Documentation technique

Base de données :

C'est le SGBDR MySQL qui a été utilisé pour l'architecture de données de cette application. Grâce au Framework JBoss-Hibernate, qui gère la persistance des objets en base de données relationnelle, il est possible de générer les classes métiers de l'application directement à partir de l'utilitaire offert par JBoss.

Framework et autres outils :

- On a utilisé le langage Java pour développer le projet GSB-Admin. (http://www.java.com/fr/)
- Comme expliquer précédemment, c'est le Framework Hibernate qui a été utilisé afin de développer de façon plus cohérente. (Les modèles d'un côté, les controllers et enfin les vues créées avec JFace). (http://www.hibernate.org/)
- JFace est une bibliothèque graphique libre Java pour le projet Eclipse qui s'appuie sur la bibliothèque SWT pour fournir des interfaces utilitaires plus structurées. Combiné à Hibernate, il est possible de relier directement les objets à des éléments dans les vues (listes déroulantes, tableaux de données …Etc.), c'est le data-binding. (http://wiki.eclipse.org/index.php/JFace)
- L'environnement de développement est ECLIPSE Juno. (http://www.eclipse.org/)
- Le serveur de développement est Mamp/Wamp, en fonction de l'OS utilisé. (http://www.mamp.info/en/index.html) (http://www.wampserver.com/)

Règles et contraintes :

Afin de gérer le Data-binding avec JFace et Hibernate, il est nécessaire de faire hériter chacune des classes métiers du projet de la classe AbstractModel, qui permet de relier des « écouteurs » aux objets. C'est à dire que l'on pourra « écouter » et voir si un objet a été ajouté, modifié ou supprimé, et ainsi mettre à jour directement l'objet dans le projet. Visuellement, par exemple, en effaçant un objet d'un tableau au moment de sa suppression de la base.

Il est aussi nécessaire de redéfinir les setteurs de chaque classe, pour repérer les modifications d'objets, de la façon suivante :

```
public void setId(String id) {
	String oldValue = this.id;
	this.id = id;
	firePropertyChange("id", oldValue, id);
}
```

Par ailleurs, il faut créer des fichiers de configurations de Hibernate :
hibernate.reveng.xml

- EXTRAIT :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-reverse-engineering PUBLIC "-//Hibernate/Hibernate Reverse
Engineering DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-reverse-
engineering-3.0.dtd" >

<hibernate-reverse-engineering>
  <table-filter match-catalog="gsbAdmin" match-name="etat"/>
  <table-filter match-catalog="gsbAdmin" match-name="familles"/>
  <table-filter match-catalog="gsbAdmin" match-name="fichefrais"/>
  <table-filter match-catalog="gsbAdmin" match-name="fichefraisarchive"/>
```

Ce fichier permet la génération des modèles depuis la base de données, grâce à l'outil offert par hibernate.

hibernate.cfg.xml

- EXTRAIT :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
            "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
            "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property
name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="hibernate.connection.password"></property>
        <property
name="hibernate.connection.url">jdbc:mysql://127.0.0.1/gsbAdmin?zeroDateTimeBehavi
or=convertToNull</property>
        <property name="hibernate.connection.username">root</property>
        <property
name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

        <mapping class="net.models.Etat"/>
        <mapping class="net.models.Familles"/>
```

/……

Dans ce de dernier, on pointe le driver utilisé, la base de données en relation, le username et le password nécessaire pour la connexion, et enfin les modèles à mapper.

Lancement de l'application :

Au lancement de l'application, une fenêtre de connexion s'affiche, le responsable ou le délégué devra entrer son login et son mot de passe pour que l'application se lance. Dans le cas contraire, il en sera informé.

Structure de l'application :

- GSB-Admin
  - src ***(Contient les fichiers sources de l'application et leurs packages)***
    - hibernate
      - HibernateUtil.java ***(Classe d'hibernate qui permet notamment de gérer les sessions)***
    - img ***(Contient toutes les images de l'application)***
    - net.controller ***(Contient les controllers de l'application)***
      - AppController.java ***(Contient le salt de sécurité de l'application ainsi que les fonctions d'initialisation des listes d'objets)***
      - LoginController.java ***(Permet de vérifier comparer un mot de passe en clair avec le mot de passe hashé présent dans la base de données …)***
      - MySHA1.java ***(Permet certaines conversion de type SHA1 ou SHA1+Salt)***
    - net.gui ***(Ensemble des fenêtres de l'application)***
      - LoginWindow.java
      - MainWindow.java
      - Praticiens_edit.java
      - Medicaments_edit.java
      - …
    - net.models ***(Ensemble des classes métiers)***
      - AbstractModel.java
      - Etat.java
      - Familles.java
      - Users.java
      - Fichefrais.java
      - …
    - net.view ***(Ensemble des composites s'intégrants dans des vues)***
      - Accueil_index.java
      - FicheFrais_index.java
      - Medicaments_index.java
      - …
    - swt
      - SWTResourceManager.java ***(notamment utile pour la mise en cache des images de l'application)***
    - hibernate.cfg.xml ***(fichier de configuration de hibernate)***
    - hibernate.reveng.xml ***(fichier de génération de classes par hibernate)***
  - JRE System Libraries
  - Referenced Libraries
  - lib