Rapport final: Organisation d'un festival

Sujet :

Notre projet portait sur le développement d 'une application permettant l'organisation d'un festival. L'objectif de ce travail était d'avoir l'application la plus fonctionnelle et la plus complète possible. En effet, les besoins d'un usager de cette dernière sont que l'application soit simple d'utilisation et guidée. Le problème principal a donc été l'optimisation de la programmation du festival en fonction des artistes invités, de leur demande en matériel et du matériel proposé par l'organisateur du festival.

Approche :

Ainsi lors de la première approche, nous nous sommes concentrés sur les fonctionnalités que devait proposer notre application afin de répondre aux attentes de l'organisateur. Les tenants et aboutissants d'une organisation réussie étant d'avoir la meilleure programmation possible tout en optimisant les dépenses de manière à être rentable au niveau financier. Dans cette optique il fallait donc se concentrer sur plusieurs points cruciaux à une optimisation générale.

Organisation :

Concernant la répartition des taches, Zacharie s'est plus occupé de la partie de la rédaction concernant les BDD et connexion entre les BDD et les classes, des méthodes des classes ainsi que la compilation complète du main en console. Hubert s'est plus occupé de la rédaction des diagrammes, de l'envoi des mails et de la partie rédaction des rapports. Cependant, bien que certains points ont été plus abordé par l'un ou l'autre, tout a été traité et analysé en commun.

Hypothèses :

Afin de mener à bien ce projet, plusieurs hypothèses ont été formulées. Ainsi nous partons du principe qu'un organisateur organise un festival à la fois, que tous les concerts ont la même durée (hors phase de préparation), que le type de matériel demandé par un artiste existe dans le matériel que possède déjà l'organisateur.

De plus on a travaillé avec des dates en String de la forme DD/MM/YYYY à défaut du type Date. Ces dates seront celles où on souhaite inviter les différents artistes. Le concert de l'artiste aura donc lieu à une date fixée par l'organisateur, afin que la disponibilité du matériel, des scènes corresponde.

Concernant l'application même on a pris l'hypothèse que chaque scène possède un style musical qui lui est propre et qui doit être rentré manuellement par l'utilisateur directement dans la base de données. Ce style de musique sera utile lors de la programmation afin que chaque artiste ayant le même style de musique soit dirigé vers la scène de son type de musique. L'artiste sera ajouté à la fin du concert d el'artiste ayant le même registre musical précédent.

Enfin, si l'artiste demande du matériel non disponible à l'horaire où il est programmé, l'application renvoie seulement à l'organisateur la quantité et le type de matériel à acheter pour satisfaire l'artiste., et ce par soucis de simplicité du code (sinon il aurait fallu créer un système de file d'attente).

Développement :

De manière générale, le développement de l'application s'est donc fait chronologiquement. Dans un premier temps il faut donc définir le *festival,* c'est à dire lui associer un nom ainsi qu'un nombre de places, qui servira par la suite à gérer la *billetterie.* Puis, comme dans tous les festivals il fallait ajouter du *matériel* proposé par l'organisation pour l'accueil des *artistes*. Une fois que le *festival* possédait une réserve de matériel type, tout était prêt pour inviter des artistes.

Donc il a fallut inviter des artistes qui seraient à chaque fois questionnés sur leurs caractéristiques tels que leur nom, leur style de musique ou encore leur disponibilité le jour du dit festival. Suite à cela, l'artiste est, ou non, ajouté à la base de donnée des artistes. Cependant avant de lui attribuer un *concert* il a fallu être capable de connaître le matériel dont il avait besoin afin de lui attribuer un horaire de passage coïncident avec la disponibilité du matériel nécessaire à son passage. En effet, si un artiste a besoin de matériel spécifique proposé par le festival, cela doit être pris en compte de manière à ce que le concert puisse se dérouler dans les meilleurs conditions possibles. Ainsi on a du demander aux artistes le matériel dont il avait besoin et, avant de leur attribuer un horaire de passage, vérifier la disponibilité de ce dernier, le matériel étant disponible s'il n'est pas utilisé à l'horaire voulu.

Dès lors, deux cas de figures sont possibles : le premier étant que le matériel n'est pas disponible à cet horaire, ce qui entraîne la recherche d'un nouvel horaire par l'application de manière à trouver la plage horaire suivante la

plus proche où le matériel est disponible. Si cette dernière n'existe pas, c'est à dire si la demande de matériel par l'artiste est trop grande par rapport à ce que propose le festival, alors il est proposé à l'organisateur de se procurer plus de matériel ou tout simplement d'annuler l'invitation de l'artiste. Le deuxième cas de figure possible à la suite de la vérification de disponibilité du matériel demandé par l'organisateur est que celui-ci soit disponible à l'horaire voulu. Dans ce cas, en fonction du type à déplacer, c'est à dire le nombre total de type de matériel différent à faire venir des autres scènes ou de la réserve afin de répondre à la demande de l'artiste, un temps de préparation plus ou moins long sera ajouté à la programmation. Ce temps permettra alors d'anticiper le transport et l'installation de ce matériel supplémentaire nécessaire à l'organisation du concert, afin que la timeline soit le mieux respectée lors du Jour J. C'est cette partie du codage de l'application qui a été la plus essentielle, l'optimisation de la programmation en dépendant. La difficulté résidant sur la simultanéité des actions à effectuer par l'application, ces derniers étant de vérifier si le matériel est proposé par le festival, s'il existe en quantité suffisante, s'il est disponible à un horaire initialement proposé pour le concert de l'artiste. Suite à ces requêtes, l'horaire initialement proposé était ou non conservé, ce en quoi selon le type de matériel nécessaire, un temps de préparation était ajouté. Ce temps de préparation dépend de la quantité de type de matériel à déplacer.

Un fois que l'horaire est attribué à un artiste en ayant pris en compte le matériel à déplacer, l'application passe au suivant et ainsi de suite jusqu'à la fin du temps alloué à ce festival ou jusqu'à ce que l'organisateur ne veuille plus ajouter d'artiste à sa programmation. Suite à cela, la base de donnée associée à l'application a donc en mémoire la timeline des artistes.

Ainsi, dans la console et l'interface, à chaque artiste ajouté est retourné le matériel dont il aura besoin, ainsi que son heure de passage qui prend en compte la phase de préparation qui lui est antérieure. Dans la base de donnée on obtient donc finalement la timeline complète avec les concerts des artistes, avec pour chacun une heure de début et de fin.

Parties exploitées mais non finalisées :

Les interfaces ont été abordées et développées en partie, cependant le main compilant et étant aussi intuitif qu'une interface pour ce type de sujet, nous ne sommes pas allé au bout du développement de l'interface (Début interface

dans la classe Festival2 ci jointe). Une interface aurait été intéressante cependant dans le cas d'une application réelle, avec une étape de connexion au départ de manière à ce que Artistes, Organisateur et Festivaliers puissent utiliser la même application. Aussi répondre à des questions est plus intuitif dans une interface que dans un main.

De la même manière, il n'y a pas d'interface ou de rappel des informations fournies par l'utilisateur. Cela aurait pu être fait par une interface faisant un rappel de tout ce qui a été rentré ou directement dans le main. Cependant toutes les informations sont disponibles dans la base de données associée et en console au fur et à mesure.

Une erreur subsiste dans la programmation, en effet il y a eu un problème avec l'utilisation du type Time, empêchant l'ajout de temps d'être effectif.

Points non exploités :

Au moment de la connexion, l'organisateur de festival peut ajouter un nouveau festival ou se connecter au dernier festival entré, cependant il ne peut pas récupérer un festival s'il en a créé un autre depuis. Ainsi nous nous sommes heurtés à un problème de taille, il aurait fallu pour toutes les classes associer un id_festival à laquelle elles sont associées de manière à distinguer plusieurs festivals à la connexion.

De plus, nous n'avons pas pris en compte la distance entre les scènes dans le temps de préparation ni même la quantité de matériel à déplacer mais seulement le type de différents matériels. Ainsi pour déplacer 1 table, 1 chaise et 1 micro, il notre programme renverra plus de temps de préparation que pour déplacer 50 caissons de basses. Cela aurait été développable en ajoutant dans la base de données une colonne « difficulté de déplacement » en Integer avec des valeurs allant de 1 à 10 par exemple et définir le temps de préparation en fonction d'un total de tous les valeurs de « difficulté de déplacement » de tout le matériel à changer de scène.

Aussi nous n'avons pas réussi à faire de plan en .svg. Il aurait pour cela intégrer de nouveaux paramètres telle que la surface totale du festival, la disposition générale de cette surface. Il aurait fallu de plus intégrer l'accès entre ces derniers, faire des hypothèses sur la capacité en fonction de la surface (pour connaître le nombre de personnes dans la public qui peuvent assister à chaque concert).

Exécution :

Afin d'exécuter le programme, il faut run le main dans la classe *Festival* et répondre ensuite en suivant les instructions données par le main. Ainsi chaque information est rentrée dans la bdd. Pour consulter toutes les informations il faut aller directement dans la bdd ou regarder directement la console.

Ce que le projet nous a appris :

Ce projet nous a appris les étapes de résolution d'un problème comme on peut en retrouver dans le monde du travail. En effet nous avons d'abord effectué une grosse phase d'Analyse, puis nous avons commencé le développement et enfin du rendre des livrables pour une échéance donnée.
De plus nous avons appris à utiliser des logiciels que Modelio, Ganttproject, PgAdminIII. Nous avons amélioré de manière non négligeable notre niveau en Java mais surtout notre manière de raisonner quant à la POO.
Enfin le travail en groupe permet de clarifier le raisonnement, de manière à ce que notre binôme puisse le reprendre en cours de route sans qu'il y ait nécessité d'interaction et donc de potentielle perte de temps. Le fait de travailler en groupe permet aussi d'avoir plusieurs points de vue de résolution d'un problème donné et donc d'être « débloqué » lorsque l'on s'engage sur une méthode de résolution trop compliquée. Le travail de groupe augmente la lucidité.