Republic of Cameroon

Peace – Work – Fatherland

----------

University of Yaoundé I

----------

National Advanced School of Engineering Polytechnic

----------

Department of Computer Engineering

République du Cameroun

Paix – Travail – Patrie

----------

Université de Yaoundé I

----------

Ecole National Supérieur Polytechnique Yaoundé

----------

Département du Génie Informatique



Présenté par :

KANA NGUEDIA FRANKLIN

GUY DANIEL NGOSSO OKALIA

KAMSU KOM FLORIAN

ARAFAT BARE

Sous la supervision de

Dr BATCHAKUI

Enseignant à L’ENSPY

COMPUCLINIC : MODULE LABORANTIN

Table des matières

[INTRODUCTION 2](#_Toc106801706)

[I- Présentation du contexte General 3](#_Toc106801707)

[II- Etude de cas : module laborantin 3](#_Toc106801708)

[A. Analyse et spécification des besoins 3](#_Toc106801709)

[1. Analyses des besoins 3](#_Toc106801710)

[2. Besoins fonctionnels 4](#_Toc106801711)

[3. Besoins non fonctionnels 5](#_Toc106801712)

[B. Spécification des besoins 5](#_Toc106801713)

[1. Diagramme de cas d’utilisation 5](#_Toc106801714)

[**Description textuelle :** 6](#_Toc106801715)

[2. Diagrammes de Séquences (Système) 7](#_Toc106801716)

[ Remplir Bulletin 9](#_Toc106801717)

[III. Conception 11](#_Toc106801718)

[1 Diagrammes de Séquences (Technique) 11](#_Toc106801719)

[2 Diagramme de classes Techniques 16](#_Toc106801720)

[IV Architecture 19](#_Toc106801721)

[2 Guide de deploiement 19](#_Toc106801722)

[1. Deploiement avec linterface WSGI 20](#_Toc106801723)

[ Déploiement de Django avec Gunicorn 20](#_Toc106801724)

[o Installation de Gunicorn 20](#_Toc106801725)

[o Django dans Gunicorn comme application WSGI générique 20](#_Toc106801726)

[2. Deploiement avec ASGW 20](#_Toc106801727)

[a. Deploiement de Django avec Daphne 20](#_Toc106801728)

[ Installation de Daphne 20](#_Toc106801729)

[ Lancement de Django dans Daphne 21](#_Toc106801730)

[b. Deploiement de Django avec Hypercorn 21](#_Toc106801731)

[ Installation de Hypercorn 21](#_Toc106801732)

[ Lancement de Django dans Hypercorn 21](#_Toc106801733)

[c. Comment utiliser Django avec Uvicorn 21](#_Toc106801734)

[ Installation de Uvicorn 21](#_Toc106801735)

[ Lancement de Django dans Uvicorn 22](#_Toc106801736)

[3 Guide d’installation 23](#_Toc106801737)

[INSTALLATION DU BACKEND 23](#_Toc106801738)

[Installer python: 23](#_Toc106801739)

[Obtenez la dernière version de Python à l’adresse https://www.python.org/downloads/ ou par l’intermédiaire du gestionnaire des paquets de votre système 23](#_Toc106801740)

[Installer Django : 23](#_Toc106801741)

[Déployer le serveur 23](#_Toc106801742)

[INSTALLATION DU FRONTEND 24](#_Toc106801743)

[Installer nodeJS: 24](#_Toc106801744)

[Obtenez la dernière version de nodeJS à l’adresse https://www.nodejs.org/ ou par l’intermédiaire du gestionnaire des paquets de votre système 24](#_Toc106801745)

[Installer le Web package npm / yarn : 24](#_Toc106801746)

[Installer les dépendances du serveur: 24](#_Toc106801747)

[Déployer le serveur 24](#_Toc106801748)

[4 Guide Utilisateur 25](#_Toc106801749)

[CONCLUSION 29](#_Toc106801750)

# INTRODUCTION

Dans le but de rendre plus complet COMPUCLINIC dans sa mission de gestion d’hôpital, le module laborantin a été ajouté à l’application déjà existante. Ce module permettra via l’application de gérer toutes les tâches liées aux activités de laboratoires, allant de la demande d’examen par le médecin jusqu’à l’envoi des résultats d’examens par le laborantin.

Dans ce document nous ferons dans un premier temps un rapport détaillé sur les phases d’analyse, de conception et d’implémentation du module et par la suite nous présenterons un guide de déploiement et un guide utilisateur.

## Présentation du contexte General

COMPUCLINIC est une application de gestion d’hopital. Cette application prend en charge le circuit d’un patient depuis son entrée jusqu’à la sortie de l’hopital. Ce patient peut etre nouveau ou ancien et sa reception se fait selon son état (ambulatoire oi interne). COMPUCLINIC fourni les fonctionnalités suivantes:

* Identifier un patient (interne et externe) : cette fonctionnalité est fourni via Le sous module « Identification du patient » qui est une interface activable en lecture sinon en saisie, qui fournit ou enregistre les informations suivantes selon que le malade est ambulatoire ou interne.
* Création d’un dossier du Patient
* Génération des quittances : Les différentes prestations offertes par la SSS ne sont accessibles aux malades que moyennant **unticket informatisé d’accès aux prestations de soins*.*** L’obtention **du TIAPS** ou « ticket d’accès aux prestations »est conditionnée par le paiement de la quittance correspondante.

Cependant l’application n’a pas une fonctionnalité permettant de gérer les activités liés au laboratoire notre mission consiste donc à developper le module qui fournira cette fonctionnalité et à l’inclure dans COMPUCLINIC.

# Etude de cas : module laborantin

Dans cette partie on doit être capable de présenter un bulletin de résultat d’un examen précis demander par un médecin.

Pouvoir retracer tous les examens supplémentaires dans le dossier technique du patient

## Analyse et spécification des besoins

Le module qu’on se propose de développer doit tenir en compte les exigences des différents utilisateurs c’est-à-dire les laborantins. Une étude des besoins de ces acteurs est alors nécessaire. Le présent chapitre s’articule autour de deux volets : le premier où nous exposerons les différents besoins fonctionnels et non fonctionnels des différents acteurs et un deuxième volet concerne la spécification de ces besoins via les diagrammes de cas d’utilisation

### Analyses des besoins

#### Définition du rôle du laborantin

La fonction du laborantin est de réaliser des analyses médicales ayant été prescrites par le médecin. Pour cela, il prélève le sang du patient, ou bien réalise un frottis ou recueille un échantillon urinaire. Il utilise ensuite le prélèvement pour l'analyser et voir s'il y a une infection, un début de grossesse ou encore un problème au niveau du métabolisme.

#### Définition du rôle du médecin

La fonction du médecin est de proposer un diagnostic complet pour la prise en charge du patient après avoir reçu le rapport d’examen qui a été fait par le laborantin.

#### Présentation du parcourt du patient

Pour s’assurer que la conception et le développement soit sur la bonne voie, il est crucial de recueillir des informations sur le projet et le client.

A cet effet nous avons effectué nos recueils d’information sur le système de gestion des examens en laboratoire auprès du cabinet médical Sainte-Thérèse de Biyem-Assi, l’objectif étant de tracer le parcourt d’un patient depuis son arrivé, au prélèvement de la fiole nécessaire pour son examen en laboratoire.

Un patient arrive à l’accueil de l’hôpital où il est reçu par un(e) infirmier(e) (secrétaire) qui relève les différents paramètres du patient tel que : sa température, son poids, etc…

Ainsi après avoir relevé ses paramètres elle envoie son dossier au médecin généraliste ou un médecin spécialiste en fonction de son emploie de temps.

Lors de la réception du patient par le médecin, celui-ci fourni un premier diagnostic et peut demander au patient de faire des examens supplémentaires pour effectivement avoir le fin mot sur la maladie et pouvoir offrir un traitement adéquat.

Ainsi le médecin peut faire une demande de traitement supplémentaire auprès des laborantins en envoyant le dossier médical du patient a laboratoire. Une fois au laboratoire le laborantin en charge des prélèvements effectue donc un premier prélèvement et étiquette pour tracer celui-ci tandis que les autres laborantins effectuent l’analyse les fioles et à la fin te ressort un **bulletin de résultat**

Le bulletin de résultat disponible, le laborantin renvoie le dossier avec le bulletin de résultat au médecin en charge du patient qui va donc fournir ces derniers diagnostics et renvoyer les bulletins aux infirmiers qui vont procéder à la prise en charge du patient

Présentation du bulletin d’examen du cabinet Sainte-Thérèse

### Besoins fonctionnels

Ce sont les actions et les réactions que le système doit faire suite à une demande d’un acteur principal. Tenant compte de la nature de l’application, on distingue les besoins par acteurs

* Laborantin : l’application doit permettre au laborantin de :
  + Remplir un rapport de résultat en fonction du type d’examen prescrit par le médecin
  + De renvoyer ce rapport au médecin concerner après avoir fini les analyses
* Médecin
  + Demander d’effectuer un examen en laboratoire
  + Consulter tous les bulletins d’examens de ces patients
  + Faire des diagnostics

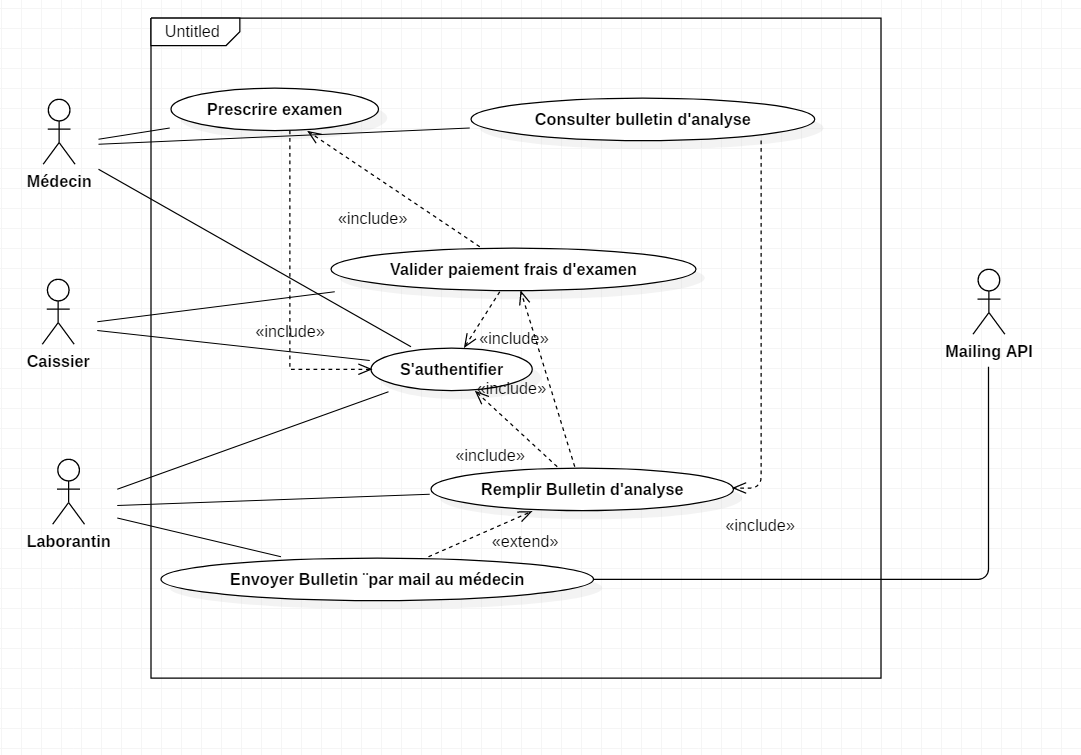
### Besoins non fonctionnels

* **L’extensibilité** : l’architecture de l’application permettra l’évolution et la maintenance (ajout ou suppression ou mise à jour) au niveau de ses différents modules d’une manière flexibles.
* **La sécurité :** l’accès aux informations n’est possible qu’après vérification des privilèges et des droits d’accès. Ainsi tout utilisateur passera par une phase d’authentification pour pouvoir consulter les services offerts.
* **L’ergonomie et convivialité :** l’application fournira une interface conviviale et simple à utiliser et qui ne requiert aucun prérequis, donc elle pourra être exploitable par tout type d’utilisateurs.

## Spécification des besoins

Dans cette partie nous présenterons les diagrammes de cas d’utilisation principaux et la description des scénarios de cas d’utilisation : cette partie représente la vue fonctionnelle du module.

### Diagramme de cas d’utilisation



### 

### **Description textuelle :**

* S’authentifier : Cas d’utilisation hérité de la première itération du projet Compuclinc

1. Le membre du personnel entre ses informations de Login.
2. Le Système vérifie la correspondance en base de donnée.
3. L’utilisateur accede à l’interface lui correspondant si la vérification de (ii) plus haut renvoie vrai, le cas contraire un messsage d’érreur lui est renvoyé.

* Prescrire Examen:

1. Le médécin choisit les examens à effectuer par le patient
2. Le médécin valide ses choix et les examens à faire sont ajoutés au dossier médical du patient

* Valider Paiement frais Examens :

1. Le patient se présente devant le/la Caissier/ère
2. Le Caissier sélectionnes la prestation
3. Le Caissier Valide le Paiement après avoir recu l’argent et la quittance est généréé

* Remplir Bulletin d’examen :

1. Le Laborantin accède à la liste des patients pour lesquels il a des examens à faire
2. Le Laborantin sélectionne un patient et effectue les examens associés, en laboratoire
3. Le Laborantin renseigne les résultats dans un bulletin d’analyse.

* Envoyer Bulletin par Mail : Ce cas étend les d’utilisation « Remplir Bulletin Examen »

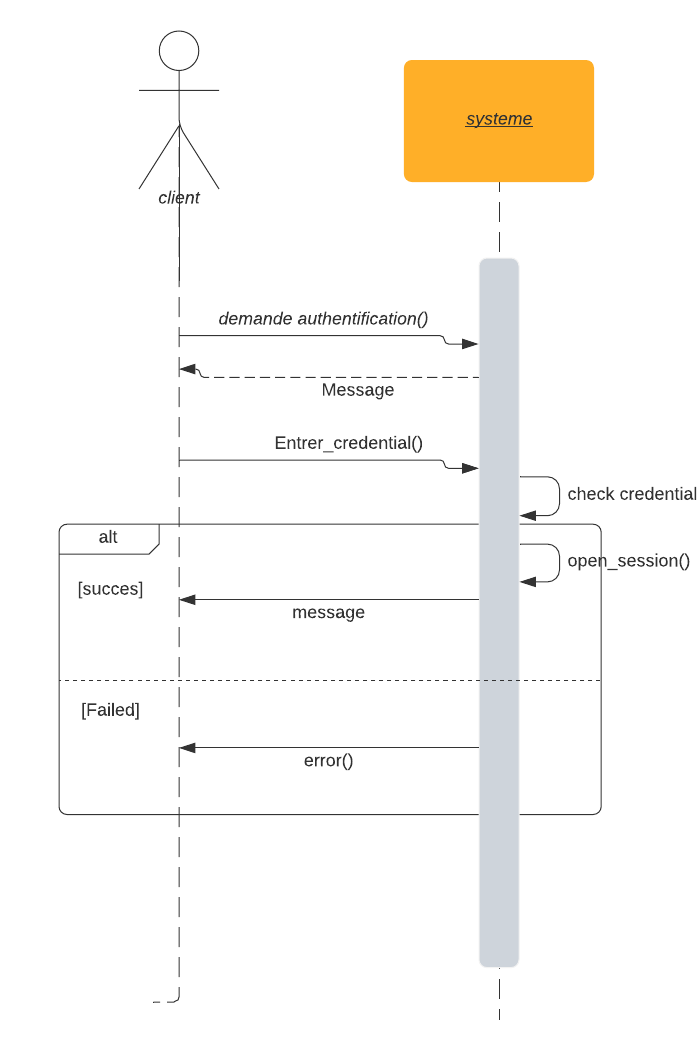
1. Le Laborantin selectionne le bulletin à envoyer par mail et le médecin destinataire
2. Le mail est envoyé au médecin grace à l’API associée.

* Consulter Bulletin d’Analyse :

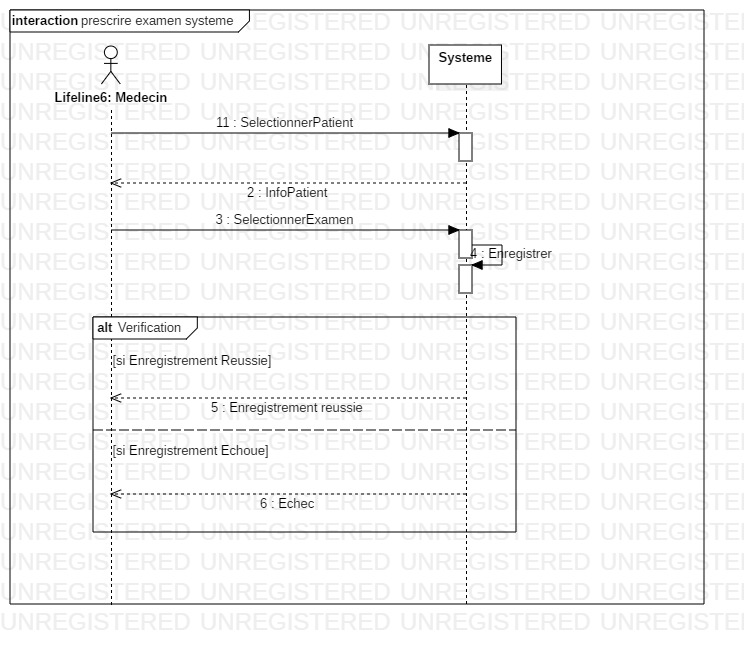
1. Le médecin accède au dossier médical du patient
2. Il consulte la section examens du dossier
3. Les examens ayant été renseignés dans les cas d’utilisation précédent, Il accède aux résultats des examens.

### Diagrammes de Séquences (Système)

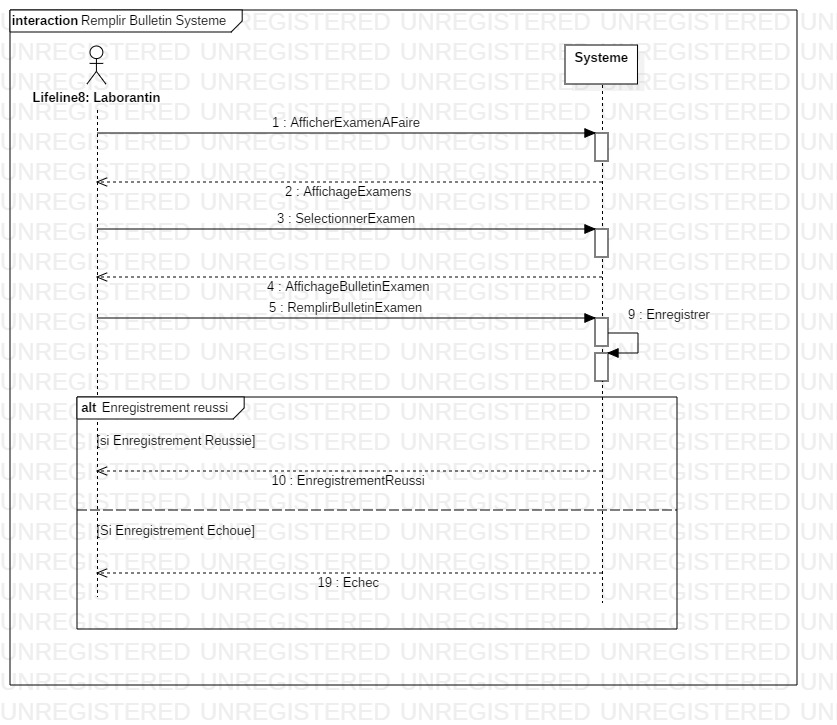
* S’authentifier



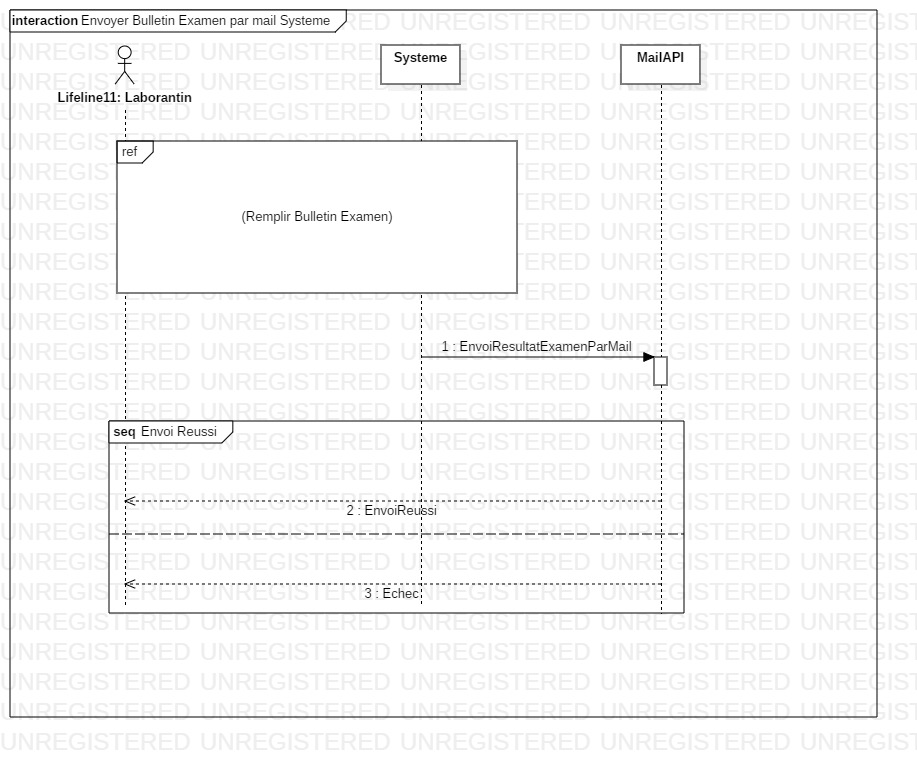
* Prescrire Examen



### Remplir Bulletin



* Envoyer Bulletin par Mail



# Conception

### Diagrammes de Séquences (Technique)

* S’authentifierUne image contenant table

  Description générée automatiquement
* PrescrireExamenUne image contenant table

  Description générée automatiquement
* Remplir bulletin examen

Une image contenant table

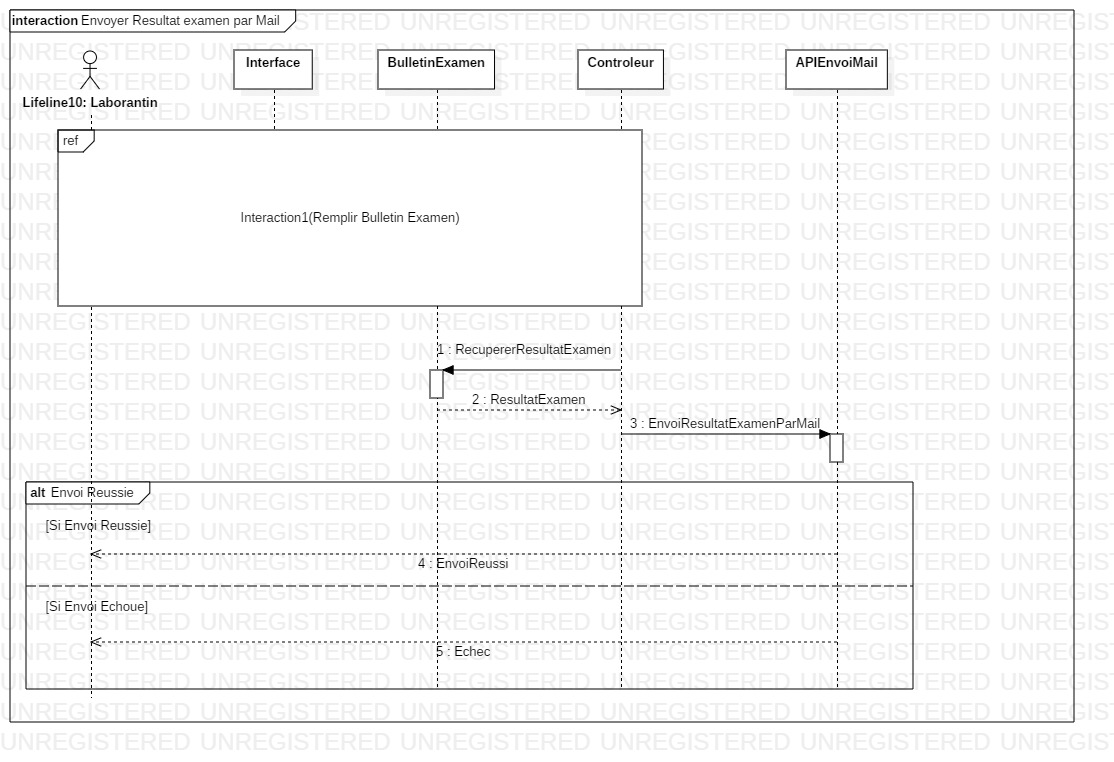
Description générée automatiquement

* Consulter Bulletin

Une image contenant table

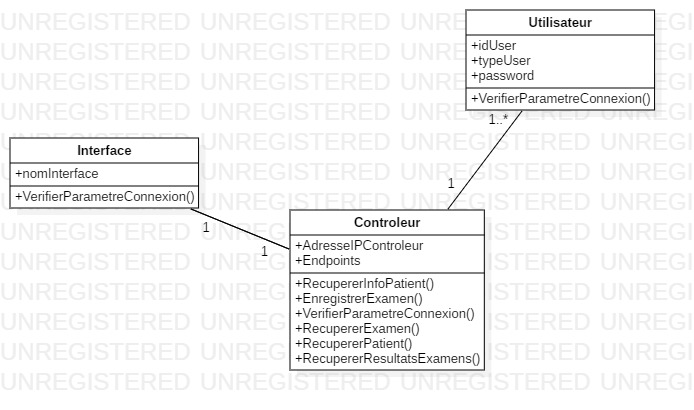
Description générée automatiquement

* Envoyer résultats

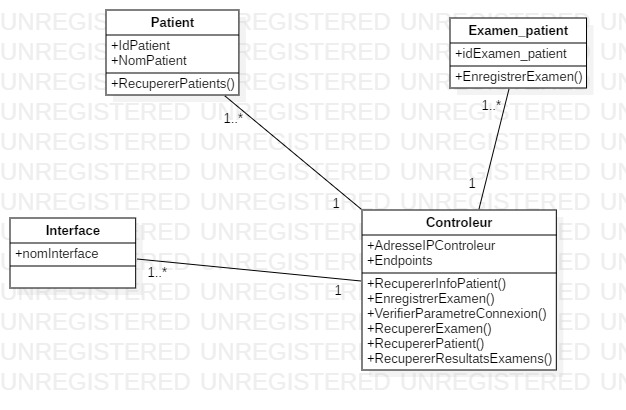


### Diagramme de classes Techniques

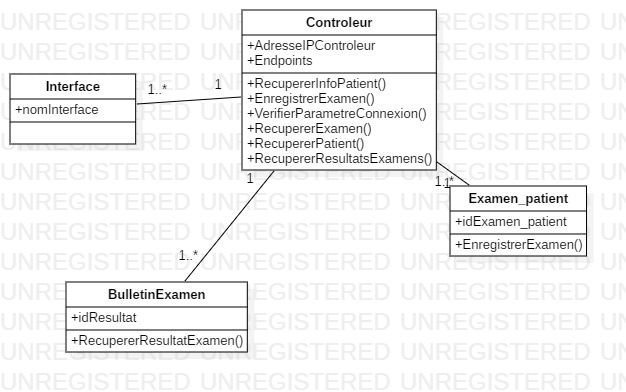
* S’authentifier



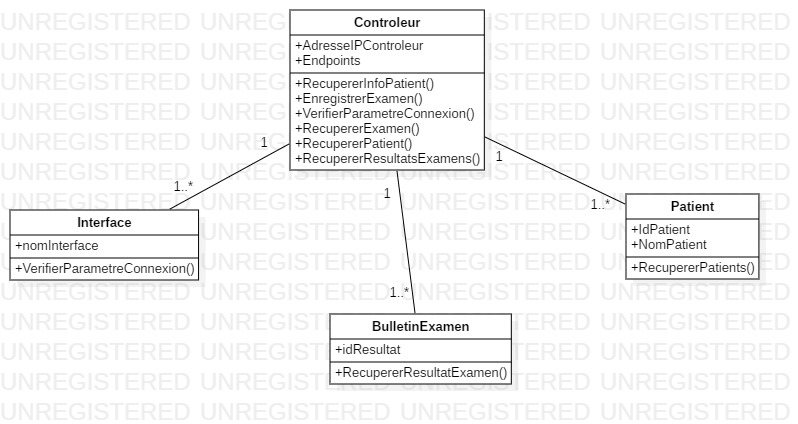
* Prescrire Examen



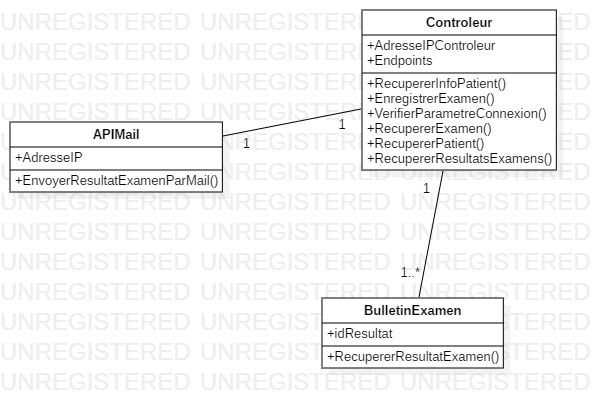
* Remplir bulletin examen



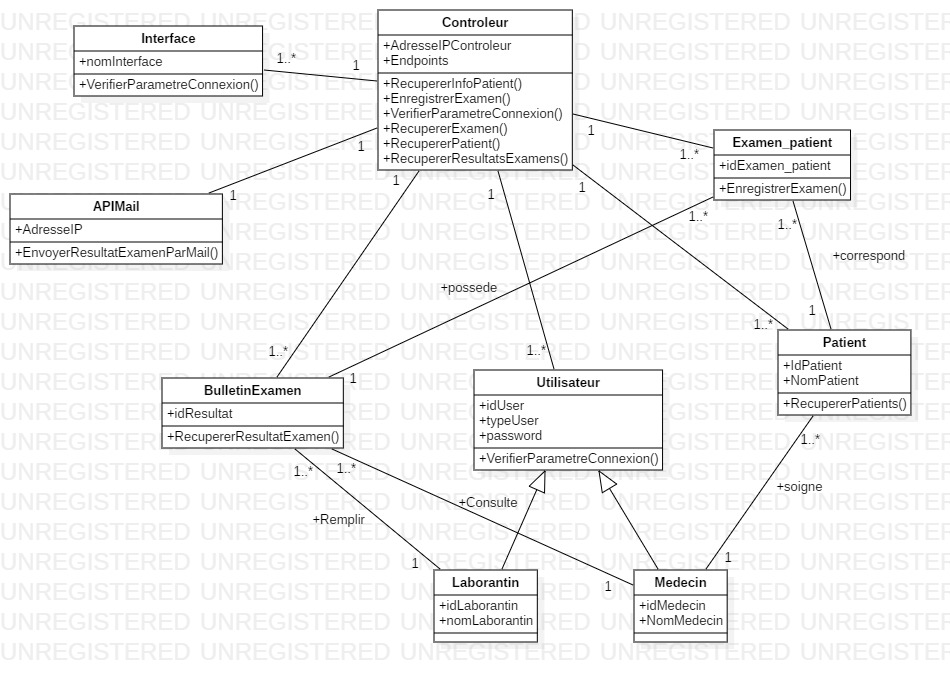
* Consulter Bulletin



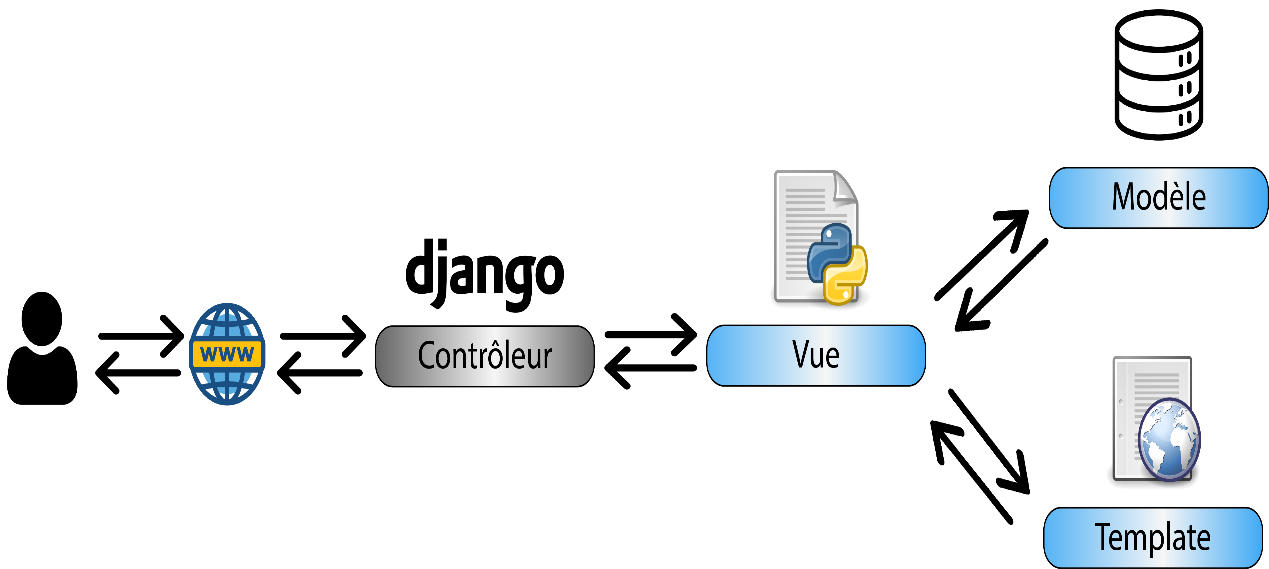
* Envoyer résultats



Le diagramme de classe Technique globale relatif à notre application est :



### IV Architecture



Dans cette architecture, nous avons le client qui va se connecter au réseau de l’application, puis le contrôleur qui va gérer les requêtes HTTP, le routage, la sécurité et l’authentification de l’utilisateur, va traiter la requête de l’utilisateur ;

La **vue** qui contient les pages dynamiques pour tous les utilisateurs (administrateurs, médecin généraliste, médecin spécialiste, réceptionniste, caissière, laborantin) ;

Le modèle qui contient les tables de notre Base de Données à savoir : Patient, Consultation, Maladie, Prescription, Médicament, Examen, Reçu, Ligne de reçu, Symptôme.

# Guide de deploiement

Django étant par nature un cadriciel Web a besoin d’un serveur Web pour fonctionner. Et comme la plupart des serveurs Web ne communiquent pas nativement avec Python, il y a besoin d’une interface pour permettre cette communication.

Django prend actuellement en charge deux interfaces : WSGI et ASGI.

WSGI est le standard Python principal pour la communication entre les serveurs Web et les applications, mais il ne gère que le code synchrone.

ASGI est le nouveau standard orienté sur le code asynchrone qui permet à un site Django de bénéficier des fonctionnalités asynchrones de Python et de Django au fur et à mesure de leur disponibilité.

Il est également nécessaire de considérer la manière de gérer les fichiers statiques de votre application ainsi que le signalement des erreurs.

Pour terminer, avant de passer une application en production, il est recommandé de passer en revue notre liste de contrôle de déploiement pour s’assurer que les configurations appliquées sont convenables.

## Deploiement avec linterface WSGI

la technique utilise ici pour le deploiement d'une application web django avec linterface WSGI sera celle de Gunicorn

## Déploiement de Django avec Gunicorn

Gunicorn (« Green Unicorn ») est un serveur WSGI en pur Python pour UNIX. Il n’a aucune dépendance et peut être installé avec pip.

## Installation de Gunicorn

Installez gunicorn en exécutant *python -m pip install gunicorn* sur le terminal. Pour plus de détails, consultez la documentation de gunicorn.

## Django dans Gunicorn comme application WSGI générique

Lorsque Gunicorn est installé, une commande gunicorn est disponible et permet de lancer le processus de serveur de Gunicorn. L’invocation la plus simple de gunicorn est de lui transmettre l’emplacement d’un module contenant un objet d’application WSGI nommé application, ce qui pour un projet Django typique devrait ressembler à ceci :

gunicorn myproject.wsgi

Cela démarrera un processus utilisant un fil d’exécution écoutant sur 127.0.0.1: 8000. Il faut que votre projet soit dans le chemin Python ; la façon la plus simple de s’en assurer est d’exécuter cette commande dans le même répertoire que votre fichier manage.py.

## Deploiement avec ASGW

## Deploiement de Django avec Daphne

Daphne est un serveur ASGI en Python pur pour UNIX, maintenu par des membres du projet Django. Il constitue le serveur de référence pour ASGI.

## Installation de Daphne

Vous pouvez installer Daphne avec pip en tapant la commande suivante sur le terminal :

*python -m pip install daphne*

## Lancement de Django dans Daphne

Lorsque Daphne est installé, une commande daphne est disponible et permet de lancer le processus de serveur de Daphne. Dans le cas le plus simple, Daphne a besoin d’être appelé avec l’emplacement d’un module contenant un objet d’application ASGI, suivi par le nom donné à l’application (séparés par un deux-points).

Pour un projet Django typique, l’invocation de Daphne pourrait ressembler à ceci :

*daphne myproject.asgi:application*

Cela démarrera un processus écoutant sur 127.0.0.1: 8000. Il faut que votre projet soit dans le chemin Python ; pour s’en assurer, exécutez cette commande dans le même répertoire que votre fichier manage.py.

## Deploiement de Django avec Hypercorn

Hypercorn est un serveur ASGI qui prend en charge HTTP/1, HTTP/2 et HTTP/3 avec un accent sur la prise en charge des protocoles.

## Installation de Hypercorn

Vous pouvez installer Hypercorn avec pip en tapant l’instruction suivant sur le terminal:

*python -m pip install hypercorn*

## Lancement de Django dans Hypercorn

Lorsque Hypercorn est installé, une commande hypercorn est disponible qui exécute des applications ASGI. Hypercorn a besoin d’être appelé avec l’emplacement d’un module contenant un objet d’application ASGI, suivi par le nom donné à l’application (séparés par un deux-points).

Pour un projet Django typique, l’invocation de Hypercorn pourrait ressembler à ceci :

*hypercorn myproject.asgi:application*

Cela démarrera un processus écoutant sur 127.0.0.1: 8000. Il faut que votre projet soit dans le chemin Python ; pour s’en assurer, exécutez cette commande dans le même répertoire que votre fichier manage.py.

## Comment utiliser Django avec Uvicorn

Uvicorn est un serveur ASGI basé sur uvloop et httptools, avec un accent sur la vitesse.

## Installation de Uvicorn

Vous pouvez installer Uvicorn avec pip:

*python -m pip install uvicorn gunicorn*

## Lancement de Django dans Uvicorn

Lorsque Uvicorn est installé, une commande uvicorn est disponible qui exécute des applications ASGI. Uvicorn a besoin d’être appelé avec l’emplacement d’un module contenant un objet d’application ASGI, suivi par le nom donné à l’application (séparés par un deux-points).

Pour un projet Django typique, l’invocation de Uvicorn pourrait ressembler à ceci :

*gunicorn myproject.asgi:application -k uvicorn.workers.UvicornWorker*

Cela démarrera un processus écoutant sur 127.0.0.1: 8000. Il faut que votre projet soit dans le chemin Python ; pour s’en assurer, exécutez cette commande dans le même répertoire que votre fichier manage.py.

# Guide d’installation

# INSTALLATION DU BACKEND

# Installer python:

# Obtenez la dernière version de Python à l’adresse <https://www.python.org/downloads/> ou par l’intermédiaire du gestionnaire des paquets de votre système

# Installer Django :

Pour installer Django, ouvrez votre terminal ou invite de commande.

* Sur Windows, taper :  *win + r et taper cmd* puis « Entrer »
  + Taper :  *py -m pip install Django*
* Sur linux ou Mac, ouvrir un terminal en cliquant sur l’icône du terminal
  + *Taper : python -m pip install Django*

# Déployer le serveur

Ouvrir un terminal dans le dossier du backend **‘backend computclinic’**

* Sur linux vous devrez activer l’environnement virtuel en tapant :  *python3 –m venv env*
* Apres l’activation de l’environnement virtuel, télécharger toutes les dépendances en tapant : pip install –r requirements.txts
* Faites migrer la base de donnee en executant les requetes suivantes :
  + Sur Windows , taper :
    - *python manage.py makemigrations*
    - *python manage.py migrate*
  + Sur Linux , taper :
    - *Python3 manage.py makemigrations*
    - *Python3 manage.py migrate*
* Demarrer le serveur python
  + Sur Windows taper : python manage.py runserver
  + Sur Lnux taper : python3 manage.py runserver

# INSTALLATION DU FRONTEND

# Installer nodeJS:

# Obtenez la dernière version de nodeJS à l’adresse <https://www.nodejs.org/> ou par l’intermédiaire du gestionnaire des paquets de votre système

# Installer le Web package npm / yarn :

Pour installer npm, ouvrez votre terminal ou invite de commande.

* Taper :  *npx install*

# Installer les dépendances du serveur:

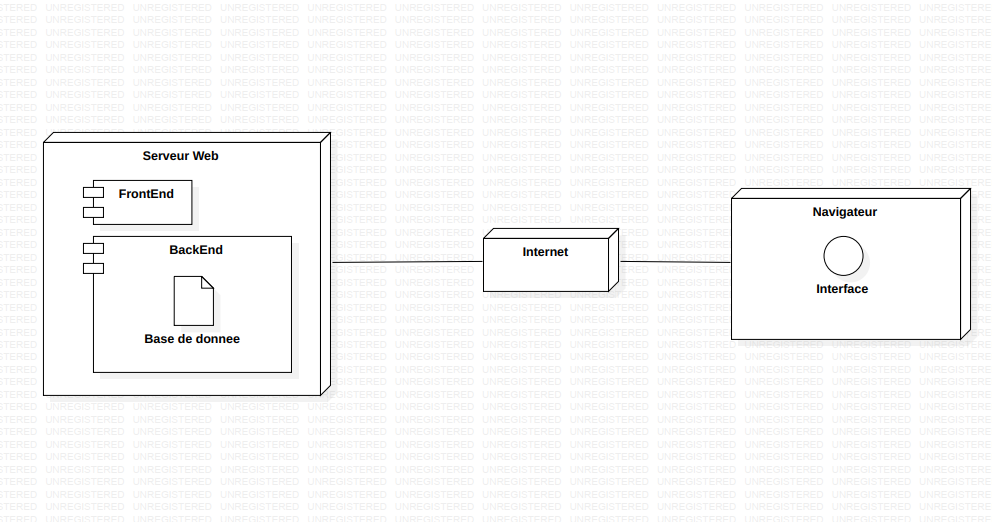
Pour installer les dependances, ouvrez votre terminal ou invite de commande.

* Taper :  *npm install*

# Déployer le serveur

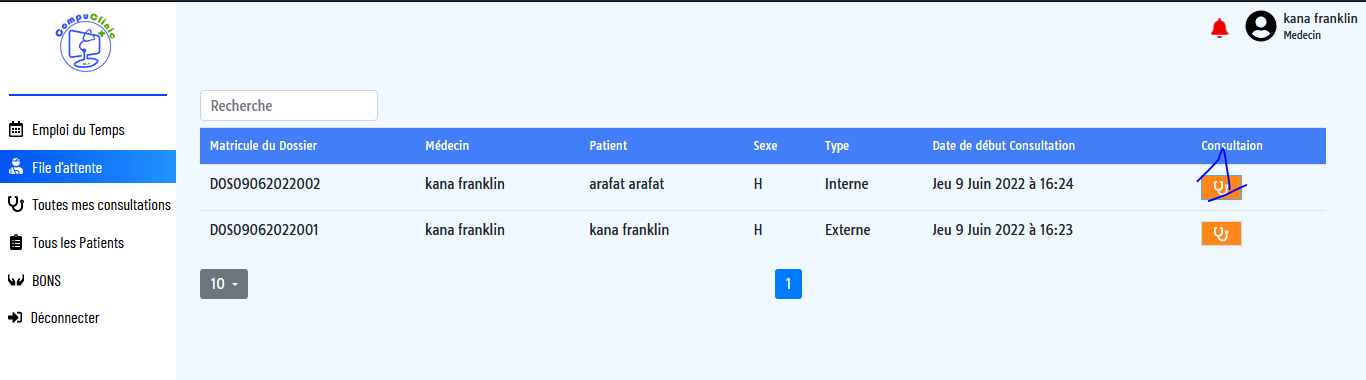
Ouvrir un terminal dans le dossier du backend **‘backend computclinic’** et taper : *npm start*

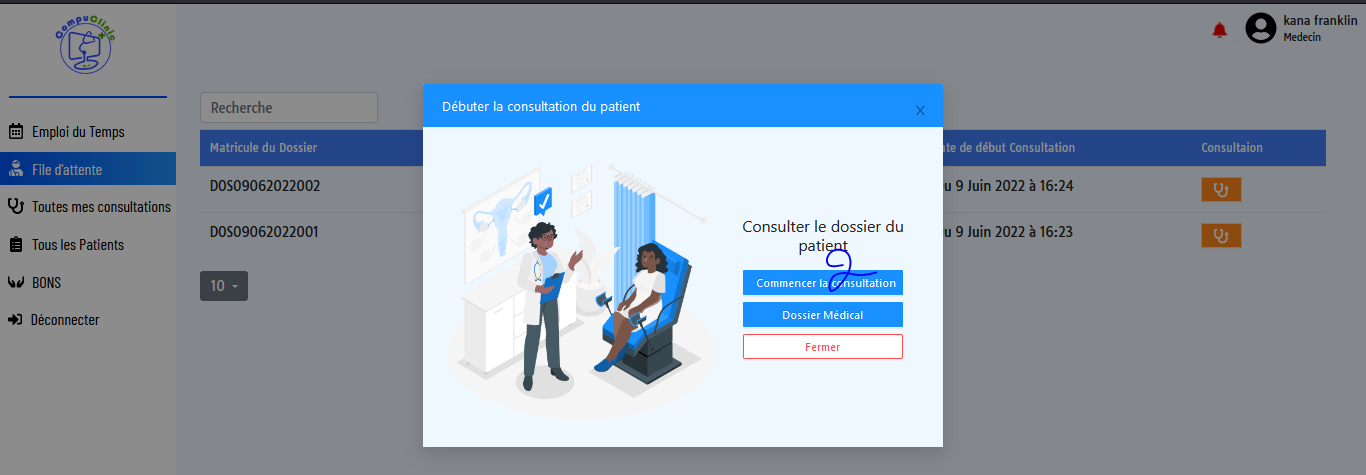
***Diagramme de déploiement***

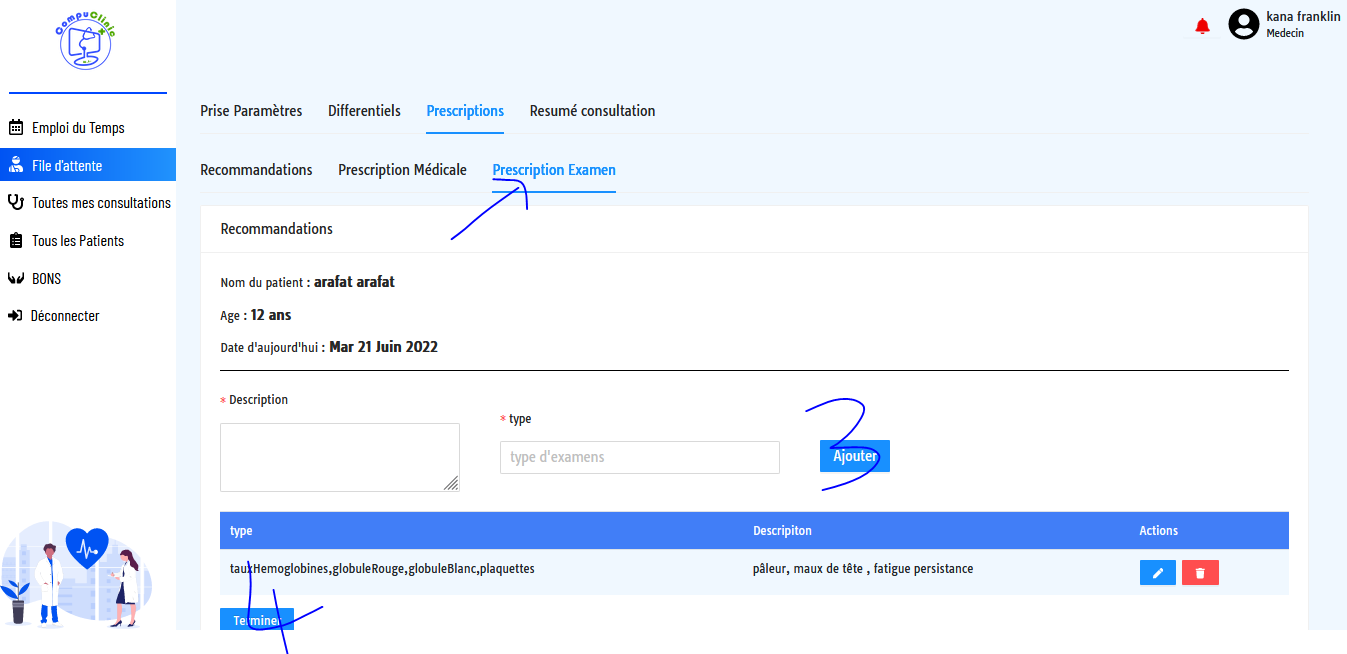


# Guide Utilisateur

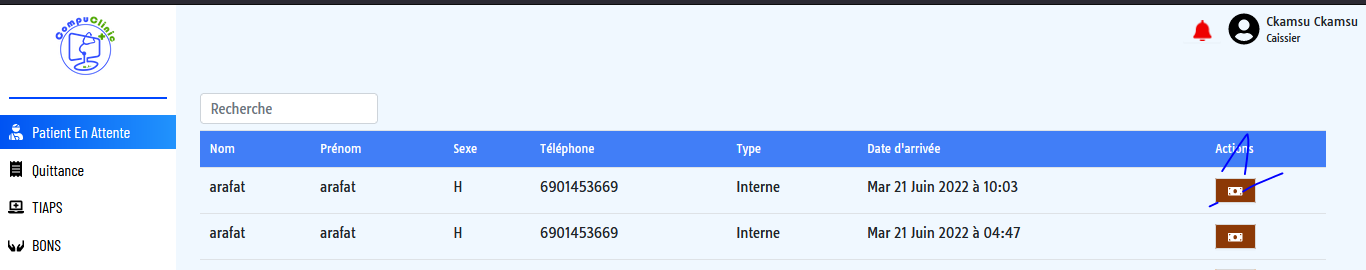
* Prescription des examens pour le patient



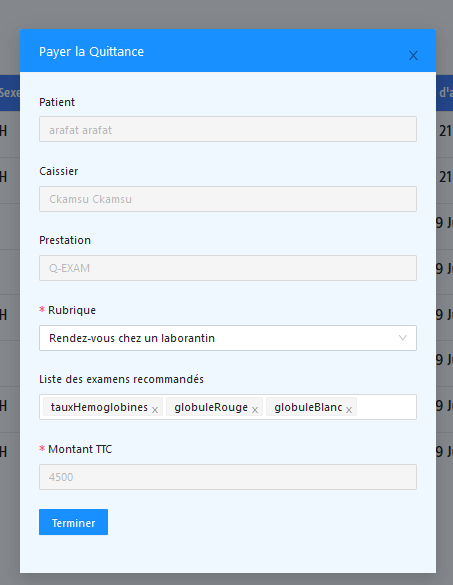




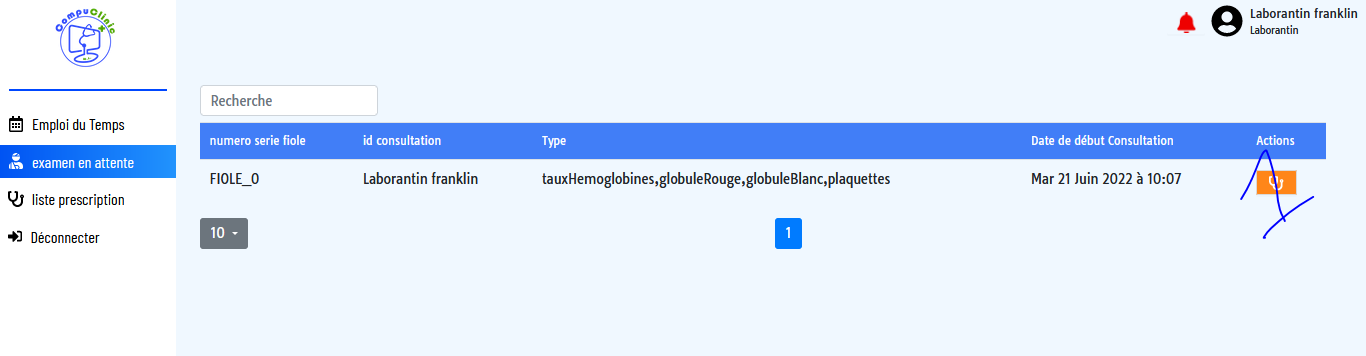
* Paiement de la quittance des examens a la caisse / liste des patients en attentes d’examen on le reconnait par la couleur marron pour les patients en attente de payement d examen et orange pour le payement des consultations



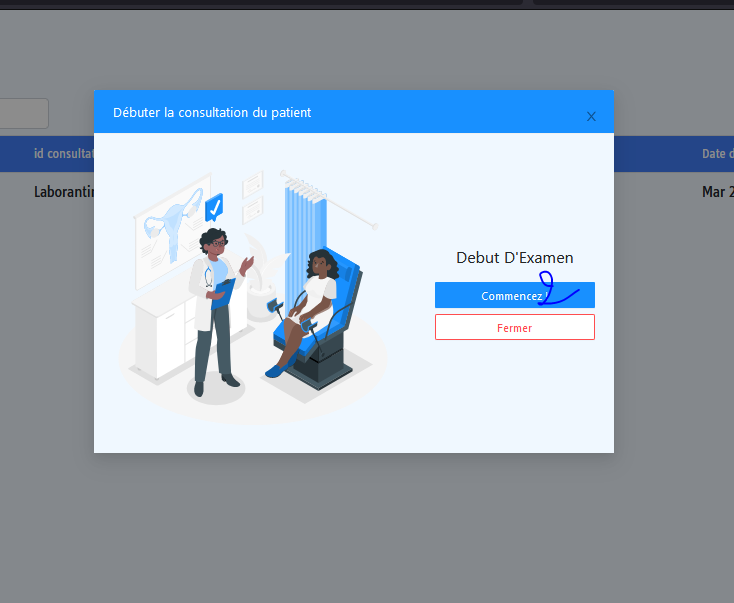
* Fenêtre de payement des quittances

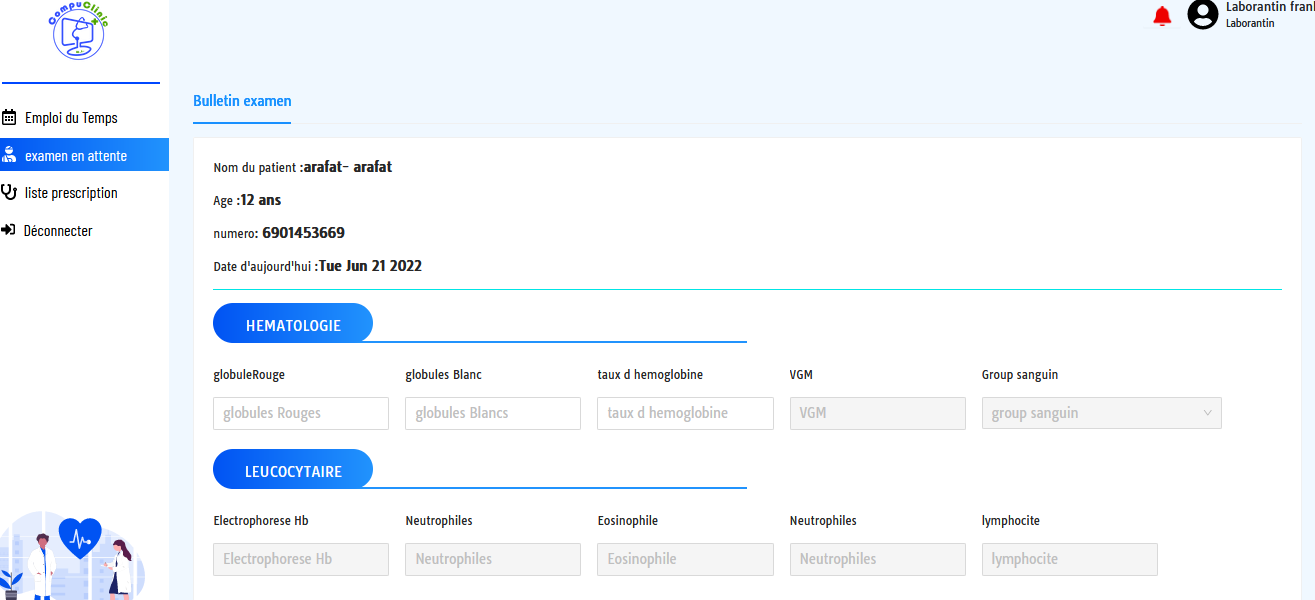


* Dashboard laborantin pour les examens



* Etablissement du bulletin d’examen





* Bulletin d’ examen final, vue du medecin



# CONCLUSION

En somme au terme de notre analyse nous pouvons dire l’ajout du module laborantin permet à COMPUCLINIC de faciliter les tâches liées aux activités de laboratoire tels que l’envoi de bulletins d’examen, l’envoi de tests à effectuer aux laborantins.

Cependant nous constatons que d’autres modules peuvent être ajouter à COMPUCLINIC dans le but de le rendre plus complet comme par exemple un module de gestion des cas d’urgence, de gestion de services transfusionnels et banques de sang des hôpitaux…