer la recherche. La class RechercheParDate est donc un élément facilitant grandement la navigation et la recherche d'informations.

La class RechercheParNom est une class similaire à la class RechercheParDate, car elle permet également de créer une interface utilisateur pour effectuer une recherche spécifique. Cependant, au lieu de rechercher des événements selon une date, cette class permet de rechercher des rendez-vous selon un nom de personne ou de rendez-vous. Elle contient un QLineEdit pour saisir le nom à rechercher et un bouton de recherche pour lancer la recherche. Cette class est utilisée dans l'application pour permettre à l'utilisateur de trouver rapidement des rendez-vous en entrant un nom dans

la barre de recherche dédiée. Le nom renseigné est ensuite utilisé dans les fonctions de recherche pour chercher les rendez-vous correspondants dans l'agenda.

La class RendezVousBox est liée à la class AfficheurRdv car elle permet d'afficher les informations d'un rendez-vous de manière esthétique et conviviale. Elle contient une méthode pour créer l'affichage d'un rendez-vous, qui consiste en un QLabel pour le titre du rendez-vous, un QLabel pour la date du rendez-vous, deux AfficheurHoraire pour l'heure de début et de fin du rendez-vous, ainsi que trois QPushButton pour afficher les participants, éditer et supprimer le rendez-vous. La méthode configure la mise en page de l'interface utilisateur en utilisant un QGridLayout.

Nous évoquions tout à l'heure la sélection de date dans l'AjouteurRdv, cela est possible grâce à la class SelectionneurDate, qui permet de créer un composant graphique pour sélectionner une date. Les éléments de sélection sont un QSpinBox pour le jour et l'année, et un QComboBox pour le mois. La class contient une méthode pour mettre à jour la date stockée en fonction des valeurs sélectionnées, à partir de trois méthode distinctes pour le jour, le mois et l'année, et une méthode pour initialiser les valeurs des éléments de sélection à une date donnée. Des slots permettent de mettre à jour la valeur maximale de d_jour en fonction du mois ou de l'année sélectionnée.

La class "SelectionneurHeure" permet de créer un composant graphique pour sélectionner une heure. Comme la class "SelectionneurDate", elle contient des QSpinBox pour les heures et les minutes et une méthode pour mettre à jour l'heure stockée en fonction des valeurs sélectionnées. Elle dispose également d'une méthode pour initialiser les valeurs des sélecteurs à une heure donnée.

La class MainWindow est la class principale de l'application, elle regroupe toutes les class du front-end et permet de les utiliser ensemble. Elle contient des slots pour interagir avec les boutons de certains composants, car ils ont un effet sur l'affichage de la MainWindow. Lorsque l'application est lancée, la MainWindow est construite et organisée grâce à la méthode creeInterface. Cette méthode met en place les différents composants de l'interface utilisateur tels que la zone scrollable pour les rendez-vous, les boutons pour naviguer entre les jours, les différents widgets permettant d'afficher les rendez-vous et les contacts, les pop-up pour les recherches, les boutons de bar de menu etc. La MainWindow possède également des méthodes permettant d'écrire et de lire les fichiers de sauvegarde pour le répertoire et l'agenda, et elle crée des instances des class listePersonne et listeRendezVous pour stocker les données de l'application. Une date est également stockée pour la navigation et plusieurs manipulations de l'interface. La MainWindow contient énormément de pointeurs vers les widgets avec lesquels elle interagit. Cela permet de les créer et les supprimer quand on en a besoin, ce qui est pratique pour la gestion de la mémoire, mais aussi de les cacher et les réafficher lorsque qu'ils sont utilisés fréquemment.

Le front-end de l'application contient une dernière chose, qui cette fois-ci n'est pas initialisé dans une class, mais dans le main de notre projet, c'est la feuille de style. La fonction ouvre un fichier style.qss, qui contient les règles de style pour l'application. Si le fichier est ouvert avec succès, son contenu est lu et utilisé pour définir la feuille de style de l'application à l'aide de la méthode setStyleSheet. Sinon, la feuille de style est définie par défaut directement dans la fonction main. Cette feuille permet d'avoir une interface plus chaleureuse et plus agréable à utiliser.

En conclusion, notre projet de gestion de rendez-vous a été élaboré en utilisant une approche de développement web basée sur une architecture en deux couches : le front-end et le back-end. Le front-end correspond à la partie visible de l'application, laquelle est responsable de la présentation de l'interface utilisateur et de l'interaction avec l'utilisateur. D'un autre côté, le back-end s'occupe de la logique métier de l'application, incluant notamment la gestion et le traitement des données principales, à savoir l'agenda et le répertoire, mais aussi les dates et les horaires.
L'utilisation de ces deux class distinctes, le back-end et le front-end, a permis à notre projet de bénéficier d'un certain nombre d'avantages. En effet, cela a permis de séparer la logique métier de l'application de la présentation de l'interface utilisateur, ce qui a facilité la mise à jour de l'une ou l'autre de ces parties sans perturber l'autre, mais aussi d'augmenter l'efficacité de l'une par rapport à l'autre.
L'utilisation de différentes class du back-end et du front-end a donc été cruciale pour le développement de notre projet de gestion de rendez-vous. Elle a permis de créer une application fonctionnelle et robuste, tout en facilitant le travail de développement et en offrant une expérience utilisateur de qualité.

Répartition des tâches et optimisation du travail d'équipe

Afin de réaliser le développement de l'agenda dans les délais imposés, nous avons organisé notre travail de manière efficace et productive. Les quatre membres de notre équipe: Arthur, David, Reda et Leon, ont chacun apporté une contribution importante au projet en se répartissant les tâches. Arthur et David se sont occupé majoritairement de la partie front-end afin de produire un rendu intuitif et lisible, pendant que Leon et Reda se sont concentrés sur la partie back-end. Dans un second temps, chacun d'entre nous sommes intervenus autant dans le back que dans le front pour permettre l'inclusion de l'un dans l'autre de manière fonctionnelle.

Arthur a été un membre essentiel de l'équipe, apportant ses compétences techniques pour améliorer l'interface utilisateur de l'application. Il a travaillé sur plusieurs QWidget personnalisés tels que "AfficheurDate", "AfficheurRdv", "SelectionneurDate", "SelectionneurHoraire", "RendezVousBox", "AjouteurRdv", et la feuille de style qss "stylesheet.qss" afin faciliter l'utilisation de l'application et rendre les fonctionnalités plus intuitives pour les utilisateurs. En outre, Arthur a grandement participé dans le processus de debugage de l'application. Il a apporté sa contribution pour détecter et résoudre les bugs dans son ensemble. Il s'est également chargé de la rédaction de l'introduction du rapport ainsi que de la première partie : "Conception et fonctionnalités de l'interface" et de la seconde : "Organisation du programme".

David a joué un rôle crucial dans le développement du projet en travaillant sur la conception de plusieurs fichiers importants tels que "ListePersonne", "ContactWidget", "AjouteurPersonneRdv", "InfosRdv", "AfficheurRdv", les procédures de sauvegarde et une partie de "MainWindow". David s'est impliqué dans le rendu graphique de l'application en développent notamment le répertoire et la QScrollArea. Il a également participé à la correction de bugs dans l'application en ajoutant ou modifiant les procédures et

méthodes manquantes ou défectueuses dans les différents fichiers. En plus de sa contribution technique, David a également pris en charge une partie de la rédaction du rapport en expliquant l'utilité de chaque fichier programmé ainsi qu'en rédigeant la documentation sur la majorité du projet.

Ainsi Arthur et David ont travaillé ensemble sur la class "MainWindow", qui est au cœur de l'application. Cette class permet de relier toutes les autres class et assure le bon fonctionnement de l'application dans son ensemble. Leur travail a permis d'avoir une interface graphique cohérente et intuitive pour l'utilisateur, tout en garantissant la stabilité et la performance de l'application.

Leon a apporté une contribution importante au projet en travaillant sur des éléments clés du back-end, notamment des fichiers clés tels que "Horaire" et "Personne". En effet, il a été impliqué dans le développement de plusieurs class qui ont permis de garantir le bon fonctionnement global de l'application. En plus de cela, Leon a été responsable de la création de class de recherche avancées pour simplifier l'expérience utilisateur en travaillant sur les fichiers "RechercheParNom", "RechercheParPersonne" et "RechercheParDate" permettant une meilleure accessibilité pour l'utilisateur dans l'interface graphique. Grâce à son expertise en programmation et sa capacité à conceptualiser les processus logiques complexes, Leon a été un élément clé dans l'optimisation du code et l'amélioration du rendu global de l'application. Il a également rédigé la partie : Répartition des tâches et optimisation du travail d'équipe.

Reda a été d'une aide précieuse dans le développement des class de gestion des rendez-vous et des dates, qui sont des éléments clés du fonctionnement de l'application. Il a travaillé sur les fichiers "ListeRdv", "RendezVous" et "Date", en mettant l'accent sur la simplicité et la performance du code. En plus de sa contribution au développement, Reda a créé une présentation PowerPoint pour le projet, qui permettra d'assurer une communication durant la soutenance de projet. Enfin, il a également apporté sa contribution au compte rendu en évaluant le travail réalisé par l'équipe et en proposant des pistes d'amélioration pour l'avenir. Reda a permis à l'équipe de développer une application robuste et conviviale, qui répond aux besoins de ses utilisateurs.

Évaluation du travail et pistes d'améliorations

L'implication de chacun au projet, nous a permis de produire une application fonctionnelle et opérationnelle sur de nombreux aspects. De ce fait, dans cette partie nous allons évaluer le travail que nous avons accompli jusqu'à présent et identifier les éventuelles pistes d'amélioration pour rendre notre application plus performante et plus conviviale.

Tout d'abord, nous avons travaillé de manière organisée et avons bien réparti les tâches selon les aptitudes de chacun, ce qui nous a permis de progresser de manière

efficace. Nous avons créé un ensemble de class pour gérer un carnet de rendez-vous, en disposant d'une structure de données pour les personnes et les rendez-vous, ainsi que de deux listes doublement chaînées pour le répertoire de personnes et l'agenda contenant les rendez-vous. Nous disposons également de deux class permettant la gestion temporelle des rendez-vous, à savoir date et horaire.
Notre code permet de vérifier si une personne est occupée lorsqu'un rendez-vous est planifié ou modifié et nous avons donc également la possibilité d'ajouter ou de supprimer des rendez-vous, des personnes (suppression possible seulement si la personne n'est pas engagée dans un rendez-vous), ainsi que de modifier les informations relatives aux rendez-vous comme la date, les horaires de début et fin, ainsi que la liste des participants et aux personnes. Nous avons choisi d'autoriser le changement de nom des personnes pour prendre en compte le changement à l'issue d'un mariage. De plus, nous avons mis en place des méthodes pour afficher tous les rendez-vous pour une date donnée par l'utilisateur et pour une personne donnée. Un rendez-vous peut être affiché à partir de son nom. Les informations relatives aux personnes participants à un rendez-vous peuvent être affichées dans une boite de dialogue.
Les éléments de nos listes chainées sont identifiés par leurs noms. Leur insertion se fait en fonction du nom des personnes pour la liste de personne, et selon l'heure et la date pour les rendez-vous. De ce fait, la majorité des manipulations sont effectuées en utilisant le nom de l'élément.
Nous avons également géré la mémoire allouée convenablement, en particulier lors de la suppression d'éléments, en supprimant les pointeurs après leur utilisation. Nous avons optimisé notre code en gérant les boucles, le parcours des tableaux, l'allocation et la libération de la mémoire. Enfin, nous avons bien séparé le code de la gestion du carnet de rendez-vous et du code de l'interface utilisateur.

Cependant, il y a toujours des pistes d'amélioration pour rendre notre application plus performante et plus conviviale. Par exemple, il serait utile de pouvoir rechercher des éléments dans les deux listes à partir d'autres critères que le nom. Nous pourrions également améliorer notre interface utilisateur pour qu'elle soit plus intuitive et plus facile à utiliser, notamment avec un sytème de couleur en fonction du type de rendez-vous ou un affichage annuel ou mensuel.

Nous avons travaillé de manière efficace et organisée pour créer une application de gestion de carnet de rendez-vous avec une structure de données efficace et optimisée. Nous avons également réussi à mettre en place des méthodes pour ajouter, supprimer, modifier et afficher des rendez-vous et des personnes ainsi que les manipuler de multiples manières. Notre application répond donc efficacement à toutes les principales et fondamentales exigences du projet. Toutefois, il y a toujours des améliorations possibles pour rendre notre application encore plus performante et conviviale.

Conclusion

En conclusion, nous sommes très satisfaits de notre projet de gestion de rendez-vous. Nous avons travaillé en harmonie pour concevoir une application fonctionnelle et agréable à utiliser, tout en répondant aux demandes. Nous avons été en mesure d'intégrer efficacement les différentes class du back-end et du front-end pour maintenir la cohérence de notre projet. Nous avons également acquis une expérience précieuse tout au long de ce projet, en apprenant à collaborer en groupe et à développer des compétences techniques clés. Chacun de nous a pu contribuer de manière significative au projet en fonction de ses compétences et de ses intérêts, tout en apprenant de nouveaux concepts et techniques.
Dans l'ensemble, ce projet de gestion de rendez-vous a été une expérience très enrichissante pour nous tous. Nous sommes convaincus que cette application sera utile pour les utilisateurs et nous sommes fiers du travail accompli.

Cependant, comme une conclusion commune ne peut jamais refléter précisément les sentiments de chacun, nous avons également pris le temps de rédiger individuellement nos ressentis sur le projet.

ARTHUR

Je suis ravis d'avoir participé à l'élaboration et au développement d'un tel projet. Après ces deux années passionnantes d'étude dans le supérieur, j'étais impatient de créer quelque chose de plus concret, quelque chose de complexe qui nous forcerai à explorer, expérimenter, tester, se tromper et recommencer afin d'obtenir finalement un résultat dont nous serions vraiment fiers.
Ce projet a été pour moi l'occasion de faire le bilan des compétences que j'ai acquises tout au long de la formation en rendant le tout bien plus réel. Cela passe par l'organisation des class dans des fichiers sources bien séparés ; par le développement d'une interface bien au de la de nos habitudes en travaux dirigés ou encore par l'interconnexion profonde de nos class et listes chainées. La complexité du projet a mis à rude épreuve ma capacité à concevoir des choses abstraites. Un si grand nombre de fichiers communicant de façon minutieuse les uns avec les autres n'étais pas simple à gérer. Mais bug après bug nous avons réussi à mettre au point une interface complexe et intuitive. J'ai eu l'occasion de découvrir le langage qss et de donner une apparence élégante et sophistiquée au projet, quelque chose d'inédit pour moi dont je suis à la fois fier et satisfait.
En résumé, ce projet était ambitieux et difficile à réaliser mais ce sont ces choses qui l'ont rendu excitant et profondément formateur.

DAVID

Au cours de ce projet, j'ai eu l'opportunité de travailler avec un groupe actif, avec lequel j'ai beaucoup appris. J'ai particulièrement apprécié l'esprit d'équipe et la collaboration de chaque membre pour la réussite du projet. En tant que développeur « polyvalent », j'ai

pu intervenir sur la grande majorité du code, que ce soit en back-end ou en front-end. Cela m'a permis d'avoir une vision globale sur le projet et une certaine connaissance du code. Ce qui m'a permis entre autre, de rapidement identifier les problèmes lors de mes séances de débugage. De plus, la découverte de bibliothèques pratiques et efficaces de Qt et C++ a été très enrichissante pour moi. Je suis donc fier d'avoir participé à la création de cette application pratique et fonctionnelle.

LEON

La programmation d'un agenda peut être un défi passionnant et enrichissant, mais cela peut aussi être une expérience frustrante et stressante. Dans l'ensemble, j'ai ressenti un mélange de ces émotions tout au long de ce projet. D'une part, j'ai apprécié la satisfaction de voir l'agenda prendre forme au fur et à mesure que je codais. Cela m'a donné un sentiment de réussite et de fierté à chaque étape du processus. J'ai également appris beaucoup de choses tout au long de cette expérience, notamment en termes de conception de logiciels et de codage. D'autre part, j'ai également ressenti des moments de frustration lorsque je rencontrais des bugs ou des erreurs qui semblaient difficiles à résoudre. Il y avait des moments où j'ai dû prendre une pause pour éviter de devenir trop stressé ou anxieux à propos de ce projet. Dans l'ensemble, la programmation de cet agenda m'a permis de mieux comprendre les complexités du développement de logiciels et de développer mes compétences en programmation. C'était une expérience enrichissante qui m'a donné un aperçu de l'importance de la patience, de la persévérance et de la collaboration lors de la création de quelque chose de nouveau et de complexe.

REDA

Je suis satisfait du rendu final de notre application. Durant ce projet, j'ai pu développer un nombre conséquent de compétences comme le travail d'équipe et la répartition des tâches comme dans une vraie entreprise. J'ai aussi apprécié le fait de voir directement à quoi servira notre structure de donnée et en quoi elle est utile au fonctionnement de notre agenda. J'ai pu comprendre davantage l'utilité et le fonctionnement de certaines structures de donnée, en particulier les listes chainées. En effet, j'ai trouvé le projet plus concret et moins théorique vis à vis de ce que l'on a pu faire en CM ou en TD.