UNIVERSITÉ MONTPELLIER 2
SCIENCES ET TECHNIQUES

Université de Montpellier 2

Rapport TER L3

Application de gestion de tâche

Tuteur : M. Seriai
Participant : Cyril BARCELO,
Mohand MAMMA,
Feng LIU

1er Fevrier 2015 —26 Avril 2015

Table des matières

1 Introduction **2**

2 Présentation du projet **3**
2.1 Contrainte et situation initiale . . . 3
2.2 Présentation du sujet . . . 4
2.3 Objectif . . . 4
2.4 Déroulement du projet . . . 4

3 Gestion de projet **6**

4 Outils existants **7**
4.1 Domaines informatique ou le projet se situe : . . . 7
4.2 Outils déjà existants dans ce domaine . . . 7

5 Conception et Réalisation **8**
5.1 Conception . . . 8
5.2 Implémention . . . 13
5.2.1 Modèle . . . 13
5.2.2 Vue . . . 15
5.2.3 Contrôleur . . . 18

6 Conclusion, Perspectives et remerciement **19**
6.1 Avenir de l'application : . . . 19
6.2 Difficulté rencontré : . . . 19
6.3 Remerciement . . . 19

7 Annexes **20**
7.1 Manuel d'utilisation . . . 20

1 Introduction

Le projet de TER EN licence 3 Informatique est très important, nombreux sont ceux qui pensent que ce dernier est l'occasion pour les étudiants de juger leurs acquis des trois années de Licence et de mieux choisir par la suite leurs orientations et spécialités en master. C'est pour cela que le choix du sujet est très important, pour nous le choix c'est porté sur ce sujet de développement d'application d'aide à l'organisation de tâches». De notre point de vue, ce projet présente deux caractéristiques fondamentales : d'une part mettre en oeuvre nos connaissances en algorithmie, programmation orienté objet, base de donnée et les différents langages déjà vu, et d'autre part se rapprocher un peu plus d'un aspect qui est très important dans notre vie quotidienne : la gestion de nos emplois du temps. L'équipe de projet est constitué de 3 membres répartie sur 2 des groupes de la Licence 3 informatique, Cyril Barcelo, Mohand MAMMA, Feng Liu, notre encadrant est Monsieur Seriai Djamel, enseignant à la faculté des sciences de Montpellier2, qui de part son expérience, nous à permis de corriger les défauts de l'application et améliorer certains aspects afin de faciliter son utilisation courante.

Et c'est dans l'objectif de vous présenter au mieux l'application et le travail réalisé le long du semestre que ce rapport a été rédigé, dans une première partie, nous vous présenterons d'une manière légère et efficace ce qu'est un agenda électronique et un planificateur intelligent, ainsi que les objectif de notre projet et ses différentes étapes de réalisation, pour ensuite vous amener à découvrir les différents aspects de la gestion de projet que nous avons employé et les outils déjà existants. Nous continuerons par les différents choix de conception et d'implémentation effectuées. Avant de terminer par l'avenir de l'application et les objectifs potentiellement réalisable sur un moyen terme.

2

Présentation du projet

Nous utilisons depuis bien longtemps l'expression "le temps c'est de l'argent", le gérer d'une manière intelligente et efficace est aujourd'hui devenue plus qu'un objectif mais une obligation.

C'est alors une aubaine qui se présente pour notre domaine de l'informatique, qui a travers divers applications offre certaines solutions et utilités au utilisateurs. Parmi elle celle développé par nos soins à travers ce projet et que nous présenterons dans cette première partie. Mais avant cela observons le schéma qui vous aidera certainement a mieux comprendre l'application et son utilité..

2.1 Contrainte et situation initiale

Des dizaines d'applications ont été développé surtout lors de cette dernière décennie, les objectifs et domaines peuvent être différents mais le point de départ reste le même : le besoin pressant de l'homme d'améliorer son mode de vie. Nous nous s'intéressons pour notre part à un aspect très important dans la vie quotidienne de chacun, la gestion du temps. Divers application informatique existent déjà (nous en citerons Deux entre autres…), qui sont Conçues pour rendre un service à l'utilisateur et qui est l'aide à la gestion de son emploi du temps.

C'est en se basant sur tout ce qu'a été déjà réalisé que nous essayerons de réaliser une application de gestion de tâche des plus complète et d'apporter le maximum d'amélioration et de nouveauté afin de la rendre la mieux possible.

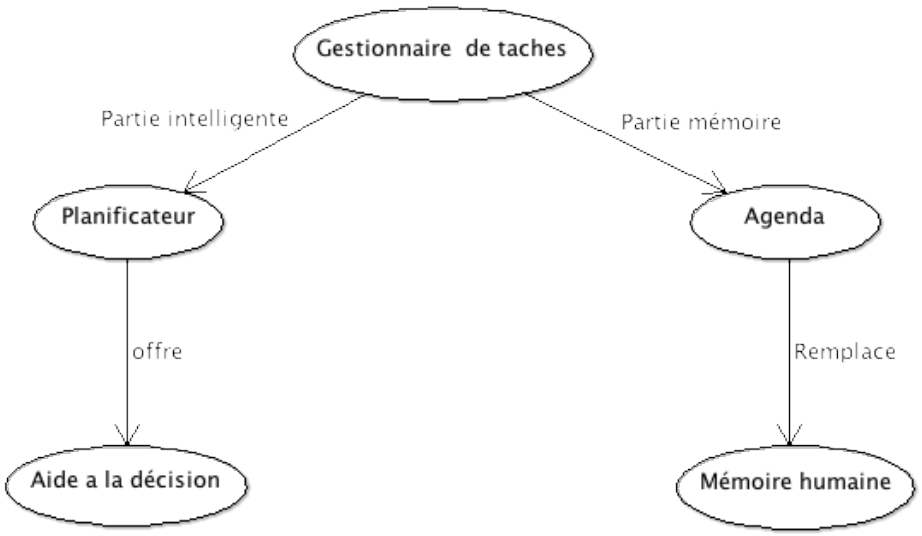


Figure 2.1 : Schéma représentant l'application

2.2 Présentation du sujet

Comme vu dans la figure 2.1, notre application de gestion de tâche est en effet divisé en deux partie importante, que sont l'agenda électronique, mais aussi le planificateur qui représente la nouveauté que nous voulions apporter aux applications déjà existante.

Commençons alors par introduire les deux partie :

1- Agenda électronique : C'est en effet un programme informatique qui permet de récupérer des informations saisi par un utilisateur, de les stocker en mémoire et de les afficher ensuite de divers manières
-Fonctionnalité de l'agenda : Permet d'ajouter une tâche dont les paramètres sont saisi par l'utilisateur dans la mémoire de la machine et de visualiser la liste des tâches précédemment ajoutées suivant l'ordre chronologique.

2- Planificateur : Représente la partie intelligente de notre application, et qui en se basant sur les différents paramètres saisie par l'utilisateur et disponibilité dans l'agenda électronique programme des tâches et les transmet à l'agenda pour ensuite les visualiser.
-Fonctionnalité du planificateur : Permet en suivant une liste de paramètres de proposer des choix de planification sur une période définie.

2.3 Objectif

En partant du simple au plus compliqué, naturellement l'objectif premier est alors la réalisation d'un agenda électronique qui peut être a la fois facile a utilisé mais surtout le plus complet et efficace possible. Pour ensuite passer à l'étape en dessus en essayant d'intégrer la partie planificateur.

2.4 Déroulement du projet

La figure ci-après est un schéma représentant le déroulement de notre projet, dont la première étape aura été de choisir un sujet. Une fois ce choix effectué nous nous sommes renseigné sur les agenda électronique, planificateur, etc, ce qui nous a permit de décidé de la démarche à adopter pour notre développement et de définir les différents composants, déterminer les classes et outils dont nous avons eu besoin. Nous avons donc commencé par faire une maquette de notre application puis le reste du développement à suivi.

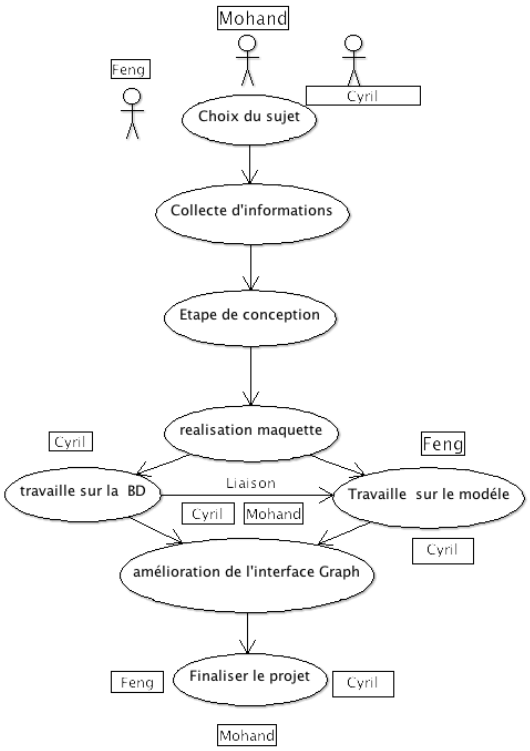


Figure 2.4.1 : Schéma représentant le déroulement du projet

3 Gestion de projet

Afin de mieux nous organiser et de gérer notre temps de manière plus optimale, nous avons, en suivant les conseils de notre encadrant, désigné un chef de projet. Cela nous a permis de prendre des décisions plus facilement sur certain desaccords entre plusieurs membres de l'équipe, c'est Cyril qui a accepté d'endosser ce rôle. Chaque membres du groupe a proposé des idées et des tâches à effectuer pour avancer dans le projet. Par la suite, Cyril réparti l'ensemble des tâches à chaque membres. En se qui concerne la gestion de version et l'échange de documents nous avons utilisé GitHub.

Les réunions avec notre encadrant se faisait chaque vendredi à 13h. Nous nous sommes réunis deux à trois fois par semaine après les cours pour mettre en commun les travaux effectués par chacun et résoudre ensemble les problèmes rencontrés. Voici le diagramme de gantt représentant notre projet :

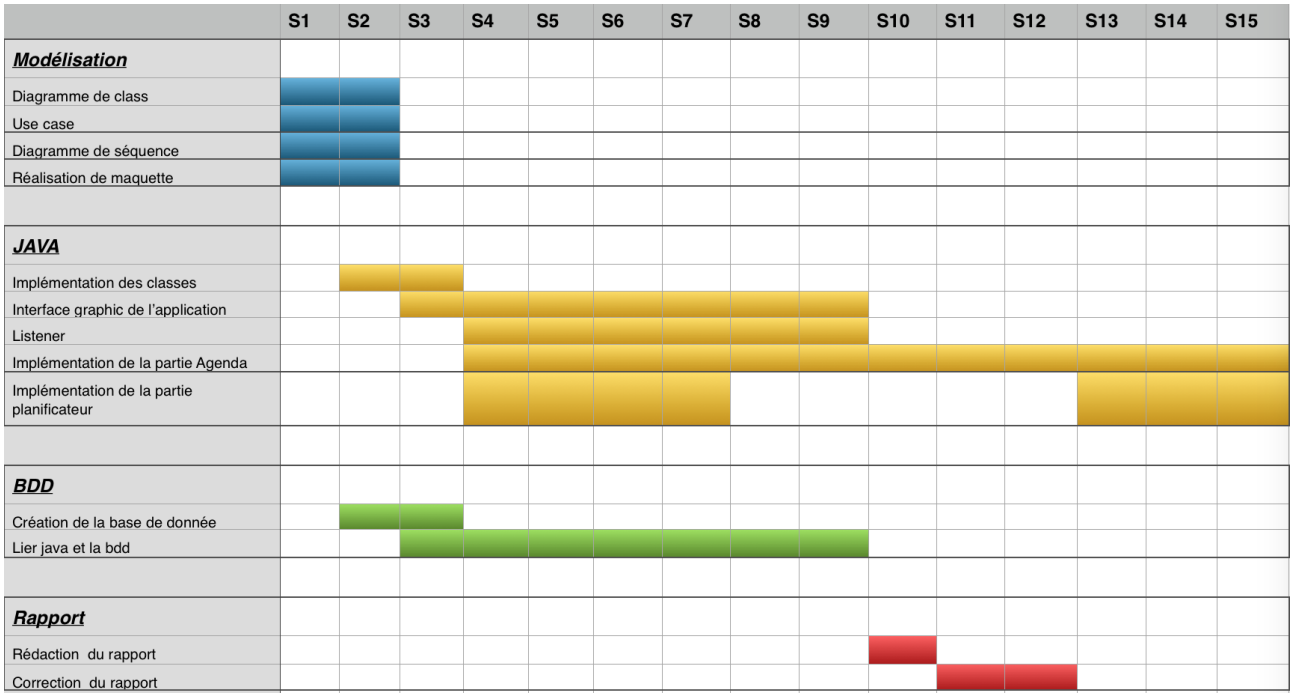


Figure 3.1 : Diagramme de Gantt du projet

4

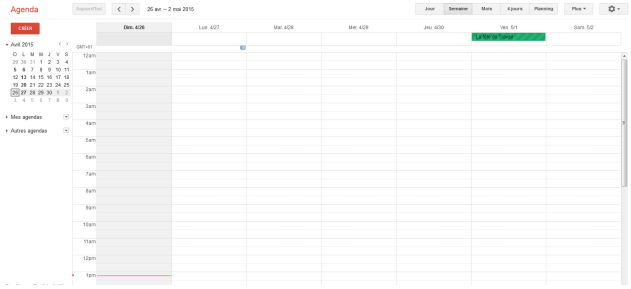
Outils existants

4.1 Domaines informatique ou le projet se situe :

Notre projet se situe dans le domaine des application utilitaire informatique, plus spécifiquement dans les utilitaires de gestion/organisation du temps.

4.2 Outils déjà existants dans ce domaine

-Google Agenda



-Calendrier Outlook

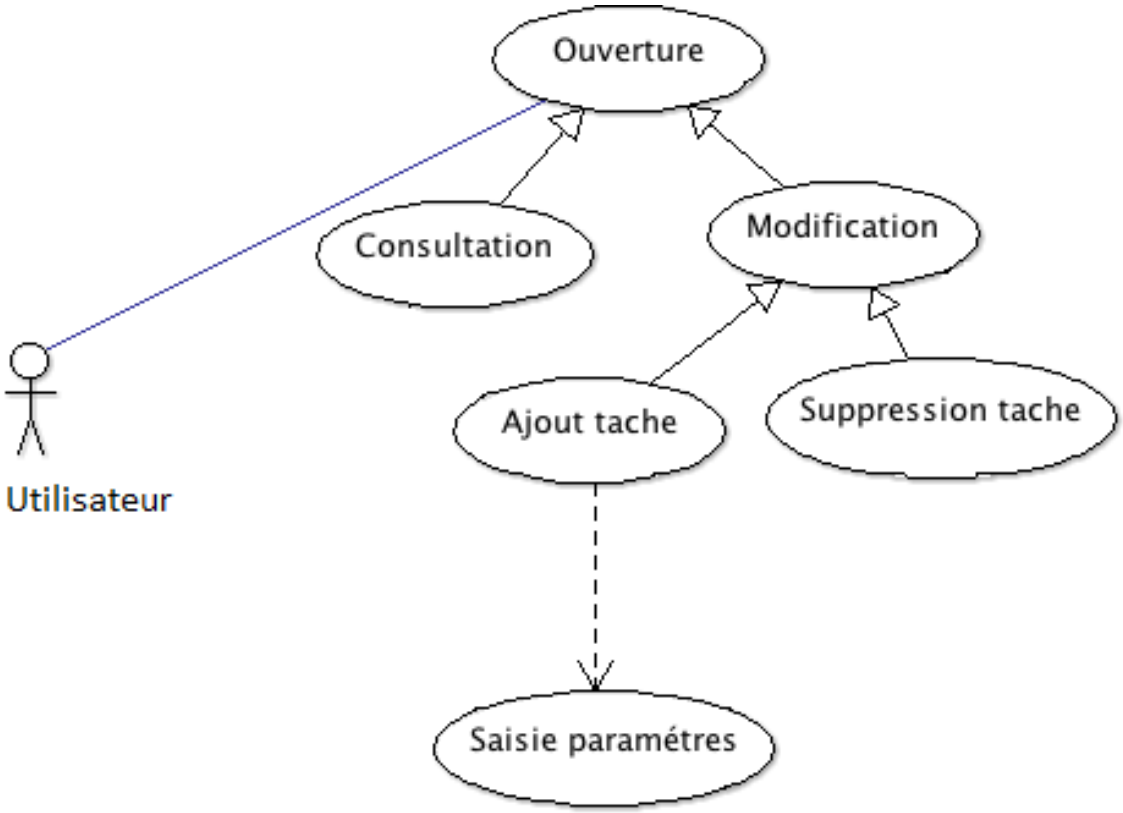
-etc...

5

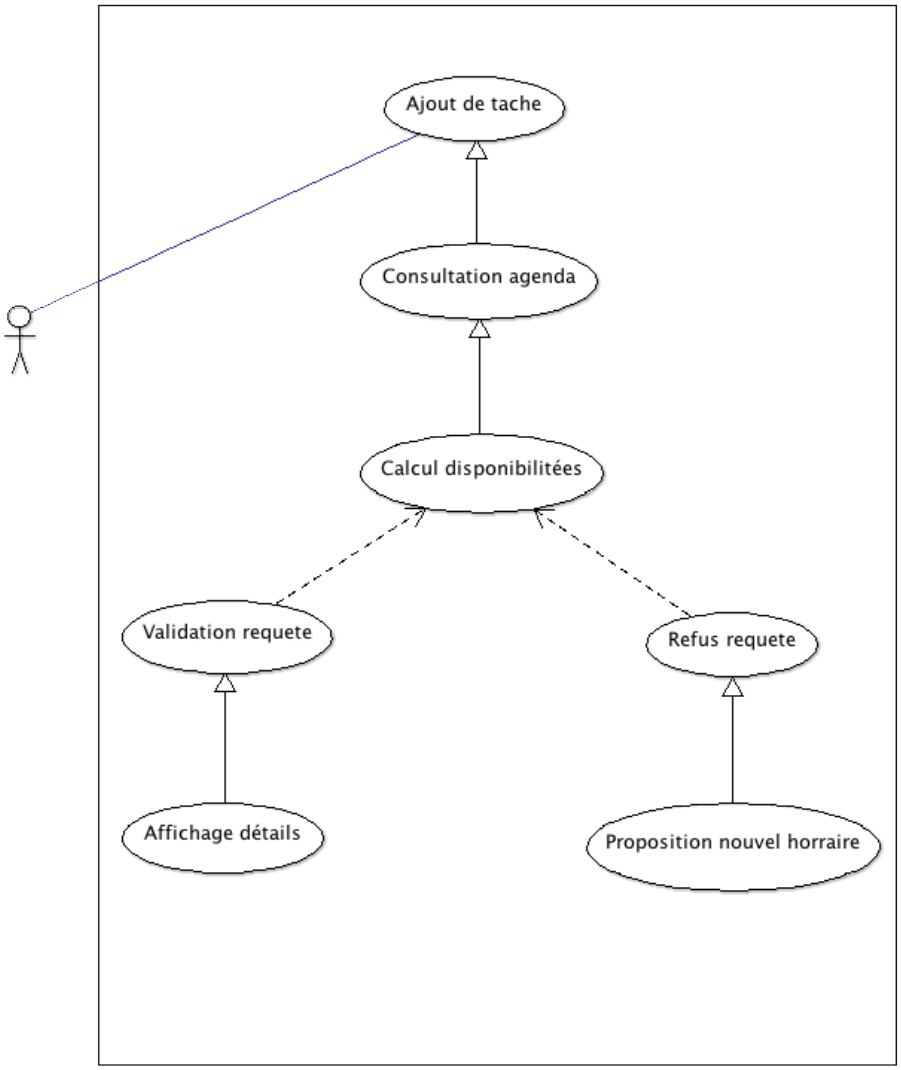
Conception et Réalisation

Nous allons à présent faire une description du travail que nous avons effectué en justifiants les choix important que nous avons dû faire. Dans cette optique, nous commencerons par discuter de l'aspect conception pour ensuite nous occuper de l'aspect implémentation.

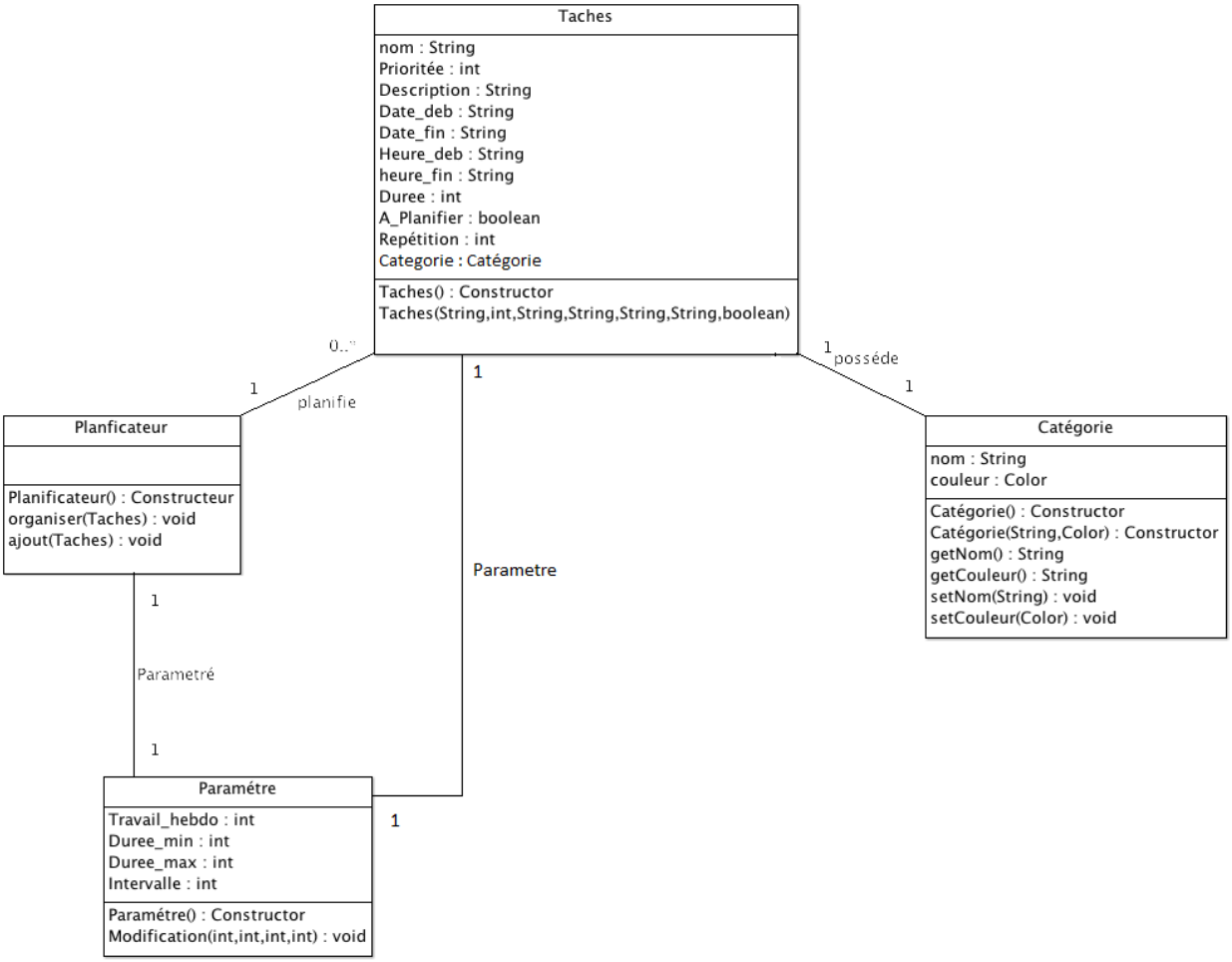
5.1 Conception



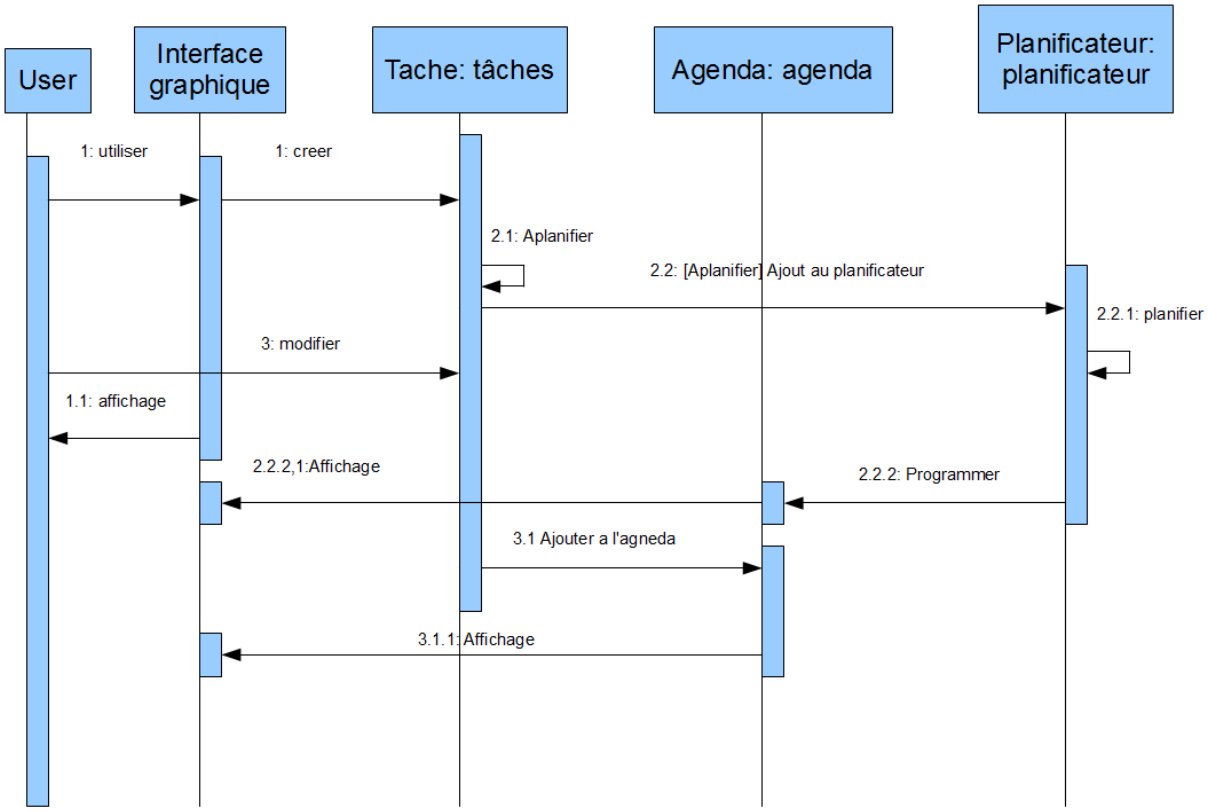
La figure ci-dessus est le diagramme de cas d'utilisation de la partie agenda, ou plutôt de l'application dans son ensemble le planifcateur mis à part. On peut donc voir les possibilités laissées à l'utilisateur une fois l'application ouverte, ce dernier ayant le choix entre une simple consultation de l'agenda ou une modification de sa liste de tâche en ajoutant ou supprimant une tâche de la base de données, dans le cadre de l'ajout de tâche, l'utilisateur doit alors donner les paramètres de cette dernière.



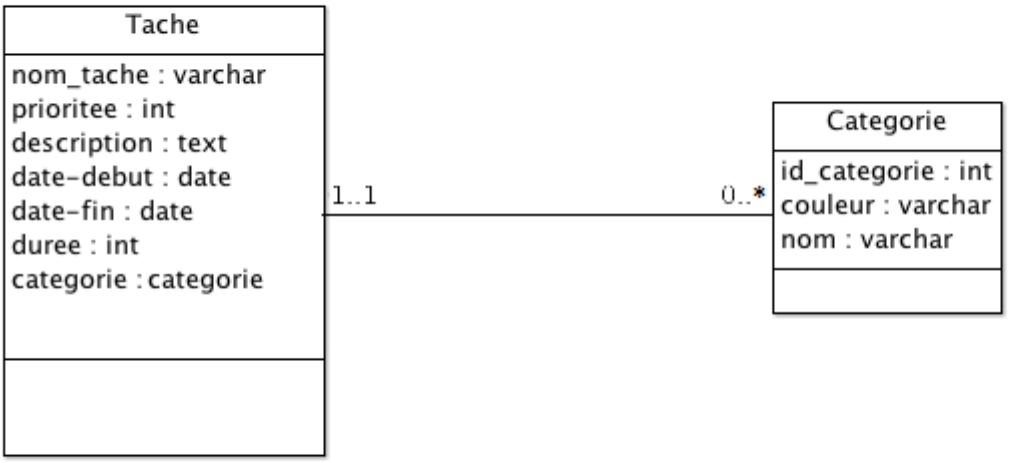
La figure ci-dessus est le diagramme de cas d'utilisation de la partie planificateur, dans cette partie, l'utilisateur sélèctionne une(ou plusieurs) tâche à planifier, l'application va alors consulter les disponibilité et proposer un arrangement à l'utilisateur, ce dernier peut accepter ou refuser la planification proposé.



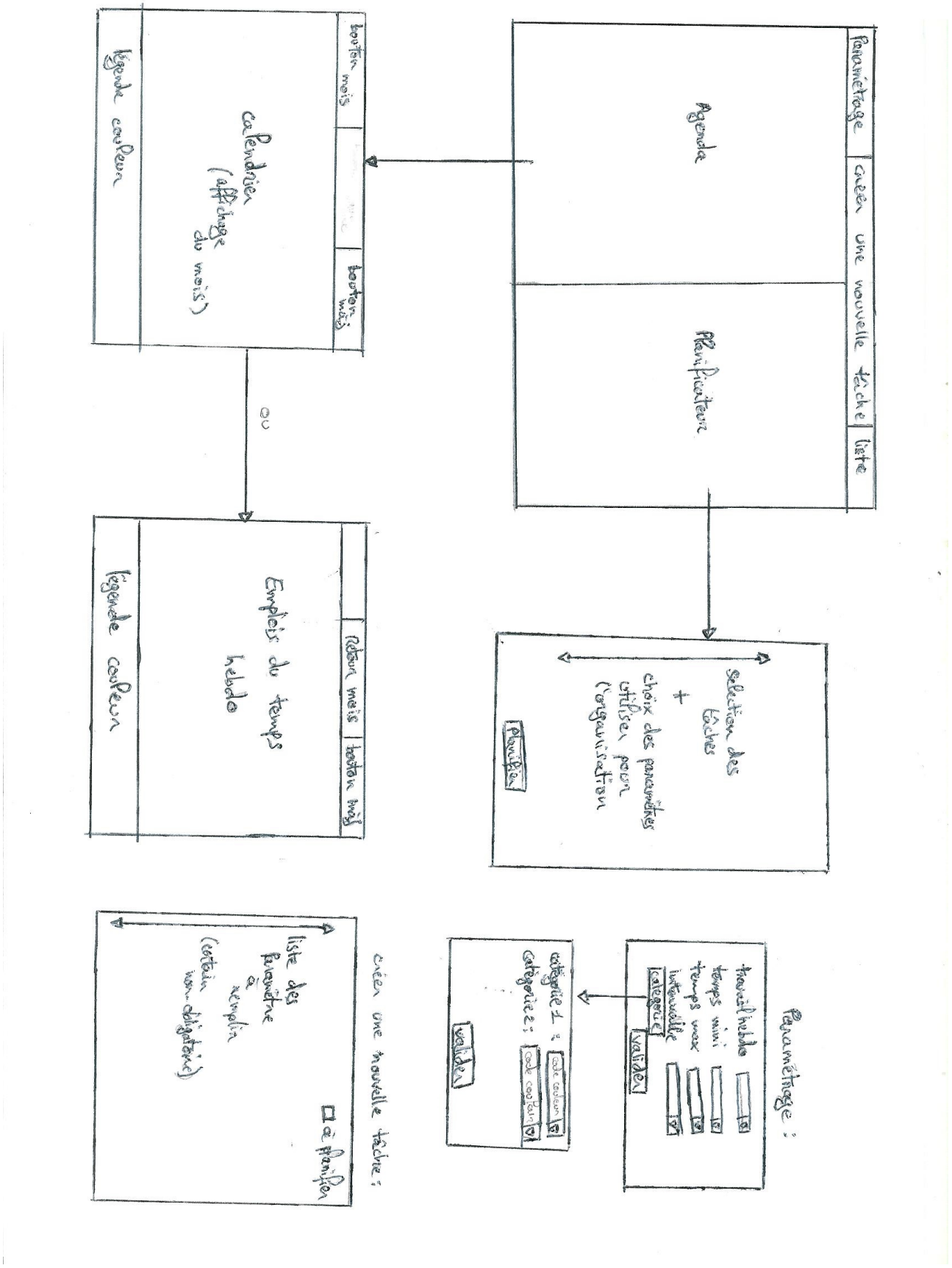
La figure ci-dessus est le diagramme des classes de l'application, cette dernière est donc composé de quatre classes principales : la classe tâches est utilisé pour contenir une tâche extraite de la base de données pour faire des traitements, elle contient un attribut de type Catégorie qui contient le nom et la couleur de la catégorie. La classe planificateur, elle peut prendre jusqu'à n tâches pour les planifier. Et la classe Paramètre permet de paramétrer l'ensemble de l'application en délimitant des durées qui sont utilisé à la création d'une tâche ou à sa planification.



La figure ci-dessus est le diagramme de séquence de notre application. L'utilisateur peut au travers de l'interface graphique créer(ou modifier) une tâche ou demander un affichage(calendrier, liste des tâches à planifier, etc). Lorsqu'une tâche est créer, l'utilisateur peut la marquer comme "à planifier", si tel est le cas, la tâche est envoyé au planificateur et ajouté à la liste des tâches à planifier qui calcule un agencement et le propose à l'utilisateur. Si en revanche, la tâche n'est pas à planifier, elle est directement ajouté à l'agenda.



La figure ci-dessus est un schéma de notre base de données.Cette dernière est composé de deux tables, une pour stocker les tâches et l'autre les catégories. La table de Tâche possède un attribut catégorie qui référence la table Categorie, on voit sur ce schéma qu'une tâche possède obligatoirement une catégorie(et seulement une), et qu'une catégorie peut correspondre à plusieurs tâches.


Paramétrage
créer une nouvelle tâche
liste
Agenda
Planificateur
bouton mois
bouton mois
Calendrier
(affichage du mois)
légende couleur
ou
Emploi du temps
hebdo
légende couleur
bouton mois
bouton mois
Sélection des tâches
+
choix des paramètres utilisés pour l'organisation
Planifier
Paramétrage :
travail hebdo
temps mini
temps max
intensité
catégorie
valider
catégorie 1 :
catégorie 2 :
code couleur
code couleur
valider
Créer une nouvelle tâche :
liste des paramètre à remplir (certain non obligatoire)
à manipuler

La figure précédente est la maquette de notre application telle que nous l'avons imaginé au commencement, l'objectif premier étant d'offrir une certaine ergonomie à la partie visuel de notre application.

2-Les différentes parties

Présentons maintenant la disposition de chaque partie :

1-créer nouvelle tâche : Cette partie est présenté sous la forme d'un formulaire comportant plusieurs champs (nom, description..), permettant à l'utilisateur de saisir toutes les informations sur la tâche à créer.

2-agenda : un calendrier s'affiche pour le choix d'une date, puis un tableau s'affiche présentant le descriptifs des différentes tâches présentes pour la date choisie.

3-Paramètres : Un formulaire est présenté avec les différents paramètres présent, nous avons aussi introduit le bouton catégorie qui nous renvoie vers une nouvelle fenêtre qui nous permet entre autre de consulter les catégories déjà existantes et d'en créer de nouvelle (avec possibilité de choisir une couleur ...)

4-Planficateur : une liste de tâches présenté en forme d'un tableau, avec possibilité de sélectionner plusieurs tâches et de les soumettre au planificateur (Partie Modèle) pour les planifier.

5.2 Implémention

Nous avons basé notre application sur une architecture MVC(Modèle-Vue-Contrôleur) afin d'apporter au logiciel une meilleure maintenabilité, et également de nous permettre un développement simplifié via un code plus clair, plus aéré et une communication plus facile. En adoptant cette architecture, nous avons donc pu contrôler les interactions entre les classes plus facilement et ainsi éviter toutes interactions direct entre le modèle et la vue, ces liaisons étant effectué dans le contrôleur.

Nous avons donc établit le découpage suivant :

-Modèle : Classes d'accès à la base de données et de traitement.

-Vue : Classes de l'interface graphique.

-Contrôleur : Listeners.

5.2.1 Modèle

La partie modèle regroupe les données, les liaisons avec la base de données et des classes de 'stockage' utilisé pour les traitements.

En ce qui concerne Les données, nous avons décider de créer une base qui est composé de deux tables, une table 'tâches' qui nous permet de stocker les tâches crée par l'utilisateur qu'elles soient planifié ou à planifier et qui contient toutes les informations de ces dernières. La seconde table est la table 'catégories' qui stock toute les catégorie définie via l'IHM par l'utilisateur. Cette dernière nous est nécessaire car les tâches possèdent toutes une catégorie ce que nous avons représenté dans la base par un attribut catégorie qui référence donc la base 'catégories'.

Pour effectuer les liaisons, nous avons décidé d'utiliser l'API JDBC(Java DataBase Connectivity) qui d'après nos recherche était la plus facile en se qui concerne l'utilisation de l'apprentissage par des débutants. Nous avons donc dû apprendre à utiliser cette API un minimum avant de commencé à l'utiliser et poursuivre son apprentissage au fur et à mesure du développement. Cet API donc, nous permet d'insérer des requêtes SQL dans notre code Java. Nous avons alors créé des classes pour effectuer les interactions, la première, ' TâcheJDBC' nous permet principalement d'insérer une nouvelle tâche dans la base, d'en supprimer une ou de récupérer un identifiant. La seconde, 'CategorieJDBC' nous permet d'ajouter une catégorie ou de récupérer un identifiant. Cependant, au fil des améliorations, nous avons était confronté à un problème, en effet certain attributs pouvant être 'null', nous obtenions des erreurs lors de l'appel au fonction comme 'AjoutTacheJDBC'. Pour remédier à ce problème nous avons donc attribué des valeurs spécifiques aux variables correspondant aux attributs null, par exemple pour un String nous utilisons la valeur d'une chaine vide "", et nous avons également ajouté des constructeurs dans nos classes JDBC ou nous effectuons des tests, si par exemple un String contient "", nous envoyons un null à la place dans la requête à l'aide de variable de classe. De cette manière nous avons pu contourné le problème en appelant d'abord un des constructeurs avant les fonctions de liaison. La figure ci-après montre notre constructeur pour les tâches.

```
public TacheJDBC(String str,int i,int j,String str2,String d1,String d2,int k){
    nom=str;
    prio=i;
    categorie=j;
    if(str2==""){
        description=null;
    }else{
        description=str2;
    }
    if(d1.equals(" 00:00")){
        dateDeb=null;
    }else{
        dateDeb=d1;
    }
    if(d2.equals(" 00:00")){
        dateFin=null;
    }else{
        dateFin=d2;
    }
    duree=k;
}
```

Figure 5.2.1.1 : Constructeur paramétré d'une classe JDBC

De la même manière, nous avons rencontré un problème vis à vis du format des dates, en effet la format date JAVA n'est pas le même que le format date de SQL, nous avons donc dû procéder à des conversions (voir la figure suivante).

```
Timestamp DF,DD;
DD=null;
DF=null;
if(dateDeb!=null||dateFin!=null){
   if(dateDeb!=null){
        //DD : datedebut      DF : datefin
        //Timestamp DD = new Timestamp(System.currentTimeMillis());
        DD = new Timestamp(System.currentTimeMillis());

        //String DDstr = dateDeb+":00";//String saisi par utilisateur
        String DDstr = dateDeb+":00";

        try{
            //DD = Timestamp.valueOf(DDstr);
            DD = Timestamp.valueOf(DDstr);
        }catch (Exception e){
            e.printStackTrace();
        }
   }
```

Figure 5.2.1.2 : Extrait de code de conversion de date
exemple de code complet en annexe 1.

Nous avons également créer des classe qui nous servent de stockage, en effet les tâches(ou catégories) étant envoyés directement dans la base de données, nous avons codé les classes leurs correspondant afin de stocker les données récupéré dans la base pour effectuer des traitement. La première classe que nous avons codé fut la classe Tâches qui nous sert à stocker une tâche de la base de données pour effectuer des traitements sur cette dernière comme la planification. Cette classe Tâches contient donc autant d'attributs qu'une tâche peut avoir de paramètre et n'a comme méthode que des constructeurs et des accèsseurs. Nous avons également créé une classe paramètre qui est une classe abstraite, de cette manière nous pouvions utiliser les accèsseurs de chaque attributs lors des tests sans avoir à créer d'instance de paramètre.

5.2.2 Vue

Le développement de la partie Vue c'est déroulé en plusieurs étapes, tout d'abord, nous avons réalisé un interface minimum afin d'avoir une base et de pouvoir commencer le reste du développement. Pour se faire nous avons décider d'utiliser la librairie graphique de java Swing

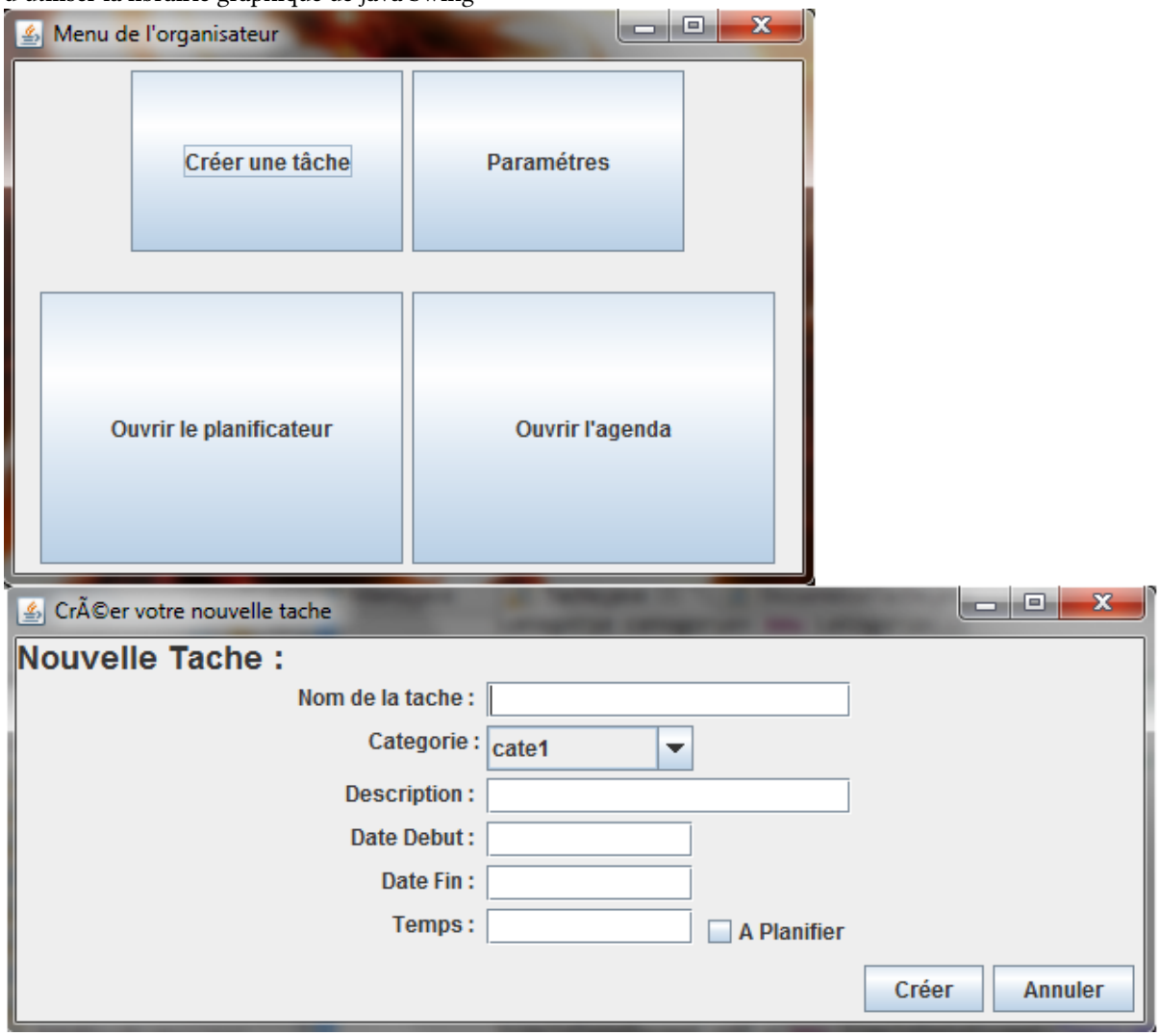


Figure 5.2.2.1 : Première version de l'interface graphique

La figure ci-dessus est un screenShot de notre première version de l'interface, le but de cette version était donc de nous permettre de créer une tâche, d'accéder au paramètre, etc. Nous nous sommes donc par la suite intérressé d'avantage à l'ergonomie qui n'était pas vraiment là sur la première version, nous avons donc changé cette fenêtre avec des gros boutons qui ouvré d'autre fenêtre et rassemblé le tout.

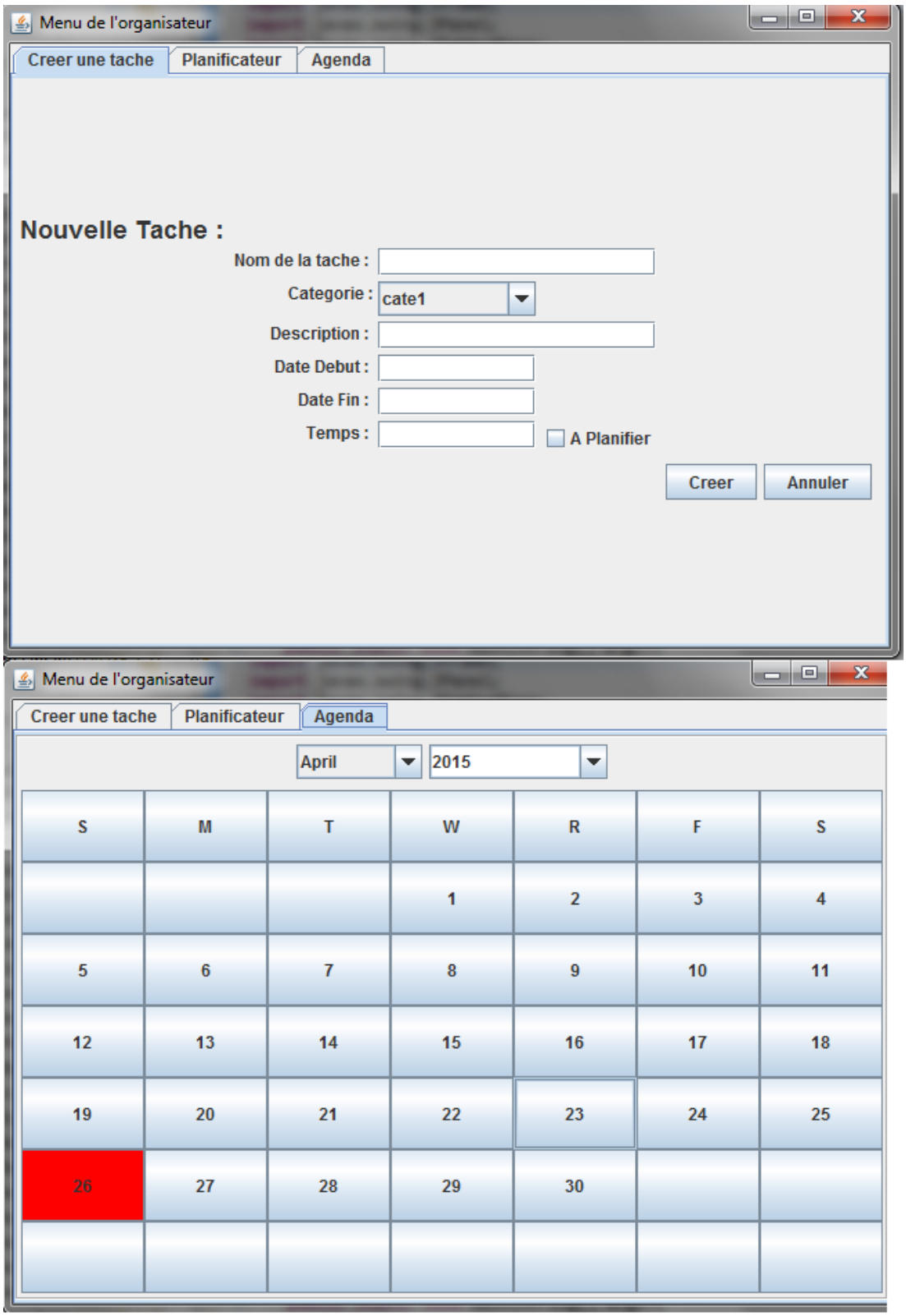


Figure 5.2.2.2 : Seconde version de l'interface graphique

Cette nouvelle version était donc plus simple d'utilisation, toute l'application se retrouve sur une seule fenêtre à onglets. Cependant, même si l'ergonomie de notre application avait était grandement amélioré, la partie agenda n'était toujours pas très amélioré et ouvrait toujours des nouvelles fenêtre superflues et la partie création de tâches manqué de détails et impliqué trop de traitements en arrière plan, nous avons donc procédé à une nouvelle amélioration.

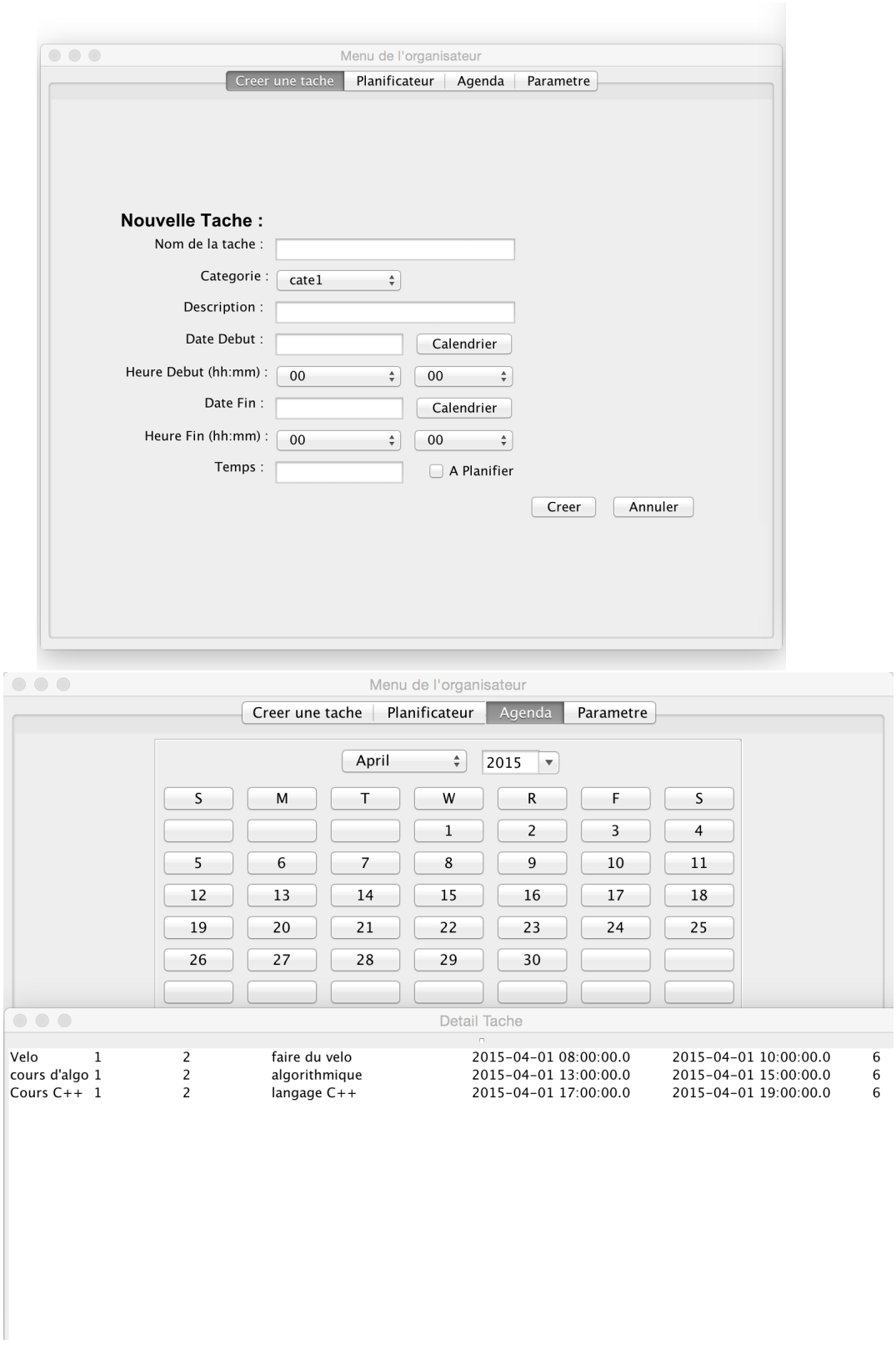


Figure 5.2.2.3 : Version finale de l'interface graphique

Cette version est donc notre version finale au moment ou nous écrivons ce rapport(et est donc susceptible d'être modifier légèrement d'ici la présentation) cette dernière version nous apporte donc l'ergonomie que nous cherchions tout en limitant les traitements et tests effectué en arrière plan. La partie agenda une nouvelle fois amélioré nous permet donc afficher les tâches prévus à une certaine date directement sous le calendrier.

5.2.3 Contrôleur

Le contrôleur a pour but de contrôler les actions effectué par l'utilisateur via la vue en appelant la méthode du modèle correspondantes et en notifiant la vue en retour pour qu'elle se mette à jour si nécessaire.

Notre contrôleur est donc composé des listeners qui permettent donc à chaque action effectué sur l'interface graphique d'appeler la méthode du modèle approprié

```
if(str.length()==0){
    JOptionPane.showMessageDialog(null, "Nom de tache obligatoire");
}else{
    if(Date_Deb==" 00:00"){
        if(!dd.before(df)){
            JOptionPane.showMessageDialog(null, "Date Debut posterieure a la date de fin");
        }
    }else{
        if(Date_Fin==" 00:00"){
                if(dd.equals(df)){
            //if()
                }
        }else{
            TacheJDBC ajout= new TacheJDBC(NameField.getText(),1,j,DescriptionField.getText(),Date_Deb,Date_Fin,i);
            ajout.AjoutTacheBDD();
        }
    }
}
```

Figure 5.2.3.1 : Extrait du code d'un listener

Voici ci-dessus une partie du code du Listener associé à la création d'une tâche, utilisé lorsque l'utilisateur appuis sur le bouton 'créer' de la fenêtre de création de tâche.

6 Conclusion, Perspectives et remerciement

Durant ce projet nous avons appris de une nouvelle technologie de liaison de donnée(JDBC). Nous avons également dû améliorer nos compétences en base de données, en LaTex et en JAVA. Nous avons pu expérimenter d'avantage ce qu'est la gestion d'un projet et le développement d'un logiciel en équipe, en effet gérer notre temps et communiquer au fur et à mesure de l'avancement c'est révélé plus que nécessaire de même que l'auto-documentation. Ce projet nous aura donc permit de devenir de meilleurs développeur avec une meilleures appréhension du développement d'application.

6.1 Avenir de l'application :

Rendre l'applications plus fonctionnelles et complètes possible est l'objectif principal de tout développeur, c'est dans cet optique qu'une fois le planificateur intégrés à l'agenda, la prochaine étape sera de pouvoir aider l'utilisateur à prendre des décisions et mieux programmé ses tâches et cela en insérant un système basé sur des statistiques qui lui permettra de juger la fréquences de ses activité sur une périodes donnée.

6.2 Difficulté rencontré :

Au cours de notre projet, nous nous sommes heurté à divers difficultés, la première fut la liaison entre notre base de données et la partie modèle de l'application en effet l'api JDBC que nous avons utilisé nous était avant cela inconnue, s'ajoutant à ça le fait que les types peuvent être différents entre java et sql notamment le type date. La seconde principale difficulté et non la moindre fut d'intégrer le planificateur au reste de l'application.

6.3 Remerciement

Nous remercions notre encadrant M.Seriai pour l'aide et les éclaircissement qu'il a pu nous apporter tout au long de notre projet.

7 Annexes

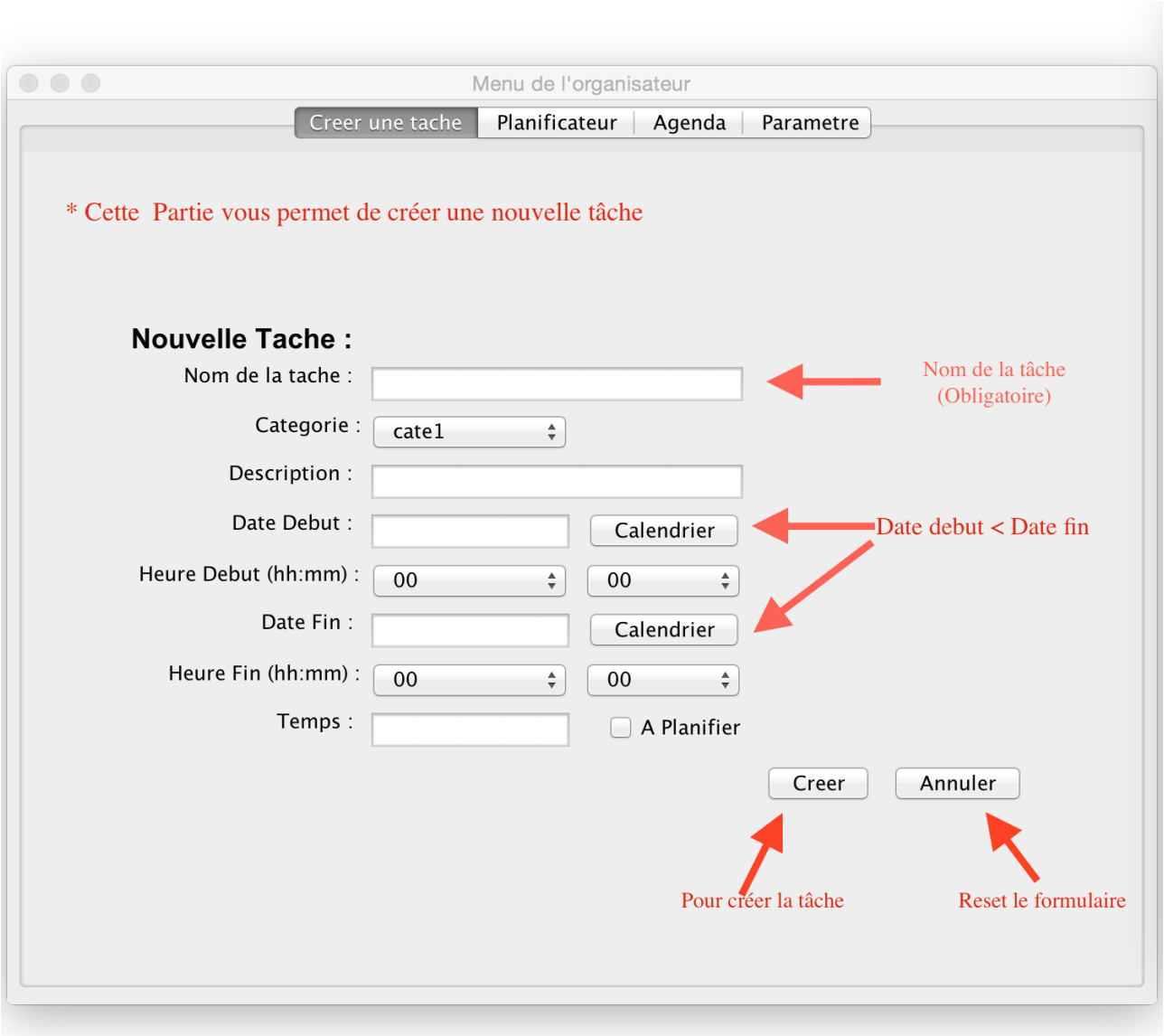
```
try {
    Class.forName("com.mysql.jdbc.Driver").newInstance();
} catch (Exception e) {}

try{
    Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/organisateur", "root","");
    java.sql.Statement pstm = con.createStatement();
    if(dateDeb==null||dateFin==null){
       if(dateDeb==null){
            Timestamp DF = new Timestamp(System.currentTimeMillis());
            String DFstr = dateFin+":00";

            try{
                DF = Timestamp.valueOf(DFstr);
            }catch (Exception e){
                e.printStackTrace();
            }
       }
       if(dateFin==null){
            Timestamp DD = new Timestamp(System.currentTimeMillis());

            String DDstr = dateDeb+":00";//String saisi par utilisateur

            try{
                DD = Timestamp.valueOf(DDstr);
            }catch (Exception e){
                e.printStackTrace();
            }
      }
    }

    pstm.execute ( "INSERT INTO taches(NOM,PRIO,CATEGORIE,DESCRIPTION,DATE_DEB,DATE_FIN,DUREE,Aplanifier,"
            "repetition) VALUES('"+nom+"','"+prio+"','"+categorie+"','"+description+"','"+dateDeb+"','"+
            dateFin+"','"+duree+","+1+","+0+")");
```

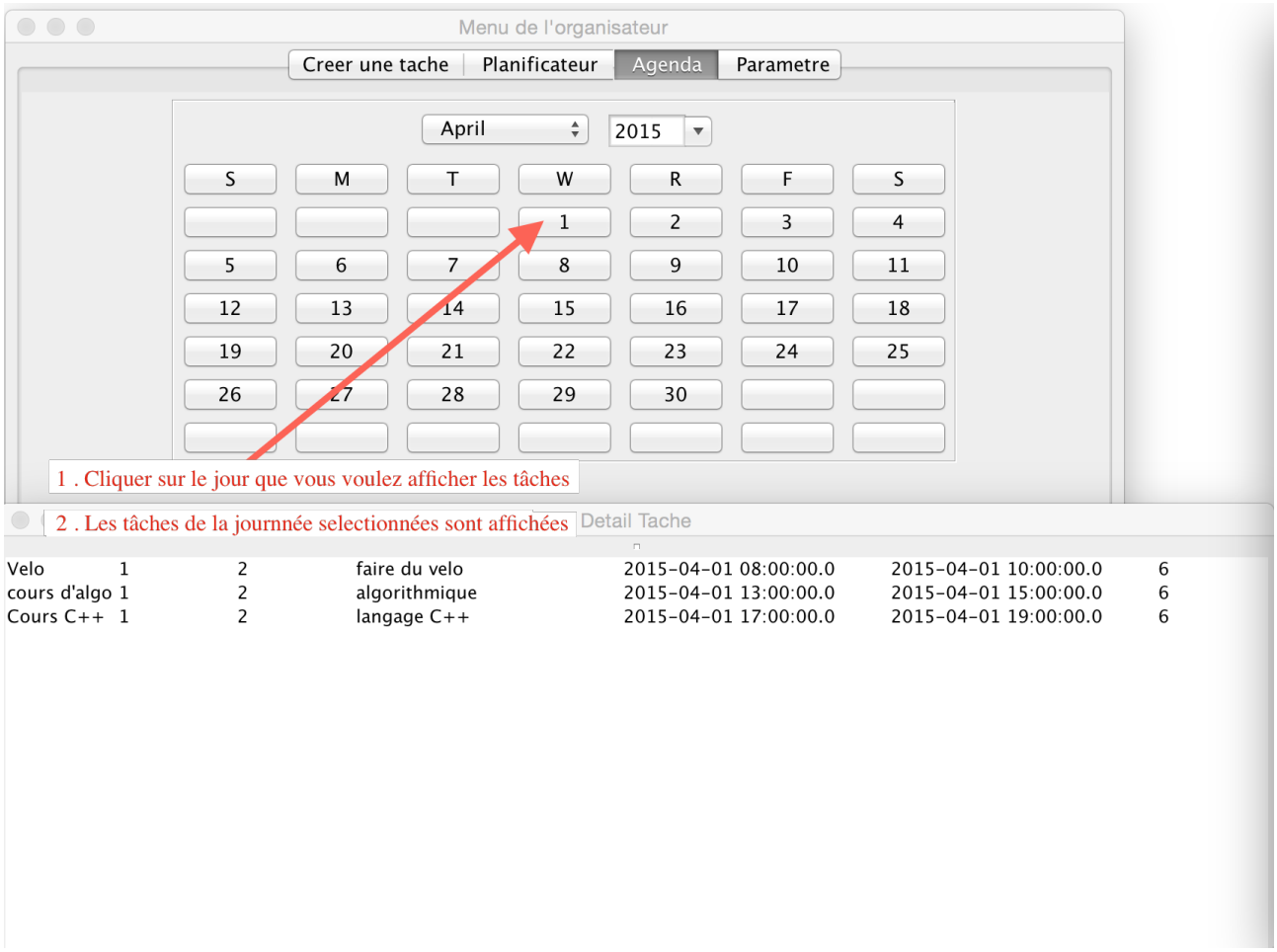
Annexe 1 : Exemple de code utilisant JDBC

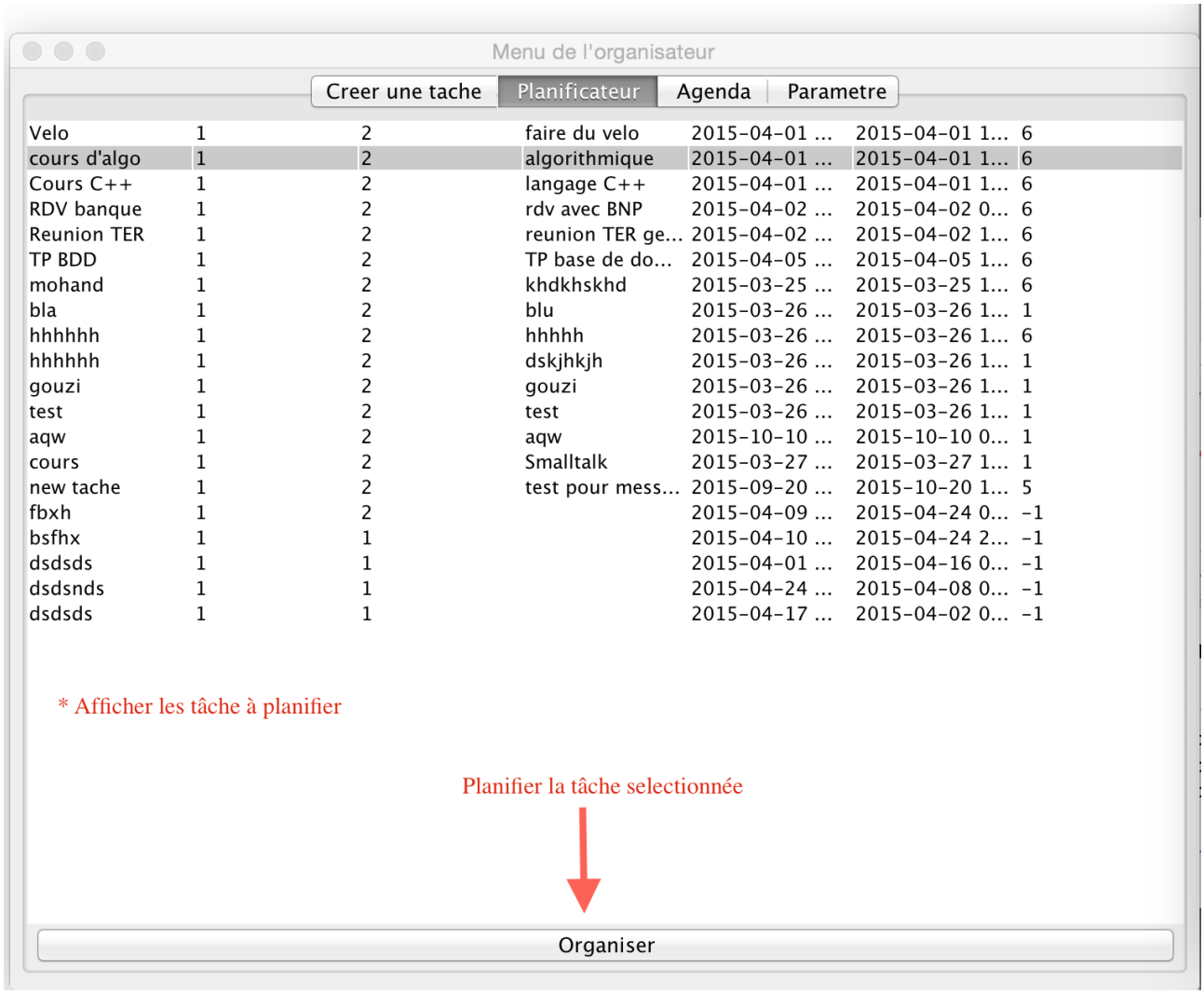
7.1 Manuel d'utilisation

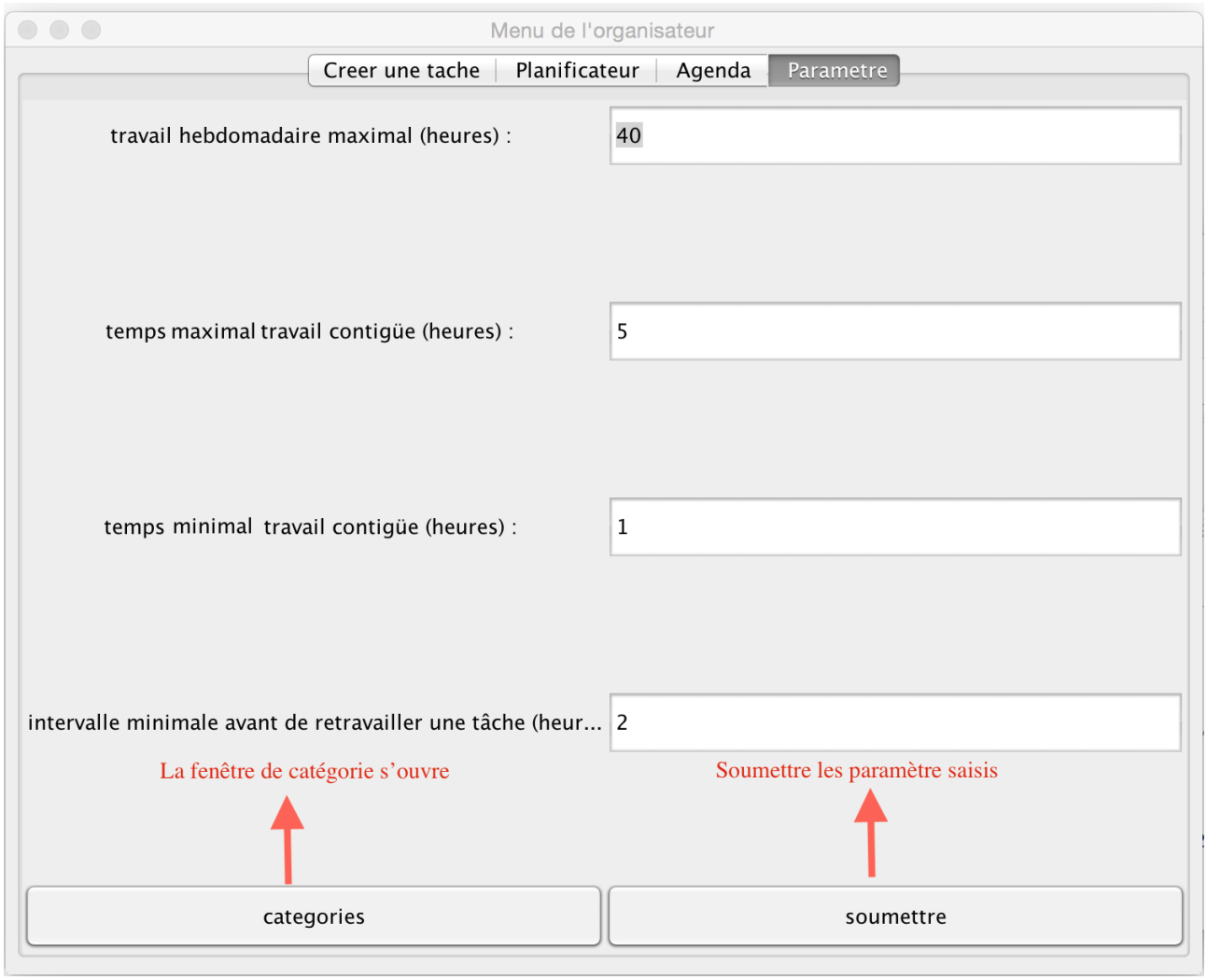
Quelques règles de gestion :

Notre application est mono-utilisateur, chaque tâche doit avoir au moins un nom, une catégorie et chaque tâche doit avoir un nom différent. Le nombre de catégorie est limité à 8.


Menu de l'organisateur
Creer une tache
Planificateur
Agenda
Parametre
* Cette Partie vous permet de créer une nouvelle tâche
Nouvelle Tache :
Nom de la tache :
Nom de la tâche (Obligatoire)
Categorie :
cate1
Description :
Date Debut :
Calendrier
Date debut < Date fin
Heure Debut (hh:mm) :
00
00
Date Fin :
Calendrier
Heure Fin (hh:mm) :
00
00
Temps :
A Planifier
Creer
Annuler
Pour créer la tâche
Reset le formulaire


Menu de l'organisateur
Creer une tache
Planificateur
Agenda
Parametre
April
2015
S M T W R F S
1 2 3 4
5 6 7 8 9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30
1 . Cliquer sur le jour que vous voulez afficher les tâches
2 . Les tâches de la journnée selectionnées sont affichées
Detail Tache
Velo 1 2 faire du velo 2015-04-01 08:00:00.0 2015-04-01 10:00:00.0 6
cours d'algo 1 2 algorithmique 2015-04-01 13:00:00.0 2015-04-01 15:00:00.0 6
Cours C++ 1 2 langage C++ 2015-04-01 17:00:00.0 2015-04-01 19:00:00.0 6


Menu de l'organisateur
Creer une tache
Planificateur
Agenda
Parametre
Velo 1 2 faire du velo 2015-04-01 ... 2015-04-01 1... 6
cours d'algo 1 2 algorithmique 2015-04-01 ... 2015-04-01 1... 6
Cours C++ 1 2 langage C++ 2015-04-01 ... 2015-04-01 1... 6
RDV banque 1 2 rdv avec BNP 2015-04-02 ... 2015-04-02 0... 6
Reunion TER 1 2 reunion TER ge... 2015-04-02 ... 2015-04-02 1... 6
TP BDD 1 2 TP base de do... 2015-04-05 ... 2015-04-05 1... 6
mohand 1 2 khdkhskhd 2015-03-25 ... 2015-03-25 1... 6
bla 1 2 blu 2015-03-26 ... 2015-03-26 1... 1
hhhhhh 1 2 hhhhh 2015-03-26 ... 2015-03-26 1... 6
hhhhhh 1 2 dskjhkjh 2015-03-26 ... 2015-03-26 1... 1
gouzi 1 2 gouzi 2015-03-26 ... 2015-03-26 1... 1
test 1 2 test 2015-03-26 ... 2015-03-26 1... 1
aqw 1 2 aqw 2015-10-10 ... 2015-10-10 0... 1
cours 1 2 Smalltalk 2015-03-27 ... 2015-03-27 1... 1
new tache 1 2 test pour mess... 2015-09-20 ... 2015-10-20 1... 5
fbxh 1 2 2015-04-09 ... 2015-04-24 0... -1
bsfhx 1 1 2015-04-10 ... 2015-04-24 2... -1
dsdsds 1 1 2015-04-01 ... 2015-04-16 0... -1
dsdsnds 1 1 2015-04-24 ... 2015-04-08 0... -1
dsdsds 1 1 2015-04-17 ... 2015-04-02 0... -1
* Afficher les tâche à planifier
Planifier la tâche selectionnée
Organiser


Menu de l'organisateur
Creer une tache
Planificateur
Agenda
Parametre
travail hebdomadaire maximal (heures) :
40
temps maximal travail contigüe (heures) :
5
temps minimal travail contigüe (heures) :
1
intervalle minimale avant de retravailler une tâche (heur...
2
La fenêtre de catégorie s'ouvre
Soumettre les paramètre saisis
categories
soumettre

* Cette partie vous permet de créer et modifier la catégorie

* ! Le nombre de catégorie est fixé à 8