Dossier

Technique

Tableau des listes des mises à jour

Date	Désignation de la modification	Responsable
29/04/18	Création des ID SERIAL pour chaque tables	MOE
03/05/18	Création d'une page tampon pour les retours aux menux	Développeur
23/05/18	Modification d'apparence pour les tableaux	Réunion Qualité
25/05/18	Rajout d'exception sur les choix « null »	Développeur
03/06/18	Mise en place d'une 3ème fonction PL/pgsql (traitement des dates)	MOE
07/06/18	Retrait des livres d'un détenteur lorsqu'il souhaite en demander un	MOA
10/06/18	Risque de prêt du même livre (1er cas test)	MOE
11/06/18	Risque que l'utilisateur détienne plus de 2 livres (2ème cas test)	MOE
12/06/18	Risque d'usurpation d'identité (3ème cas test)	Développeur

Table des matières

1 Configuration du poste informatique

1.1 Utilisation de PostGreSQL 9.4

1.1.1 Ajouter une base de donnée : Publio (avec un 'P' majuscule)

```
CREATE DATABASE "Publio"
WITH OWNER = postgres
ENCODING = 'UTF8'
TABLESPACE = pg_default
LC_COLLATE = 'French_France.1252'
LC_CTYPE = 'French_France.1252'
CONNECTION LIMIT = -1;
```

Création d'un rôle de connexion : child
Avec un mot de passe : Wxcvbn

Toutes les connexions à la base de donnée se font via le compte child avec comme mot de passe 'Wxcvbn'. C'est incontournable, pour la simple et bonne raison que la plus part des connexion se font dans les fonctions de l'application. C'est pas super bien optimisé ; car le jour où je change de serveur, ça va en faire de la correction !

1.1.2 Utilisation du fichier : table.sql

On retrouve dans ce fichier :

- La création des tables qui sont nécessaires au bon fonctionnement du programme.
- La répartition des droits nécessaires
- La mise en place de droit particulier dans le cadre de l'utilisation de 'SERIAL'
- La création des fonctions (PL)

1.1.3 Utilisation du fichier : RAZ_SERIAL

Ce sont des requêtes SQL, permettant la mise à zéro des SERIAL de chaque tables. Attention, on ne peut pas lancer toutes les requêtes en même temps dans Query de PostgreSQL, il faut les exécuter les unes à la suite des autres. Il a été souvent utilisé lors de la phase de test.

1.1.4 Utilisation du fichier : incrémentation_Publio

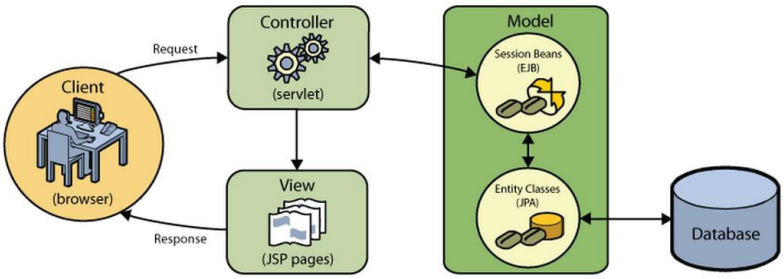
Permet l'ajout, dans les tables crées, des informations permettant l'exploitation du programme et surtout sert de support durant la phase des tests.

1.2 Utilisation d'Eclipse

JRE System library : jre1.8.0.161
GlassFish System Libraries
Referenced libraries : postgresql-42.2.0.jar

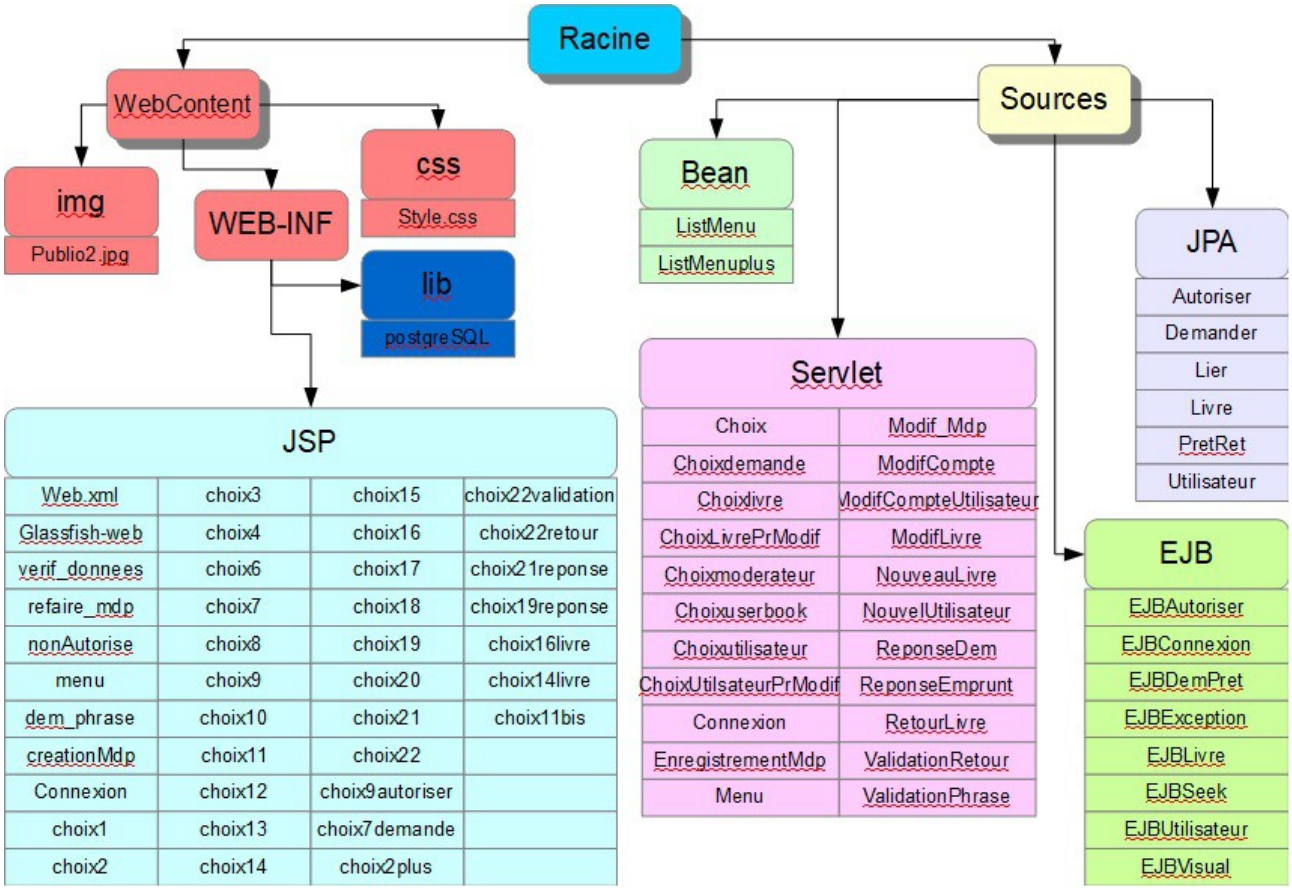
2 Agencement du programme en J2E

On est dans un schéma MVC (modèle, vue, contrôle), les pages JSP sont en relation avec les Servlets, qui déclenchent des fonctions qui sont dans les fichiers EJB. On utilise le langage JPQL (Java Persistence Query Language) pour communiquer avec la base de données via les fichiers JPA.



2.1 Représentation graphique du programme

Au cœur d'Eclipse



2.2 Composition et fonctionnement des fichiers du programme

2.2.1 Section Connexion

Composition :

- JSP : connexion, creationMdpPhrase, dem_phrase, refaire_mdp
- Servlet : Connexion, EnregistrementMdpPhrase, ValiderPhrase, Modif_Mdp,
- EJB : EJBConnexion, EJBUtilisateur,
- JPA : Utilisateur,

Fonctionnement :

Elle permet,comme son nom l'indique, la connexion de l'utilisateur. Celui-ci à besoin d'un Nom et d'un Prénom, lors d'une première connexion, l'utilisateur sera dirigé vers un module de création mot de passe et de phrase (répondre à la question du lieu de naissance). Pour l'utilisateur ayant fait une erreur de frappe lors de la saisie de son mot de passe, il aura droit à un deuxième essaie. En cas de deuxième échec, on demandera à l'utilisateur sa ville de naissance et si il échoue encore, il lui sera demandé de contacter son modérateur (un détenteur de bibliothèque). Si le nom et le prénom ne se trouve pas dans la base de données (BDD), une information lui sera délivrée. Pour l'utilisateur réussissant à se connecter, il arrivera sur la page JSP servant de tampon: 'verif_donnees.jsp'.

2.2.2 Section Menu

Composition :

- JSP : verif_donnees, menu,
- Servlet : toutes sauf connexion,
- EJB : EJBUtilisateur,
- JPA : Utilisateur,
- BEAN : ListMenu, ListMenuplus

Fonctionnement :

On part de la carte d'identité de l'utilisateur, on retrouve sur cette page un élément essentiel, au bon déroulement de l'affichage du menu. Car si l'utilisateur est détenteur d'une bibliothèque, il aura le menu modérateur et pour un utilisateur sans bibliothèque, il aura un menu standard.A l'issus de la consultation de sa carte identité, l'utilisateur est convié à cliquer sur l'accès menu, qui le redirigera sur une liste de choix avec des boutons de type « radio ».

2.2.3 Section : MenuPlus

Composition :

- Extention du fichier menu
- 1 constructeur avec la liste des action pour le modérateur
- 1 méthode : affichage du

Fonctionnement :

Le constructeur continue de chargé dans ArrayList toutes les actions que peut effectuer un modérateur
la méthode affiche = pour faire apparaître module utilisateur + module modérateur

2.2.4 Section Choix

Composition :

- JSP : choix de 1 à 22
- Servlet : Menu,
- EJB : toutes sauf EJBConnexion,
- JPA : toutes,

Fonctionnement :

Le constructeur charge dans une ArrayList toutes les actions que peut effectuer l'utilisateur
la méthode affiche = pour faire apparaître sur l'écran le menu.

2.2.5 EJB : Visual

Composition : 10 fonctions

Fonctions	*Déclenchée par*	*Fichier racine du déclencheur*
TousLeslivres	Menu : Choix 3 + 6	Choix
pret_encours	Menu : Choix 4	Choix
dem_pret	Menu : Choix 8	Choix
dem_autorisation	Menu : Choix 10	Choix
ChoixUtilisateur	Menu : Choix 13 + 16 + 17	Choix
Seslivres	Menu : Choix 14 + 15	Choix
dem_pret_A_Traiter	Choix 18 +19	Choix
dem_autorisationA_Ttraiter	Choix 20 + 21	Choix
dejalu2	Choix16 > choix16livre	Choixuserbook
ChoixDuLivre	Choix6 > choix6livre Choix14 > choix14 livre	Choixlivre ChoixlivrePrModif

Fonctionnement :

Regroupe toutes les fonctions de visualisation correspondant à toutes les sorties du cahier des charges, lors de l'évaluation de complexité de chaque composant.

2.2.6 EJB : Seek

Composition :

2fonctions : DEMautorisation, choix 9, pour tous les utilisateurs
Dejalus, choix 5, pour tous les utilisateurs.

Fonctionnement :

Visualiser les livres empruntés, même si le menu parle de visualisation, ça reste tout de même une rechercher des livres déjà lus et donc rendus.
Demander l'autorisation, dans le choix 9, consiste à rechercher dans la BDD, tous les détenteur de bibliothèque, afin d'en sélectionner un.

2.2.7 EJB : Utilisateur

Composition et fonctionnement :

- 1 fonction PL/pgsql : Averto--> avertissement état du livre
- load (id user) = super pratique pour récupérer de la BDD la fiche complète d'un utilisateur ou modérateur.
- Modif tel, modif adresse sert à la modification de l'adresse et du téléphone de l'utilisateur et de ses abonnés.
- créer biblio,
- vérif NomPrenom lors de la création d'un utilisateur, on ne veut pas de doublon dans la BDD
- VerifLien (table Lier) sert dans 2 servlets :
  - Choixdemande : car on vérifie avant une demande de livre que l'utilisateur et bien lié avec le détenteur du livre.
  - Choixmoderateur : lorsqu'un utilisateur demande à être lié à une bibliothèque d'un détenteur, on vérifie que l'utilisateur n'est pas déjà lié, pour éviter les doublons.
- EnregistrementUser : pour la création d'un nouvel utilisateur.
- Update Nb livre = mise à jour du nombre de livre lors d'une validation d'une demande de livre et lors d'un retour de livre.
- Update Mdp = mise à jour du mot de passe lors de la phase de connexion.

2.2.8 EJB : Livre

Composition :

- ✔ 1 fonction PL/pgsql : Calculretourlivre (retourdate)
- ✔ 12 fonctions : SansSeslivres, load, verifLien, enregistrementDEM, verifTitreAuteur, enregistrementBook, modiftitre, modifsoustitre, modiftome, modifauteur, modifdate, modifetat.

Fonctionnement :

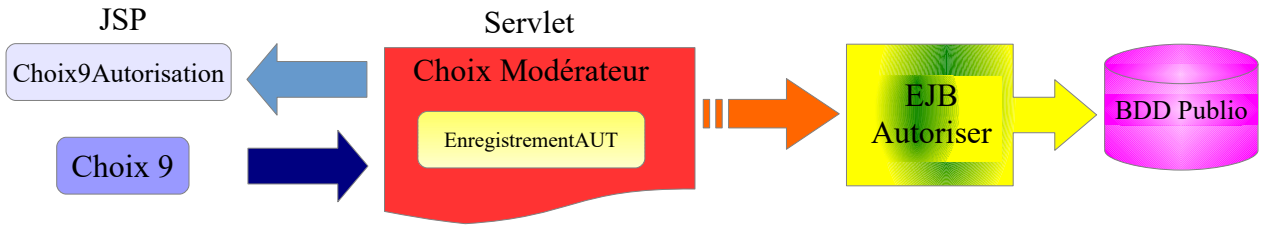
- On utilise la fonction PL/pgsql , pour calculer la date prévue du retour du livre, lors de l'acceptation de la demande de prêt du livre pour un des utilisateurs du détenteur.
- SansSesLivres : permet la visualisation des livres disponible pour un prêt, mais sachant qu'un détenteur peut être aussi un utilisateur de bibliothèque. On ne veut pas qu'un utilisateur emprunte un de ses propre livre.
- Load : permet le chargement de tous les paramêtres d'un livre par son numéro d'identification.
- VerifLien : lors d'un choix de livre à emprunter, on vérifie préalablement, que le demandeur et le détenteur sont liés. (Choixdemande)
- EnregistrementDEM fonction non utilisée, car elle a été remplacée par la même fonction mais dans EJB DemPret (Choixdemande)
- VerifTitreAuteur : évite les doublons d'un livre. On ne s'arrête pas qu'au titre et à l'auteur cette fonction prend en compte aussi le soustitre et le tome.
- EnregistrementBook, finalise la création d'un nouveau livre
- modif titre, soustitre, tome, auteur c'est 4 fonctions permettent de corriger des erreurs de saisie.
- Modif date retour, on ne sait jamais un utilisateur peut avoir un empêchement pour rendre un livre, ou le modérateur autorise que l'utilisateur garde le livre plus longtemps, pour cela le modérateur peut modifier la date de retour manuellement.
- Modif état ce n'est pas généralement pas une bonne nouvelle, lors d'un retour de livre le modérateur saisie un retour en mauvais état du livre.

2.2.9 EJB : autoriser

Composition :

✔ 5 fonctions : enregistrementAUT, load, reponseacces, EnregistrementLier, enregistrementAUTO

Représentation : pour la fonction EnregistrementAUT



Fonctionnement :

- Choix 9 Demande d'autorisation d'accès à une bibliothèque, utilisateur choisit parmi la liste des modérateurs, vérification que le lien n'existe pas déjà avec fonction vérifLien. Si ce n'est pas le cas, création de la demande d'autorisation, par la méthode : EnregistrementAUT

- Load, permet le chargement d'une ligne de la BDD en fonction du numéro d'ID

- reponseacces : le modérateur choisit le choix 21 du menu, première étape choisir la demande d'autorisation à traiter par Visual.dem_acces_atraiter. On contrôle que le choix, qui fait correspond à ses demandes d'autorisation, puis on utilise la fonction Load pour les autorisations, ainsi on récupère les informations de la BD, le modérateur accepte ou refuse.
- la création du lien entre l'utilisateur et lui. Utilisation de la méthode Enregistrement Lier,

2.2.10 EJB : DemPret

Composition :

- ✔ 1 fonction PL/pgsql : difdatepravertissement, calcule de différence de date entre le retour prévu et la saisie du retour du détenteur (compare_date)
- ✔ 2 méthodes : enregistrementDEM, enregistrementPret
- ✔ 7 fonctions :load, , reponsedem, retour_pretA_traiter, (PretRet) charger, MajDateRetour, SaisiEtatlivre

Fonctionnement :

- Enregistrement DEM : vient de l'utilisateur qui demande le prêt d'un livre
- La reponsedem : c'est le détenteur qui l'active, si réponse positive déclenchement de EnregistrementPret
- load : sert dans la BDD pour la table Demander
- charger sert dans la BDD pour la table PretRet
- retour_pretA_traiter : se déclenche lorsqu 'un détenteur veut réintégréer un livre dans sa bibliothèque, il doit choisir dans la liste
- MajDateRetour : Update date retour par la date du jour
- SaisiEtatlivre : Update de l'attribut Livre, booléen etat livre.

2.2.11 Table : Lier

Composition et fonctionnement :

- Cette table permet de recenser les personnes qui sont liés.
- On a un modérateur et un utilisateur.
- Particularité cette table n'a pas eu besoin EJB ; grâce aux clé secondaires, on peut facilement trouver sont chemin à travers la BDD.

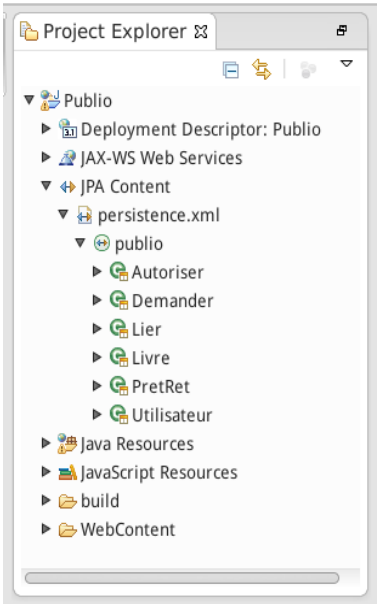
3 Conclusion

3.1 Retex (RETour _ d' EXercice)

3.1.1 Contexte

Dans un premier temps, la principale difficulté du projet résidait, dans le fait que, je n'ai jamais programmé en J2E. Il a fallu estimer le mieux possible le temps nécessaire à la formation et à la réalisation. J'ai passé beaucoup de temps au début, à « choisir » quoique je préfère le terme appréhender, la persistance. J'ai commencé des cours, en autodidacte, sur « Hibernate », mais je n'ai pas accrocher je ne trouvais pas cela intuitif. Donc je suis revenu sur un modèle DAO et puis je me suis rapproché petit à petit au modèle Object-Relationnal-Mapping (ORM). Suite à de nombreuses lectures sur internet, je savais que TopLink était fourni dans Eclipse, que Glassfish était un bon serveur d'application, donc je me suis lancé et j'ai importé ma base de données existante, dans Eclipse via la création d'une JPA, et l'aventure a commencé...

3.1.2 Situation



Dans Eclipse, avec l'importation de la Base de donnée, on visualise le JPA Content, à l'intérieur de celui-ci on retrouve le fichier de configuration persistence.xml, qui gère l'accès à la Base de données par le serveur d'application GlassFish. Lorsque l'on consulte l'arborescence de persistence.xml, il apparaît les nom de la BDD ainsi que toutes ses tables. Par les annotations des JPA, on a accès à toutes les ramifications grâce aux clés étrangères, je me suis parfois amuser à passer par des chemins détournés pour arriver à obtenir l'information nécessaire.

3.1.3 Méthodologie

Malheureusement avec de nombreux jours de retard … Il a fallu changé de méthode de travail sur le projet, pour arriver à rattraper le retard, tout en continuant d'apprendre, par exemple la gestion des fonctions PL en SQL ne pose pas de problème mais en JPQL, il faut connaître ! On peut dire que je suivais une méthode Merise linéaire, classique et donc là j'ai pris la décision de suivre une méthode agile en Extreme Programming.

Si on suit la définition de Wikipedia : « .. est une méthode agile plus particulièrement orientée sur l'aspect réalisation d'une application, sans pour autant négliger l'aspect gestion de projet. XP est adapté aux équipes réduites avec des besoins changeants. XP pousse à l'extrême des principes simples. »

On retrouve une programmation simple, c'est pour cela qu'il y a autant de Servlet, les pages JSP n'ont pas de Head. Tout est programmé de manière linéaire, à la main avec l'aide du copier/coller.

Le programme tourne, pas de complexité dans la programmation, les délais sont respecter. L'avantage c'est de pouvoir retoucher l'application pour la rendre plus fonctionnelle.

3.2 Résultat et Perspectives

3.2.1 Analyse

La base du SI Publio n'est pas optimisé. Maintenant, il y a un chemin à définir pour le faire évoluer vers les standard du J2E. Cela fera l'objet du prochain project sous forme Merise, prendre le temps à la rédaction de mon cahier des charges et surtout bien effectuer mon dossier d'analyse. Je me dis à l'heure d'aujourd'hui mon logiciel est petit et malléable, autant le formater le mieux possible pour qu'il persiste dans le temps.

3.2.2 Perspectives

Pour l'évolution de l'application, je compte intégrer des fichiers .do (je n'ai pas encore abordé le sujet, mais j'en ai vu), améliorer l'architecture des pages JSP en collant plus au format HTML5, avec des div container, header, footer, etc … et sans oublier de créer une version mobile.

Pour l'évolution du SI Publio, tout en gardant le serveur d'application GlassFish, on peut imaginer un système de messagerie avec création et utilisation de « template ». Après avoir optimiser l'application, on pourra envisagé un notre mode d'hébergement, tel que le « cloud » et pourquoi un concept de « Make @Home » avec un serveur hébergé à la maison. Le contenu du SI Publio a le temps d'évoluer, aujourd'hui les livres et demain ...