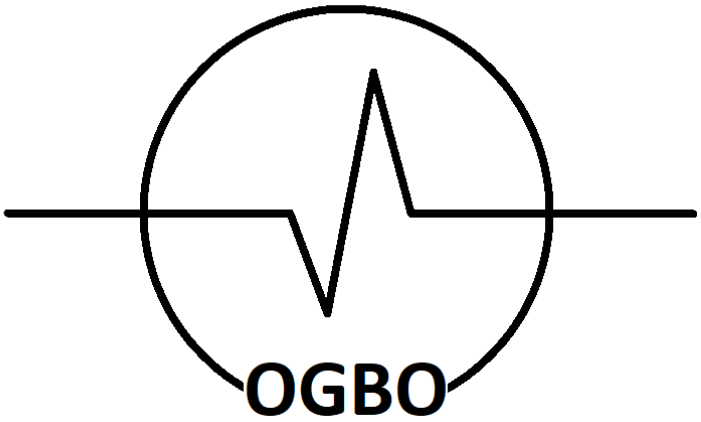
Lycée LA FAYETTE Clermont-Ferrand



Outil de Gestion de Bloc Opératoire en milieu hospitalier

Projet ISN Terminale S

Lucile VELUT Mathias BALLAND Paul NAUTRE
TS2

Table des matières

Présentation d'ensemble du projet

L'Outil de Gestion de Bloc Opératoire est un utilitaire multiplatforme (windows, OSX, Linux) permettant la coordination de l'exploitation d'un parc de blocs opératoires dans une structure hospitalière régulière. Il a été créé selon les besoins de la clinique de Saint Omer du groupe ELSAN mais est théoriquement adaptable à tout hopital ou clinique français.

Plus qu'un unique programme, OGBO est un ensemble d'utilitaires assistant l'exécution de tâches administratives jusqu'alors traitées en papier. Il se découpe en trois parties, ditsétages,correspondant aux différentesétapes d'une procédure d'affectation de salles à un praticien.

Le premiere étage consiste en une interface servant de formulaire dedemande de réservation de bloc. Celui-ci est rempli par le chirurgien prévoyant d'opérer un patient. Le second étage est un outil graphique de construction de planning assisté par ordinateur. Il traite automatiquement les demandes collectées pour planifier l'exploitation des salles sur quatre semaines glissantes puis permet au cadre de bloc de l'établissement de vérifier, modifier et valider ce programme préconstruit. Le troisieme étage correspond au suivi en temps réel de l'avancement des interventions en cours dans l'établissement. Il se divise en deux interfaces. L'une, manipuléepar un infirmier de salle dans un bloc, sert à collecter les informations sur l'intervention en cours dans la salle. L'autre, ouverte en vue, affiche le statut de chaque salle. Cette partie du système est comparable à un système de tableau d'affichage d'aéroport. A l'issue d'une intervention, les informations utiles la concernant sont archivéesafin de servir àune étude des performances de l'ensemble de la procédure.

Etude du besoin

Notre client

Lorsqu'un exercice scolaire appelle à développer un projet, une difficulté majeure est de décider de ce projet. Cela est tout particulièrement vrai dans le cas où le projet en question doit présenter un intérêt à l'usage et une contrainte technique. Si une idée soudaine peut sembler brillante, il est très compliqué d'en faire un projet réalisable et réaliste sur le plan fonctionnel. Dans notre cas, orienter notre logiciel vers le milieu hospitalier était très risqué puisqu'en tant que lycéen nous n'avons qu'une connaissance très limitée des exigences de ce milieu. Pour éviter tout éparpillement nous avions besoin d'un œil expert en la matière, un client. Nous avions, parmi nos contacts, Benoit NAUTRE, directeur de la clinique de Saint Omer du groupe leader de l'hospitalier privé en France ELSAN. Nous savions que celui-ci travaillait avec ses équipes au développement d'un projet semblable et il a accepté de nous cadrer en nous partageant les attentes qu'il adressait au service informatique de son établissement. Durant nos trois mois de travail, nous avons beaucoup échangé avec monsieur NAUTRE. Dans un premier temps nous lui avons demandé de nous exposer la commande du bloc de direction, les solutions techniques envisageables (en termes de stockage des données notamment) et les premières projections et maquettes de l'outil. Cela s'est fait au cours d'une visioconférence de groupe organisée sur un temps de travail en classe. Dans un deuxième temps nous avons fait vérifier et valider un ensemble de maquettes que nous avions constitué, en essayant d'insuffler une touche d'amélioration fonctionnelle aux exigences déjà très complètes de la clinique. A l'issue de notre travail de développement, nous avons présenté et fait valider à nouveau notre produit, cette fois achevé, à monsieur NAUTRE, qui nous a encore offert de son temps pour nous expliquer, en sa qualité de docteur en gestion hospitalière, les enjeux de la création d'outils dans ce secteur.

Monsieur NAUTRE nous a été d'une aide inestimable : toutes les informations qui suivent dans cette section étude du besoin n'ont pu être collectées que grâce à lui.

Procédure hospitalière classique

En France, on critique beaucoup les procédures administratives. « Procédurier » est même un reproche dans la langue française. Pourtant une procédure a par définition vocation à cadrer une action courante afin de l'optimiser. Plutôt que de laisser chacun essayer de résoudre une situation de son côté, on crée un protocole englobant le groupe entier et menant à une résolution à tous les coups et en un nombre

D'étapes connu. Le problème n'est pas la procédure mais les individus qu'elle concerne : les humains sont créatifs, ingénieux mais trop maladroits pour exécuter ces tâches prédéfinies. De plus, le support papier qu'ils utilisent n'est pas adapté : il s'abîme, se perd… C'est là qu'intervient le miracle de l'informatique. Pour ce qui est du facteur humain, il ne le remplace pas mais l'accompagne. Quant au papier, il le remplace pour le coup tout à fait, pour un format plus solide, et plus sûr : la donnée numérique.

Dans la clinique de Saint Omer à Blendecques, la répartition des salles entre praticiens fait l'objet d'une procédure : le cadre de blocs prévoit un planning bihebdomadaire de répartition des salles, il définit qu'un jour j1 d'une semaine s1, un bloc b1 est attribué à un chirurgien c1 d'une heure h1 à h2. Un chirurgien qui compte opérer un patient en informe le cadre de bloc. Celui-ci trouve alors une salle à une date adaptée pour placer cette intervention. Le patient est contacté, l'intervention a lieu au jour et à l'heure et dans le délai prévu. Ainsi la salle est libérée selon le planning pré établi. En théorie cet enchaînement est simple et fluide, en pratique il est d'une réelle complexité. En effet, de nombreux facteurs d'erreurs et de confusions interviennent, à tous les niveaux : le chirurgien peut fournir des informations incomplètes au cadre de bloc, le cadre de bloc peut omettre une demande, la durée de l'intervention peut être incertaine… De plus, la tâche du cadre de bloc est très lourde, sujet à conflits et constitue une perte de temps récurrente.

Informatisation de la procédure

Le processus d'informatisation implique une transition technique mais les acteurs de la procédure doivent être conservés. Ici, la procédure n'affiche aucune incohérence. Par conséquent, elle ne doit pas être remise en cause en elle-même et c'est l'outil numérique qui doit s'y adapter. D'où l'architecture en trois étages de notre outil, pensé selon l'organisation préexistante. Pour le reste, les fonctionnalités de l'outil correspondent au besoin d'accompagnement de l'individu qui le manipule. Pour le premier étage, il apparait que le chirurgien ne sait pas nécessairement précisément quelles sont les informations exactes à transmettre lors de sa demande de réservation. Pour cela, le formulaire le guide. Sont demandés : le nom, numéro de sécurité sociale du patient, l'acte opératoire envisagé, le nom du praticien lui-même. Le formulaire propose également un champ facultatif « note au cadre de bloc » dont celui-ci pourra tenir compte au moment venu. De cette façon, l'usage d'un outil informatique ne limite pas la communication entre membres des équipes. Une observation du fonctionnement des salles d'opérations montre que si un praticien en particulier met un temps toujours similaire pour une même opération, il n'est pas dit

Qu'un autre nécessite un temps égal pour cette même opération. Ainsi, la durée de l'intervention dépend de l'intervention et du praticien. Lorsqu'il remplit le formulaire, le praticien n'a pas à préciser le temps qu'il prévoit pour l'intervention car celui-ci est automatiquement déduit du reste des informations données.

Pour le second étage, la construction du planning se fait dans un premier temps automatiquement : le programme place les interventions dans le temps en respectant les plages attribuées à chaque praticien. Ce premier travail entièrement automatique épargne au cadre de bloc une tâche longue et fastidieuse. Cela dit, il est trop aléatoire pour constituer un outil autonome et le cadre de bloc doit pouvoir le modifier. Il a à sa disposition une interface. Celle-ci lui permet d'une part de décider des plages attribuées aux praticiens, qui sont les paramètres de l'algorithme de traitement, d'une autre part de lire, modifier et valider le planning automatique. Ici l'initiative de notre groupe a été de proposer pour cela un outils graphique coloré de type « drag and drop ». Par cette interface, le cadre de bloc est le seul acteur du processus ayant tous les droits. C'est-à-dire qu'il peut déplacer, replacer, supprimer une intervention et outrepasser les plages prédéfinies.

Pour le troisième étage, les exigences de la clinique étaient simples : ergonomie, lisibilité. Dans toute équipe professionnelle, la circulation de l'information est primordiale, mais lorsque cette équipe est plongée dans un environnement stressant, comme peut l'être un service de chirurgie, l'information se perd rapidement. D'où l'intérêt d'étendre l'outil à cette dernière phase de la procédure.
La première interface que nous avons développée permet à l'infirmier de bloc d'informer sur le statut de l'intervention à laquelle il assiste. Sur un seul écran, il doit alors savoir qui est le patient, qui est le chirurgien, quel type d'intervention est pratiqué. Par un système de clic minimal, il doit pouvoir informer l'extérieur sur l'avancement de l'acte : installation, induction, incision, fermeture, sortie. Si besoin, il doit avoir la possibilité de prévenir d'un retard, d'en donner une estimation et une raison. Enfin, si le patient ou le chirurgien de se présente pas, il a la responsabilité d'annuler l'intervention, ce qui condamne le temps prévu pour celle-ci. Il est précisé par la direction que l'outil n'a pas à prévoir les cas d'incidents majeurs qui déclenchent d'autres protocoles concernant d'autres acteurs. A l'extérieur des salles, une autre interface doit afficher toute ces informations collectées. C'est là que monsieur NAUTRE nous a donné l'exemple d'un tableau d'affichage et c'est sur ce modèle que nous avons travaillé. La direction voulait un affichage coloré, schématique et complet. C'est à partir de ces informations que nous avons mis en place les premiers modèles et maquettes.

Outils et stratégie

Bien répondre à un besoin, cela commence par le comprendre. Au-delà des spécificités propres au milieu hospitalier, Monsieur NAUTRE nous a fait comprendre le besoin omniprésent de la création d'outils en entreprise. Un outil correctement conçu pour une entreprise devient lui-même une part de l'entreprise : il participe à son fonctionnement et mesure son efficacité. Dans notre cas par exemple, l'outil OGBO participe à l'organisation du service de chirurgie et en mesure les performances en communicant et en sauvegardant les informations le concernant. Si l'outil avait été conçu pour une autre entreprise ou pour d'autres besoins, la participation et les mesures n'auraient pas correspondu aux réalités de la clinique et auraient par conséquent eut un impact négatif sur le service. D'où l'intérêt pour une structure de développer un outil pour elle-même plutôt que d'en adapter un autre, si performant soit-il dans son environnement propre. A présent, sur les caractéristiques de l'outil, les spécialistes du management NORTON et KAPLAN ont montré qu'un outil de gestion en entreprise devait présenter trois axes : l'assistance à l'optimisation financière, le support de processus internes et l'assurance de la satisfaction client. Evidemment, ces aspects sont intimement liés. Soit, OGBO dans ses meilleurs jours : un patient est convoqué à 08:00. Le service étant tout à fait coordonné, l'opération débute à 08:30 et se termine au bout de 1h selon le planning. La fluidité du process a satisfait le patient et n'a entrainé aucun surcoût. On peut affirmer que l'outil est parfaitement fonctionnel, il s'inscrit dans le *BalancedScorecard,* la méthode de gestion équilibrée développée par les deux chercheurs cités plus tôt.

Collaboration et répartition des taches

Travail Commun

Le projet étant avant toute une action de groupe, il est nécessaire qu'il soit longuement réfléchi communément. Après le choix du sujet, cette réflexion se fait principalement autour de l'étude du besoin. Il s'agit là que chacun d'entre nous se fasse une idée claire est précise de ce à quoi aboutirait notre collaboration, de quelles en seraient les difficultés majeures et de prévoir comment faire face aux défis techniques. La composition de notre groupe n'appelait pas à la mise en place d'une hiérarchie mais plutôt à celle d'un dialogue actif mû par une transparence totale. Pour cela, nous avons dans un premier temps longuement discuté de vive voix entre nous, puis nous avons synthétisé nos aboutissements à l'écrit. Nous avons créé un groupe sur la plateforme Messenger pour échanger sur nos avancements individuels et nous nous sommes assurés d'échanger au maximum le contenu que nous créions au fur et à mesure. Au cours du développement, nous nous sommes assurés que les recherches que nous menions chacun de notre côté, sur les interfaces graphiques notamment, profitaient à tous. Etant donné le volume de code que nous avions à produire, il était capital que chacun respecte dans son travail un ensemble de normes d'écriture de sorte à rendre chaque production tout à fait lisible et exploitable par chacun. En ce sens même nos tâches individuelles étaient imprégnées de leur inscription dans une démarche de groupe.

Tâches individuelles

Notre projet étant complexe et le temps imparti relativement faible, la division du travail en était un aspect essentiel. Puisque nous étions trois, il nous a semblé dans un premier temps cohérent de nous attribuer à chacun un étage de l'outil tel que nous les définitions plus tôt. Mais cette première organisation a vite montré ses limites : en premier lieu nos niveaux de pratique très hétérogènes impliquaient une efficience peu probante de nos temps de travail. Ensuite il fallait à chacun engranger beaucoup de connaissances sur toutes les solutions techniques envisagées (base de données composante graphique, POO…) ce qui était évidemment peu optimal. Pourtant, une seconde répartition qui aurait voulu que chacun se concentre sur l'intégration de l'une de ses solutions à chaque étage aurait obligé un travail déséquilibré et chronologiquement incohérent. Nous avons finalement opté pour une répartition plus réfléchie et correspondant plus exactement à l'équipe que nous formions. Lucile VELUT a usé de son sens de l'esthétisme pour designer et développer l'interface

D'affichage d'utilisation des salles. Cela impliquait pour elle une maîtrise de la bibliothèque TKinter et de l'architecture algorithmique particulière dite orientée objet. Elle a également produit le formulaire de demande de réservation de salle de l'outil. Mathias BALLAND, qui ne s'attaquait à la programmation que depuis le début de l'année a eu à faire de rapides progrès pour manier les agencements graphiques et l'algorithmie évènementielle de sorte à écrire l'interface de communication propre à ce même troisième étage selon la maquette qu'il avait préalablement produit. Paul NAUTRE, fort d'une expérience plus approfondie, s'est concentré sur l'architecture des données traitées par l'utilitaire et a développé l'outil graphique de création de planning, second étage de l'ensemble OGBO.

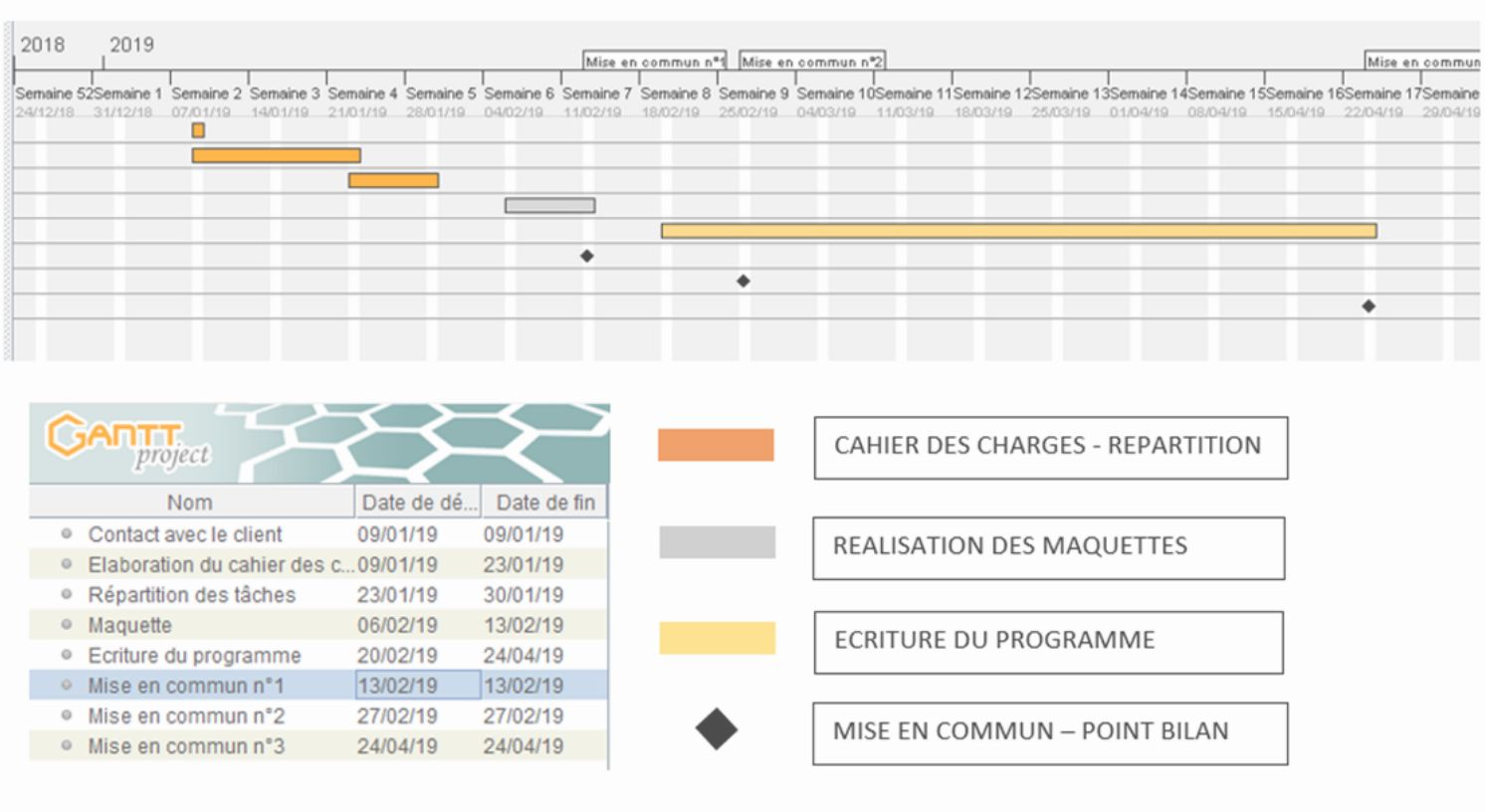


Diagramme de Gantt du Projet

Gestion des données

Contrainte sur les informations traitées

Le fonctionnement de l'OBGO implique l'exploitation de très grandes quantités de données. Celles-ci peuvent être classées en trois catégories selon leurs utilités : les informations « de base » sont celles propres à l'établissement et sont peu variables dès lors du déploiement de l'outil : liste des praticiens et des interventions principalement. Il est nécessaire de créer des fenêtres de modifications pour ces valeurs mais celles-ci sont en marge de la procédure.

Les informations « paramètres » sont celles qui régissent le fonctionnement de l'outil. Elles sont définies par l'utilisateur mais uniquement lues par le programme. Elles correspondent aux caractéristiques variables de l'établissement, comme les horaires de fonctionnement des blocs. Si ces valeurs venaient à ne plus être accessibles ou lisibles par les programmes, alors ils dysfonctionneraient, d'où l'importance de prévoir une vérification de la cohérence des paramètres. On y applique la loi de Murphy : « *Tout ce qui est susceptible d'aller mal, ira mal* » ; ici, si un paramètre peut être incohérent, il finira nécessairement par l'être.

Enfin les informations « circuit » sont celles au cœur du système : demandes de réservations, contenus du planning, statuts des salles… Elles constituent un flux permanent et constant transitant entre les étages du système. Contrairement aux paramètres, elles ne sont individuellement pas nécessaires au fonctionnement de l'algorithme mais la procédure est telle qu'on ne peut tolérer aucune perte : si une demande de réservation est omise, le programme ne plantera pas mais sa tâche ne sera pas remplie.

Une dernière contrainte commune à ces trois types d'informations est l'accessibilité généralisée : puisque l'outil est un ensemble de programmes fonctionnant simultanément sur plusieurs postes, les données doivent être centralisées sur un poste du réseau local, un serveur tel qu'on en trouve dans la plupart des structures hospitalières. Cela impose l'utilisation d'une base de données, nous avons choisi MySQL.

Notre solution MySQL

MySQL est un système de gestion de base de données, c'est-à-dire un logiciel permettant de stocker et manipuler de grands volumes d'informations organisées en tableaux. Concrètement, un programme connecté comme client à une base de données peut créer de nouveaux tableaux, supprimer, modifier ou lire ceux existant.

Cela se fait à l'aide de requêtes formulées dans un langage particulier. MySQL

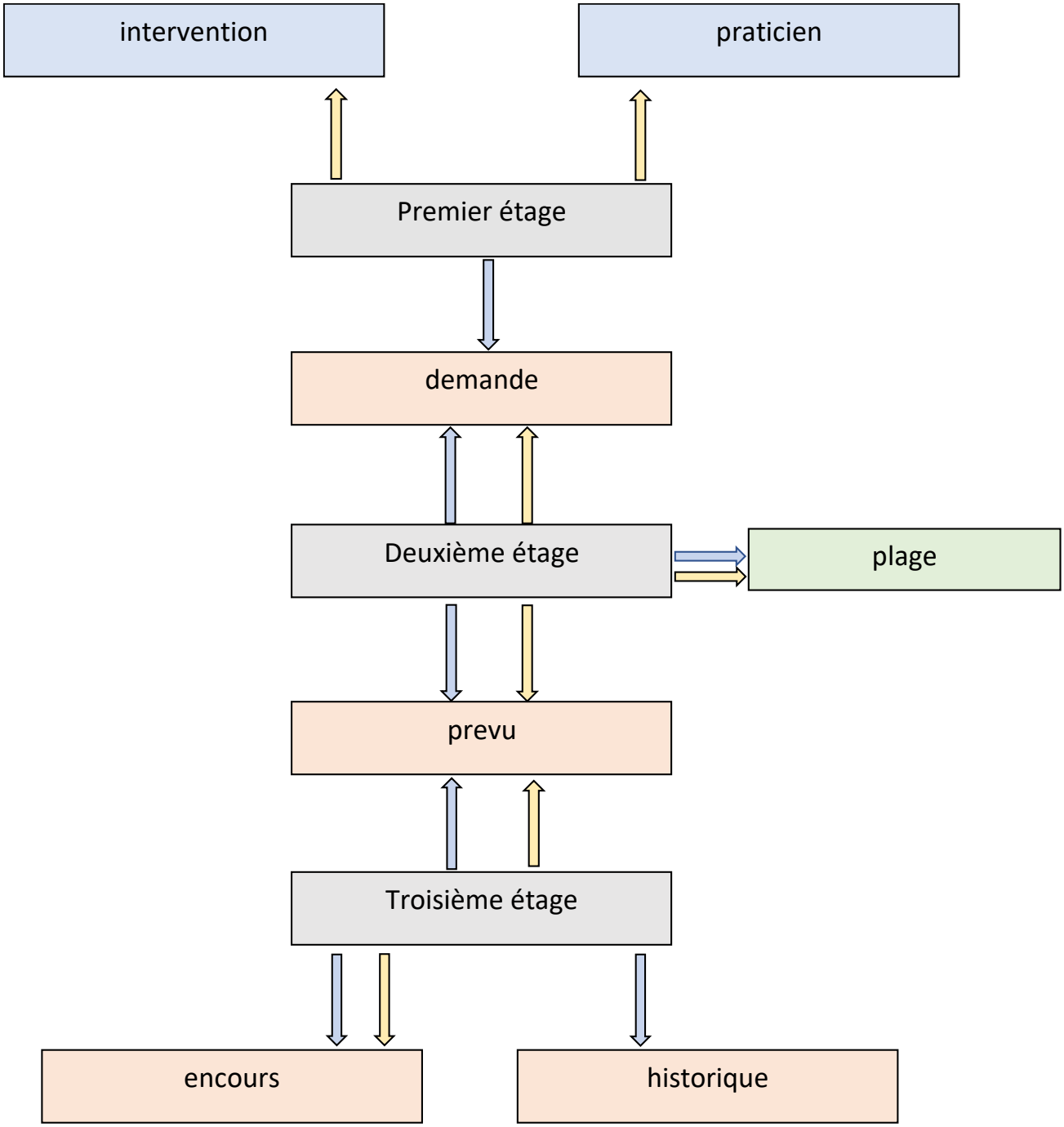
Présente les avantages d'être libre (utilisable gratuitement temps que le programme n'est pas vendu), compatible avec python via une bibliothèque officielle, relativement simple d'utilisation, et suffisamment courant pour jouir d'une large communauté ce qui facilite la prise en main. Une fois le logiciel configuré sur un réseau, un programme peut s'y connecter à l'aide d'un couple utilisateur/mot de passe puis y travailler, via les tables des bases de données en mémoire.

Architecture de données

En théorie, il serait possible de travailler en concentrant toutes les informations dans un seul tableau. Mais celui-ci serait alors extrêmement confus et les programmes l'exploitant devraient l'être aussi. La réflexion autour de l'architecture des données est donc primordiale et doit précéder tout développement. Un schéma permet de détailler celle que nous avons pensée.

Les cadres gris représentent les programmes de l'outil et les flèches leurs interactions avec les tableaux de données. En jaune, la lecture, en bleu l'écriture. Les tableaux bleus contiennent les informations que nous définissions comme de base. Le vert représente les paramètres et le orange les informations circuits. Ainsi, un coup d'œil permet de comprendre comment chaque type d'information est lié à l'utilitaire. En lisant les types d'interventions et les praticiens de l'établissement et en mettant cela en corrélation avec les formulaires remplis, le premier étage écrit la liste des demandes. Celle-ci est lue et vidée par le second étage qui les traite pour les injecter dans le planning, tableau « prévu ». Le traitement prend pour paramètres les plages définies part le cadre de bloc via cette même interface (d'où la double lecture/écriture). De même, le tableau des interventions prévues est continuellement lu et vidé par le troisième étage qui inscrit leurs évolutions dans le tableau « encours » avant de les archiver définitivement dans l'historique. Il est à noter que ce schéma ne répertorie que le programme directement intégré à la procédure, nos trois étages. En vérité ils sont entourés d'un nuage de petits programmes de déploiement ou de marge, comme ceux permettant de compléter les listes d'interventions et de praticiens ou celui chargé de la lecture de l'historique. Nous reviendrons dessus dans la prochaine partie.

OGBO


intervention
praticien
Premier étage
demande
Deuxième étage
plage
prevu
Troisième étage
encours
historique

Déploiement et intégration

Après cette approche théorique, il s'agissait d'acquérir, via quelques recherches, la maitrise de MySQL et des portes qu'il existe entre ce logiciel et l'univers Python, les bibliothèques. En programmation, une bibliothèque est un ensemble de fonctions et de modèles d'objets informatiques souvent sous licence permettant d'utiliser des fonctionnalités variées sans avoir à les créer entièrement. Elles sont très variées (bibliothèques graphiques, systèmes, matérielles…) et généralement incontournables (ici, réécrire les outils intégrés à la bibliothèque aurait probablement pris une année de plus). Celle qui nous intéresse ici, MySQL Connectors est distribuée par Oracle et est très facile de prise en main du fait de sa popularité qui implique un grand nombre de supports sur le web.

Après l'installation du logiciel et la création d'une base de données pour le projet, la première chose à faire est logiquement de créer tous les tableaux nécessaires. Pour cela on écrit les « programmes de déploiement ». Comme leur nom l'indique, ils servent au déploiement de l'outil soit une seule fois dans un même établissement. Leur exécution est capitale car sans eux l'outil ne peut littéralement pas démarrer. En fait ils sont même nécessaires dès le développement. En effet, la création des programmes de traitement entraine de nombreux tests et ceux-ci doivent être faits avec des tableaux non seulement existants mais aussi souvent remplis. Aussi dans un premier temps les programmes de déploiement remplissent les tableaux créés d'informations exemples puis, plus le développement avance, moins ces informations sont nécessaires car l'outil les crée et les complète de lui-même. En somme, ces programmes sont les seuls à réduire lorsque le projet avance, jusqu'à être minimaux à la clôture.

Pour le premier étage et les deux interfaces du troisième étage, les aspects graphiques ont été développés par Lucile VELUT et Mathias BALLANT et la « connexion » à l'outil a été réalisée par Paul NAUTRE. Il s'agissait donc bien d'un travail d'intégration dans le sens où les solutions de gestion des données devaient être adaptées aux programmes existants. On retrouve ici l'intérêt des normes d'écriture en programmation : un code anarchiquement écrit aurait peut-être pu s'exécuter mais l'intégration aurait été impossible. Dans notre cas, elle n'appelait qu'à un peu de méthode. Dans un premier temps, un pont doit être ouvert entre le programme et la base de données. Ensuite, tous les ordres utiles doivent être contenus dans des variables de sorte à ne pas avoir à être réécrits constamment. Enfin les ordres doivent être complétés et lancés suivant l'algorithme prévu. Aucune autre interaction que celle prévue ne doit avoir lieu dans le schéma précédent. Les erreurs à éviter sont de perdre une information, doubler une information ou corrompre une information (en la rendant fausse ou illisible).

Algorithme d'optimisation du moteur de planning

Objectif

Nous voulons d'OGBO qu'il intègre un assistant informatique au cadre de bloc capable de générer une esquisse de planning. La tâche est relativement simple : lire une demande de réservation, la placer dans la liste des interventions prévues en y attribuant un unique triplet salle, jour, heure en fonction des disponibilités et des plages définies, supprimer la demande de réservation et passer à la suivante. Après cela, le cadre de bloc n'aura qu'à ouvrir l'interface du deuxième étage pour relire et réarranger l'ensemble, en prenant notamment en compte les notes laissées par les praticiens que l'assistant aura ignorées. Les droits du cadre de bloc seront supérieurs à ceux de l'assistant, puisqu'il pourra supprimer définitivement une intervention et outrepasser les plages.

Algorithme

Une difficulté ici, au-delà de l'aspect mathématique de l'algorithme, est le format de la donnée. Pour l'ordinateur, les dates, les horaires et les noms contenus dans la base de données ne sont que des suites de caractères. C'est très humain de savoir que le 31 décembre 2018 vient avant le 1er janvier 2019. En revanche, assurer la compréhension des informations par le programme exige beaucoup de traitements, de décompositions de chaines, etc… Cela dit, cette partie est relativement peu intéressante, nous détaillerons donc l'algorithme en considérant cette partie comme acquise.

D'une part nous avons donc un ensemble de demandes, chacune caractérisée par un praticien, une date de formulation et une durée prévue. D'une autre part, nous avons un ensemble de plages, chacune caractérisée par une salle, un praticien, une heure et une durée. À partir de cela nous cherchons à constituer un ensemble d'interventions chacune caractérisée par une salle, un praticien, une heure et une durée. Le traitement s'effectue alors ainsi :

```
Considerer la demande la plus ancienne
parcourir les plages attribuées au même praticien et dont la durée est plus longue que la durée prévue
    Si il existe une intervention sur cette plage
        Si la fin de cette intervention est prevue avant la fin de la plage
            Si le delai entre la fin de l'intervention et la fin de la plage est plus grand que la durée demandée
                Prévoir une nouvelle intervention avec le praticien, la salle, l'heure de fin de l'intervention et la durée demandée
                Supprimer la demande
    Sinon
        Prévoir une nouvelle intervention avec le praticien, la salle, l'heure de début de la plage et la durée demandée
        Supprimer la demande
```

Simulation et test

Par manque de temps, nous n'avons pas développé de version fonctionnelle du moteur de planning. En revanche, nous avons écrit un programme similaire, exécutant exactement la même tache mais sur un format de données simplifié. Ce programme a ainsi permis de vérifier la cohérence de notre algorithme en contournant toute la complexité des bases de données. Ce moteur miniature travaille avec trois types d'objets très simples simulant les propriétés des entités demandes, rendez-vous et plages contenues dans la base de données. L'algorithme précédemment proposé est résumé en une fonction exécutée en boucle un grand nombre de fois. A l'issu de ces exécutions successives, une partie des demandes a été supprimée et de nouveaux rendez-vous ont été créer. On vérifie le bon fonctionnement du tout en comparant le modèle optimal calculé par le moteur à celui que nous déterminons manuellement. La grande difficulté ici n'est pas de traduire l'algorithme précédant en langage informatique mais de vérifier la cohérence des résultats : pour une seule exécution, il existe plusieurs issues (aucune plage trouvée, plage trouvée mais déjà occupée, plage trouvée mais trop courte etc…). Plus l'information traitée est dense, plus les issus sont nombreuses. La phase de test à elle toute seul à donc pris plus d'une heure mais a mis en évidences les défauts de l'algorithme qui ont pu être corrigés.

```
========== RESTART: C:\Users\paul\Desktop\moteur de planning.py ==========
plage trouvée
aucune interference
plage suffisante: insertion d'un nouveau rendez vous
Suppression de la demande



plage trouvée
interference rdv
plage partiellement libre
plage trouvée
aucune interference
plage suffisante: insertion d'un nouveau rendez vous
Suppression de la demande



plage trouvée
interference rdv
plage partiellement libre
plage trouvée
interference rdv
plage partiellement libre
plage suffisante: insertion d'un nouveau rendez vous
Suppression de la demande



plage trouvée
aucune interference
plage suffisante: insertion d'un nouveau rendez vous
Suppression de la demande



plage trouvée
interference rdv
plage partiellement libre
plage suffisante: insertion d'un nouveau rendez vous
Suppression de la demande



fin de traitement
```

Console d'exécution du moteur miniature

Perspective et diffusion

Idée de développement

Il est rare qu'un outil informatique, si bien développé soit-il, ne subisse quelques mises à jour après sa première diffusion. Par ailleurs, rappelons que OGBO dispose d'une fonction de mémorisation : toutes ses opérations sont gardées en mémoire dans un historique et l'analyse de cet historique après quelques mois de fonctionnements en conditions réelles, à l'aide d'un moteur statistique adapté, pourrait mettre en évidence, si ce ne sont des défaillances, des défauts d'optimalité corrigeables. Les corrections qui suivraient viseraient probablement principalement le moteur de planning : en premier lieu, il faudrait imaginer un algorithme allant plus loin dans la tâche réservée au cadre de bloc, de sorte à la restreindre à une simple relecture. En second lieu et pour aller dans ce sens, il faudrait étendre les paramètres de cet algorithme pour le rendre plus « intelligent ». Par exemple lui faire prendre en compte les lieux d'habitation des patients pour les placements des interventions. Enfin, on ferait véritablement un pas vers l'intelligence artificielle en intégrant l'outil statistique à cet algorithme de sorte à lui faire corriger ses propres lacunes. On parlerait alors de Machine Learning mais ce type d'amélioration exigerait une maitrise bien supérieure des réalités informatiques et mathématiques.

En outre, on pourrait envisager quelques modifications d'ordre esthétique où organiser une étude auprès des acteurs de la procédure pour axer nos améliorations sur les besoins comme nous avons axé le développement sur les exigences de l'établissement.

Question de la licence

Du fait de son ampleur, notre projet pose la question de sa diffusion. Internet regorge de communautés de développement et il serait intéressant de savoir ce que notre outil deviendrait entre leurs mains. A ce stade il serait probablement trop immature pour être commercialisé. Mais il pourrait présenter un intérêt éducatif ou bien être achevé et exploité. Il serait alors regrettable que nos noms ne soient plus mentionnés et oubliés, d'où l'intérêt de le distribuer sous licence. Ainsi, tous ceux qui travailleront dès lors sur OGBO seront liés à nous par un contrat stipulant que nous sommes à l'initiative du projet. Il existe un grand nombre de contrats de licence pré rédigés, il s'agirait donc pour nous d'en choisir un par lequel OGBO et ses variantes seraient libres à la consultation et à la modification mais interdits à la commercialisation. On parlerait alors de licence libre restrictive.

Conclusion

Ce projet aura été l'occasion de diverses expériences et de nombreuses découvertes. Dans le domaine informatique, nous avons abordé et approfondi plusieurs notions. En entrant en communication avec un client et en menant une étude du besoin, nous avons approché de plus près le monde de l'entreprise. Le partenariat avec monsieur NAUTRE, professeur de gestion, nous a ouverts aux champs des sciences de gestion de projet et de la stratégie. Enfin, pour la première fois, nous sommes allés ensemble au bout d'un projet suffisamment complet pour que sa diffusion puisse être étudiée.