**Classes Préparatoires Intégrées (CPI)**

**2ièmeannée**

**PROJET**

SUIVI DES MARCHES ESI

**Sujet N° : 06**

**Equipe N° 32 :**

1. HADJER NARJAS (CE)
2. SALHI YOUNES
3. AIBOUD KATIA
4. HADDADI MAZIGH
5. BRAHNA LYNDA

**Encadrée par :**

* Mme SI TAYEB
* Mr KHELOUAT
* **Clients :**
* Le comptable

**ANNEE** : 2020 /2021

**Remerciements**

 La réalisation de ce projet n’aurait jamais été possible sans la contribution de maintes personnes à qui nous voudrions témoigner toutes nos reconnaissances.

Tout d’abord, nous tenons à remercier profondément nos deux encadreurs : Mme Si Tayeb et Mr Khelouat pour la qualité de leur encadrement, leur orientation, leurs conseils, ainsi que leur soutien consistent. Tout ceci nous a permis de mener ce projet à son terme.

Nous voudrions également remercier tous nos enseignants à l’ESI pour la qualité des enseignements durant les deux dernières années.

Enfin, nous tenons à témoigner toute notre gratitude envers nos familles et nos amis qui nous ont apporté un grand support moral tout au long de cette démarche.

**Sommaire**

[Introduction 7](#_Toc40541550)

[Chapitre 1 : Présentation du projet 8](#_Toc40541551)

[1. Résumé du projet : 8](#_Toc40541552)

[2. Problématique : 8](#_Toc40541553)

[3. Objectifs : 8](#_Toc40541554)

[4. Pourquoi Suivi Marché ESI: 9](#_Toc40541555)

[Chapitre 2 : Conception 10](#_Toc40541556)

[1. Introduction : 10](#_Toc40541557)

[2. Solution proposée : 10](#_Toc40541558)

[3. Diagramme des classes : 11](#_Toc40541559)

[4. Définition des classes : 11](#_Toc40541560)

5. Prototype de l'application …………………………………………………………………………………………………….14

[6. Conclusion : 26](#_Toc40541583)

[Chapitre 3 : Réalisation 27](#_Toc40541584)

[1. Introduction : 27](#_Toc40541585)

[2. Choix des outils : 27](#_Toc40541586)

[3. Méthode de travail : 28](#_Toc40541587)

[4. Implémentation des fenêtres : 29](#_Toc40541588)

[5. Conclusion : 45](#_Toc40541600)

[Chapitre 4: Gestion du projet 47](#_Toc40541605)

[1. Introduction : 47](#_Toc40541606)

[2. Organisation du travail : 47](#_Toc40541607)

[3. Planning de travail : 47](#_Toc40541608)

[a) Phase d’analyse du cahier des charges et de la documentation : 47](#_Toc40541609)

[b) Phase de la conception : 47](#_Toc40541610)

[c) Phase de la réalisation : 47](#_Toc40541611)

[d) Préparation des présentations : 48](#_Toc40541612)

[e) Rédaction du rapport et guide d’utilisation : 48](#_Toc40541613)

[f) Phase de finalisation : 48](#_Toc40541614)

[Conclusion 48](#_Toc40541615)

[***Références*** 49](#_Toc40541616)

1. Introduction

Nous sommes des étudiants en deuxième année classe préparatoire à l’Ecole Nationale Supérieure d’Informatique d’Alger. Passionnés du monde de l’informatique, nous essayons de faire en sorte que cette dernière, évoluant de jour en jour comme une science de l’utile et de l’agréable, réponde aux besoins quotidiens de l’Homme.

Pour gérer à bien une entreprise ou un établissement scolaire, il est primordial de bien gérer ses marchés et son commerce et de bien veiller sur l’offre et la demande des biens et des services.

  De ce fait et dans le cadre du projet pluridisciplinaire 2CP, il nous a été demandé de créer une application Web permettant le suivi des Marchés de l’ESI (Ecole Nationale Supérieure d'Informatique d’Alger).

* 1. Présentation du projet
     1. Résumé du projet :

Ce projet pluridisciplinaire représente une étape importante dans la formation d’un ingénieur en informatique au sein de l’école nationale supérieure d’informatique durant le cycle préparatoire. En plus du contenu technique, qui consiste à mettre en œuvre les compétences acquises durant les 2 années préparatoires dans le cadre du développement d’un petit logiciel, l’accent sera mis sur l’acquisition et l’application des aspects organisationnels et relationnels entre les membres de l’équipe, orientés par deux encadreurs pour ainsi satisfaire le client.

De divers points essentiels seront traités : analyse et découpage du travail, répartition des charges de travail entre les membres du groupe par le chef d’équipe, circulation de l’information entre les membres de l’équipe, mise en place d’un planning de travail, exposé hebdomadaire de l’avancement du projet.

Dans notre cas, le client est le fournisseur des biens demandés par l'établissement. Quand un manque est signalé, une demande sera lancée et plusieurs offres seront présentées à l’établissement. Une fois le choix s’est porté sur une demande (selon des critères établis au préalable), un nouveau dossier sera alors créé et suivi d’un service à un autre (service Marché, service Commandes, service Budget, service Comptabilité)

Enfin, pour mettre en œuvre tout cela, notre projet a pour but de concevoir et de mettre en place une application Web permettant le suivi des dossiers.

* + 1. Objectifs :

Après avoir bien lu et analysé le problème en question et le cahier des charges transmis par nos deux encadreurs, et après s'être mis d’accord sur plusieurs points jugés ambigus par les membres de l’équipe, nous sommes arrivés à regrouper les problèmes récurrents qui sont:

o   La difficulté de manipuler les fichiers excel volumineux ainsi que leur difficulté à les gérer et les risques liés à leur confidentialité, d’où la nécessité de créer une base de données.

o   L'incapacité de savoir le service ayant entraîné un retard.

o   La difficulté de l’organisation des dossiers et des différents services.

* Tous ces problèmes ont nécessité la création de l’application Web pour une meilleure gestion et une meilleure organisation.
  + 1. Objectifs :

Les objectifs visés peuvent être résumés comme suit :

* Créer une base de données contenant tous les types de marchés.
* Proposer une codification significative pour les différents marchés.
* Permettre au personnel des différents services qui gèrent les marchés de pouvoir suivre chaque dossier.
* Mettre en place des alertes pour informer les structures de l’évolution du traitement d’un dossier.
* Permettre la sauvegarde périodique (et la restauration) de la base de données.
* Afficher les statistiques selon différents critères.
  + 1. 4.Pourquoi Suivi Marché ESI :

Ainsi, nous voulons satisfaire les attentes des clients en développant une application Web répondant aux qualités suivantes:

**-Simplicité : SuiviMarchéEsi** est une application Web très simple à utiliser avec une interface intuitive complète permettant à l'utilisateur de suivre l’évolution d’un dossier par chaque structure en coordination avec les autres structures et en temps réel.

-**Puissance :** Avec **SuiviMarchéEsi**, On peut suivre et gérer plusieurs dossiers à la fois, lancer une nouvelle année commerciale et la clôturer et avoir plusieurs comptes dans différents services en même temps.

-**Gestion des comptes  :** L'utilisateur peut se connecter à son compte ou bien à un de ses comptes s'il en a plusieurs et effectuer toutes les opérations relatives à ce compte ( Administrateur, ordonnateur, consulteur, éditeur)

-**Disponibilité d’un guide en ligne :** Avec **SuiviMarchéEsi,** vous ne serez jamais en face d’un problème car vous trouverez sa solution dans le guide en ligne qui explique toute fonctionnalité et répond à tous vos besoins en détails. (voir bouton “Aide en ligne” de l'application.

Tous ces problèmes ont été gérés par l’équipe fondatrice de l’application **SuiviMarchéEsi**.Les points cités seront détaillés ainsi que d’autres options ajoutées dans la suite du rapport.

* 1. Conception
     1. Introduction :

L’étude conceptuelle d’un projet est une étape clé pour laquelle temps, effort et rigueur doivent être consacrés. La conception est sans contestation la partie la plus importante car c’est elle qui implique la réussite ou la non-réussite du projet. C’est pour cela que notre équipe a pris son temps pour bien analyser et comprendre le problème posé et les objectifs attendus afin de s’en sortir par une bonne conception générale et accordé par toute l’équipe .Dans ce chapitre nous allons exposer la solution générale proposée , les différents diagrammes bien détaillés, et le prototype initial des pages de notre application

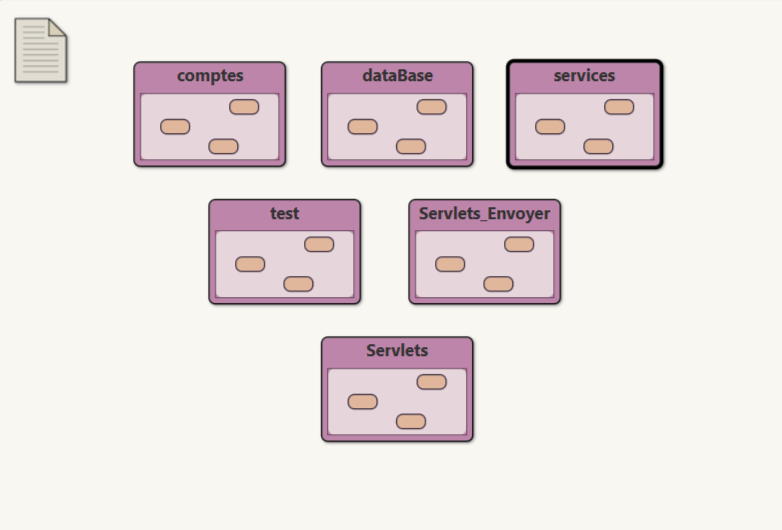
* + 1. Solution proposée :

Notre solution est basée sur l’approche orientée objet, donc, après l’analyse du problème donné, nous avons été d’accord sur le point que les classes principales seront ceux qui permettent la connexion à la base de données , et la manipulations des différentes tables de la base de données : tables des marchés et la table des comptes utilisateurs . Ensuite , pour chaque classe on a défini les attributs et les méthodes nécessaires qui forment les piliers de notre solution .

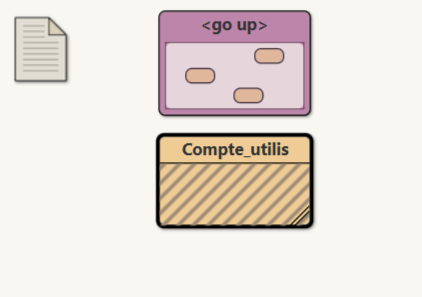
* + 1. Diagramme des classes :

Pour le diagramme des classes, nous l’avons réalisé par BluJ qui est un [environnement de développement intégré](https://fr.wikipedia.org/wiki/Environnement_de_d%C3%A9veloppement_int%C3%A9gr%C3%A9) créé principalement pour l'[enseignement](https://fr.wikipedia.org/wiki/Enseignement) de la [programmation orientée objet](https://fr.wikipedia.org/wiki/Programmation_orient%C3%A9e_objet). Il permet d'écrire des programmes [Java](https://fr.wikipedia.org/wiki/Java_(langage)) en utilisant le langage [Java](https://fr.wikipedia.org/wiki/Java_(langage_de_programmation)) ou (depuis la version 4) le langage Stride (langage utilisant un éditeur à base de cadres)

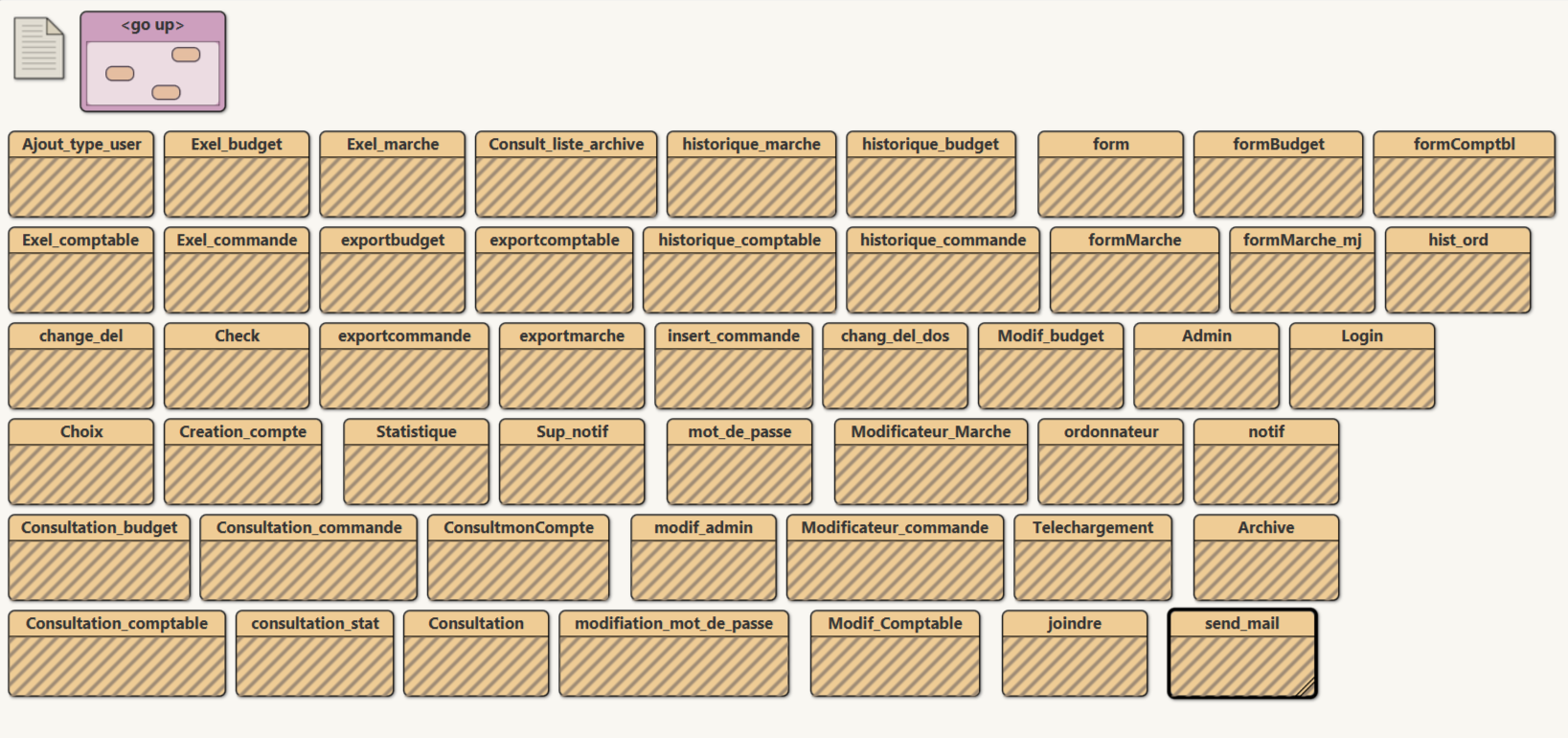
1) Diagramme des Packages :



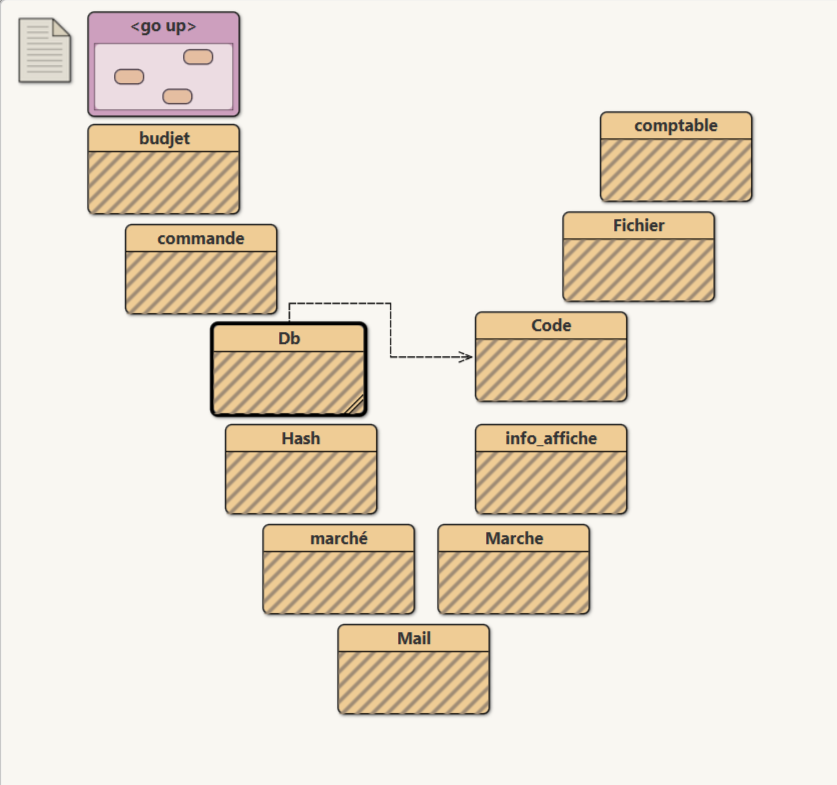
1) Diagramme Package : comptes :



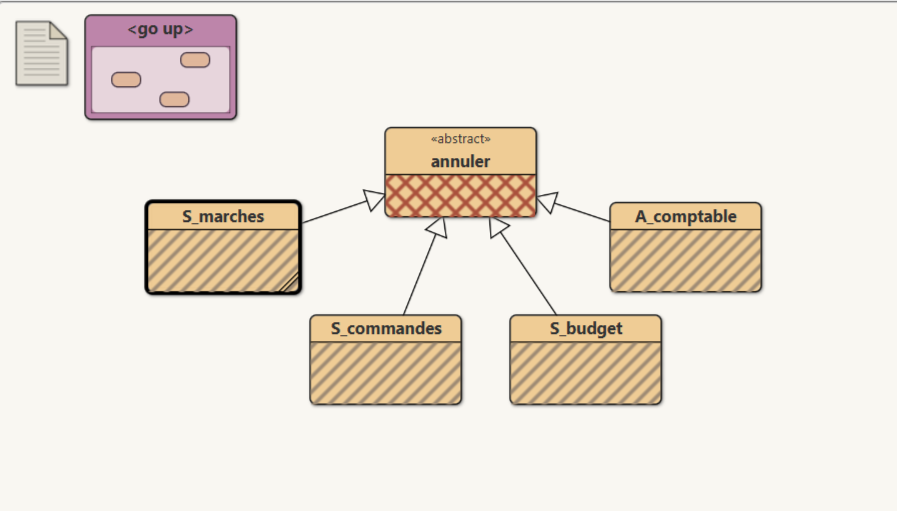
2) Diagramme Package : Servlets :



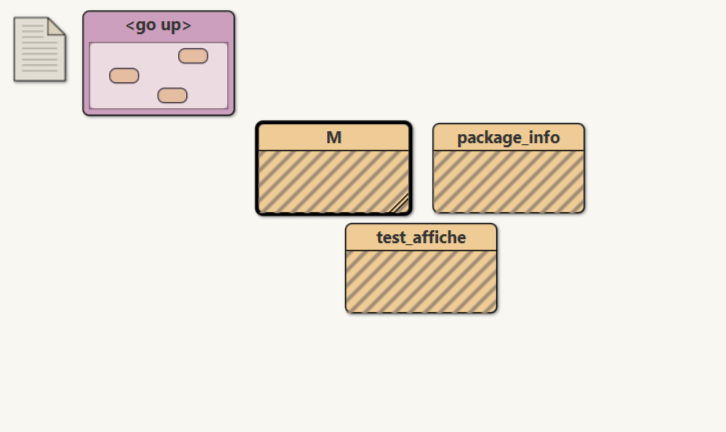
3) Diagramme Package : dataBase :



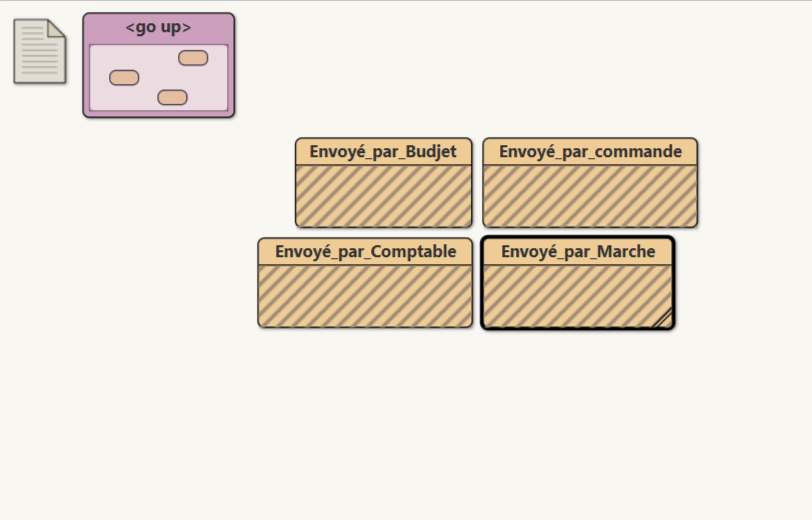
4) Diagramme Package : services :



5) Diagramme Package : test:



5) Diagramme Package :Servlets\_Envoyer:



* + 1. Définition des classes :
       1. Package : comptes :

1)Classe Compte\_utilis : Cette classe contient des attributs qui représentent les

Colonnes de la table des utilisateurs dans la base de données .

|  |  |
| --- | --- |
| **Class Compte\_utilis** | |
| **Liste des attributs** | |
| **Attribut** | **Signification** |
| private String Nom\_utilis | Le nom de l’utilisateur qu’il va insérer lors de la connexion |
| private String nom | Le nom d’utilisateur |
| private String prenom\_utilis | Le prénom de l’utilisateur |
| private String mot\_pass | Mot de passe de compte utilisateur |
| private String mail | L’adresse mail de l’utilisateur |
| private String type\_utilis | Le service de l’utilisateur |
| private String [] notif | Tableau de chaine de caractères qui va contenir les numéros de dossiers que l’utilisateur doit traiter |
| private String [] notifdate | Tableau de chaine de caractères qui va contenir les dates de réception de la notification que l’utilisateur doit traiter |
| private byte[] image | Garde en format byte le photo de profil utilisateur |
| Private String URL | Contient le chemin de répertoire ou l’utilisateur peut sauvegarder les fichier exportés ou importés |
| **Liste des méthodes** | |
| **Méthode** | **Rôle** |
| Constructeur, Getters et Setters | Récupérer et modifier les attributs |

* 1. Package : dataBases :
* 1)Classe Code : Cette classe contient les requêtes SQL permettant la connexion et la manipulation de la base de données , ainsi les opérations de bases pour la tables des marchés et des comptes : recherche , insertion , suppression et mise à jour sous forme de chaine de caractères qui vont être utilisées par la classe Db .

|  |  |
| --- | --- |
| **Class Code** | |
| **Liste des attributs** | |
| **Attribut** | **Signification** |
| private String up\_ntf | Met à jour les notifications |
| private String update\_del | L’instruction sql pour mettre à jour la colonne délai dans la table marche |
| private String creat\_db | L’instruction sql qui permet de créer la base de données |
| private String code\_recherche\_marche | L’instruction sql qui permet de rechercher et récupérer les champs de service marche de la table marché |
| private String code\_recherche\_commande | L’instruction sql qui permet de rechercher et récupérer les champs de service commande de la table marché |
| private String code\_recherche\_budget | L’instruction sql qui permet de rechercher et récupérer les champs de service budget de la table marche |
| private String code\_recherche\_comptable | L’instruction sql qui permet de rechercher et récupérer les champs de service comptable de la table marche |
| private String code\_Insert\_marche | L’instruction sql qui permet l’insertion d’une nouvelle ligne dans la table qui correspond à un nouveau dossier inséré dans la table |
| private String code\_mis\_jour\_commandes | L’instruction sql qui permet de mettre à jour une ligne dans la table marché (champs de service commande seulement) |
| private String code\_mis\_jour\_budget | L’instruction sql qui permet de mettre à jour une ligne dans la table marché ( champs service budget seulement ) |
| private String code\_mis\_jour\_comptable | L’instruction sql qui permet de rechercher et récupérer les champs de service comptable de la table marche |
| private String code\_recherche\_cpt | L’instruction sql qui permet de rechercher avec le nom d’utilisateur et récupérer les champs d’un compte utilisateur |
| private String code\_mis\_jour\_compte | L’instruction sql qui permet de mettre à jour une ligne dans la table compte correspondante à un compte utilisateur |
| private String code\_insertion\_compte | L’instruction sql qui permet l’insertion d’une nouvelle ligne dans la table comptes qui correspond à un nouvel utilisateur inséré |
| private String code\_insertion\_commande | L’instruction sql qui permet l’insertion d’une nouvelle ligne dans la table qui correspond à un nouveau dossier inséré dans la table (le dossier commence par le service commande) |
| private String code\_recherche\_liste\_commande | L’instruction sql qui permet de rechercher tous les dossiers dans le service commande répondant aux critères de la recherche |
| private String code\_recherche\_globale | L’instruction sql qui permet de rechercher tous les dossiers (renvoie les champs de tous les services) répondant aux critères de la recherche |
| private String code\_recherche\_marchea | L’instruction sql qui permet de rechercher tous les dossiers dans le service marche répondant aux critères de la recherche |
| private String code\_recherche\_budgeta | L’instruction sql qui permet de rechercher tous les dossiers dans le service budget répondant aux critères de la recherche |
| String code\_recherche\_comptablea | L’instruction sql qui permet de rechercher tous les dossiers dans le service comptable répondant aux critères de la recherche |
| String code\_recherche\_marche\_trt | L’instruction sql qui permet de rechercher tous les dossiers traités |
| String code\_annul | L’instruction sql qui permet de rechercher tous les dossiers annulés |
| **Liste des méthodes** | |
| **Méthode** | **Rôle** |
| Constructeur , Getters | // |

* 2)Classe Db: Cette classe contient principalement les variable et les méthodes permettant la connexion et la manipulation de la base de données , ainsi les opérations de bases pour la tables des marchés et des comptes : recherche , insertion , suppression et mise à jour .

|  |  |
| --- | --- |
| **Class Db** | |
| **Liste des attributs** | |
| **Attribut** | **Signification** |
| private static Connection cn | Une variable de type Connexion permet de sauvegarder la connexion à la base de données |
| private String name\_tb | Le nom de la table marché dans la base donnée |
| private String name\_tb\_cpt | Le nom de la table compte dans la base données |
| private Code cd | Contient un objet de type Code qui va contenir toute les instructions sql à utiliser |
| **Liste des méthodes** | |
| **Méthode** | **Rôle** |
| Constructeur , Getters et Setters | // |
| public Db(Connection cn ) | Crée une instance Code à travers la connexion cn |
| public Db( ) | // constructeur  Etablie la connexion à la base de données et la met dans |
| public String Duree(Timestamp t1 , Timestamp t2 ) | Calcule le différence de temps entre deux dates et renvoie une chaine de caractères qui contient le nombre de jours , heures et secondes |
| public S\_marches recherche\_marche( String num\_dossier) | Retourne un objet de type S\_marche qui contient des champs dans le service marche de dossier numéro num\_dossier |
| public boolean rech(String num\_dos) | Retourne vrai si le dossier numéro num\_dos existe dans la bdd |
| public String pourcentage\_Commande(S\_commande sc ) | Retourne le taux de remplissage de dossier dans le service commande |
| public S\_commandes recherche\_commande ( String num\_dossier) | Renvoie un objet de type S\_commandes qui contient des champs dans le service commande de dossier numéro num\_dossier |
| public String pourcentage\_budget (S\_budget sb ) | . Retourne le taux de remplissage de dossier dans le service budget |
| public S\_budget recherche\_budget ( String num\_dossier) | Retourne un objet de type S\_budget qui contient des champs dans le service budget de dossier numéro num\_dossier |
| public String pourcentage\_comptable (A\_comptable sc ) | Retourne le taux de remplissage de dossier dans l’agence comptable |
| public A\_comptable recherche\_comptable ( String num\_dossier) | Renvoie un objet de type A\_comptable qui contient des champs dans le service comptable de dossier numéro num\_dossier |
| Public void Insert\_marche( tous les champs de service marché sous forme String ) | Crée un nouveau dossier dans la table des marchés |
| Public void Insert\_commandes ( tous les champs de service commande sous forme String ) | Crée un nouveau dossier dans la table des marchés commençant par le service commande |
| Public void mis\_jour \_commandes ( tous les champs de service commande sous forme String , String num\_dos ) | Modifie les champs de la table correspondants au service commande dans la table des marchés |
| Public void mis\_jour \_budget ( tous les champs de service budget sous forme String, String num\_dos ) | Modifie les champs de la table correspondants au service budget dans la table des marchés |
| public void mis\_jour\_cmptabl (tous les champs de service comptable sous forme String, String num\_dos ) | Modifie les champs de la table correspondants au service comptable dans la table des marchés |
| public void mis\_jour\_marche(tous les champs de service marche sous forme String, String num\_dos ) | Modifie les champs de la table correspondants au service marche dans la table des marchés |
| public ArrayList<S\_commande> liste\_commande( ) | Retourne la liste des dossiers en cours de traitement dans le service commande |
| public ArrayList<S\_commande> liste\_commande\_trt( ) | Retourne la liste des dossiers traités dans le service commande |
| Public String Code\_couleur (Timestamp t1 , Timestamp t2 , String nb\_jour ) | Retourne le code couleur selon l’état de dossier par rapport au délai |
| Public int moins\_un\_jour(Timestamp t1 , int nb\_jour ) | Retourne 1 s’il reste un jour avant fin de délai pour traitement de dossier 0 sinon |
| ArrayList<Compte\_utilis> recherche\_Cpt( String nom\_utilis ) | Retourne une liste qui contient tous les comptes d’un utilisateur donné |
| Public void suppression\_parNomUtilis( String nomutilis , String typecompte , String service ) | Supprime de la table des utilisateurs un compte utilisateur dans un service donné |
| Public boolean existe\_user(String user\_name ) | Retourne vrai si le nom d’utilisateur donné possède au moins un compte utilisateur dans la table, faux sinon |
| boolean mis\_jour\_Compte( String nv\_nom , et tous les champs d’un compte utilisateur ) | Modifie les champs d’un compte utilisateur selon les valeurs données en entrée |
| public void insertion\_compte(tous les champs d’un compte utilisateur ) | Crée une nouvelle ligne dans la table des comptes utilisateurs (nouveau copte utilisateur) |
| àublic String ajout\_compte( tous les champs d’un compte utilisateur ) | Crée une nouvelle ligne dans la table (nouveau compte utilisateurs |
| Up\_ntf(String user\_name , String service , String[] donneeTableau , String[] donneTableaudate ) | Ajoute une nouvelle notification dans les deux tableaux en entrée |
| prublic void delet\_notif(String name\_user , String Service , String num\_dos ) | Supprime la notification du tableau des notifications dans la table |
| Public void add\_notif( String service , String num\_dos) | Ajoute notification à tous les utilisateurs du service en entrée |
| public String ajout\_type\_utilisateur(String user\_name , String type\_utilis , String service ) | Crée une nouvelle ligne dans la table avec un nom d’utilisateur existant mais le type et service différent |
| public ArrayList<Object> RechDossCourTrait()  Remarque : Cette fonction existe aussi pour retourner les dossiers en cours de traitement dans un seul service seulement.  public ArrayList<S\_marche>RechDossCourTraitMarche()  publicArrayList<S\_commandes>RechDossCourTraitCommande ()  public ArrayList<S\_budget>RechDossCourTraitBudget ()  public ArrayList<A\_comptable  >RechDossCourTraitComptable() | Renvoie une liste contenant tous les services de tous les services en cours de traitement |
| public ArrayList<S\_commandes> get\_historique\_commande(ArrayList<String> liste\_dos )  Remarque : Cette fonction existe pour tous les autres services . |  |
| public ArrayList<Object> get\_his\_ord(ArrayList<String> liste\_dos ) |  |
| public void lancer\_anne ( String filename ) | Ecrit le nom de la nouvelle base de données dans le fichier filename |
| public void update (String filename , String date ) |  |
| public ArrayList<ArrayList<Object>> recup\_dos\_cour\_trt() | Retourne une liste de liste de services des dossiers en cours de traitement ( Afin de les copier dans la nouvelle base de données ) |
| public ArrayList<ArrayList<Object>> ecrire\_dos\_c\_trt() | Ecrit dans la nouvelle base de données les dossiers en cours de traitement copiés de l’ancienne base de données |
| public void update\_del(String del\_m , String del\_c , String del\_b , String del\_ac , String num\_dos ) | Met à jour les délais des quatre services |
| public void cloturer\_dos(String filename ) | Met la date actuelle dans la date de clôture de dossier |
| public void cloturer\_anne(String filename , String archive , String file\_del ) | Clôture l’année actuelle et archive les dossier de la bdd actuelle traités et copie les dossiers non clôturés dans la nouvelle bdd |
| public ArrayList<Object> FiltreDate (String dateDebut , String dateFin ) | Retourne une liste d’objets contenant les services des dossier répondant aux critères non nul de filtre date (Date de lancement c’est la dateDebut et la date de clôture c’est la dateFin ) |
| public ArrayList<Object> FiltreEtatDossier(String Etat ) | Retourne une liste d’objets contenant les services des dossier répondant aux critères non nul de filtre Etat Dossier (remit en temps, en retard ou annulé) |
| public ArrayList<Object> FiltreChamps(String Service , String RespoDoss , String typeMarch ) | Retourne une liste d’objets contenant les services des dossier répondant aux critères non nul de filtre date (String Service, String RespoDoss , String typeMarch ) |
| public String exporter\_formatDoc(String nom\_ut , String numdos ) | Crée un fichier pdf contenant les champs de chaque service et retourne le chemin de fichier |
| public String exporter\_formatDoc\_marche(String nom\_ut , String numdos )  Remarque : Cette fonction existe pour chaque service seul | Crée un fichier pdf contenant les champs service marche et retourne le chemin de fichier |
| public void init\_ntf( ) | Réinitialise le tableau de notification des comptes utilisateurs à nul |
| public ArrayList<ArrayList<Object>> liste\_archive( ) | Retourne une liste de liste de tous les services des dossiers archivés |
| public void ecrire\_num\_dos ( String filename , String num\_dos ) | Ecrit le numéro de dossier dans le fichier filename |
| public String gen\_numdos ( String filename ) | Retourne |
| public void close ( ) | Ferme la connexion avec la base de données |

c)Package :Servlets  :

* 1)Classe Admin : Contient les méthodes nécessaires pour exécuter les opérations ( fonctionnalitées ) se trouvant dans le profil administrateur .

|  |  |
| --- | --- |
| **Class Admin (Servlet )** | |
| **Liste des attributs** | |
| **Attribut** | **Signification** |
| Private static HttpSession session = nul | Une variable session permettant de faire passer des paramètres d’une servlet à une autre ou une jsp |
| private Compte\_utilis cmp = nul | Garder les informations de l’utilisateur connecté |
| private Db bd = nul | Variable de type Db pour garder la connexion à la base de données |

|  |  |  |
| --- | --- | --- |
| **Liste des méthodes** | | |
| **Méthode** | | **Rôle** |
| private static void affiche\_img(Compte\_utilis , String chemin , HttpServletRequest request ) | | Affiche la photo de profil utilisateur qui se trouve dans le chemin donné |
| Protected void doGet( HttpServletRequest request , HttpServletResponse response ) | | Affiche les informations du profil, et le résultat de l’opération ou la fonctionnalité que l’utilisateur à choisit d’exécuter dont : Afficher mes information, ajouter un compte utilisateur, Ajouter au service, Consulter les comptes utilisateurs) |
| Protected void doPost (HttpServletRequest request, HttpServletResponse response ) | | Récupère à travers des formulaires les informations que l’utilisateur doit insérer pour effectuer une opération |
| **Class Ordonnateur (Servlet )** | | |
| **Liste des attributs** | | |
| **Attribut** | **Signification** | |
| Private static HttpSession session = nul | Une variable session permettant de faire passer des paramètres d’une servlet à une autre ou une jsp | |
| private Db bd = nul | Variable de type Db pour garder la connexion à la base de données | |
| private ArrayList<Object> liste | Une liste pour garder les dossiers en cours de traitement | |
| private Compte\_utilis cmp = nul | Garder les informations de l’utilisateur connecté | |
| **Liste des méthodes** | | |
| **Méthode** | **Rôle** | |
| Public S\_marches get\_dos\_marche ( String num\_dos )  Remarque : Cette fonction existe pour tous les services | Retourne un objet de type S\_marches résultat de la recherche de dossier numéro num\_dos dans liste | |
| Public ArrayList<S\_marches> get\_march( )  Remarque : Cette fonction existe pour tous les services existants | Retourne une liste d’objet de type S\_marches de tous les dossiers en cours de traitement se trouvant dans liste | |
| Public void affiche\_c\_trt (HttpServletRequest request ) | Selon le service choisit par l’utilisateur , affiche la liste des dossiers en cours de traitements dans ce service | |
| Public void cons\_glob(HttpServletRequest request , String num\_dos , boolean choix ) | Affiche la liste des dossiers en cours de traitement lorsque l’utilisateur choisit cette option | |
| Public void consult\_service(HttpServletRequest  request , String service , boolean choix ) | Affiche les champs de service choisit du dossier choisit | |
| Protected void doGet( HttpServletRequest request , HttpServletResponse response ) | Affiche les informations du profil, et le résultat de l’opération ou la fonctionnalité que l’utilisateur à choisit d’exécuter dont : Afficher mes information, Consulter un dossier , mettre à jour un dossier , consulter la liste des dossiers en cours de traitement ) | |
| Protected void doPost (HttpServletRequest request, HttpServletResponse response ) | Récupère à travers des formulaires les informations que l’utilisateur doit insérer pour effectuer une opération :  Numéro de dossier pour consulter , mettre à jour , exporter sous format pdf ou format excel , joindre un fichier , télécharger un fichier ) | |

2)Classe Consultation\_marche: Permet de consulter les champs d’un dossier dans le service marché .

|  |  |
| --- | --- |
| **Class Consultation\_marche(Servlet )** | |
| **Liste des attributs** | |
| **Attribut** | **Signification** |
| Private static HttpSession session = nul | Une variable session permettant de faire passer des paramètres d’une servlet à une autre ou une jsp |
| private Db bd = nul | Variable de type Db pour garder la connexion à la base de données |
| private ArrayList<S\_marches > liste\_marche | Une liste pour garder le service marche de tous les dossiers en cours de traitement |
| **Liste des méthodes** | |
| **Méthode** | **Rôle** |
| Protected void doGet( HttpServletRequest request , HttpServletResponse response ) | Recherche les dossier en cours de traitement dans le service marché , et met la liste dans une variable session ainsi la jsp de la consultation marche |

Remarque : La classe Consultation existe pour chaque service ( Commande , budget et comptable ) ainsi une classe pour la consultation globale ( Voir les champs de tous les services ) .

3)Classe Consultation\_marche: Permet de consulter les champs d’un dossier dans le service marché .

|  |  |
| --- | --- |
| **Class Ajout\_type\_user (Servlet )** | |
| **Liste des attributs** | |
| **Attribut** | **Signification** |
| **Liste des méthodes** | |
| **Méthode** | **Rôle** |
| Protected void doPost (HttpServletRequest request, HttpServletResponse response ) | Récupère à travers des formulaires les informations que l’administrateur doit insérer pour ajouter un nouveau compte utilisateur avec un nom d’utilisateur déjà existant :  ( Les champs d’un compte utilisateur ) et crée une nouvelle ligne dans la table des utilisateur |

Remarque : Il existe aussi la classe Creation\_compte : Même principe avec la classe Ajout\_type\_user or le nom d’utilisateur n’existe pas déjà .

4)Classe chang\_del\_dos : Permet à l’ordonnateur de changer le délai d’un dossier dans un service spécifique .

|  |  |
| --- | --- |
| **Class chang\_del\_dos(Servlet )** | |
| **Liste des attributs** | |
| **Attribut** | **Signification** |
| Private static HttpSession ses = nul | Une variable session permettant de faire passer des paramètres d’une servlet à une autre ou une jsp |
| private Db bd = nul | Variable de type Db pour garder la connexion à la base de données |
| Private String num\_dos | Contient le numéro de dossier que l’ordonnateur souhaite modifier les délais |
| Private String del | Contient le nombre de jours que l’utilisateur va insérer à travers un formulaire qui lui sera affiché lors de choix de cet opération |
| **Liste des méthodes** | |
| **Méthode** | **Rôle** |
| Protected void doPost (HttpServletRequest request, HttpServletResponse response ) | Récupère à travers un formulaire le délai et le service pour le mettre à jour dans ce service pour un dossier spécifié |
| Protected void doGet( HttpServletRequest request , HttpServletResponse response ) | Affiche le formulaire dans le profil ordonnateur pour récupérer le délai et le message après l’exécution de l’opération pour indiquer s’il y a une erreur ou elle a été exécuté avec succès |

Remarque : Il existe aussi la classe change\_det : Qui récupère un délai commun pour tous les dossiers et pour tous les services et le met dans un fichier texte, ces délais seront utilisé par défaut lors de la création d’un dossier pour les utilisateurs qui ont ce droit .

5)Classe exportpdf\_marche : Permet d’importer les information d’un dossier du service marché dans un fichier pdf .

|  |  |
| --- | --- |
| **Class exportpdf\_marche (Servlet )** | |
| **Liste des attributs** | |
| **Attribut** | **Signification** |
| Private static HttpSession session = nul | Une variable session permettant de faire passer des paramètres d’une servlet à une autre ou une jsp |
| private Db bd = nul | Variable de type Db pour garder la connexion à la base de données |
| private String num\_dos = nul | Contient le numéro de dossier à exporter sous format pdf |
| **Liste des méthodes** | |
| **Méthode** | **Rôle** |
| Protected void doPost (HttpServletRequest request, HttpServletResponse response ) | Récupère à travers un formulaire le numéro de dossier pour l’exporter sous format pdf ( sauf les champs de service marché ) et demander à l’utilisateur de choisir ou enregistrer le fichier |
| Protected void doGet( HttpServletRequest request , HttpServletResponse response ) | Affiche à l’utilisateur le formulaire pour récupérer le numéro de dossier à exporter |

Remarque : Cette classe existe pour chaque service ainsi pour l’exportation globale ( tous les services ) .

5)Classe form\_marche : Permet d’insérer un nouveau dossier dans le service marché par l’utilisateur qui a ce droit .

|  |  |
| --- | --- |
| **Class form\_marche(Servlet )** | |
| **Liste des attributs** | |
| **Attribut** | **Signification** |
| Private static HttpSession session = nul | Une variable session permettant de faire passer des paramètres d’une servlet à une autre ou une jsp |
| private Db bd = nul | Variable de type Db pour garder la connexion à la base de données |
| private S\_marches sm = null | Contient les champs de service marché de dossier à insérer |
| **Liste des méthodes** | |
| **Méthode** | **Rôle** |
| Protected void doPost (HttpServletRequest request, HttpServletResponse response ) | Fait appel à la fonction insert\_marche , et affiche un message après l’exécution |
| public static ger\_del(String file\_name ) | Récupère du fichier des délais file\_name , le délai correspondant au service marché |
| public static void insert\_marche(HttpServletRequest request ,Timestamp time\_tr , String file\_name ) | Récupère à travers un formulaire les champs du service marché pour insérer un nouveau dossier dans la table |

Remarque : Cette classe existe pour le service commande, car un dossier peut être lancé à partir du service commande .

6)Classe form\_marche\_mj : Permet de mettre à jour les champs d’un dossier dans le service marché .

|  |  |
| --- | --- |
| **Class form\_marche\_mj(Servlet )** | |
| **Liste des attributs** | |
| **Attribut** | **Signification** |
| Private static HttpSession session = nul | Une variable session permettant de faire passer des paramètres d’une servlet à une autre ou une jsp |
| private Db bd = nul | Variable de type Db pour garder la connexion à la base de données |
| private S\_marches sm = null | Contient les champs de service marché de dossier à mettre à jour |
| **Liste des méthodes** | |
| **Méthode** | **Rôle** |
| Protected void doPost (HttpServletRequest request, HttpServletResponse response ) | Fait appel à la fonction updatem, et affiche un message après l’exécution |
| public static void set\_datem(HttpServletRequest request , Timestamp time\_date\_lanc,Timestamp time\_date\_ouv,Timestamp time\_modif,Timestamp time\_tr ) | Récupère les dates à travers le formulaire sous format String et les transformer en Timestamp pour les insérer dans la table ) |
| public static void updatem\_marche(HttpServletRequest request ,Timestamp date\_tr , String file\_ecrire) | Récupère à travers un formulaire les champs du service marché à modifier pour mettre à jour le dossier et rajoute le numéro de dossier à l’historique des modifications de l’utilisateur |

Remarque : Cette classe existe pour tous les services .

7)Classe Statistique : Permet de visualiser la liste des dossiers répondant aux filtres et critères choisirent .

|  |  |
| --- | --- |
| **Class Statistique (Servlet )** | |
| **Liste des attributs** | |
| **Attribut** | **Signification** |
| Private static HttpSession session = nul | Une variable session permettant de faire passer des paramètres d’une servlet à une autre ou une jsp |
| private Db bd = nul | Variable de type Db pour garder la connexion à la base de données |
| **Liste des méthodes** | |
| **Méthode** | **Rôle** |
| Protected void doPost (HttpServletRequest request, HttpServletResponse response ) | Fait appel à la fonction insert\_marche , et affiche un message après l’exécution |
| Protected void doGet( HttpServletRequest request , HttpServletResponse response ) | Affiche les résultats ( liste des dossiers ) répondants au critères choisit selon le filtre choisirent |

8)Classe Envoyé\_par\_marché : Permet de mettre à jour la date d’envoi de dossier dans le service ou son traitement a été terminé , ainsi la date de réception dans le service suivant et notifier les utilisateur concernés .

|  |  |
| --- | --- |
| **Class Envoyé\_par\_marché (Servlet )** | |
| **Liste des attributs** | |
| **Attribut** | **Signification** |
| Private static HttpSession session = nul | Une variable session permettant de faire passer des paramètres d’une servlet à une autre ou une jsp |
| **Liste des méthodes** | |
| **Méthode** | **Rôle** |
| Protected void doPost (HttpServletRequest request, HttpServletResponse response ) | Met la date actuelle dans la date d’envoi du dossier dans service marché au service commande , ainsi dans la date de réception de dossier dans le service commande , lorsque l’utilisateur clique sur envoyer vers service suivant |

Remarque : Cette classe existe pour tous les services ( marché , budget , commande et comptable ) .

*9)-Classe Modificateur\_Marche :*

La classe pour le modificateur marché qui contient toutes les fonctions que peut réaliser un modificateur dans le service marché.

|  |  |
| --- | --- |
| **Class Modificateur\_Marche** | |
| **Liste des attributs** | |
| **Attribut** | **Signification** |
| private ArrayList<S\_marches> liste | Liste des dossiers en cours de traitement dans le  Service marché |
| private HttpSession ses | La variable session ou on garde toutes les informations de la connexion |
| private Compte\_utilis cmp | La variable ou est sauvegardée le compte de l’utilisateur (compte modificateur marché) |
| private Db bd | La variable pour effectuer des opérations sur la base de données |
| private int choix | La variable pour savoir si l’utilisateur recherche à partir de la liste de son historique ou celle des dossiers en cours de traitement ou à partir de la barre de recherche, pour pouvoir après facilité la recherche du dossier |
| **Liste des méthodes** | |
| **Méthode** | **Rôle** |
| Constructeur, Getters et Setters | Récupérer et modifier les attributs |
| public S\_marches get\_from\_hist(String num\_dos) | Récupérer un dossier de la liste de l’historique de l’utilisateur |
| public S\_marches get\_from\_liste(String num\_dos) | Récupérer un dossier de la liste des dossiers en cours de traitement dans le service |
| public void consult\_service\_modif(HttpServletRequest request,String num\_dos , int oper ) | Consulter un dossier (num\_dos) et afficher les champs du dossier, elle prend en paramètre un entier pour savoir si l’opération est une modification ‘1’ ou consultation ‘0’ |
| protected void doGet(HttpServletRequest request, HttpServletResponse response) | La fonction pour effectuer les requêtes de type Get comme consultation, affichage des champs… |
| protected void doPost(HttpServletRequest request, HttpServletResponse response) | La fonction pour effectuer les requêtes de type Post, qui sont les requêtes d’envois des formulaires pour les écrire en base de données |

**Remarque :**

Il existe aussi 3 classes pour les modificateurs commande, budget, et comptable, ces dernières ont le même comportement même attributs et même fonctions que la classe Modificateur\_Marche.

*10)-Classe Consultation :*

La classe pour le consulteur elle contient les fonctions tels que consultation globale, consulter un seul dossier, consulter archive... (toutes les opérations de consultation).

|  |  |
| --- | --- |
| **Class Consultation** | |
| **Liste des attributs** | |
| **Attribut** | **Signification** |
| private ArrayList<Object> liste | Liste des dossiers en cours de traitement dans le tous les services, Objet peut être de type service marché de type service commande, budget et comptable |
| private HttpSession ses | La variable session ou on garde toutes les informations de la connexion |
| private Compte\_utilis cmp | La variable ou est sauvegardée le compte de l’utilisateur (compte modificateur marché) |
| private Db bd | La variable pour effectuer des opérations sur la base de données |
| **Liste des méthodes** | |
| **Méthode** | **Rôle** |
| Constructeur, Getters et Setters | Récupérer et modifier les attributs |
| public String get\_i(String service) | Retourne une chaine de caractère pour afficher de manière claire a l’utilisateur le service qu’il choisit, ‘1’ pour dire que l’utilisateur a choisit le service marché, ‘2’ commande… |
| public S\_marches get\_dos\_marche(String num\_dos) | Récupérer les dossiers en cours de traitement dans le service marché via la liste d’objet |
| public S\_commandes get\_dos\_commande(String num\_dos) | Récupérer les dossiers en cours de traitement dans le service commande via la liste d’objet |
| public S\_budget get\_dos\_budget(String num\_dos) | Récupérer les dossiers en cours de traitement dans le service budget via la liste d’objet |
| public A\_comptable get\_dos\_comptable(String num\_dos) | Récupérer les dossiers en cours de traitement dans le service comptable via la liste d’objet |
| public void consult\_service(HttpServletRequest request,String num\_dos,String service,Boolean choix) | Afficher le numéro de dossier que l’utilisateur veut consulter, le Boolean est pour savoir su l’utilisateur a utilisé la barre de recherche ou la liste déroulante |
| protected void doGet(HttpServletRequest request, HttpServletResponse response) | La fonction pour effectuer les requêtes de type Get comme consultation, affichage des champs… |
| protected void doPost(HttpServletRequest request, HttpServletResponse response) | La fonction pour effectuer les requêtes de type Post, lorsque l’utilisateur utilise la barre de recherche |

*11)-Classe historique\_comptable :*

La classe pour l’historique du comptable ou il peut consulter tous sont historique.

|  |  |
| --- | --- |
| **Class historique\_comptable** | |
| **Liste des attributs** | |
| **Attribut** | **Signification** |
| private ArrayList<String> dos | Liste ou récupère les noms de dossiers récemment modifiés par l’utilisateur sous forme de chaine de caractère |
| private HttpSession ses | La variable session ou on garde toutes les informations de la connexion |
| private ArrayList<A\_comptable> hist | Liste ou on sauvegarde les dossiers que l’utilisateur a récemment modifiés |
| private Db bd | La variable pour effectuer des opérations sur la base de données |
| **Liste des méthodes** | |
| **Méthode** | **Rôle** |
| protected void doGet(HttpServletRequest request, HttpServletResponse response) | La fonction qui est appelée quand l’utilisateur clic sur bouton historique, cette fonction recherche les dossiers récemment modifiés par l’utilisateur et elle les affiches |
| protected void doPost(HttpServletRequest request, HttpServletResponse response) | La fonction pour afficher la liste des dossiers si l’utilisateur effectue une requête de type Post |

**Remarque :**

Il existe 4 autre classes identique historique marché, historique commande, historique budget, historique ordonnateur qui ont les même attributs, méthodes et même fonctionnent mais le type de retour et de recherche n’est pas le même.

*12)-Classe Excel\_commande :*

La classe pour générer le fichier Excel pour le service commande.

|  |  |
| --- | --- |
| **Class Exel\_commande** | |
| **Liste des attributs** | |
| **Attribut** | **Signification** |
| private final String[] time | Tableau ou on sauvegarde les attributs de type date ou time |
| private HttpSession ses | La variable session ou on garde toutes les informations de la connexion |
| **Liste des méthodes** | |
| **Méthode** | **Rôle** |
| protected void setBordersToMergedCells(HSSFSheet sheet, CellRangeAddress rangeAddress) | La fonction pour afficher les bordures dans le fichier Excel entre les diffèrents attributs |
| public Boolean contain(String name\_col) | Pour savoir su un attribut est de type time ou non |
| protected void doGet(HttpServletRequest request, HttpServletResponse response) | La fonction qui est appelée quand l’utilisateur clic sur Télécharger Excel, elle génère le fichier Excel pour le service commande et le télécharge pour l’utilisateur |

**Remarque :**

Il existe 4 autre classes Excel budget, Excel marché, Excel globale, Excel comptable, elles ont les mêmes attributs et méthodes, chacune d’elle génère le fichier Excel pour son service et elle le télécharge, sauf pour le global ou on génère le fichier Excel pour tous les services (tous les attributs de tous les services).

*13)-Classe Archive :*

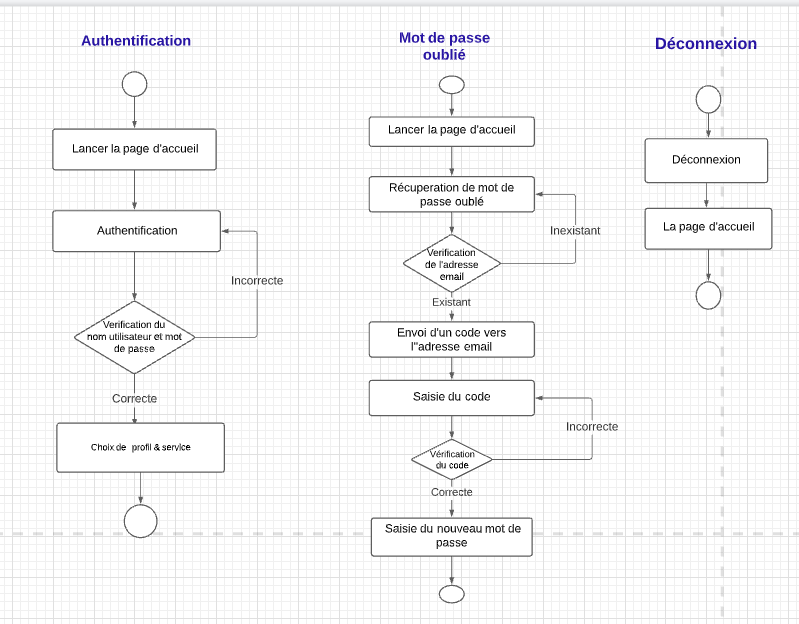
La classe pour consulter les archives.

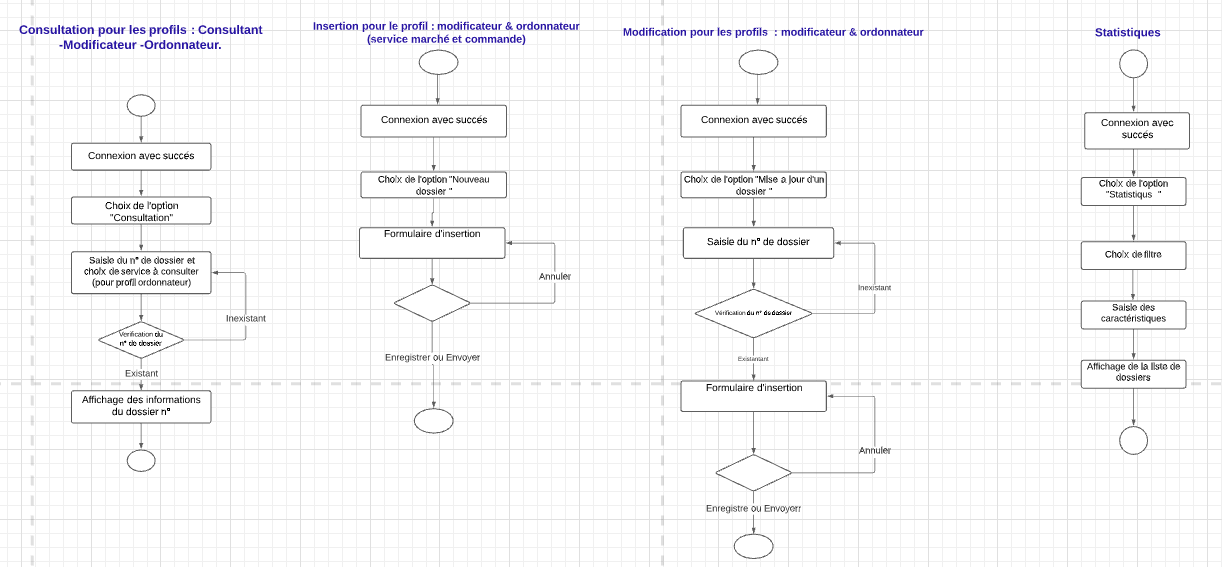
|  |  |
| --- | --- |
| **Class Archive** | |
| **Liste des attributs** | |
| **Attribut** | **Signification** |
| private HttpSession ses | La variable session ou on garde toutes les informations de la connexion |
| **Liste des méthodes** | |
| **Méthode** | **Rôle** |
| public String convert(String l) | Convertir une chaine de caractères pour qu’elle respecte les critères de cette classe |
| protected void doGet(HttpServletRequest request, HttpServletResponse response) | La fonction qui est appelée quand l’utilisateur clic sur Archive, elle affiche toutes les anciennes bases de données que l’utilisateur a archivé pour qu’il puisse choisir une pour la consulter |
| protected void doPost(HttpServletRequest request, HttpServletResponse response) | La fonction qui est appelée quand l’utilisateur a choisi une base de données à consulter, elle charge la base de données et toutes les informations qu’elle contient, ensuite elle les affiches sous forme de liste de dossiers que l’utilisateur pourra consulter |

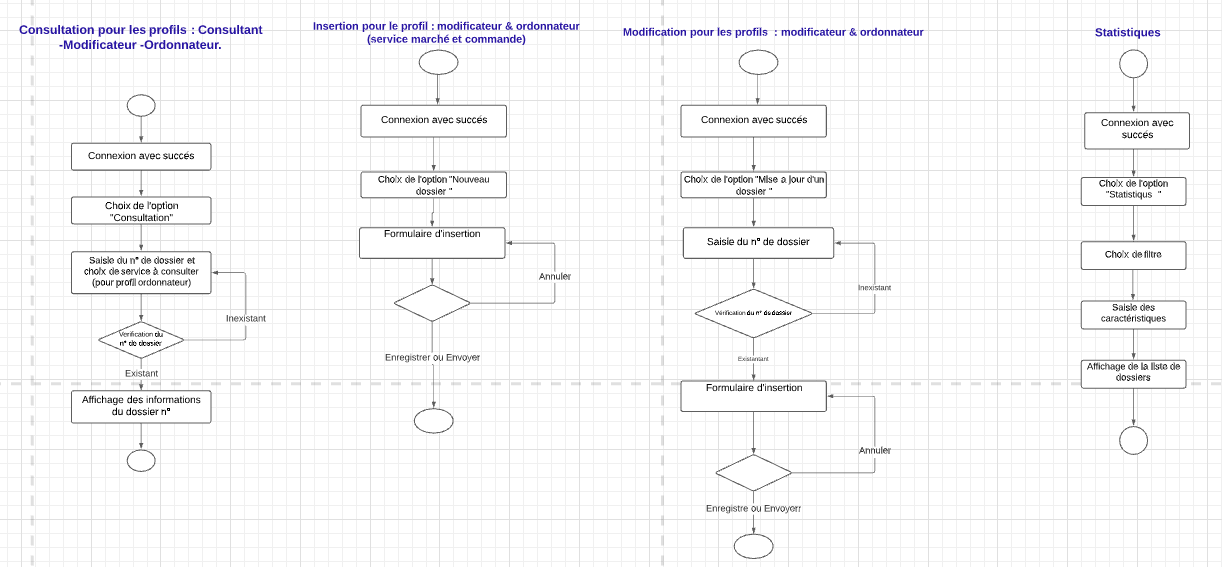
*14)-Classe Fichier :*

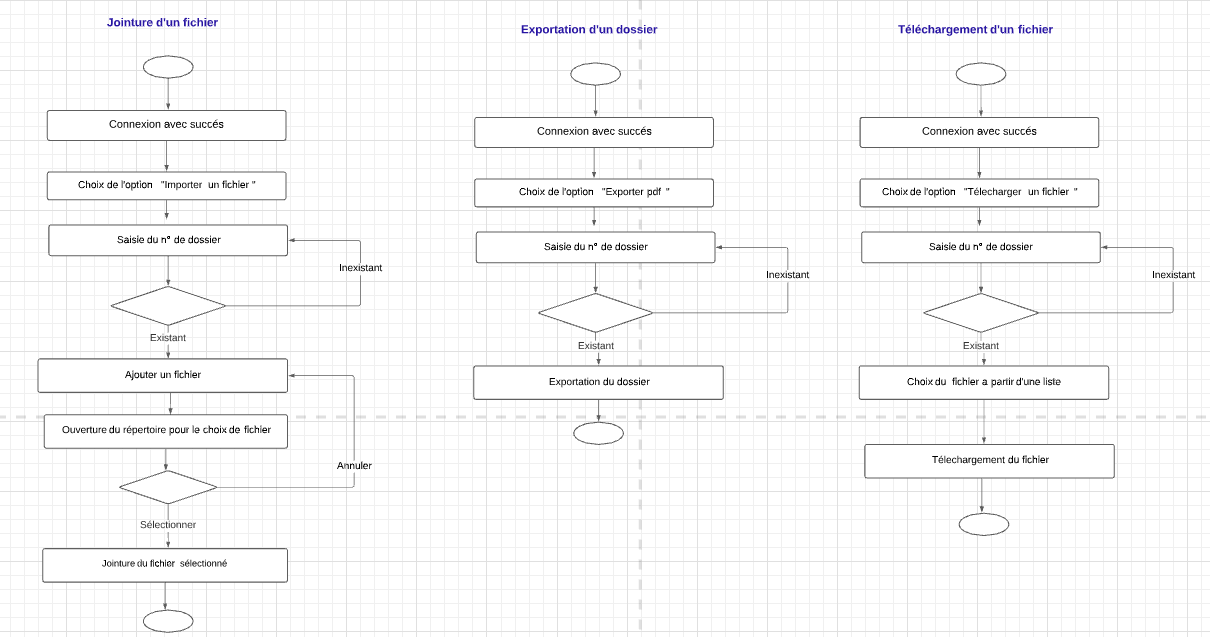
La classe pour gérer le fichier historique (ajout, suppression, récupérer des informations…),ce fichier contient l’historique pour chaque compte a chaque fois qu’un utilisateur modifie un dossier ce dernier est ajouter dans ce fichier, et si il veut consulter sont historique on récupère les numéros de dossier du compte et on les recherches dans la base de données.

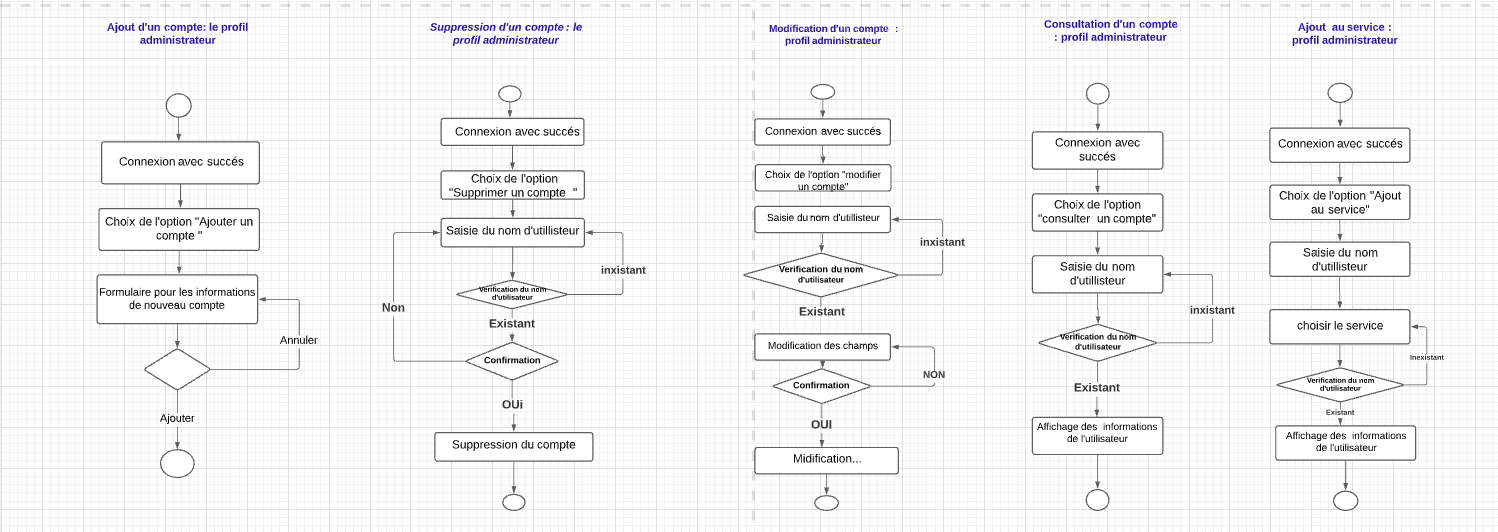
|  |  |
| --- | --- |
| **Class Fichier** | |
| **Liste des attributs** | |
| **Attribut** | **Signification** |
| **Liste des méthodes** | |
| **Méthode** | **Rôle** |
| public static ArrayList<String> get\_dos\_from\_string(String ma\_chaine) | Elle prend une chaine de caractère contentent les numéros de dossiers concaténer qui ont été extraits partir d’un fichier texte qui contient pour chaque ligne un utilisateur et les fichiers qu’il a récemment modifiés (son historique) et retourne une liste des dossiers pour pouvoir après les rechercher en base de données |
| public static int exsist(String ma\_chaine,String chaine) | Pour vérifier si un dossier existe dans la liste des dossiers si oui elle retour sa taille (le nombre de caractères sinon -1) |
| public static int get\_avant\_dernier(String ma\_chaine) | Récupère l’indice de l’avant dernier dossier dans la chaine des dossiers |
| public static String add(String ma\_chaine,String dos) | Cette fonction ajoute un dossier dans la chaine des dossiers et elle retourne la nouvelle chaine |
| public static void init(String filename) | Cette fonction initialise le fichier historique (fichier vide) |
| public static ArrayList<String> lire(String filename) | Cette fonction lit le fichier de l’historique des utilisateurs et elle le retourne sous format d’une liste de chaine de caractères ou chaque indice représente une ligne dans le fichier |
| public static void ecrire(String filename,String num\_dos,String user\_name,String type\_compte,String service) | Ajout d’un dossier dans le fichier historique |
| public static String sup\_dos(String ma\_chaine,String num\_dos) | Supprime un dossier de la chaine des dossiers et retourne la nouvelle chaine |
| public static void update\_all\_user(String filename,String name,String nv\_name) | Elle met à jour le fichier en remplacent tous les noms d’utilisateurs qui contiennent le paramètre nom par le nouveau nom (nv\_nom) |
| public static void update(String filename,String name,String nv\_name,String type\_compte,String service) | Elle met à jour le fichier en remplacent la ligne qui contient le nom le type et le service par le nouveau nom |
| public static void supp(String filename,String num\_dos,String user\_name,String type\_compte,String service) | Elle supprime un dossier dans le fichier historique |
| public static ArrayList<String> recup\_info(String filename,String user\_name,String type\_compte,String service) | Elle récupère l’historique d’un compte via le fichier historique sous forme d’une liste qui contient les numéros de dossiers |

1. 
   * 1. Diagramme d’activité :

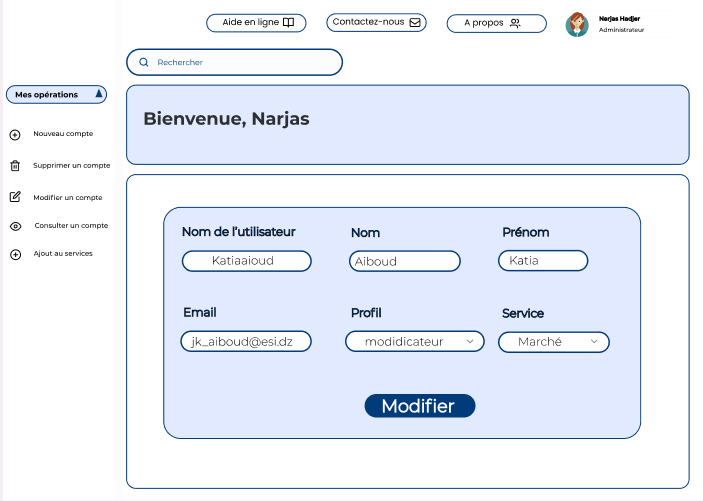
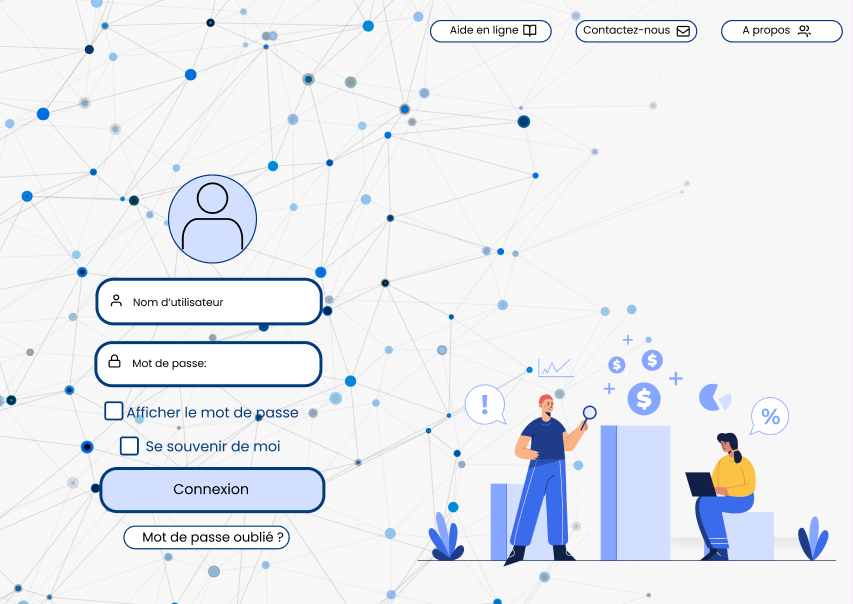




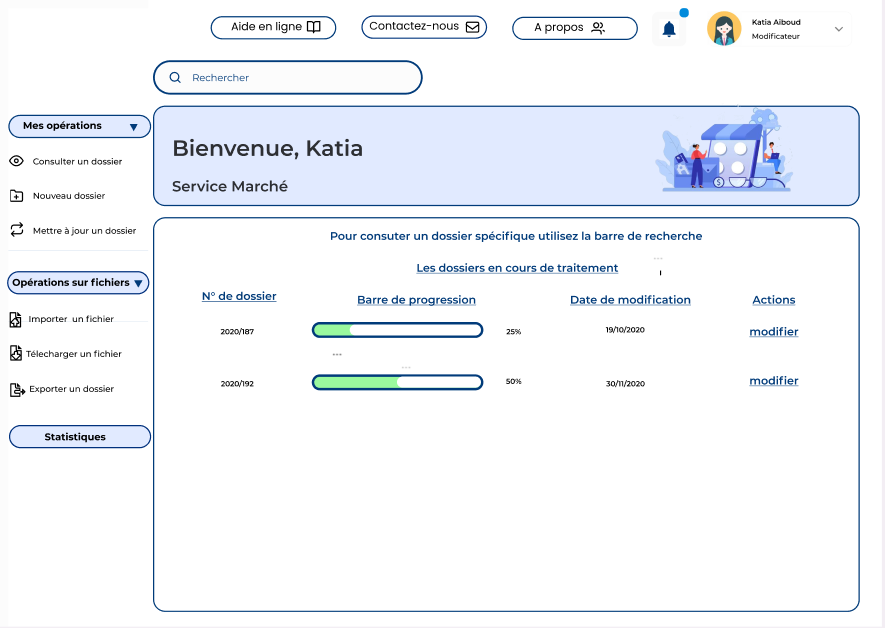




* + 1. Le prototype de l’application :







* + 1. Conclusion :

Après une bonne analyse du problème en se basant sur l’approche orienté objet, on a pu conclure avec

une conception générale validée par l’équipage tout en respectant les objectifs du projet. Une

conception dans laquelle on a schématisé notre solution et défini les différentes classes nécessaires

pour la réalisation du projet et qu’on implémentera dans le chapitre suivant.

* 1. Réalisation
     1. Introduction :

Dans ce chapitre, on va présenter le choix des outils, l’aspect programmation et implémentation des algorithmes de notre application, ainsi les pages web réalisés. La réalisation de notre application web s’est divisée en deux parties : la partie « noyau » et la partie graphique. La partie « noyau » consiste à implémenter les principales classes définies dans le chapitre précèdent. La partie graphique consiste à réaliser les différentes pages définis auparavant dans le prototype, les intégrer avec le noyau et assurer leur fonctionnement.

* + 1. Choix des outils :

**2.1.JAVA :** Pour la programmation et l’implémentation des algorithmes des différentes fonctionnalités de notre application web, on a opté pour le fameux langage JAVA. Java est un langage de programmation orienté objet, dynamique qui est indépendant de toute plateforme et assure une bonne gestion du mémoire. Ainsi, on a utilisé le **JAVA EE** qui est une extension de la plateforme standard java qui facilite le développement d’applications web distribuées, déployées et exécutées sur un serveur.

**2.2.PostgreSQL :** Comme notre projet consiste à créer une application web qui permettrait aux utilisateurs de suivre l’évolution d’un dossier dans chaque service en coordination avecles autres services ,on a besoin d’utiliser une base de donnée afin de sauvegarder et gérer les différentes informations des dossiers dans chaque service , notamment les informations des comptes d’utilisateurs .Pour ceci nous avons choisi **PostgreSQL**qui  est un [système de gestion de base de données](https://www.oracle.com/fr/database/systeme-gestion-base-de-donnees-sgbd-definition.html) relationnelle orienté objet puissant et open source qui est capable de prendre en charge en toute sécurité les charges de travail de données les plus complexes.

**2.3.HTML5 :** Pour la structuration des différentes principales et secondaires pages de notre application web, on a utilisé le **HTML5** qui est un langage de balisage des pages web qui est destiné uniquement à représenter la structure d’une page.

**2.4.CSS :**Pour styler les différentes pages de l’application web définis auparavant dans le prototype, on a utilisé le **CSS**qui est un langage informatique utilisé sur le web pour la mise e, forme des fichiers HTML ou XML.

**2.5.JavaScript :** JavaScript est un langage de script orienté objet principalement utilisé dans les pages HTML. Ce langage permet une interaction avec l’utilisateur en fonction de ses actions ce qui rend les pages web plus dynamiques.

**2.6.JQuery :** JQuery est une bibliothèque [JavaScript](https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1203585-javascript/) gratuite, compatible avec l'ensemble des navigateurs Web (Internet Explorer, Safari, Chrome, Firefox, etc.),. Il permet d'agir sur les codes HTML, CSS, JavaScript et AJAX et s'exécute essentiellement côté client.

**2.7.Figma :** C’est un outil en ligne destiné au designers .il permet de concevoir l’interface graphique et d’élaborer les prototypes. Nous avons utilisé cet outil lors de la conception des maquettes et la réalisation de prototype .il nous a été très bénéfique car il nous a permet de travailler tous ensemble sur le même projet Figma.

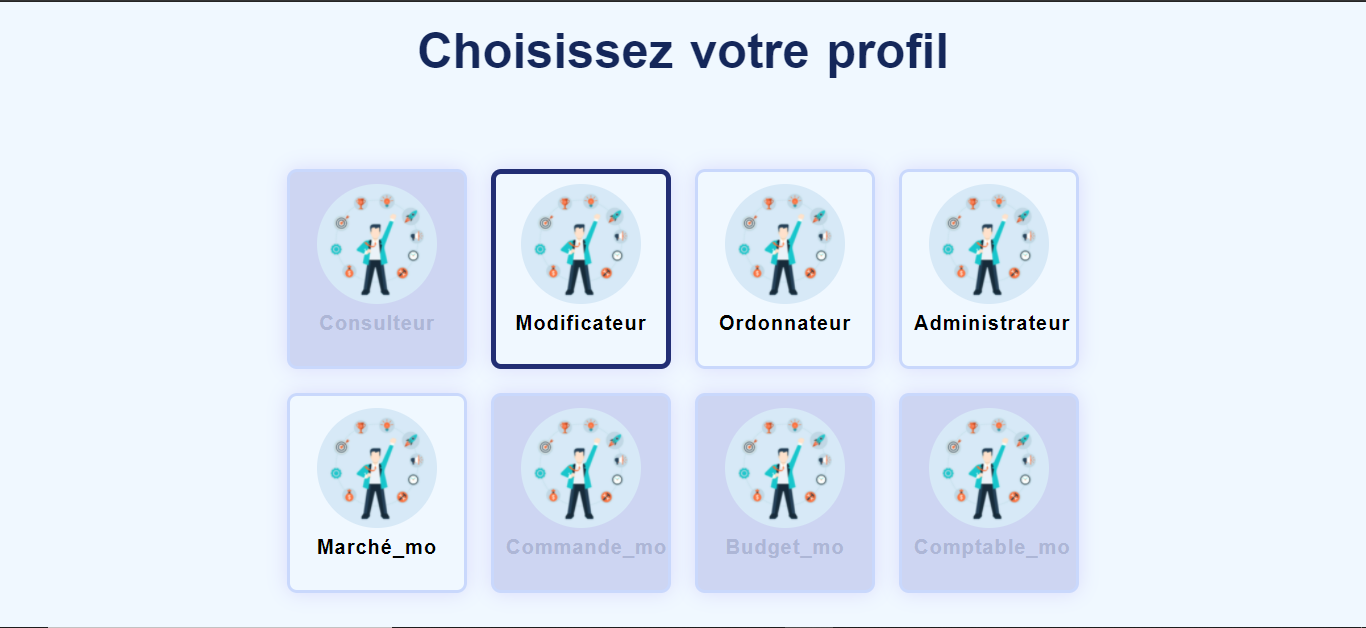
**2.8.Adobe Illustrator & adobe Photoshop :**Ce sont deux logiciels de création graphique vectorielle. Ils permettent aux utilisateurs d’éditer et de composer des graphiques vectoriels de façon interactive. On les a utilisés pour la création du logo de l’application.

**2.9.LucidChart :** C’est une plateforme de collaboration, basée sur le cloud, permettant la création des différents types de diagrammes. Elle a été utilisé pour la création de diagramme d’activités de l’application web.

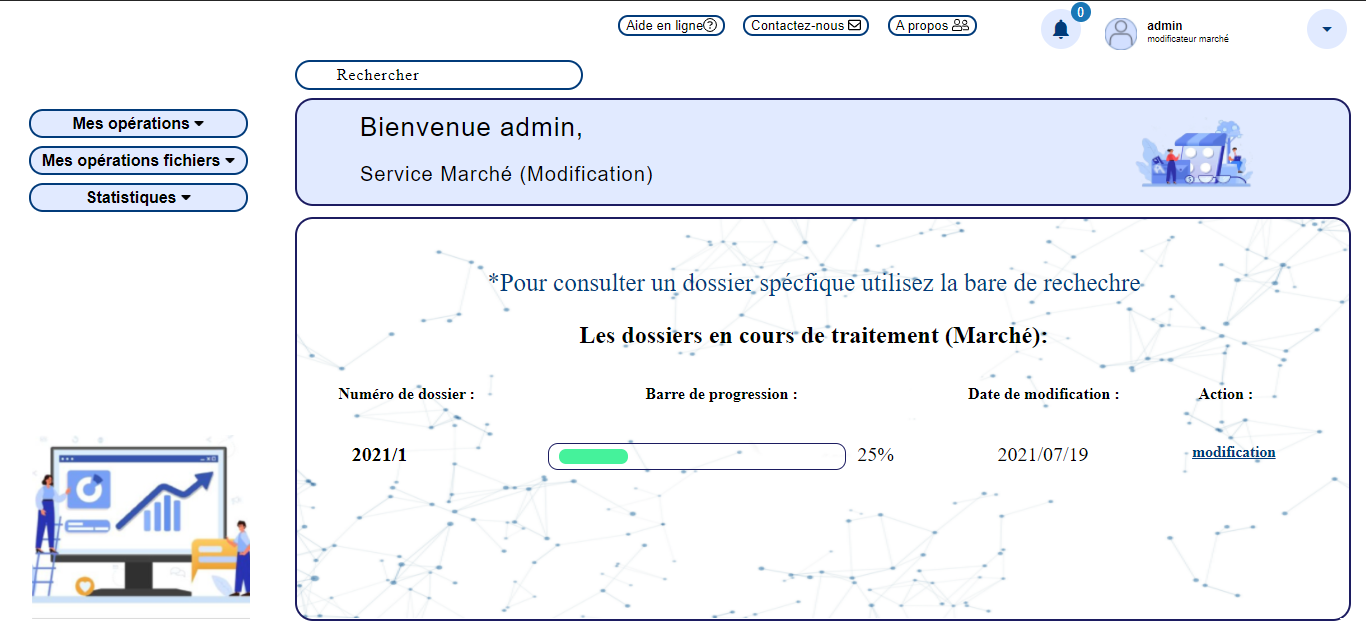
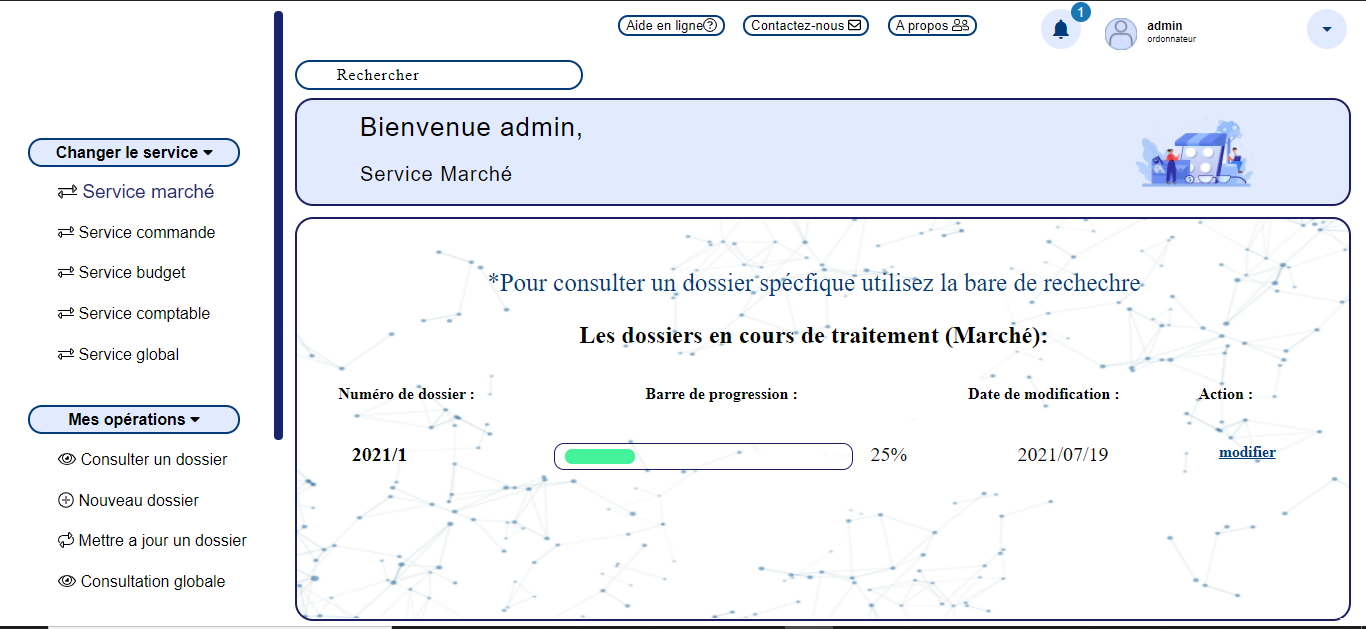
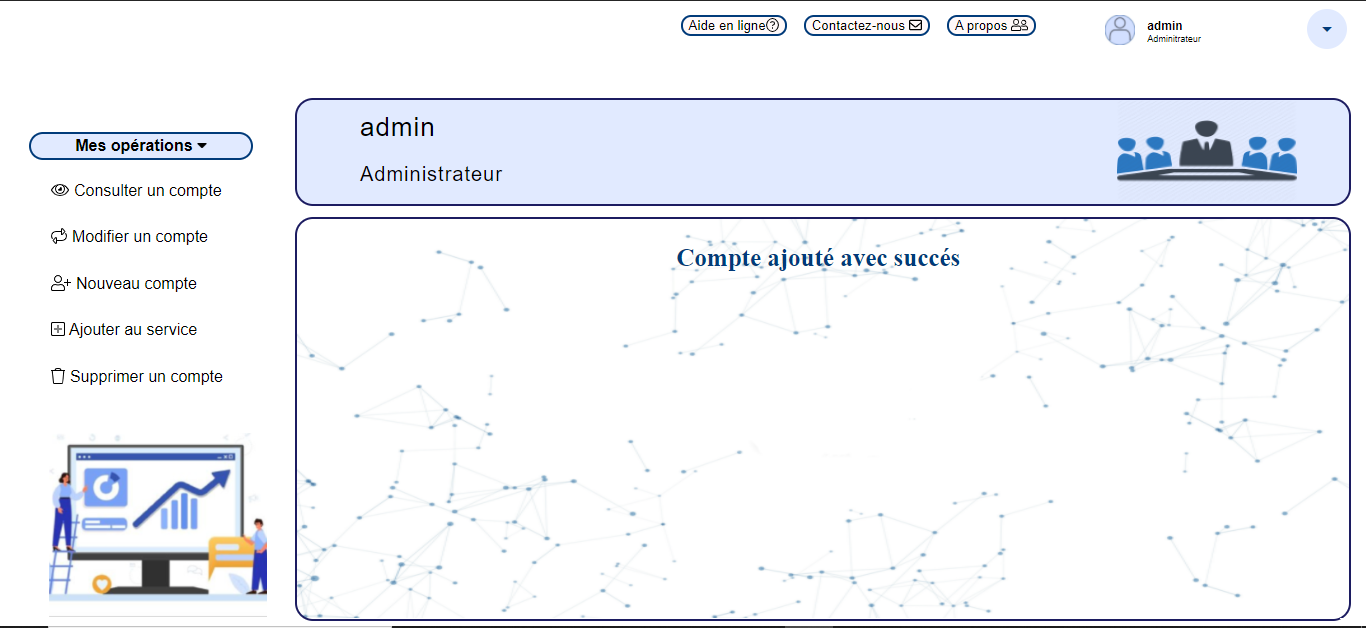
**2.10.Environnements de développement :**

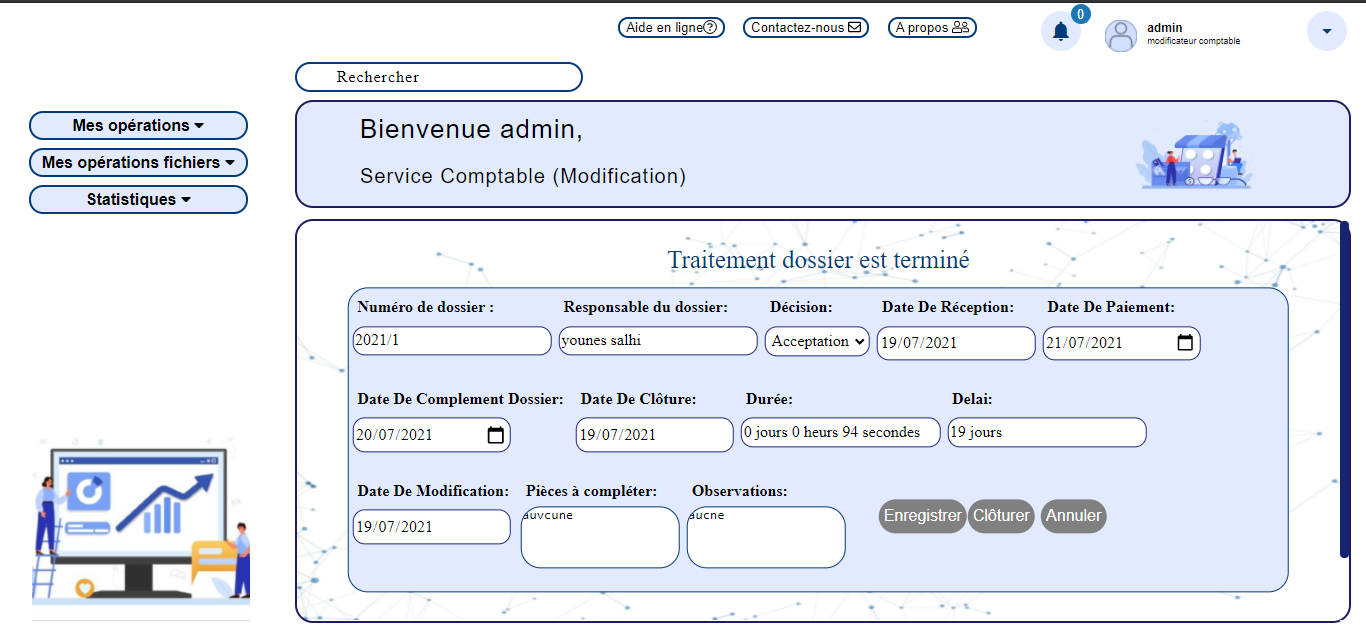
* **Eclipse IDE :** L’un des plus puissants environnements de développement utilisé pour le langage JAVA. C’est un IDE gratuit et open source avec pleins de fonctionnalités.
* **Visual Studio Code :** C’est un éditeur de code source qui peut être utilisé avec une variété de langages de programmation, notamment HTML5, CSS et JavaScript.

**2.11.Collaboration et coordination :** Afin d’organiser notre télétravail, nous avons utilisé **Google Meet** pour organiser régulièrement des réunions entre les membres d’équipe afin d’évaluer l’avancement de la réalisation du projet. Nous avons aussi créé un dossier pour les membres d’équipe sur **Google drive** pour pouvoir partager les documentations et les différentes parties du projet.

1. **Les pages de l’application  :**
   1. **La page d’accueil :**
   2. **La page de choix de profil & service:**

****

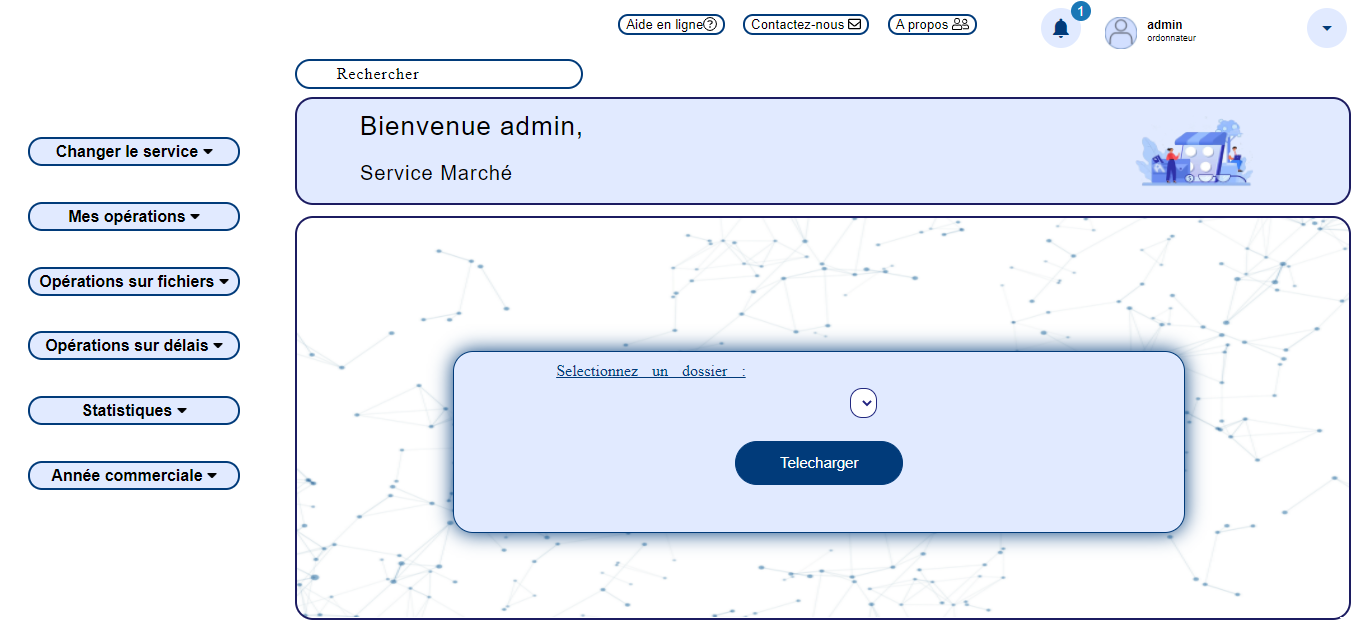
* 1. **Compte Consulteur :**
  2. **Compte modificateur :**
  3. **Compte ordonnateur :**
  4. **Compte administrateur**

1. **Les fonctionnalités de l’application :** 
   1. **Les opérations sur les dossiers :**
      1. **Consultation d’un dossier :** Tout utilisateur disposant d’un profil consulteur, modificateur et ordonnateur peut consulter les informations d’un dossier spécifique dans son service .Une barre de progression apparait au dessous des informations du dossier : elle indique l’état d’avancement du traitement de dossier.
      2. **Insertion d’un nouveau dossier :** Tout utilisateur disposant d’un profil modificateur dans le service marché et commande ou bien un profil ordonnateur peut insérer un nouveau dossier en remplissant un formulaire.
      3. **Mise à jour d’un dossier :** Tout utilisateur disposant d’un profil modificateur ou ordonnateur peut insérer les informations d’un dossier déjà crée dans le service marché ou commande. Il peut aussi les modifier tant qu’il n’a pas envoyer le formulaire au prochain service ou si le dossier est clôturé dans le service .comptable .
      4. **Les statistiques :** Les utilisateurs des profils consulteur, modificateur et ordonnateur ont la possibilité de faire des statistiques sur les dossiers c'est-à-dire en choisissant un filtre et en précisant les caractéristiques, une liste de dossiers va apparaitre selon le filtre et les caractéristiques choisis.

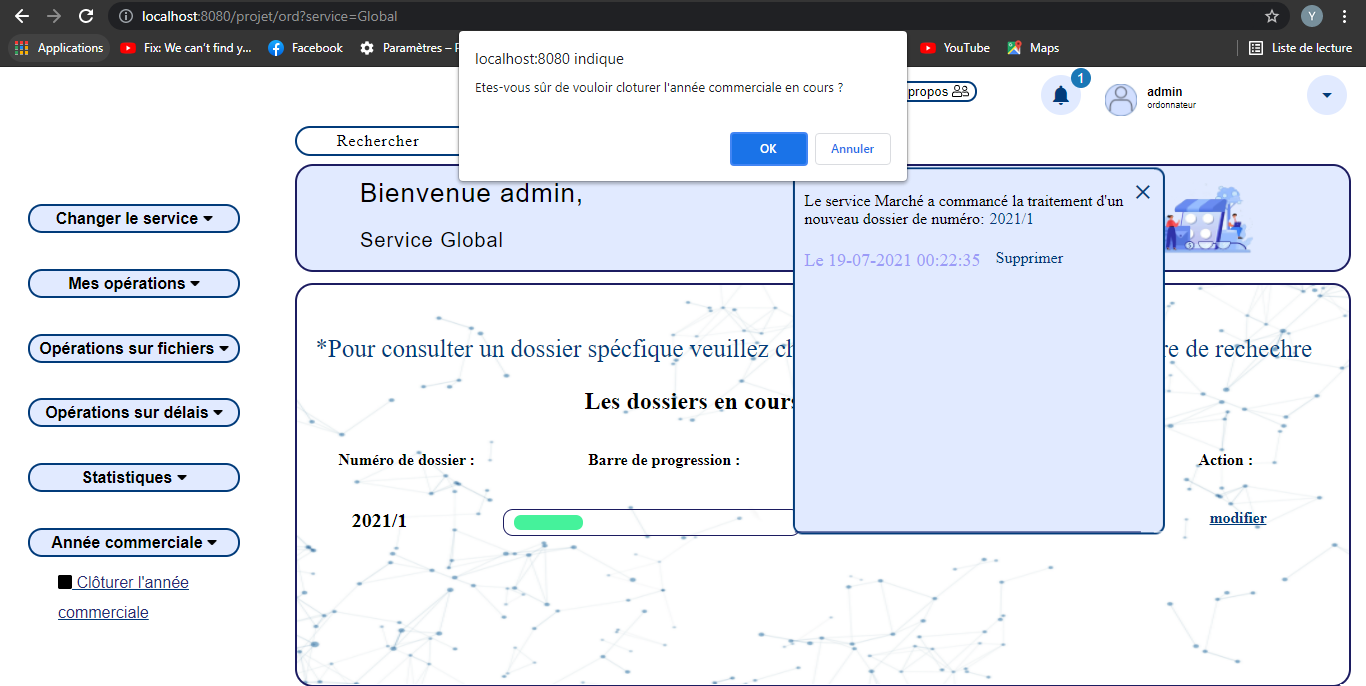


* 1. **Les opérations sur les fichiers :** Ces fonctionnalités sont disponibles pour tous les utilisateurs disposant d’un profil consulteur, modificateur ou ordonnateur.
     1. **Importation d’un fichier :** Cette fonctionnalité permet à l’utilisateur de joindre un fichier à partir de son ordinateur.

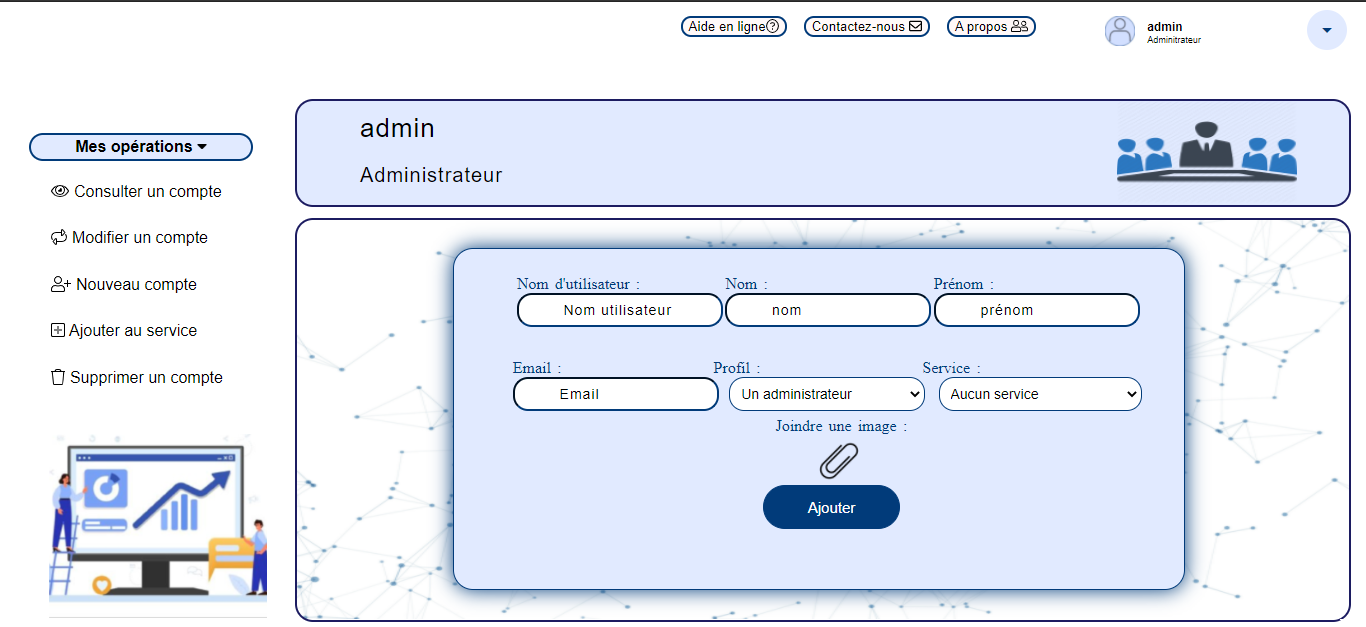
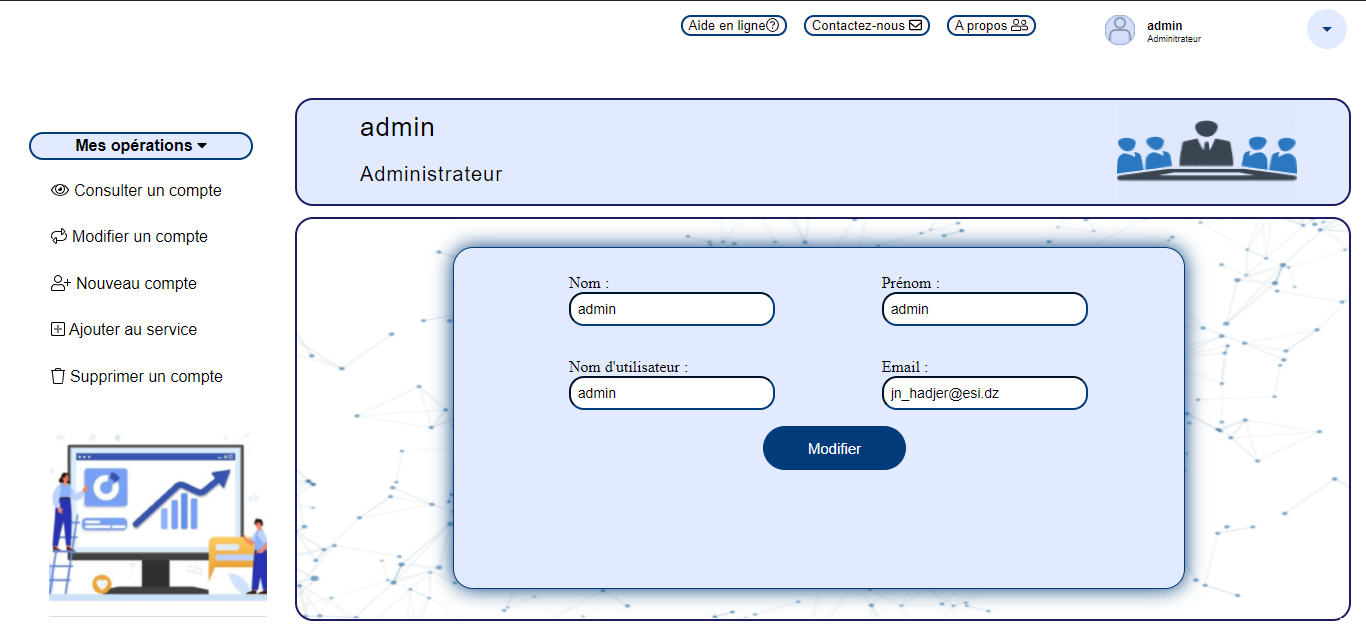


* + 1. **Téléchargement d’un fichier :** Cette option offre à l’utilisateur la possibilité de télécharger des dossiers déjà importés à partir d’une liste.
    2. **Exportation d’un dossier :** Cette fonctionnalité permet à l’utilisateur de télécharger les informations concernant un dossier spécifique sous format PDF ou sous format EXCEL  pour un profil ordonnateur.
  1. **Les opérations sur les délais :** Ces opérations sont disponibles pour le profil ordonnateur .Elles lui permettent de modifier le délai de traitement d’un dossier .Tous dossier transmis après cette durée est considérer comme un dossier en retard
     1. **Changement de délai pour un dossier dans un service :**
     2. **Changement de délai pour tous les services :**



* 1. **L’année commerciale :**
     1. **Clôture d’une année commerciale :** Vers la fin de l’année commerciale, l’ordonnateur va clôturer l’année et qu’une autre se lance . Tous les dossiers traités durant cette année seront archivés.

****

* + 1. **Les archives :** Un ordonnateur peut consulter les dossiers d’une année archivé.
  1. **Les fonctionnalités d’un profil administrateur** 
     1. **Ajout d’un compte :**
     2. **Modification d’un compte :**

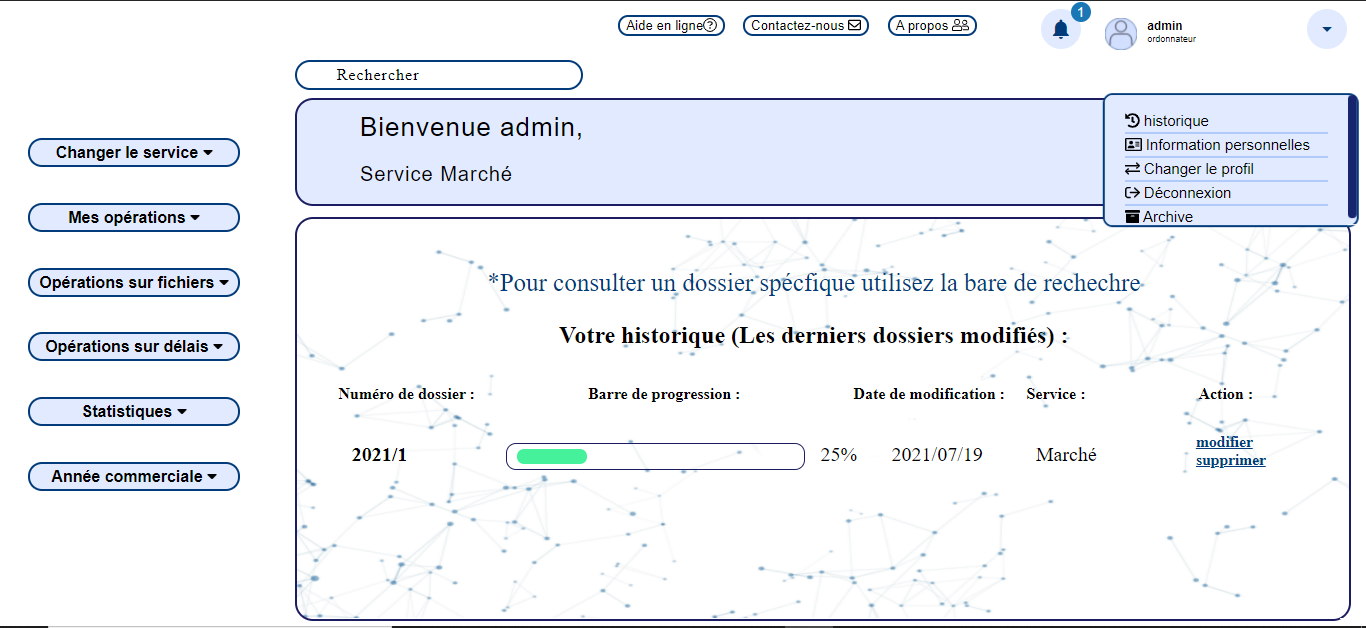


* + 1. **Ajout au service :**
    2. **Consultation d’un compte :**



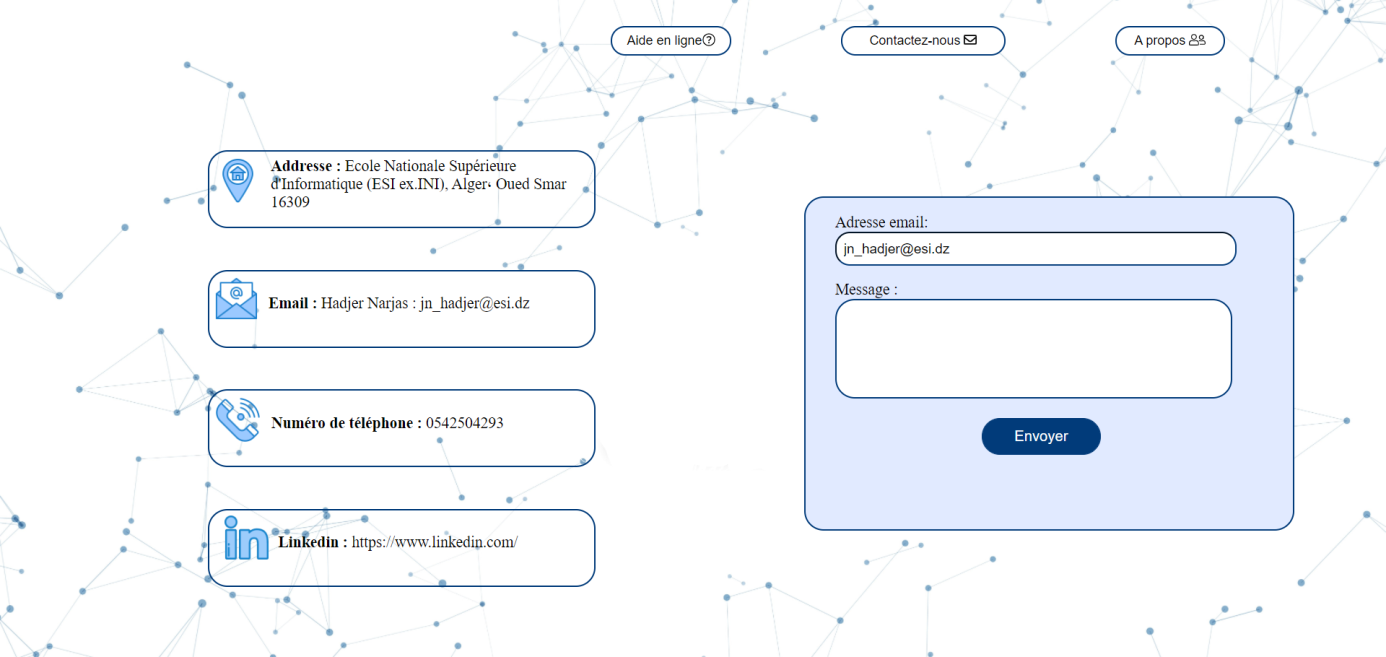
* + 1. **Suppression d’un compte :**



* 1. **Autres fonctionnalités :** 
     1. **Le bouton des notifications :** A chaque fois un formulaire d’un dossier est envoyé, l’utilisateur de service suivant sera notifié pour qu’il puisse consulter ou modifier le dossier.
     2. **Les informations personnelles :** Un utilisateur peut consulter ses informations personnelles et ainsi les modifier.
     3. **Historique :**Un utilisateur peut consulter son historique ,ou il trouvera une liste de ces actions récentes avec la possibilité de les supprimer .
  2. **La page d’aide**

Notre application web dispose d’une page d’aide en ligne afin d’assister l’utilisateur durant son premier contact avec l’application. Cette page consiste un espace d’assistance pour répondre aux éventuels questionnements du client lors de l’utilisation de l’application.

Elle contient également une section « Contactez-nous » dans le cas où le client souhaiterait obtenir davantage d’informations.

****

1. **Conclusion :**

Après mûre réflexion, longue entente et une bonne analyse du problème en se basant sur l’approche orientée objet, on a pu conclure avec une conception générale validée par toute l’équipe tout en respectant les objectifs du projet. Une conception dans laquelle on a schématisé notre solution et défini les différentes classes nécessaires pour la réalisation du projet.

 Chapitre 4 : Gestion du projet

* + 1. Introduction :

Le succès d’un projet dépend de la direction de l’équipe et de la motivation de ses membres d’orienter leurs efforts vers l’objectif du projet et d’adapter une démarche qui vise à modéliser, garantir et optimiser son déroulement. Pour cela, les membres de l’équipe doivent suivre un plan de travail à court terme, moyen terme et à long terme.

Pour assurer le bon déroulement et atteindre l’objectif du projet notre équipe a veillé à faire un stricte plan de travail, qui nous a été très utile pour assurer le suivi de l’état d’avancement de chaque membre et les tâches réalisées , ainsi qu’une bonne coordination entre les membres de l’équipe.

* + 1. Organisation du travail :

Pour la bonne gestion du projet durant les dix semaines, nous avons veillé à ce que notre travail soit régulier tout au long de la période du projet .Pour cela deux méthodes de travail ont été adaptées, une pour la période en ligne et l’autre en présentiel. En ligne presque chaque deux jours les membres de l’équipe font une réunion (en ligne) après que chaque membre déposera son travail dans un Drive partagé entre les membres de l’équipe, pour discuter sur les points réalisés et l’état d’avancement de chaque membre ainsi que les tâches à réaliser pour la prochaine réunion. En présentiel chaque mardi les membres de l’équipe se réunissent en présences des encadreurs afin de présenter l’état d’avancement du projet et les tâches réalisées par chaque membre et pour poser les différentes questions préparées aux encadreurs.

Durant la réunion (En ligne /présentiel) avec les encadreurs, plusieurs points sont abordés :

* Discussion sur l’état d’avancement et les points réalisés par chaque membre de l’équipe.
* Répartition des nouvelles tâches entre les membres de l’équipe.
* Discussion avec les encadreurs de toute éventuelle incompréhension ou problèmes rencontrés.
* Mise à jour des différentes options et profils selon les revendications des encadreurs.

Des réunions urgentes ont été organisées en cas de problèmes qui nécessite la présence de tous les membres de l’équipe afin de discuter les propositions de chaque membre et trouver une solution.

Affin d’assurer la bonne communication entre les membres de l’équipe, des groupes (un groupe pour l’équipe Front, un autre pour l’équipe back et un groupe qui englobe tous les membres) ont été créés en utilisant l’application Messenger, on a utilisé aussi autres outils de contact tel que : La messagerie de L’ESI, Google Meet, Zoom et d’autres outils de partage et tels que Drive, Git Hub pour le partage des différents travaux.

* + 1. Planning de travail :

Notre projet s’est déroulé comme suit :

Notre projet s’est déroulé en plusieurs phases :

*a)Phase d’analyse du cahier de charge et de la documentation :*

La durée de cette phase est d’une semaine, on l’avait consacré pour la bonne lecture, l’analyse et la compréhension du cahier de charge .Après avoir fixé nos idées nous avons fait un schéma global (Workflow) représentatif de notre application et dont on a suivi tout au long de notre période de travail .En parallèle, les membres de l’équipe ont fixé les outils qui vont être utilisé en Front end et back end par la suite.

*b) Phase de la conception :*

Après avoir fait le schéma global de l’application et la bonne maîtrise des outils nécessaires par les membres de l’équipe, nous avons entamé la conception. Dans cette partie, nous avons finalisé notre maquette détaillées du projet et comme notre solution est basée sur une base de données, nous avons commencé à faire la BDD.

*c) Phase de la réalisation :*

Après avoir finit la conception, on a entamé la réalisation. Cette phase a été consacrée pour la programmation : une partie qui concerne le back end, dans cette partie on a finalisé la base de données et la programmation des différentes méthodes et fonctionnalités nécessaires pour la gestion de notre application, la deuxième partie concerne tous ce qui est en relation avec le design de l’application ainsi que la programmation de ses différentes pages ( profils et comptes ) .Après avoir terminé ces deux parties on a commencé l’intégration et la liaison entre les pages de l’application et ses fonctionnalités .

*d)Préparation des présentations :*

Comme y’avait des séances de présentation chaque semaine ainsi que des séances d’évaluation, on a consacré une partie de notre temps pour les préparer. Dans chaque présentation on a présenté l’état d’avancement du projet et de chaque membre de l’équipe ainsi que les différentes tâches réalisées, y compris les tâches communes et ce qu’on doit préparer pour la prochaine présentation.

*e)Rédaction du rapport :*

La rédaction du rapport a été faite progressivement, lorsqu’un membre termine une partie du projet il rédigera la partie qui lui est dédié. Après la rédaction de chaque partie les autres membres de l’équipe s’occupent de la corriger et l’améliorer. Au fur et à mesure qu’on avance dans le projet et qu’on fasse quelques changements le rapport va être mis à jour, En suivant cette méthode de travail on a pus terminer notre rapport à temps.

*f) Phase de finalisation :*

On a consacré les dernières semaines de travail pour finaliser l’intégration des différentes pages avec ses fonctionnalités et pour faire les retouches finales de l’application y compris le test des différentes fonctionnalités et l’esthétique de l’application et sa sécurité.

1. Conclusion

Arrivé à la fin de ce rapport, on conclut que le projet intitulé « SUIVI DES MARCHES DE L’ESI » s’est déroulé comme prévu dans le respect des délais fixés et des objectifs attendus.

L’application réalisée est d’une part dédiée aux étudiants en informatique en général et en développement Web spécialement, et de l’autre aux clients voulant présenter leurs offres à l’établissement. Comme convenu, cette application permet le suivi des dossiers dans différents services, c'est-à-dire depuis son introduction au marché jusqu’à sa clôture chez le comptable.

Malgré les contraintes de temps et les difficultés techniques rencontrées, ce travail nous a été très instructif et nous a permis d’approfondir les connaissances acquises durant le cycle préparatoire notamment dans l’apprentissage de nouvelles technologies tel que GitHub. Il nous a également permis d’apprendre à travailler en équipe, sous pression, et bien planifier les tâches à faire en respectant les délais.

Pour conclure, à travers ce projet nous avons vécu la vie d’un ingénieur en informatique et ce fut une première expérience pour tous les membres de l’équipe, qui s’est déroulée dans les mêmes conditions de l’entreprise. Nous souhaitons fortement qu’il soit à la hauteur des exigences des encadreurs.

 On pourra éventuellement enrichir les fonctionnalités de « **SuiviMarchéEsi**» au fur et à mesure selon les besoins des clients. En effet, d’autres perspectives sont envisageables tel que :

o   L’ajout d’autres critères et filtres pour l’affichage des statistiques.

o   Des améliorations sur le design et la mise en forme.

o D’autres améliorations selon les remarques et les recommandations des clients.

·

# Références

|  |  |
| --- | --- |
| [1] | «Lucidchart,» [En ligne]. Available: https://fr.wikipedia.org/wiki/Lucidchart. |
| [2] | «Prezi» [En ligne]. Available: https://fr.wikipedia.org/wiki/Prezi. |
| [3] | «Figma» [En ligne]. Available: https://fr.wikipedia.org/wiki/Figma\_(logiciel). |
| [4] | «GitHub» [En ligne]. Available: https://fr.wikipedia.org/wiki/GitHub\_(entreprise). |