REPUBLIQUE TOGOLAISE

Travail-liberté-patrie

MINISTERE DE LA PLANIFICATION DU DEVELOPPEMENT ET DE LA COOPERATION

Institut Africain d'Informatique
Représentation du TOGO
(IAI-TOGO)

Tél : 22 20 47 00
E-mail : iaitogo@iai-togo.tg
Site Web: www.iai-togo.tg
07 **BP** 12456 Lomé 07, TOGO

Nouvelles Générations des Solutions Technologiques Adaptées à vos Réalités

Contact : +228 91 16 50 36 / 90 03 16 51
Mail : contacts@ng-stars.com

RAPPORT DE STAGE PRATIQUE
Type de stage : Programmation

THEME :

REINGENIERIE D'UNE SOLUTION LOGICIELLE DE PLANIFICATION, SUIVI ET CONTROLE DES ACTIVITES EN ENTREPRISE

Période : Du 06 juin 2022 au 05 août 2022

Rédigé et présenté par :
BLAKIME Christianna
Étudiante en deuxième année tronc commun
2021-2022

Superviseur :
M. BATAZI Kpaka
Ingénieur en informatique à la CEET

Maître de stage :
M. ATOKOU Abdel-Khafid
Développeur d'application

REMERCIEMENTS

Avant tout développement sur cette période d'apprentissage, je voudrais commencer par des remerciements, à ceux qui ont, d'une quelconque manière, contribué à l'aboutissement de ce stage. Ainsi, je remercie Monsieur Rachid DERMAN, directeur de NG-STARs, qui m'a accueillie dans sa structure.

Je remercie également mon maître de stage, Monsieur Abdel-Khafid ATOKOU, qui avec le concours du staff de NG-STARs, m'a guidée et prodiguée des conseils au cours de ces deux mois.

Je remercie enfin mon superviseur, Monsieur Kpaka BATAZI, qui a usé de son temps pour assurer l'acheminement de mon projet.

SOMMAIRE

LISTE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION

L'informatique est un domaine révolutionnaire et qui continue de se perfectionner au fil du temps. Il s'étend à tout domaine et devient indispensable pour l'aboutissement des activités en entreprise. Il est alors nécessaire d'en connaître les bases. Ainsi, le besoin de former des cadres capables de maîtriser et de s'adapter à tout changement du métier de l'informatique, a fait naître des centres de formation de référence tels que l'IAI-TOGO (Institut Africain d'Informatique Représentation Togo). Cet institut offre en effet des parcours en génie logiciel, administration systèmes et réseau et éventuellement en multimédia pendant trois ans. Pour rendre complet l'apprentissage de ses étudiants, l'IAI-TOGO inclut dans son programme éducatif, un stage en entreprise, en programmation ou maintenance, d'une durée de deux mois, pour les étudiants de deuxième année. Durant cette période, il est attendu de tout étudiant de mettre en pratique les connaissances acquises lors de son parcours à l'IAI. Le présent document fera l'objet de rapport de stage au sein de la société NG-STARs (Nouvelle Génération des Solutions Technologiques Adaptées à vos Réalités) qui se situe à Hédrzanawé, non loin du consulat du Burkina faso. Il prendra en compte la réalisation du cahier des charges, la phase de pré programmation et la mise en œuvre de la solution.

1 PARTIE 1 : CAHIER DES CHARGES

1.1 Présentation du sujet

Afin d'améliorer les performances d'une entreprise, il est primordial de prendre contrôle de la planification et de l'exécution des opérations. En effet pour une entreprise, l'atteinte des objectifs opérationnels et financiers permet d'augmenter sa productivité et de planifier les actions à mener conformément à ses indicateurs de mesure de la performance.

C'est dans ce cadre que la société NG-STARs nous propose de réimaginer et d'optimiser l'architecture d'un logiciel ayant pour principaux buts : la planification, le suivi et le contrôle en entreprise.

1.2 Problématique du sujet

Il est question d'étudier une application déjà existante et d'en tirer les informations importantes telles que : la raison d'être de cette application, les différentes fonctionnalités, les données à enregistrer, les niveaux d'accès propres aux utilisateurs, etc. Il faudra, ensuite, reconstruire ladite application tout en améliorant entre autres son ergonomie, les fonctionnalités défaillantes pour rendre l'utilisation agréable et intuitive pour tout utilisateur.

1.3 Intérêt du sujet

- **Objectif général**

Le but principal de ce projet est de doter une entreprise d'un outil regroupant de façon concise les informations essentielles à son fonctionnement.

1.3.1 Objectifs spécifiques

Plus précisément, il faudra :

- analyser la structure du code existant ;
- identifier les relations entre entités décrivant la base de données ;
- identifier les insuffisances du diagramme de classe et apporter les correctifs ;
- améliorer l'expérience utilisateur et les interfaces de la solution.

1.3.2 Résultats attendus

Les résultats attendus sont les suivants :

- L'analyse du code est réalisée ;
- les relations entre entités décrivant la base de données sont identifiées ;
- les insuffisances du diagramme de classe sont identifiées et les correctifs appliqués ;
- l'expérience utilisateur et les interfaces sont améliorées ;

2 PARTIE 2 : PRE-PROGRAMMATION

2.1 Etude de l'existant

Dans le présent cas, il existe déjà une application pas tout à fait aboutie, mais utilisable. Voici son fonctionnement.

L'utilisateur accède en premier à une interface de connexion où il entre ses identifiants. Une fois la connexion établie, il a accès à un tableau de bord sur lequel il naviguera pour ses différents travaux. Il peut saisir des actions, des activités et des tâches concernant un programme et accéder à leur liste totale ou par département. Les données saisies concernent deux volets : la prévision et l'exécution. A l'aide des données recueillies, un tableau des écarts est établi. Celui-ci permet à l'utilisateur de comparer les valeurs réelles requises pour l'exécution d'une activité et celles envisagées lors de sa prévision. L'utilisateur accède aussi à l'évolution de toutes les exécutions ou de celles de chaque département en cours. Chaque action, activité ou tâche dispose d'un financement pour assurer des dépenses ; ces dépenses sont classées dans un tableau. L'utilisateur accède alors à la répartition de ce financement. En effet, le financement provient de deux sources, à savoir : l'Etat et les partenaires. Il est également possible d'accéder aux statistiques de réalisations des actions, activités et tâches par statut : non réalisée, en cours ou réalisée. Enfin, l'utilisateur peut mettre à jour son profil.

Il existe également un volet paramètre. Seul l'administrateur est habileté à y faire des actions. Il peut créer et mettre à jour les programmes, les responsables, les paragraphes, les natures de dépenses, les lignes, les réalisations, les taux de réalisations. Les responsables désignent les départements.

Suite à une analyse du code existant, le diagramme de classe suivant est extrait.

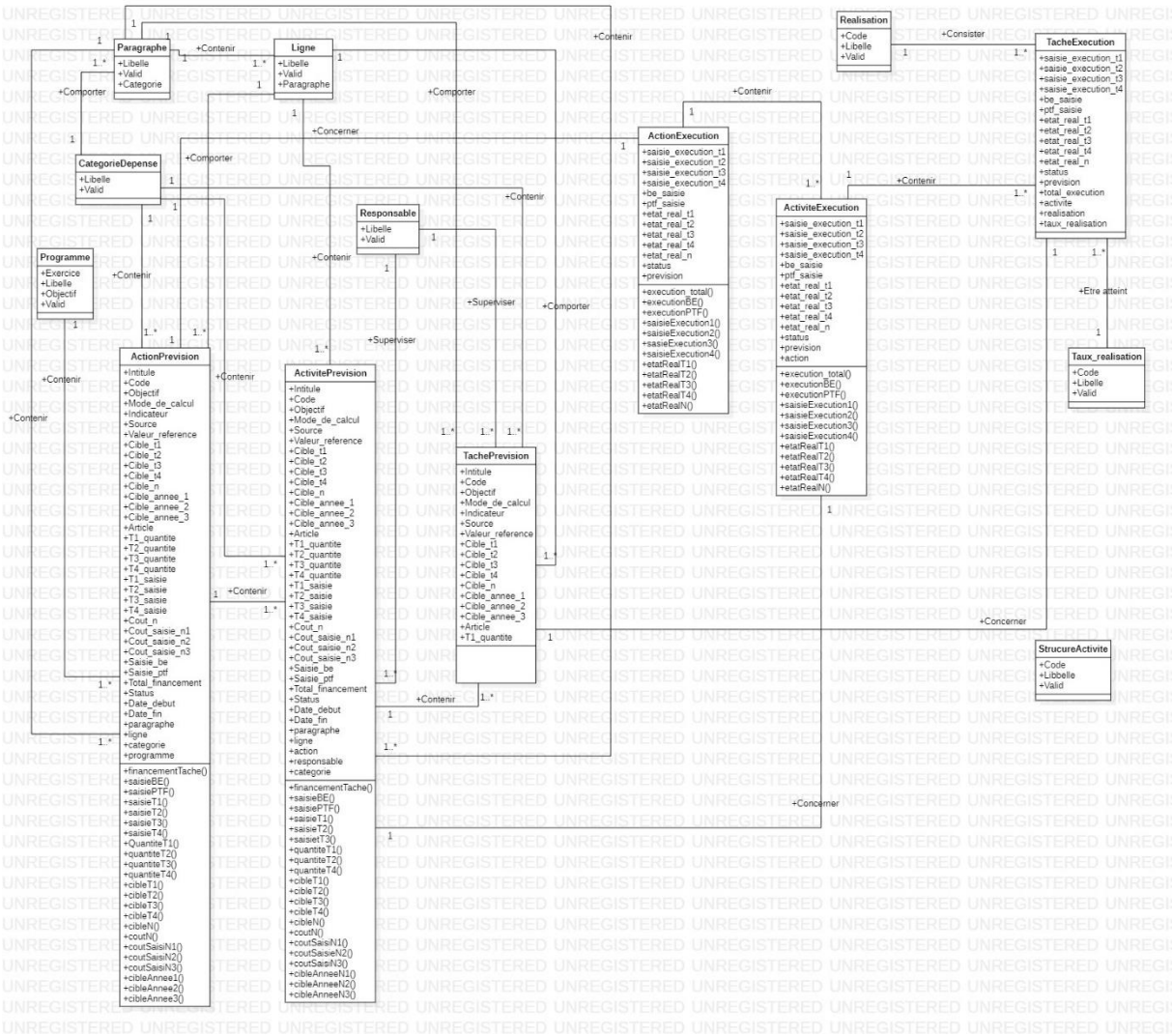
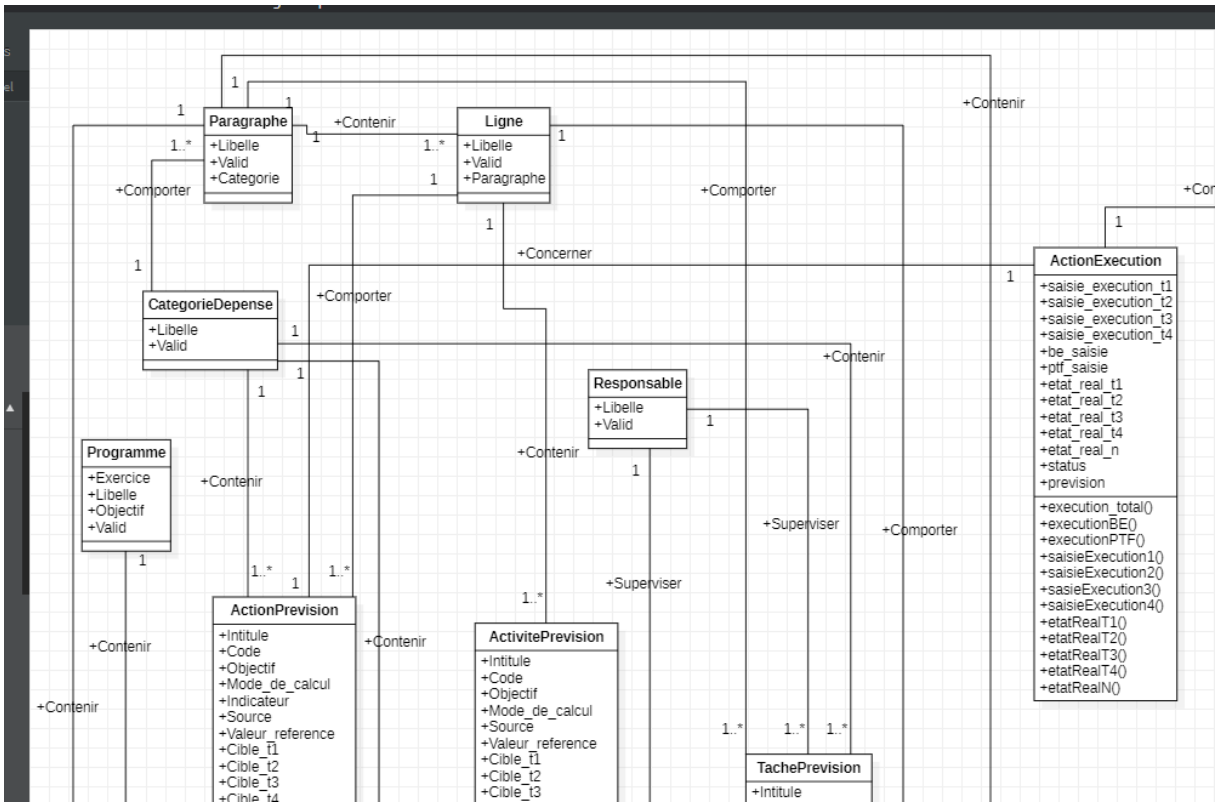
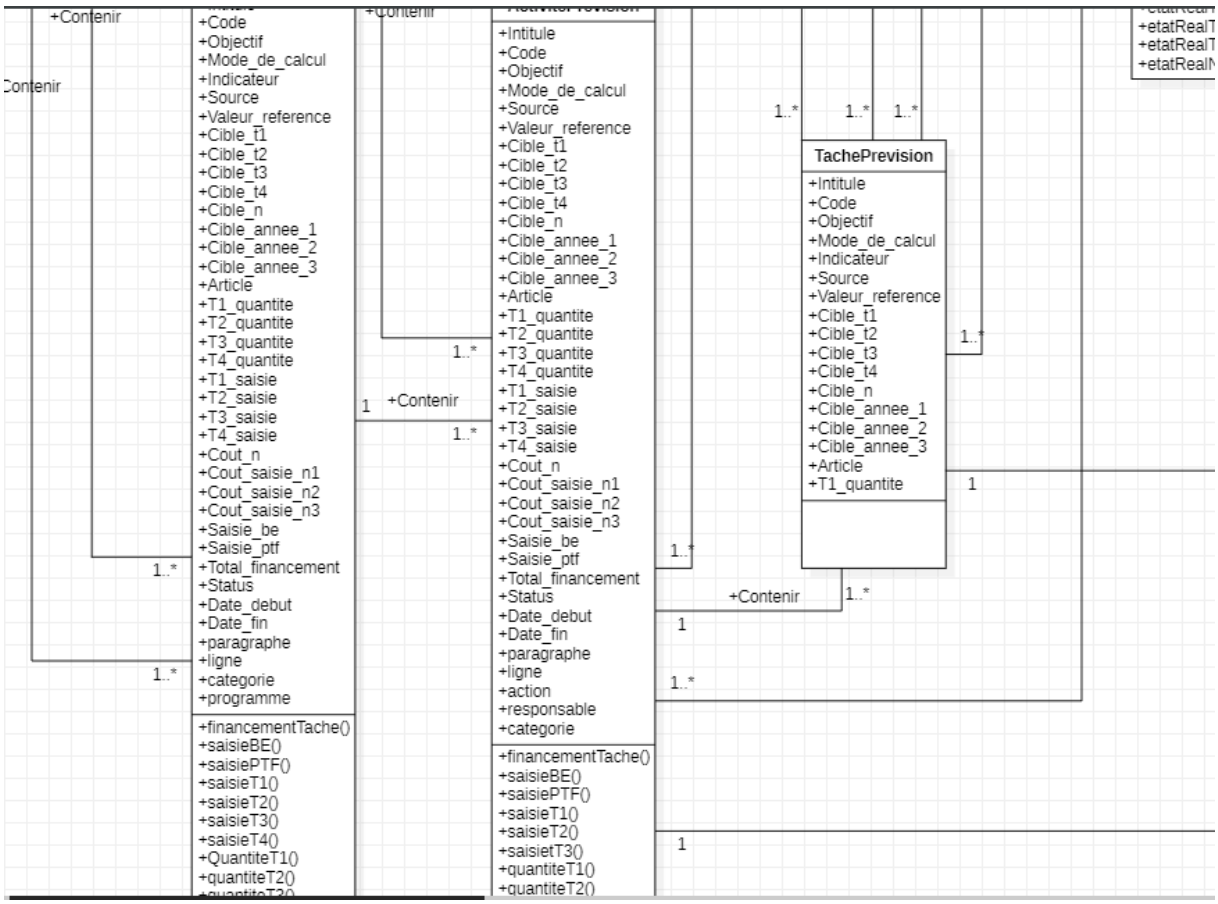
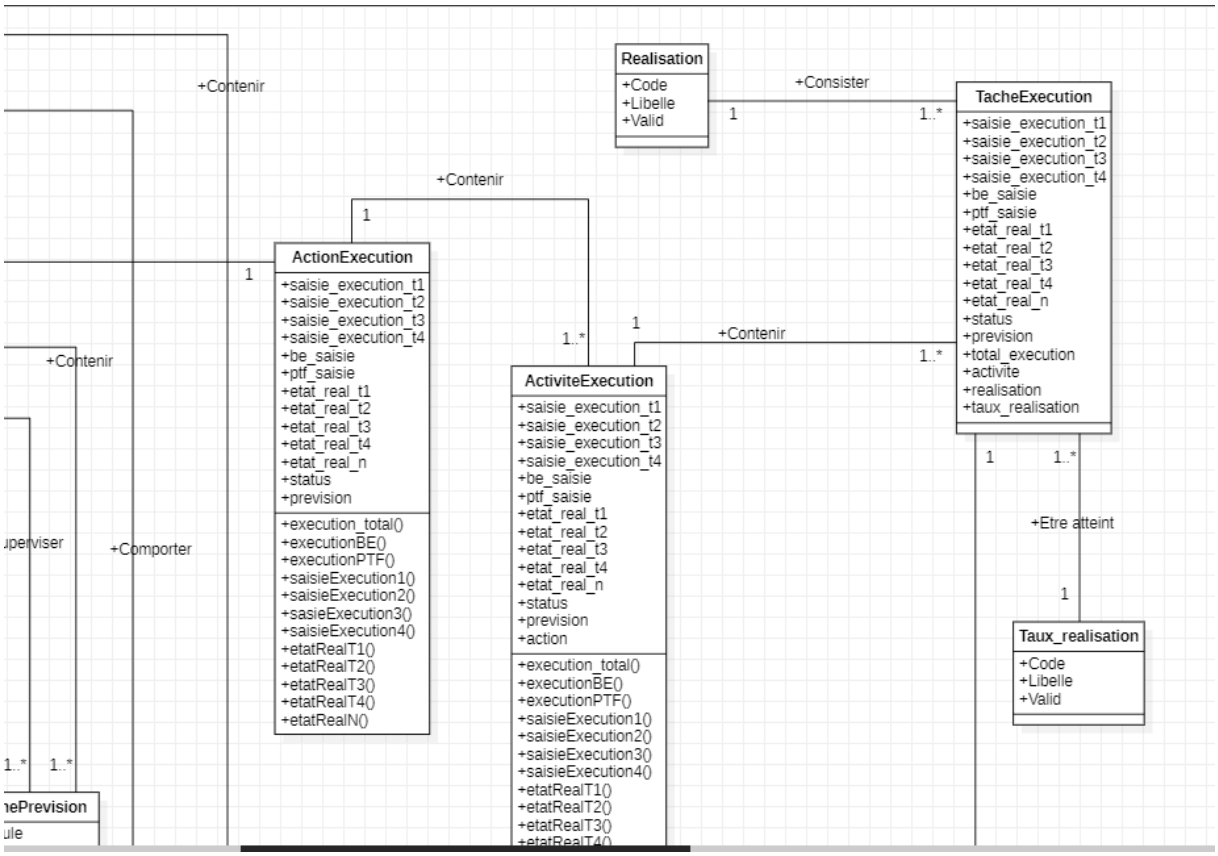


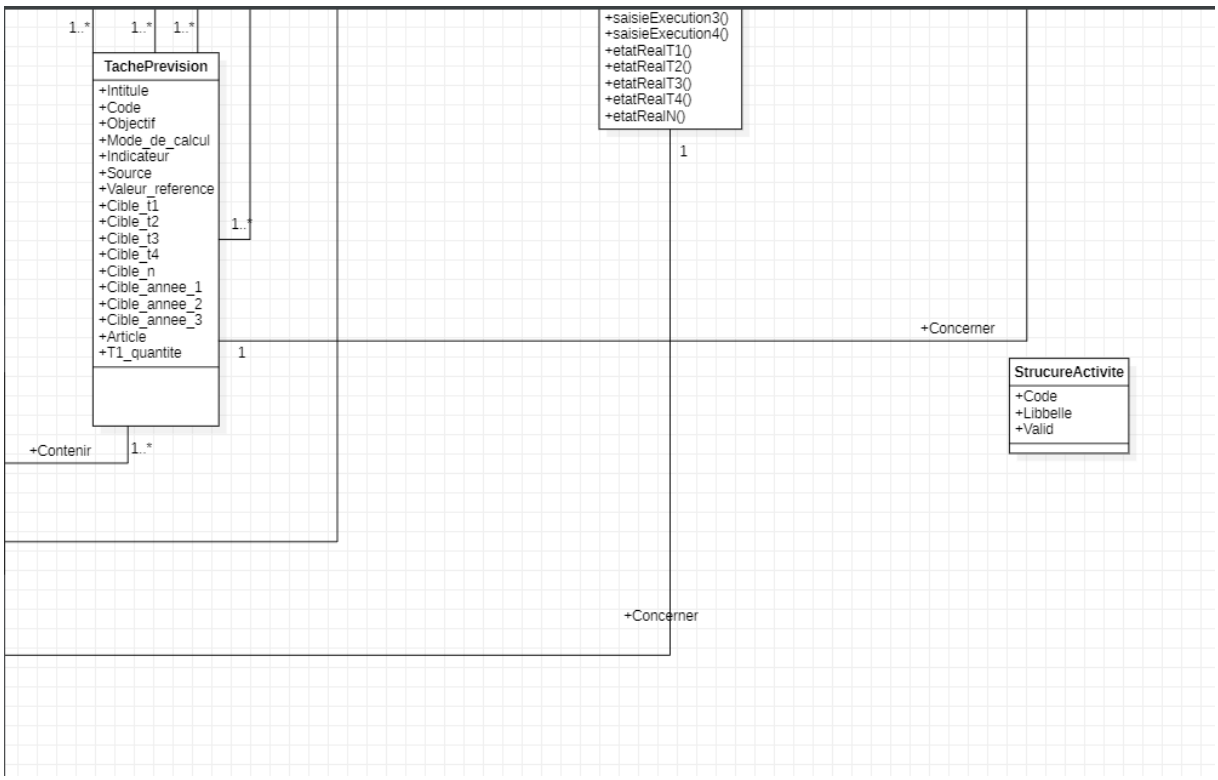
Figure 1:Diagramme de classe extrait.

Les images suivantes ont pour but de montrer en détails la figure précédente.


Paragraphe
+Libelle
+Valid
+Categorie
Ligne
+Libelle
+Valid
+Paragraphe
CategorieDepense
+Libelle
+Valid
Programme
+Exercice
+Libelle
+Objectif
+Valid
Responsable
+Libelle
+Valid
ActionExecution
+saisie_execution_t1
+saisie_execution_t2
+saisie_execution_t3
+saisie_execution_t4
+be_saisie
+ptf_saisie
+etat_real_t1
+etat_real_t2
+etat_real_t3
+etat_real_t4
+etat_real_n
+status
+prevision
+execution_total()
+executionBE()
+executionPTF()
+saisieExecution1()
+saisieExecution2()
+sasieExecution3()
+saisieExecution4()
+etatRealT1()
+etatRealT2()
+etatRealT3()
+etatRealT4()
+etatRealN()
ActionPrevision
+Intitule
+Code
+Objectif
+Mode_de_calcul
+Indicateur
+Source
+Valeur_reference
+Cible_t1
+Cible_t2
+Cible_t3
+Cible_t4
ActivitePrevision
+Intitule
+Code
+Objectif
+Mode_de_calcul
+Source
+Valeur_reference
+Cible_t1
+Cible_t2
+Cible_t3
TachePrevision
+Intitule
+Contenir
+Comporter
+Concerner
+Superviser


+Code
+Objectif
+Mode_de_calcul
+Indicateur
+Source
+Valeur_reference
+Cible_t1
+Cible_t2
+Cible_t3
+Cible_t4
+Cible_n
+Cible_annee_1
+Cible_annee_2
+Cible_annee_3
+Article
+T1_quantite
+T2_quantite
+T3_quantite
+T4_quantite
+T1_saisie
+T2_saisie
+T3_saisie
+T4_saisie
+Cout_n
+Cout_saisie_n1
+Cout_saisie_n2
+Cout_saisie_n3
+Saisie_be
+Saisie_ptf
+Total_financement
+Status
+Date_debut
+Date_fin
+paragraphe
+ligne
+categorie
+programme
+financementTache()
+saisieBE()
+saisiePTF()
+saisieT1()
+saisieT2()
+saisieT3()
+saisieT4()
+QuantiteT1()
+quantiteT2()
+Intitule
+Code
+Objectif
+Mode_de_calcul
+Source
+Valeur_reference
+Cible_t1
+Cible_t2
+Cible_t3
+Cible_t4
+Cible_n
+Cible_annee_1
+Cible_annee_2
+Cible_annee_3
+Article
+T1_quantite
+T2_quantite
+T3_quantite
+T4_quantite
+T1_saisie
+T2_saisie
+T3_saisie
+T4_saisie
+Cout_n
+Cout_saisie_n1
+Cout_saisie_n2
+Cout_saisie_n3
+Saisie_be
+Saisie_ptf
+Total_financement
+Status
+Date_debut
+Date_fin
+paragraphe
+ligne
+action
+responsable
+categorie
+financementTache()
+saisieBE()
+saisiePTF()
+saisieT1()
+saisieT2()
+saisietT3()
+quantiteT1()
+quantiteT2()
TachePrevision
+Intitule
+Code
+Objectif
+Mode_de_calcul
+Indicateur
+Source
+Valeur_reference
+Cible_t1
+Cible_t2
+Cible_t3
+Cible_t4
+Cible_n
+Cible_annee_1
+Cible_annee_2
+Cible_annee_3
+Article
+T1_quantite
+Contenir


Realisation
+Code
+Libelle
+Valid
+Consister
1
1..*
TacheExecution
+saisie_execution_t1
+saisie_execution_t2
+saisie_execution_t3
+saisie_execution_t4
+be_saisie
+ptf_saisie
+etat_real_t1
+etat_real_t2
+etat_real_t3
+etat_real_t4
+etat_real_n
+status
+prevision
+total_execution
+activite
+realisation
+taux_realisation
+Contenir
+Contenir
1
1..*
ActionExecution
+saisie_execution_t1
+saisie_execution_t2
+saisie_execution_t3
+saisie_execution_t4
+be_saisie
+ptf_saisie
+etat_real_t1
+etat_real_t2
+etat_real_t3
+etat_real_t4
+etat_real_n
+status
+prevision
+execution_total()
+executionBE()
+executionPTF()
+saisieExecution1()
+saisieExecution2()
+sasieExecution3()
+saisieExecution4()
+etatRealT1()
+etatRealT2()
+etatRealT3()
+etatRealT4()
+etatRealN()
+Contenir
1
1..*
ActiviteExecution
+saisie_execution_t1
+saisie_execution_t2
+saisie_execution_t3
+saisie_execution_t4
+be_saisie
+ptf_saisie
+etat_real_t1
+etat_real_t2
+etat_real_t3
+etat_real_t4
+etat_real_n
+status
+prevision
+action
+execution_total()
+executionBE()
+executionPTF()
+saisieExecution1()
+saisieExecution2()
+saisieExecution3()
+saisieExecution4()
+etatRealT1()
+etatRealT2()
+etatRealT3()
+etatRealT4()
+Etre atteint
1
1..*
Taux_realisation
+Code
+Libelle
+Valid
+Comporter
1..*
1..*


1..*
1..*
1..*
TachePrevision
+Intitule
+Code
+Objectif
+Mode_de_calcul
+Indicateur
+Source
+Valeur_reference
+Cible_t1
+Cible_t2
+Cible_t3
+Cible_t4
+Cible_n
+Cible_annee_1
+Cible_annee_2
+Cible_annee_3
+Article
+T1_quantite
1..*
1
+saisieExecution3()
+saisieExecution4()
+etatRealT1()
+etatRealT2()
+etatRealT3()
+etatRealT4()
+etatRealN()
1
+Concerner
StrucureActivite
+Code
+Libbelle
+Valid
+Contenir
1..*
+Concerner

2.2 Critique de l'existant

Le diagramme de classe extrait comporte certaines insuffisances.

- Il existe des associations superflues : En prenant l'exemple des classes ActionPrevision, Ligne et Paragraphe, on remarque qu'il y a un lien entre ActionPrevision et Ligne et entre ActionPrevision et Paragraphe ; or il y a déjà un lien entre Paragraphe et Ligne.

```
paragraphe = models.ForeignKey(Paragraphe, models.DO_NOTHING, default=None,
blank=True, null=True, related_name='actions')
  ligne = models.ForeignKey(Ligne, models.DO_NOTHING, default=None,
blank=True, null=True)
    programme = models.ForeignKey(Programme, models.DO_NOTHING)
 categorie = models.ForeignKey(CategorieDepense, models.DO_NOTHING,
default=None, blank=True, null=True)
```

- Des champs calculés sont pris comme attributs ; tels que : cibleN , coutN.
- Sur le plan interface, certains formulaires comportent des champs grisés, l'affichage des tableaux n'est pas fameux ; il y a beaucoup d'informations affichées

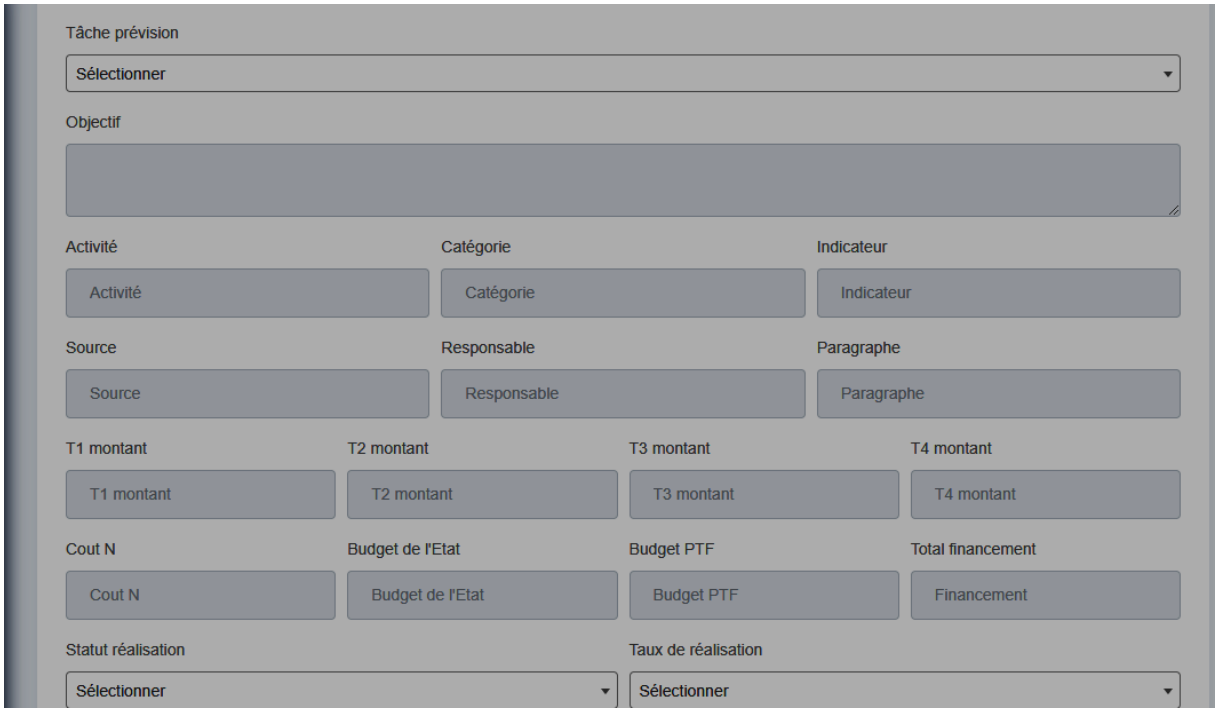


Figure 2: Formulaire de saisie existant.

- La partie administrateur est accessible à tous. Il suffit de cliquer sur le lien paramètre et tous est visible et personnalisable.

Figure 3:Espace administrateur.

2.3 Planning prévisionnel de réalisation

Tableau 1:Planning prévisionnel.

N° d'ordre	Description	Période
1	Imprégnation du cadre de stage Recherches sur l'utilisation de Django	15 – 23 juin
2	Etablissement du cahier des charges	22 – 24 juin
3	Recherche sur Django rest framework et Angular	25 – 30 juin
4	Recherche sur la création d'api avec Django	1 – 6 juillet
5	Analyse du code source	7 – 12 juillet
6	Déduction du diagramme de classes	8 - 8 juillet
7	Application des correctifs au diagramme de classes et établissement de la base de données	9 – 10 juillet
8	Validation des diagrammes	13 – 20 juillet
10	Implémentation de l'application	22 juillet – 3 Août
11	Finalisation du rapport de stage	4 – 6 Août

2.4 Etude détaillée de la solution

Pour optimiser la solution existante, des modifications ont été apportées sur le diagramme de classe et la présentation des interfaces utilisateur. Les fonctionnalités prises en compte sont la configuration d'un programme et le paramétrage des dépenses.

2.4.1 Les acteurs

Ils interagissent avec l'application en fournissant et/ou recevant des informations.

Il en existe trois acteurs, dans le cas étudié :

- L'utilisateur qui est un employé de la société et habileté à fournir les données nécessaires,
- L'administrateur qui se charge de la gestion des données statiques et non accessible à l'utilisateur lambda,
- Le système qui intervient si une vérification est nécessaire.

Leurs différentes actions sont réunies dans le diagramme de cas d'utilisation au volet suivant.

2.4.2 Diagramme de cas d'utilisation

Le diagramme des cas d'utilisation constitue la première étape de l'analyse UML en :

- Modélisant les besoins des utilisateurs.
- Identifiant les grandes fonctionnalités et les limites du système.
- Représentant les interactions entre le système et ses utilisateurs.

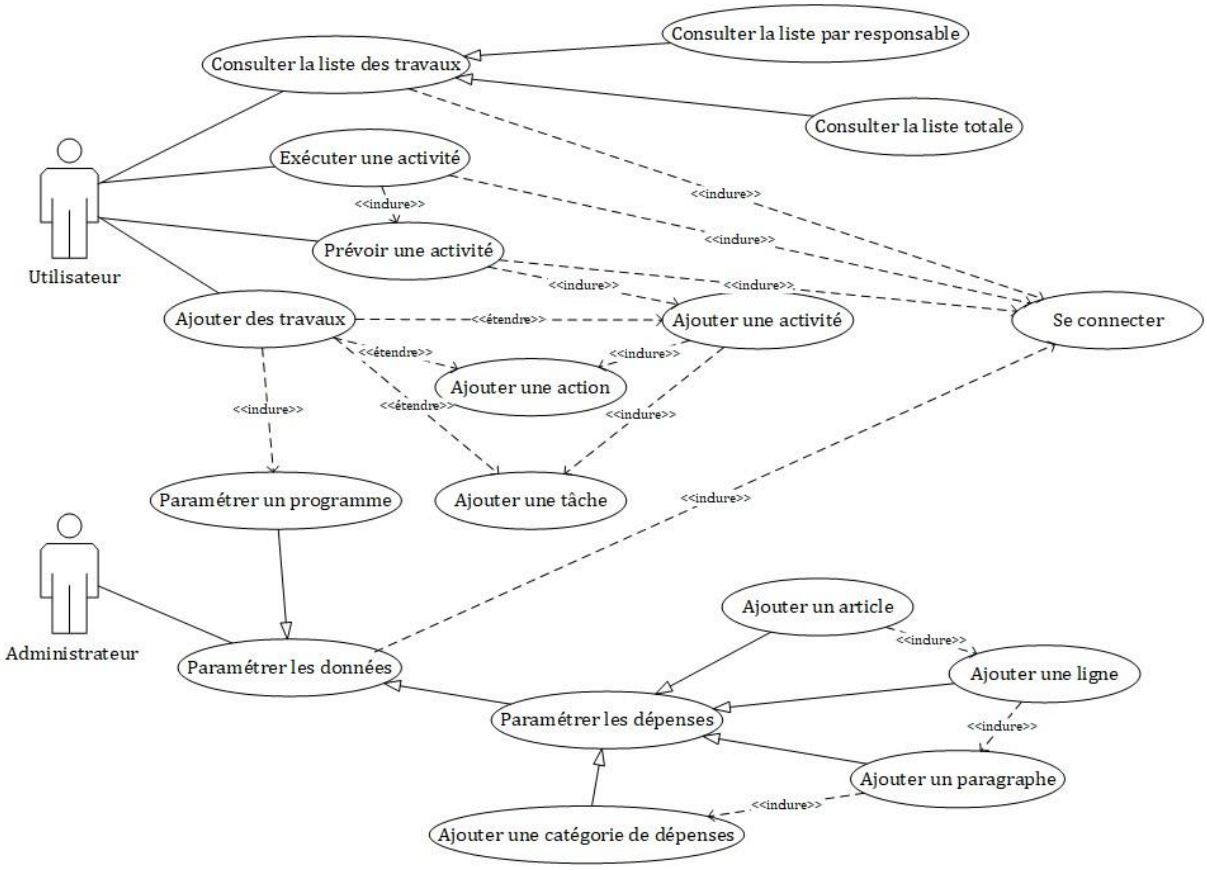


Figure 4:Diagramme de cas d'utilisation.

Les données, dont il est question au niveau de l'administrateur, concernent toutes celles que les utilisateurs n'ont pas à renseigner. Certes, l'administrateur a accès à toutes les données de l'application.

2.4.3 Description textuelle

- **Cas d'utilisation** : se connecter

Titre : Se connecter
Acteur principal : Utilisateur
Résumé : accéder à l'application à travers un compte existant
Version : 1.0
Responsable : BLAKIME Christianna

Préconditions :

1. Le site est disponible
2. Accéder à l'interface de connexion

Scénario nominal :

1. L'utilisateur entre ses coordonnées
2. L'utilisateur valide
3. Le système vérifie l'existence de ces cordonnées
4. L'utilisateur accède à la page d'accueil de l'application

Scénario alternatif
SA1 : coordonnées incorrectes
Il débute au numéro 3 du scénario nominal.
Dans Ce cas l'utilisateur peut soit rectifier ceux-ci et accède à l'application ou décide de s'arrêter.

SA2 : l'utilisateur n'a pas de compte.
Il débute avant le numéro 1 du scénario nominal.
Dans ce cas, il lui est possible de s'enregistrer via un lien sur la page de connexion. Lorsqu'il l'a fait, il peut retourner sur la page de connexion via un lien. A ce moment le scénario reprend au numéro 1 et se termine normalement.

Post conditions :

1. L'application est opérationnelle ;
2. L'utilisateur accède à sa session.

- **Cas d'utilisation** : ajouter une action

  Titre : Ajout d'une action
  Acteur principal : Utilisateur
  Résumé : Créer un nouveau volet lié à la mise en œuvre d'un programme
  Version : 1.0
  Responsable : BLAKIME Christianna

  Préconditions :
  1. Le site est disponible
  2. Se connecter
  3. Accéder à la page d'accueil de l'application
  4. Existence des programmes
  5. Existence des responsables

  Scénario nominal :
  1. L'utilisateur remplie le formulaire
  2. L'utilisateur valide
  3. Le système vérifie la validité des données entrées
  4. L'utilisateur reçoit une notification lui signifiant la réussite de l'ajout
  5. L'utilisateur est redirigé à la liste des actions

  Scénario alternatif
  SA1 : données incorrectes
  Il débute au numéro 3 du scénario nominal.
  Dans ce cas le système renvoie une notification signalant à l'utilisateur que son formulaire est erroné. L'utilisateur reprend alors la saisie au numéro 1 du scénario nominal.

  Post conditions :
  1. L'application reste opérationnelle ;
  2. L'action saisie s'enregistre dans la base de données et est ajoutée à la liste des actions.

2.4.4 Diagramme d’activité

Le diagramme d’activité fait partie des diagrammes comportementaux. Il est utilisé pour modéliser les aspects dynamiques d'un système. Il s'agit de représenter les opérations d'un processus et leurs conséquences sur les objets (logiciels ou matériels). La modélisation peut être utilisée pour décrire le déroulement d'un cas d'utilisation ou d'une méthode. Les diagrammes d'activité affichent le flux de travail d'un point de départ à un point d'arrivée en détaillant les nombreux chemins de décision existant dans la progression des événements contenus dans l'activité. Les éléments du diagramme d’activité sont :

- Les activités : Une activité définit un comportement décrit par un séquencement organisé d'unités dont les éléments simples sont les actions. Le flot d'exécution est modélisé par des nœuds reliés par des arcs (transitions). Le flot de contrôle reste dans l'activité jusqu'à ce que les traitements soient terminés.
- Les nœuds activités : Un nœud d'activité est un type d'élément abstrait permettant de représenter les étapes le long du flot d'une activité. Il existe trois familles de nœuds d'activités :
  - ✓ Les nœuds d'exécutions
  - ✓ Les nœuds objets
  - ✓ Les nœuds de contrôle.
- Les transitions : Le passage d'une activité vers une autre est matérialisé par une transition. Elles sont déclenchées dès que l'activité source est terminée et provoquent automatiquement et immédiatement le début de la prochaine activité à déclencher (l'activité cible).
- Les nœuds d’actions : Un nœud d'action est un nœud d'activité exécutable qui constitue l'unité fondamentale de fonctionnalité exécutable dans une activité. • Les nœuds de contrôle : Un nœud de contrôle est un nœud d'activité abstrait utilisé pour coordonner les flots entre les nœuds d'une activité. Il existe plusieurs types de nœuds de contrôle :
  - ✓ Nœud initial
  - ✓ Nœud de fin d'activité
  - ✓ Nœud de fin de flot
  - ✓ Nœud de décision
  - ✓ Nœud de fusion
  - ✓ Nœud de bifurcation
  - ✓ Nœud d'union

Les schémas suivant représentent quelques modélisations des diagrammes de séquences de notre projet.

- **Cas d’utilisation** : se connecter

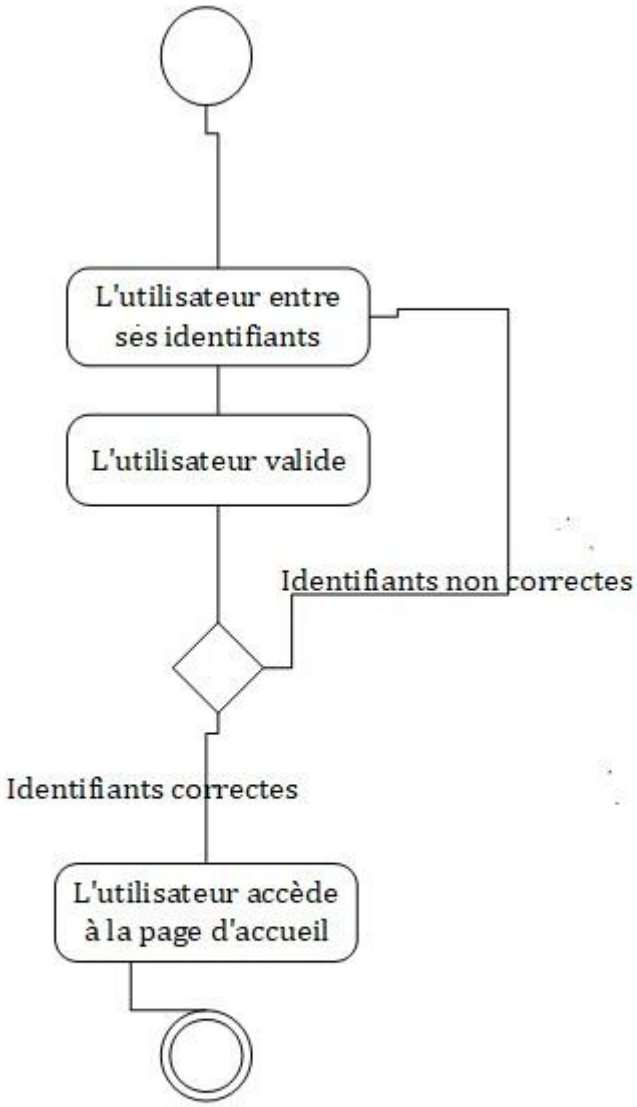


Figure 5:Diagramme d'activité (se connecter).

- **Cas d'utilisation** : ajouter une action

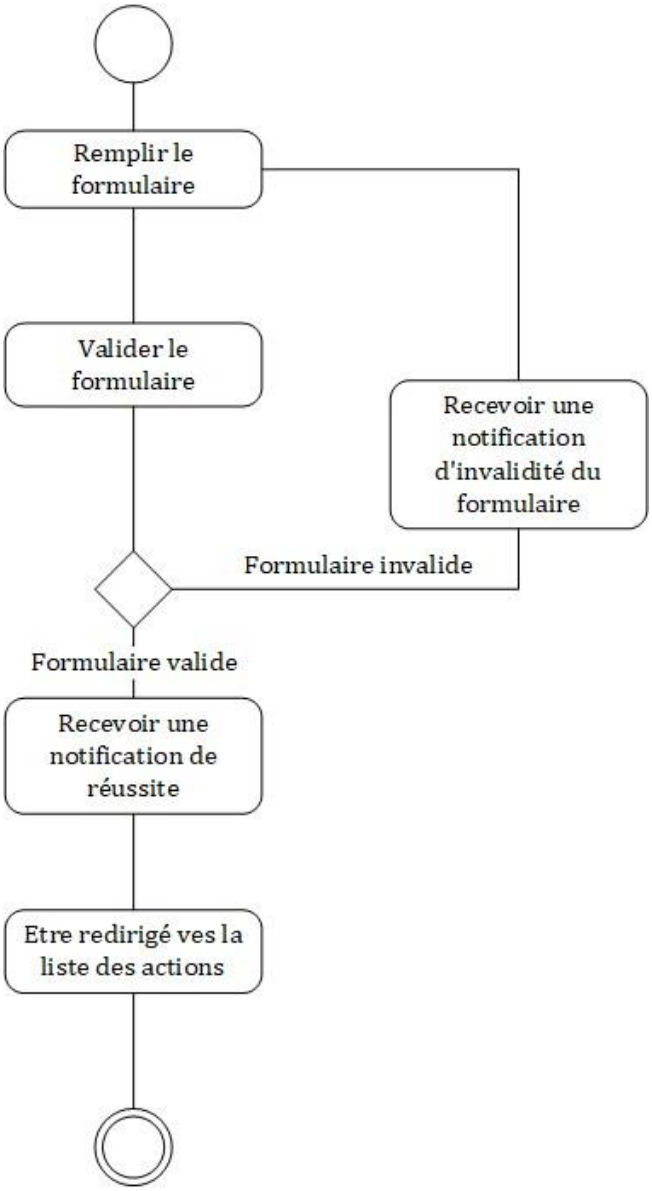


Figure 6:Diagramme d'activité (ajouter une action).

2.4.5 Diagrammes de séquences

Le diagramme de séquence fait parties des diagrammes comportementaux (dynamique) et plus précisément des diagrammes d'interactions. Il permet de représenter des échanges entre les différents objets et acteurs du système en fonction du temps.

Les éléments du diagramme de séquence sont :

- L'objet : il représente l'instance d'une classe
- Ligne de vie : La ligne de vie indique les périodes d'activité de l'objet (généralement, les moments ou l'objet exécute une de ces méthodes). Lorsque l'objet est détruit, la ligne de vie s'achève par une croix.

- Les messages : Un message définit une communication particulière entre des lignes de vie. Ainsi, un message est une communication d'un objet vers un autre objet. La réception d'un message est considérée par l'objet récepteur comme un événement qu'il faut traiter (ou pas). Plusieurs types de messages existent, les plus communs sont
  - ✓ L'invocation d'une opération : message synchrone (appel d'une méthode de l'objet cible).
  - ✓ L'envoi d'un signal : message asynchrone (typiquement utilisé pour la gestion événementielle).
  - ✓ La création ou la destruction d'une instance de classe au cours du cycle principal.
- Fragments d'interactions combinés : Un fragment d'interactions est une partie du diagramme de séquence (délimitée par un rectangle) associée à une étiquette (dans le coin supérieur gauche). L'étiquette contient un opérateur d'interaction qui permet de décrire des modalités d'exécution des messages à l'intérieur du cadre. Les schémas suivants représentent quelques modélisations des diagrammes de séquences de l'étude qui m'a été soumise.

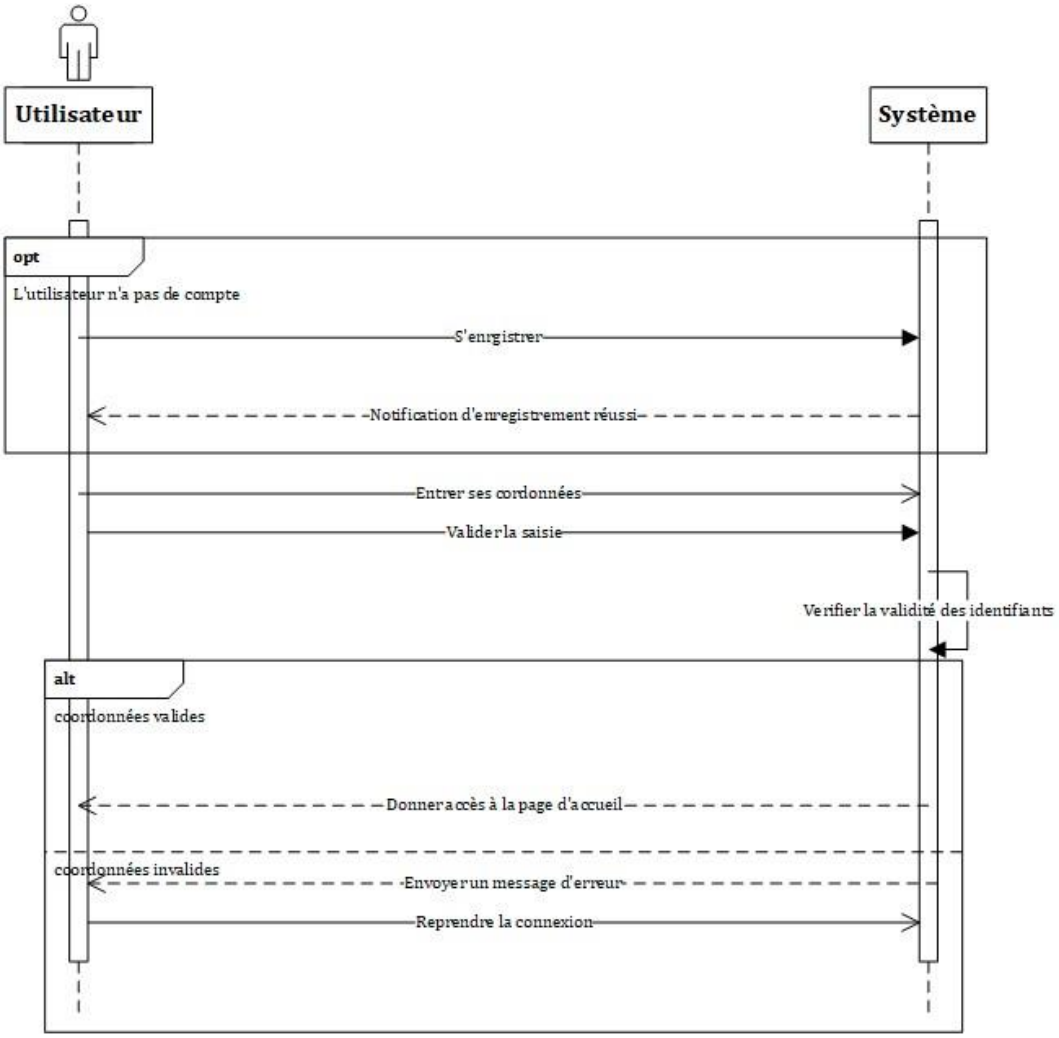


Figure 7:Diagramme de séquence (se connecter).

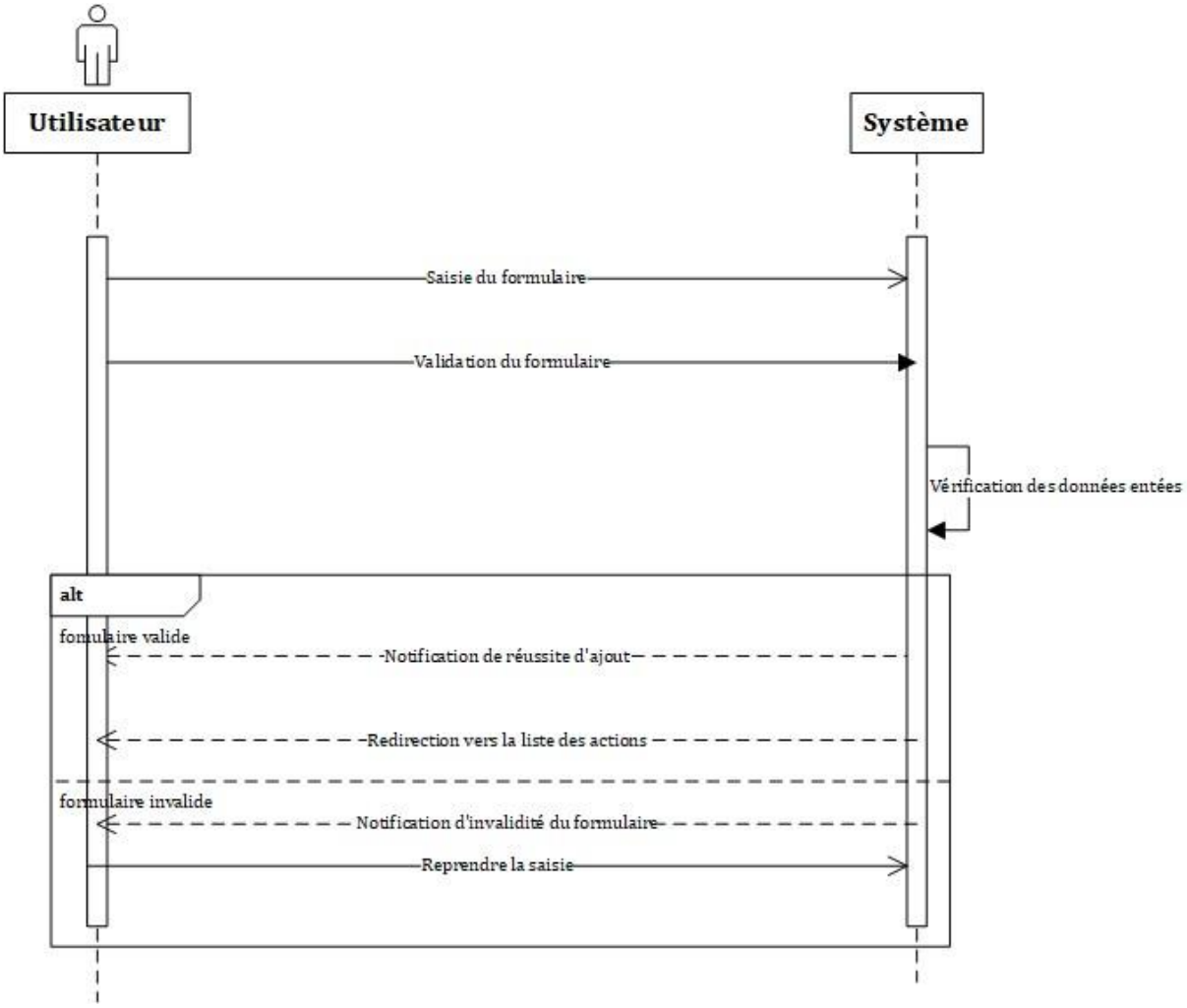


Figure 8:Diagramme de séquence (ajouter une action).

2.4.6 Diagramme de classe

Le diagramme des classes est un diagramme structurel (statique) qui permet de représenter :

- Les classes (attributs + méthodes)
- Les associations (relations) entre les classes.

Après modification du diagramme extrait, et en utilisant les principes de normalisation de bases de données, est obtenu le diagramme suivant.

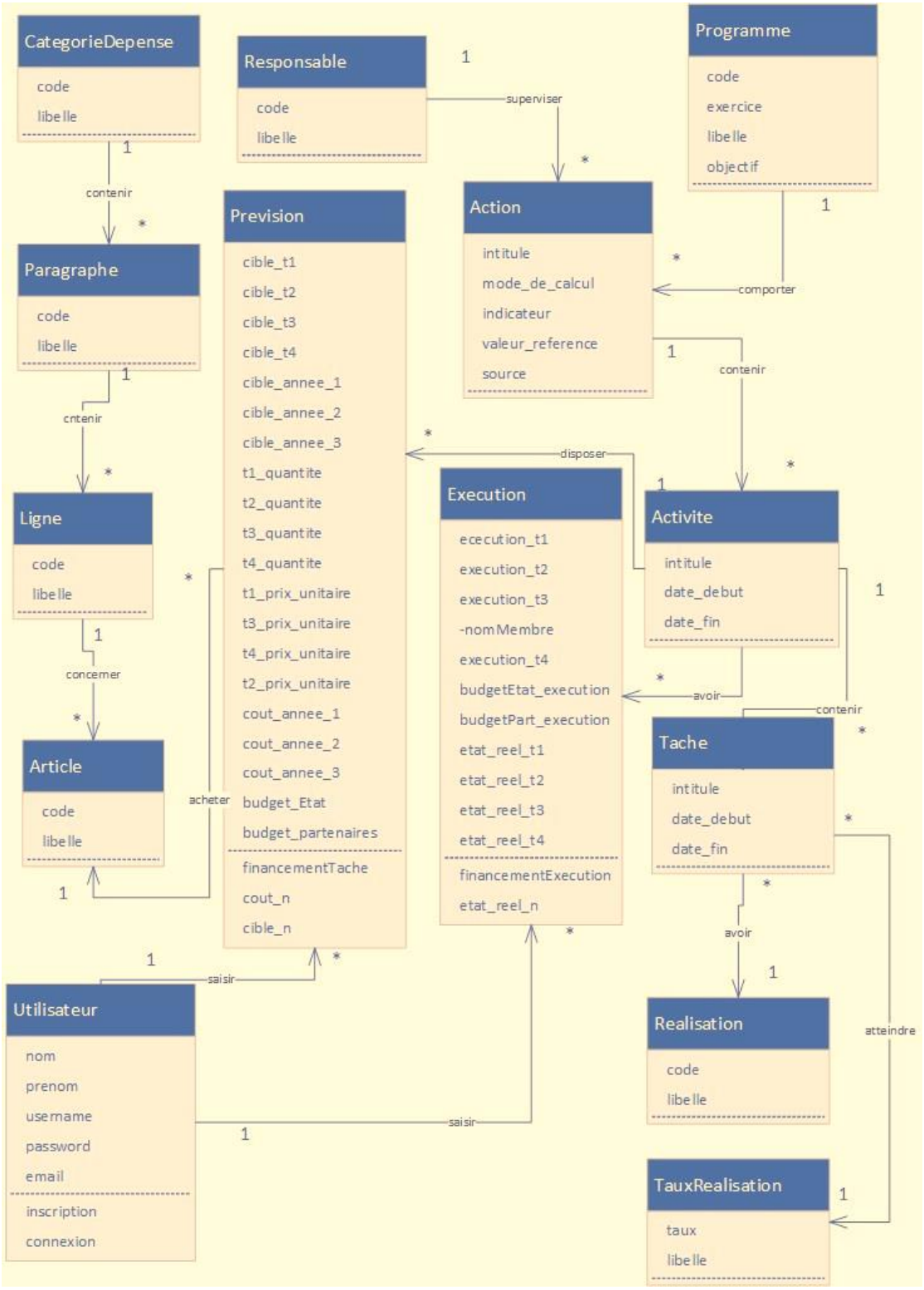


Figure 9:Diagramme de classe.

3 PARTIE 3 : REALISATION ET MISE EN ŒUVRE

3.1 Matériels et logiciels utilisés

3.1.1 Matériels

Le matériel utilisé durant cette période de stage est un ordinateur portable de caractéristiques suivantes :

Nom de l'appareil : DESKTOP-A7JOEK8

Processeur : Intel(R) Celeron(R) CPU N3060 @ 1.60GHz 1.60 GHz

Mémoire RAM installée : 4,00 Go (3,84 Go utilisable)

ID de périphérique : 2EE7644B-71CC-451C-B614-7368E97AD17E

ID de produit : 00331-10000-00001-AA831

Type du système : Système d'exploitation 64 bits, processeur x64

3.1.2 Logiciels

- **Editeur de texte**

Visual Studio Code, également communément appelé **VS Code**, est un éditeur de code source créé par Microsoft pour Windows, Linux et macOS. Les fonctionnalités incluent la prise en charge du débogage, de la coloration syntaxique, de la complétion intelligente du code, des extraits de code, de la refactorisation du code et de Git intégré. Les utilisateurs peuvent modifier le thème, les raccourcis clavier, les préférences et installer des extensions qui ajoutent des fonctionnalités supplémentaires.

Figure 10:Logo de Visual Studio Code.

- **Langage programmation**

Python est un langage de programmation généraliste interprété de haut niveau. Sa philosophie de conception met l'accent sur la lisibilité du code grâce à l'utilisation d'une indentation significative. Python est typé dynamiquement. Il prend en charge plusieurs paradigmes de programmation, y compris la programmation structurée (en particulier procédurale), orientée objet et fonctionnelle. Il est souvent décrit comme un langage "piles incluses" en raison de sa bibliothèque standard complète.

Figure 11:Logo de Python.

- **Frameworks utilisés**

Le framework **Django REST** est une boîte à outils puissante et flexible pour la création d'API Web.

Quelques raisons pour lesquelles vous pourriez vouloir utiliser le framework REST :

- L'API navigable sur le Web est un énorme avantage en termes de convivialité pour vos développeurs.
- Politiques d'authentification, y compris les packages pour OAuth1a et OAuth2.
- Sérialisation prenant en charge les sources de données ORM et non ORM.
- Personnalisable jusqu'au bout - utilisez simplement les vues basées sur les fonctions habituelles si vous n'avez pas besoin des fonctionnalités les plus puissantes.
- Une documentation complète et un excellent support communautaire.
- Utilisé et approuvé par des sociétés de renommée internationale telles que Mozilla, Red Hat, Heroku et Eventbrite.

Figure 12:Logo de Django Rest Framework.

Angular est un framework d'application Web gratuit et opensource basé sur TypeScript dirigé par l'équipe Angular de Google et par une communauté d'individus et d'entreprises. Angular est une réécriture complète de la même équipe qui a construit AngularJS.

Angular regroupe les notions d'HTML, CSS, TypeScript.

Figure 13:Logo de Angular

- **Système de gestion de base de données**

PostgreSQL, également connu sous le nom de **Postgres**, est un système de gestion de base de données relationnelle gratuit et open source mettant l'accent sur l'extensibilité et la conformité SQL. Il s'appelait à l'origine POSTGRES, faisant référence à ses origines en tant que successeur de la base de données Ingres développée à l'Université de Californie à Berkeley. En 1996, le projet a été renommé en PostgreSQL pour refléter son support pour SQL. Après un examen en 2007, l'équipe de développement a décidé de conserver le nom PostgreSQL et l'alias Postgres.

Figure 14:Logo de PostgreSQL

- **Langage et logiciel de modélisation**

Le **langage de modélisation unifié** (**UML**) est un langage de modélisation de développement à usage général dans le domaine du génie logiciel qui vise à fournir un moyen standard de visualiser la conception d'un système. La création d'UML a été motivée à l'origine par le désir de normaliser les systèmes de notation et les approches disparates de la conception de logiciels.

Figure 15:Logo d'UML.

3.2 Sécurité de l'application

Une application crédible nécessite un certain niveau de sécurité. Il existe en effet un grand nombre de risques auxquels est exposée tout logiciel :

- L'usurpation d'identité ;
- L'accès aux données confidentielles ;
- Les injections de code.

Pour remédier à cela, il a été mis en place :

- Un système d'authentification ne donnant l'accès qu'à tout utilisateur inscrit ;
- Une redirection vers une page d'erreur pour toute modification de l'url ;
- Une authentification personnalisée camouflée pour l'administrateur.

3.3 Evaluation financière de la solution

La conception d'une solution logicielle nécessite nécessairement la mise à disposition d'un budget qui couvrira toutes les dépenses nécessaires. Ainsi, pour l'application qui a été développée, les informations financières sont consignées dans le tableau ci-après.

Tableau 2:Evaluation financière

Conception du projet			
Description	Tarif horaire	Nombre d'heures en raison de dix heures par jour	Montant total
Développement de l'application	2000	400	800000
Formation du personnel			
Description	Tarif journalier	Nombre de jours	Montant total
Session de formation du personnel	20000	3	60000
Intégration de l'application au système de l'entreprise (pour un an)			
Description	Tarif mensuel	Nombre de mois	Montant total
Hébergement	10000	12	120000

3.4 Présentation de l'application

L'application réalisée se présente en différents volets. La partie principale contient l'espace utilisateur Lambda pour la configuration des programmes pré saisis par l'administrateur. Il existe un lien permettant à l'administrateur d'accéder à son espace, où il configure les données statiques.

3.4.1 Mise en place de la base de données

Etant donné que Django est utilisé, la base de données s'établit comme suit :

- Création des modèles qui constitueront les tables

```
from django.db import models

# Create your models here.
# -*- encoding: utf-8 -*-


class Programme_n(models.Model):
    code = models.CharField(max_length=5)
    exercice = models.CharField(max_length=30)
    libelle = models.CharField(max_length=100)
    objectif = models.CharField(max_length=100)

    def __str__(self):
        return f"{self.code}, {self.libelle}"

class Responsable_n(models.Model):
    code = models.CharField(max_length=5)
    libelle = models.CharField(max_length=30)

    def __str__(self):
        return f"{self.code}, {self.libelle}"


class Realisation_n(models.Model):
    code = models.CharField(max_length=30)
    libelle = models.CharField(max_length=30)

    def __str__(self):
        return f"{self.code}, {self.libelle}"


class TauxRealisation_n(models.Model):
    taux = models.FloatField()
    libelle = models.CharField(max_length=200)

    def __str__(self):
        return f"{self.taux}, {self.libelle}"


class CategorieDepense_n(models.Model):
    code= models.CharField(max_length=5)
    libelle = models.CharField(max_length=100)
```

```
    def __str__(self):
        return f"{self.code}, {self.libelle}"


class Paragraphe_n(models.Model):
    code= models.CharField(max_length=5)
    libelle = models.CharField(max_length=100)
          categorie=                                models.ForeignKey(CategorieDepense_n,
on_delete=models.CASCADE)

    def __str__(self):
        return f"{self.code}, {self.libelle}"


class Ligne_n(models.Model):
    code= models.CharField(max_length=5)
    libelle = models.CharField(max_length=100)
    paragraphe = models.ForeignKey(Paragraphe_n, on_delete=models.CASCADE)

    def __str__(self):
        return f"{self.code}, {self.libelle}"

class Article(models.Model):
    code= models.CharField(max_length=5,unique=True)
    libelle= models.CharField(max_length=100)
    ligne= models.ForeignKey(Ligne_n, on_delete=models.CASCADE)

    def __str__(self):
        return f"{self.code}, {self.libelle}"


class Action(models.Model):
    intitule = models.CharField(max_length=200)
    code = models.CharField(max_length=10,unique=True)
    objectif = models.CharField(max_length=300)
    mode_de_calcul = models.CharField(max_length=200)
    indicateur = models.CharField(max_length=200)
    source = models.CharField(max_length=200)
    valeur_reference = models.CharField(max_length=200)
    responsable= models.ForeignKey(Responsable_n, on_delete=models.CASCADE)
    programme= models.ForeignKey(Programme_n, on_delete=models.CASCADE)
    status = models.BooleanField(default=True)
    valid=models.BooleanField(default=False)

    def validate(self):
        self.valid=True

    def supprimer(self):
```

```
        self.status=False

    def restaurer(self):
        self.status=True




class Activite(models.Model):
    intitule = models.TextField(max_length=200)
    date_debut = models.DateField()
    date_fin = models.DateField()
        action = models.ForeignKey(Action, on_delete=models.CASCADE,
related_name='activites')
    status = models.BooleanField(default=True)
    valid=models.BooleanField(default=False)
    def validate(self):
        self.valid=True

    def supprimer(self):
        self.status=False

    def restaurer(self):
        self.status=True



class Tache(models.Model):
    intitule = models.TextField(max_length=200)
    date_debut = models.DateField()
    date_fin = models.DateField()
        activite = models.ForeignKey(Activite, on_delete= models.CASCADE,
related_name='taches')
    status = models.BooleanField(default=True)
    valid=models.BooleanField(default=False)
    def validate(self):
        self.valid=True

    def supprimer(self):
        self.status=False

    def restaurer(self):
        self.status=True




class Prevision(models.Model):
```

```
    cible_t1 = models.FloatField()
    cible_t2 = models.FloatField()
    cible_t3 = models.FloatField()
    cible_t4 = models.FloatField()
    t1_quantite = models.FloatField()
    t1_saisie = models.FloatField()
    t2_quantite = models.FloatField()
    t2_saisie = models.FloatField()
    t3_quantite = models.FloatField()
    t3_saisie = models.FloatField()
    t4_quantite = models.FloatField()
    t4_saisie = models.FloatField()
    saisie_be = models.FloatField()
    saisie_ptf = models.FloatField()
    status = models.BooleanField(default=True)
    valid=models.BooleanField(default=False)
    article=models.ForeignKey(Article, on_delete=models.CASCADE)
            activite=      models.ForeignKey(Activite,      on_delete=models.CASCADE,
related_name='previsions')
    def validate(self):
        self.valid=True

    def supprimer(self):
        self.status=False

    def restaurer(self):
        self.status=True

class Execution(models.Model):
    execution_t1 = models.FloatField()
    execution_t2 = models.FloatField()
    execution_t3 = models.FloatField()
    execution_t4 = models.FloatField()
    be_execution = models.FloatField()
    ptf_execution = models.FloatField()
    total_execution = models.FloatField()
    etat_real_t1 = models.FloatField()
    etat_real_t2 = models.FloatField()
    etat_real_t3 = models.FloatField()
    etat_real_t4 = models.FloatField()
    etat_real_n = models.FloatField()
    status = models.BooleanField(default=True)
    valid=models.BooleanField(default=False)
        activite    =    models.ForeignKey(Activite,    on_delete=    models.CASCADE,
related_name='execution', default=0)
    prevision=models.OneToOneField(Prevision, on_delete=models.CASCADE)
```

```
def validate(self):
    self.valid=True

def supprimer(self):
    self.status=False

def restaurer(self):
    self.status=True
```

- Créer la base de données avec les commandes : « python manage.py makemigrations » et « python manage.py migrate ».

Le classe utilisateur n'est pas créée. Celle de Django est utilisée.

3.4.2 Plan de navigation

La conception d'une application ne suffit pas à la rendre utilisable. Il faut que tout utilisateur puisse l'utiliser et ainsi tirer profit de ses services. Dans cette optique, un plan de navigation doit impérativement mis à disposition.

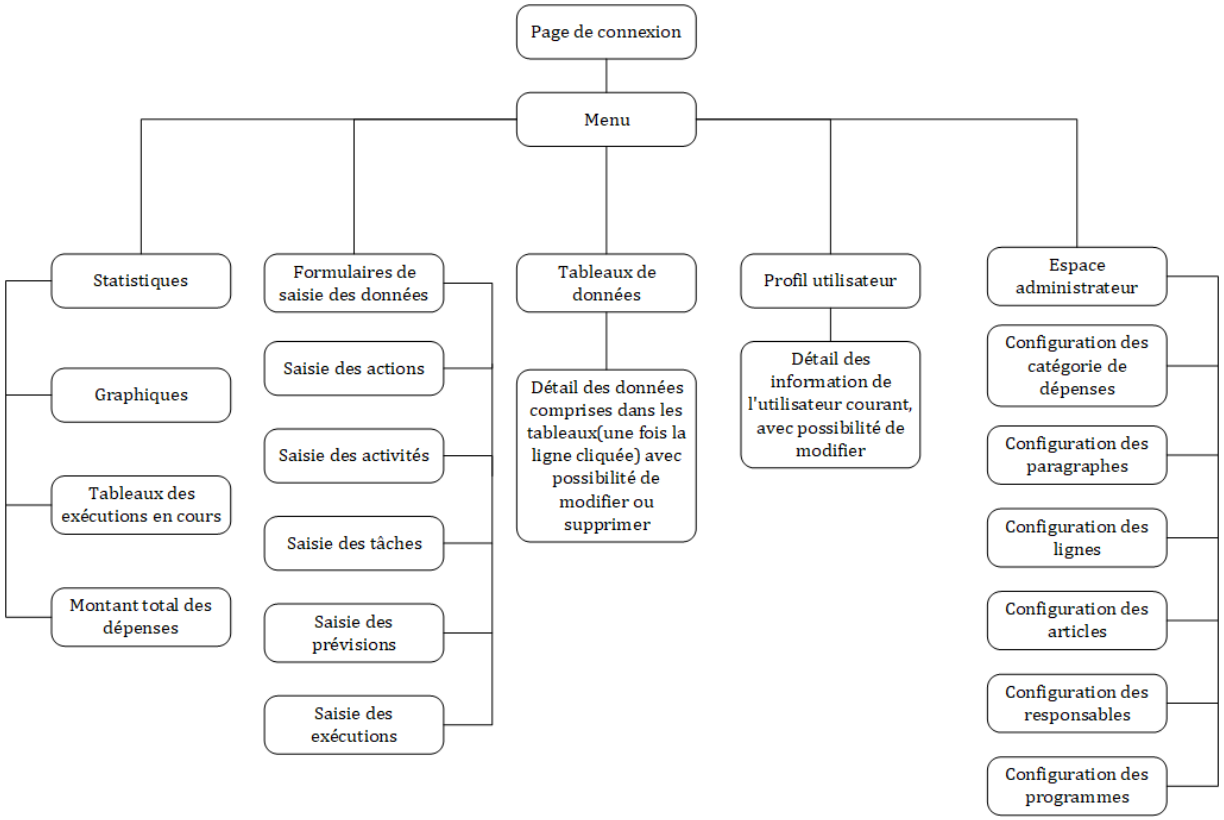


Figure 16:Plan de navigation.

3.4.3 Quelques masques de saisie

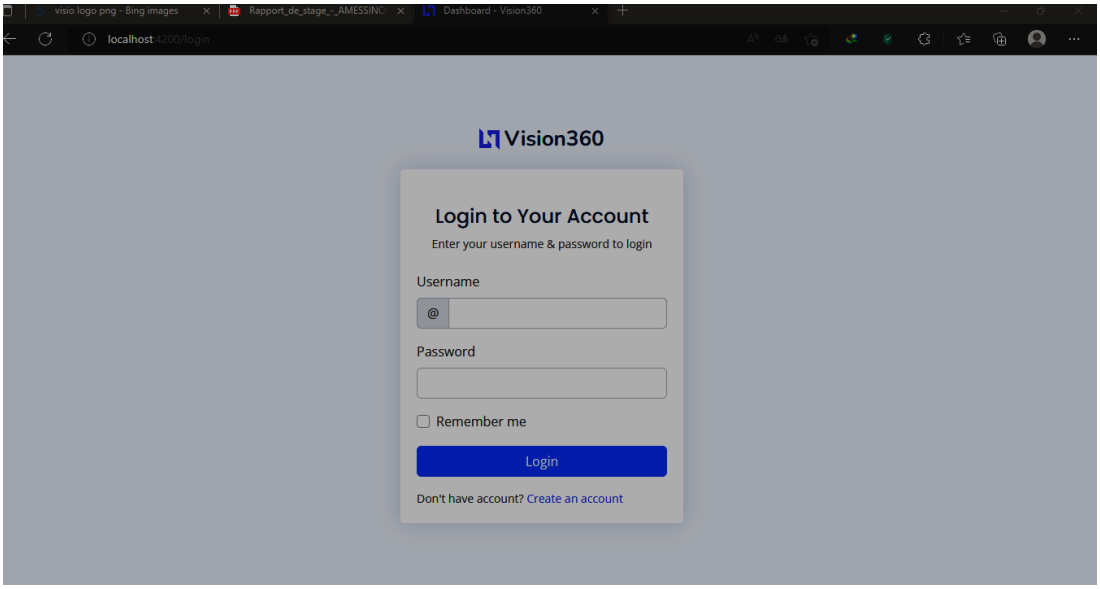


Figure 17:Page de connexion

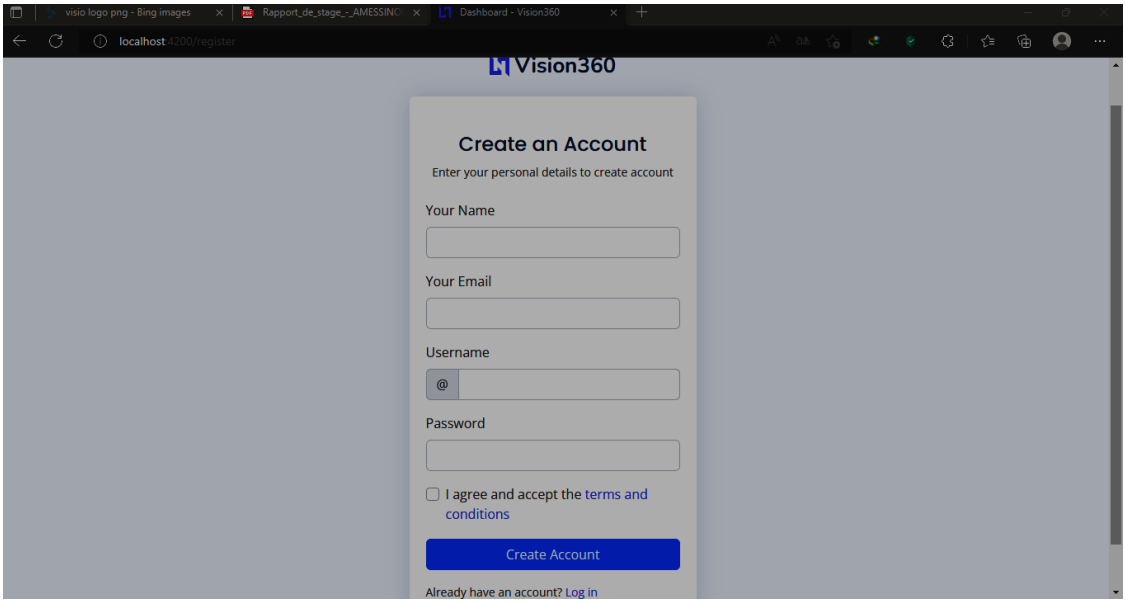


Figure 18:Page d'inscription

CONCLUSION

A titre de conclusion, ce stage a permis la mise en pratique des connaissances acquises au cours de l'année scolaire, son objectif étant l'optimisation d'une solution logicielle. J'ai pu me familiariser avec le procédé de reverse engineering avec le framework Django ; j'ai acquis des connaissances sur Django Rest Framework, et Angular que je ne maîtrisais pas mais qui m'ont servi efficacement.

BIBLIOGRAPHIE INDICATIVE

Ouvrages

- Documentation Django sur Google
- Documentation Django Rest Framework sur Google
- Documentation Angular sur Google

Cours

- Cours d'UML avec Monsieur KETOGLO Komlavi Alfred
- Cours de python avec Monsieur AMEVOR et Monsieur OURO BANG 'NA Badiou
- Initiation à l'utilisation de PostgreSQL avec Monsieur OURO BANG'NA Badiou
- Cours sur les bases de données avec Monsieur TCHANTCHO Léri Damigouri
- Initiation à la programmation Web avec Monsieur AGBOKA Komlan

WEBOGRAPHIE INDICATIVE

Software Re-Engineering - GeeksforGeeks

create an api with django - YouTube

Build a Backend REST API with Python & Django (Beginner Course) - YouTube

Build And Deploy A REST API With Django REST Framework. Full Project Tutorial. - YouTube

Django Tutorial => PostgreSQL

How to Create a Login System in Python Using Django? | Python Projects | GeeksforGeeks - YouTube

Customize the Django Admin With Python – Real Python

https://www.youtube.com/watch?v=0eWrpsCLMJQ&list=PLC3y8-rFHvwhBRAgFinJR8KHIrCdTkZcZ&index=1

https://docs.djangoproject.com/en/4.0/topics/db/models/

https://krakensystems.co/blog/2020/custom-users-using-django-rest-framework

DOCUMENTS ANNEXES

Rapport de stage de ASSIMENOU Komi Salvation Kaleb, Gestion automatisée de la facturation : cas de la clinique « Maison d'Italie ».

TABLE DES MATIERES