

Rapport de Stage

De

Mme Assia GHELIS

Association EURAFRIQUE

Développement d'une application de gestion de tâches

Soutenu le 19 septembre 2023

JURY

Président : METAIS Elisabeth (CNAM)

Membres : LAMMARI Ilhem (CNAM)

Remerciements

Je tiens à exprimer ma profonde gratitude pour le soutien inestimable que mon référant au niveau de l'association EurAfrique, Monsieur Dominique CONTARD m'a apporté tout au long de mon travail. Son encadrement attentif, ses conseils éclairés et son engagement constant ont grandement contribué à la réussite de ce projet.

Je tiens également à remercier chaleureusement Madame Elisabeth Metais, qui a su me guider avec patience et expertise, m'aidant ainsi à affiner mes idées et à progresser dans ma démarche.

Un grand merci s'adresse également à Madame Ilhem LAMMARI, membre du jury, pour sa précieuse contribution et ses retours constructifs qui ont enrichi mon travail et m'ont permis de le peaufiner davantage.

Enfin, je tiens à exprimer ma reconnaissance envers Monsieur Florian ROGER qui m'a apporté des conseils et du soutien, dont les conseils avisés et le soutien indéfectible ont été des éléments clés dans la réalisation de ce projet.

Je suis consciente que ce projet n'aurait pas été possible sans leur appui constant et leurs confiances en mes capacités, leur bienveillance m'a été une source d'inspiration pour moi.

Table des matières

1. Introduction	5
1.1. Présentation de l'Association EurAfrique	5
1.2. Contexte et Objectifs du Stage	5
2. Présentation des Technologies	7
2.1. Flask	7
2.2. Principales Caractéristiques et Avantages de Flask :	7
3. Travail Réalisé :	8
3.1. Développement de l'application de gestion des tâches avec Flask	8
3.2. Conclusion	12
4. Développement de mon application et résultat obtenu	13
4.1. Présentation de l'interface principale	13
4.1.1. Authentification des utilisateurs :	13
4.1.2. Implémentation des opérations ajouter, modifier, supprimer et lister :	14
5. Gestion de la base de données MySQL	21
5. 1. 2 Droits d'Accès et Limitation de Privilèges	21
Sauvegarde et Restauration :	21
5.2.1. Sauvegarde et Restauration :	22
5.3. Conclusion	23
6. Enseignements et Compétences Acquis	24
7. Conclusion Générale	25
8. Annexe	26.

Résumé

Mon stage au sein de l'Association Eurafrique a été une opportunité exceptionnelle d'acquérir une expérience pratique dans le domaine du développement d'applications et de la gestion de projets. En utilisant des outils tels que Flask, Python, HTML, CSS et MySQL, j'ai pu développer une application avec des fonctionnalités interactives et efficaces pour les applications web.

Cette expérience m'a permis d'explorer un autre aspect de la gestion d'entreprise et de contribuer activement à l'efficacité de l'équipe. J'ai acquis une expérience précieuse en développement d'applications web, en gestion de bases de données et en gestion de projets.

Ce stage m'a offert une opportunité unique de fusionner mes compétences académiques avec une expérience pratique, me préparant ainsi à relever avec confiance les défis du monde professionnel.

Mots clés : Flask, Python, HTML, CSS et MySQL

1.Introduction

1.1. Présentation de l'Association EurAfrique

L'association EurAfrique est une organisation à vocation sociale et culturelle, œuvrant pour le rapprochement et la coopération entre les continents européen et africain. Fondée en 2013, l'association a pour objectif de promouvoir l'échange culturel, l'éducation, le développement durable et la solidarité entre les pays des deux continents.

L'association EurAfrique a plusieurs objectifs, elle aspire à créer des liens culturels solides entre les peuples européens et africains. Elle s'efforce de mettre en place des projets éducatifs visant à améliorer l'accès à l'éducation et à encourager les échanges entre les étudiants et les enseignants des deux continents. Elle s'engage en faveur du développement durable en encourageant des pratiques respectueuses de l'environnement, de la biodiversité et des ressources naturelles. Des projets de sensibilisation, de reforestation et de préservation de l'écosystème sont souvent menés en partenariat avec des organisations locales.

L'association travaille également sur des initiatives de solidarité en faveur des populations les plus vulnérables. Elle mène des actions humanitaires, offre un soutien aux communautés marginalisées et participe à des programmes visant à améliorer les conditions de vie des populations défavorisées.

1.2. Contexte et Objectifs du Stage

Au cours de ce stage mon rôle a été de contribuer à la création d'une application afin de renforcer la présence en ligne de l'association et d'améliorer sa visibilité. Cette mission s'inscrit dans la volonté de l'association de moderniser ses moyens de communication et de faciliter l'accès à ses activités et ressources pour un public plus large, tout en respectant ses valeurs de coopération intercontinentale et de solidarité.

La mission qui m'a été confiée a été encadrée par Monsieur Dominique CONTARD, qui a joué un rôle central dans la définition des objectifs du projet et dans la supervision de mes activités. Mon travail sur les applications web a été complémentaire aux efforts de l'association pour promouvoir ses idéaux et ses actions à travers les continents, en ligne avec sa vision de renforcement des liens entre l'Europe et l'Afrique.

Contexte

Mon objectif principal en entamant ce stage était de développer une compréhension approfondie du processus de développement d'applications web, des meilleures pratiques en matière de conception et de programmation, ainsi que de la gestion de bases de données. J'ai initialement envisagé d'utiliser WordPress pour développer des interfaces, mais après avoir constaté que ses fonctionnalités étaient limitées pour répondre aux besoins complexes de l'association, j'ai rapidement identifié la nécessité de maîtriser des outils de développement plus polyvalents.

Mon choix pour cette entreprise était motivé par mon intérêt pour la technologie et mon désir d'acquérir une expérience pratique dans le domaine du développement.

Mon rôle au sein de l'équipe de développement était de contribuer à la création et à la personnalisation d'applications web, en mettant en œuvre des technologies telles que Python, Flask, HTML, CSS et MySQL.

Objectifs

Mon objectif principal était de développer une compréhension approfondie du processus de développement, d'applications, ainsi que la gestion de bases de données, et d'améliorer la gestion des tâches, la productivité, et d'assurer une meilleure collaboration et communication au sein d'une équipe.

Je me suis donc fixé les objectifs suivants :

Apprentissage du Développement Web Avancé :

Création d'applications web personnalisées et fonctionnelles. Je souhaitais acquérir des compétences en développement front-end (HTML, CSS) et en back-end (Python, Flask), me permettant ainsi de créer des solutions sur mesure pour les besoins spécifiques d'une entreprise.

Expérience en Gestion de Bases de Données :

Étant donné que les applications web nécessitent souvent une gestion de données efficace, mon objectif était de comprendre comment concevoir et manipuler des bases de données. Je voulais être capable de créer des schémas de bases de données adéquats, de mettre en place des requêtes complexes et de garantir la sécurité des données.

Immersion dans la Gestion de Projet :

Mon stage m'a également offert l'opportunité de participer à des projets de gestion d'entreprise. Mon objectif était de comprendre comment implémenter des solutions logicielles fonctionnelles, pour répondre aux besoins spécifiques des clients et comment collaborer au sein d'une équipe pour mener à bien ces projets.

2. Présentation des Technologies

2.1. Flask

Flask est un framework web puissant construit en Python. Il est conçu pour faciliter le développement rapide d'applications web en fournissant les outils essentiels nécessaires tout en restant flexible et extensible. Flask adopte une approche "micro", ce qui signifie qu'il fournit uniquement les composants de base nécessaires pour construire une application web, laissant aux développeurs la liberté de choisir les bibliothèques et les modules complémentaires à utiliser en fonction de leurs besoins.

2.2. Principales Caractéristiques et Avantages de Flask :

Légèreté : Flask est simple et léger. Il n'impose pas de structures rigides ni de conventions strictes, ce qui signifie que les développeurs ont la liberté de structurer leur application selon leurs préférences.

Flexibilité : Grâce à son approche modulaire, Flask permet d'ajouter des extensions et des bibliothèques tierces au besoin. Cela signifie qu'on peut choisir les composants spécifiques dont mon application a besoin, ce qui la rend plus légère et plus efficace.

Routing : Flask facilite la définition de routes pour les différentes parties de mon application. Les routes sont utilisées pour déterminer quel code doit être exécuté lorsque certaines URL sont atteintes.

Gestion des Templates : Flask intègre un moteur de template Jinja2 qui permet de générer des pages web dynamiques en incorporant des variables et des logiques dans les modèles HTML.

Intégration de Bases de Données : Bien qu'il ne vienne pas avec une couche de gestion de bases de données intégrée, Flask permet d'intégrer facilement des bases de données en utilisant des extensions comme Flask-SQLAlchemy pour travailler avec des bases de données SQL.

Sessions et Gestion de l'État : Flask prend en charge la gestion des sessions pour conserver l'état entre différentes requêtes, ce qui est essentiel pour la création d'applications interactives.

Extensibilité : Les extensions sont des bibliothèques tierces qui ajoutent des fonctionnalités spécifiques à Flask. Cela permet aux développeurs de personnaliser et d'étendre les capacités du framework en fonction de leurs besoins.

Déploiement Facile : Flask facilite le déploiement d'applications sur divers serveurs web, ce qui permet de mettre rapidement en ligne des applications fonctionnelles.

3. Travail Réalisé :

3.1. Développement de l'application de gestion des tâches avec Flask

Cette application représente une solution moderne et pratique pour gérer efficacement les tâches, les projets et les responsabilités au sein d'un environnement professionnel ou personnel. L'application de gestion des tâches que j'ai contribué à développer tire parti des fonctionnalités puissantes de Flask pour offrir une interface utilisateur conviviale et des fonctionnalités avancées de gestion de tâches.

Dans cette section, je vais vous présenter en détail les différentes étapes du développement de l'application, notamment la conception de la base de données, la création des fonctionnalités de gestion des tâches, l'implémentation de la sécurité et des autorisations, ainsi que d'autres aspects clés de mon projet.

Étape 1 : Configuration de l'environnement

1. Installation de Python : Après avoir installé Python sur mon système,
2. Création d'un environnement virtuel : J'ai utilisé **venv** pour créer un environnement virtuel isolé où j'ai installé les dépendances spécifiques à ce projet.

Étape 2 : Initialisation du projet Flask

1. Installation de Flask : Dans mon environnement virtuel, j'ai installé Flask en utilisant la commande : **pip install Flask**.
2. Création d'un dossier pour le projet : J'ai créé un dossier nommé "env" pour mon projet, où j'ai placé tous les fichiers liés à mon projet grâce à la commande : **py -m venv env**

3. Création d'un fichier d'application : J'ai créé un fichier nommé 'app.py' pour démarrer mon application Flask.

Dans app.py, j'ai importé Flask et créé une instance de l'application :

```
from flask import Flask
app = Flask(__name__)
```

4. J'ai configuré une clé secrète pour gérer les sessions et la sécurité avec **app.config**:

```
import os
app.secret_key = os.urandom(24)
#app.config['SECRET_KEY'] = 'my_secret_key_here'
db = SQLAlchemy(app)
bcrypt = Bcrypt(app)
```

Image 1 : Code de la création de la clé_secrète.

Étape 3 : Configuration de la base de données SQLAlchemy:

1. Installation de SQLAlchemy : SQLAlchemy est un outil populaire pour interagir avec les bases de données en Python. Je l'ai installé via : **pip install Flask-SQLAlchemy**

2. Configuration de la base de données : J'ai configuré la connexion à la base de données Eurafrigue dans mon fichier **app.py** :

```
app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql://admin:123@localhost/gestion_taches'
```

Image 2 : Code de configuration de la connexion à la base de données gestion_tâches.

3. Intégration de la Base de Données MySQL

J'ai opté pour MySQL en tant que système de gestion de base de données, en raison de sa réputation en matière de performances et de fiabilité. J'ai intégré une base de données MySQL pour stocker et gérer les informations de mon application. J'ai conçu le schéma de la base de données en fonction des besoins spécifiques de l'Association EurAfrique. L'ORM SQLAlchemy a été utilisé pour faciliter les interactions avec la base de données, assurant ainsi une gestion sécurisée des données.

Étape 5 : Création du modèle Task:

J'ai défini un modèle 'Task' en utilisant SQLAlchemy. Ce modèle représente une tâche avec un identifiant ('id') et un contenu ('content') :

```
class Task(db.Model):
    __tablename__ = 'task'
    id = db.Column(db.Integer, primary_key=True)
    nom = db.Column(db.String(255), nullable=False)
    description = db.Column(db.Text)
    attribue_a = db.Column(db.String(255), nullable=False)
    date_attribution = db.Column(db.Date)
    date_echeance = db.Column(db.Date)
    statut = db.Column(db.String(20))
```

Image 3 : Code de création de la classe Task.

Étape 6 : Création des formulaires TaskForm:

J'ai créé un formulaire 'TaskForm' à l'aide de Flask-WTF et WTForms. Ce formulaire permet aux utilisateurs d'ajouter ou de modifier une tâche :

```
class TaskForm(FlaskForm):
    nom = StringField('Nom', validators=[DataRequired()])
    description = TextAreaField('Description')
    attribue_a = StringField("Attribué à", validators=[DataRequired()])
    date_attribution = DateField("Date d'attribution", validators=[DataRequired()])
    date_echeance = DateField("Date d'échéance", validators=[DataRequired()])
    statut = SelectField("Statut", choices=[('En cours', 'En cours'), ('Terminée', 'Terminée'), ('En attente', 'En attente')])
```

Image 4 : Code de création de la Classe TaskForm.

Migration de la base de données : Lorsque j'ai modifié la structure de la base de données, j'ai utilisé les migrations pour assurer une transition. J'ai utilisé un outil comme Flask-Migrate pour générer les scripts de migration et mettre à jour la base de données sans perdre de données pour cela j'ai exécuté ces commandes depuis le terminal :

Installation de Migrate grâce à la commande : **pip install flask-Migrate**

Initialisation de la base de données avec la commande : **flask db init**

Migration de la base de la base de données et création de la table task avec la commande : **flask db migrate -m «create user table»**

Étape 7 : Gestion des Utilisateurs et des Autorisations : Interface Utilisateur

J'ai pris en charge la création et la gestion des comptes utilisateurs au sein des bases de données. Pour garantir un accès sécurisé et approprié, j'ai déployé des autorisations selon les besoins spécifiques de chaque utilisateur.

La gestion des utilisateurs et des autorisations a été mise en œuvre pour réguler l'accès aux différentes fonctionnalités du site. À cette fin, j'ai mis en place des mécanismes essentiels tels que l'inscription, la connexion et la réinitialisation des mots de passe.

Pour assurer une gestion précise des droits d'accès, j'ai instauré un système de rôles et de permissions. Cette approche a permis de définir des rôles correspondant aux différentes responsabilités des utilisateurs. Les rôles ont ensuite été associés à des permissions spécifiques, autorisant ainsi un contrôle fin sur les actions qu'un utilisateur peut entreprendre au sein de l'application comme suite :

1. Inscription et Connexion : J'ai mis en place un processus d'inscription pour permettre aux nouveaux utilisateurs de créer un compte. De même, j'ai élaboré un mécanisme de connexion sécurisé pour que les utilisateurs puissent accéder à leur compte en toute sécurité.

2. Réinitialisation de Mot de Passe : Pour offrir une expérience utilisateur fluide, j'ai implémenté un mécanisme de réinitialisation de mot de passe. Cela permet aux utilisateurs d'obtenir un nouveau mot de passe en cas d'oubli ou de besoin de réinitialisation.

3. Rôles et Permissions : J'ai défini différents rôles en fonction des responsabilités des utilisateurs. Ces rôles ont été assortis de permissions spécifiques, déterminant quelles actions peuvent être effectuées par chaque utilisateur. Par exemple, un rôle d'administrateur peut avoir des autorisations étendues par rapport à un utilisateur standard.

Grâce à cette mise en place, l'application a pu garantir un accès contrôlé et approprié aux fonctionnalités pour chaque utilisateur. Les mécanismes d'inscription, de connexion et de réinitialisation de mot de passe ont facilité l'expérience utilisateur, tandis que la définition de rôles et de permissions a permis une gestion fine des droits d'accès en fonction des rôles assignés aux utilisateurs.

Étape 8 : J'ai créé les routes pour les fonctionnalités : J'ai défini les routes qui permettront aux utilisateurs d'accéder aux différentes pages de l'application :

- `'/'` : Cette route affiche la liste des tâches existantes.
- `'/add_task'` : Cette route permet aux utilisateurs d'ajouter de nouvelles tâches en soumettant le formulaire.
- `'/edit_task/<int:task_id>'` : Cette route permet aux utilisateurs de modifier une tâche existante en soumettant le formulaire :

```
@app.route('/add_task', methods=['GET', 'POST'])
def add_task():
    form = TaskForm() # Créer une instance du formulaire
    if form.validate_on_submit():
        # Créer une nouvelle instance de la classe Task en utilisant les données du formulaire
        new_task = Task(
            nom=form.nom.data,
            description=form.description.data,
            attribue_a=form.attribue_a.data,
            date_attribution=form.date_attribution.data,
            date_echeance=form.date_echeance.data,
            statut=form.statut.data
        )
        db.session.add(new_task)
        db.session.commit()
        return redirect(url_for('index')) # Rediriger vers la liste des tâches après l'ajout

    return render_template('add_task.html', form=form)

@app.route('/edit_task/<int:task_id>', methods=['GET', 'POST'])
def edit_task(task_id):
    task = Task.query.get_or_404(task_id) # Récupérer la tâche à éditer depuis la base de données
    form = EditTaskForm(obj=task) # Créer un formulaire de modification en utilisant les données de la tâche

    if form.validate_on_submit():
        form.populate_obj(task) # Mise à jour des données de la tâche avec les données du formulaire
        db.session.commit() # Enregistrer les modifications dans la base de données
        return redirect(url_for('index')) # Rediriger vers la page principale

    return render_template('edit_task.html', form=form, task=task)

@app.route('/delete_task/<int:task_id>', methods=['GET'])
def delete_task(task_id):
    task = Task.query.get_or_404(task_id) # Récupérer la tâche à supprimer depuis la base de données
    db.session.delete(task) # Supprimer la tâche de la base de données
    db.session.commit() # Enregistrer les modifications dans la base de données
    return redirect(url_for('index')) # Rediriger vers la page principale
```

Image 5 : Code Python (création des différentes routes avec la définition des fonctions correspondantes).

Étape 9 : Utilisation de templates et Création des vues :

J'ai utilisé le moteur de templates de Flask pour rendre les pages HTML dynamiques en injectant des données.

J'ai développé les vues pour afficher les listes de tâches, les détails des tâches et les formulaires pour ajouter de nouvelles tâches.

Code pour afficher la liste des tâches :

```
<!DOCTYPE html>
<html>
<head>
  <title>Liste des Tâches</title>
  # Ajoutez le lien vers les styles de Bootstrap
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body>
  # Conteneur Bootstrap pour aligner le contenu
  <div class="container mt-4">
    # Titre de la page
    <h1 class="mb-4">Liste des Tâches</h1>
    # Liste de tâches sous forme de groupe
    <ul class="list-group">
      # Boucle à travers chaque tâche dans la liste de tâches
      {% for task in tasks %}
        # Élément de la liste pour chaque tâche
        <li class="list-group-item">
          # Titre de la tâche
          <h4>{{ task.content }}</h4>
          # Information sur la date d'affectation
          <p><strong>Date d'affectation :</strong> {{ task.assignment_date }}</p>
          # Information sur le destinataire de la tâche
          <p><strong>Assignée à :</strong> {{ task.assigned_to }}</p>
          # Actions possibles pour la tâche
          <p>
            Lien pour modifier la tâche
            <a href="{{ url_for('edit_task', task_id=task.id) }}" class="btn btn-primary btn-sm mr-2">Modifier</a>

            # Vérification de l'état de la tâche
            {% if task.status == "Terminée" %}
              # Badge indiquant que la tâche est terminée
              <span class="badge badge-success">Terminée</span>
            {% else %}
              # Badge indiquant que la tâche est en cours
              <span class="badge badge-warning">En cours</span>
            {% endif %}

          </p>
        </li>
      {% endfor %}
    </ul>
    Lien pour ajouter une nouvelle tâche
    <a href="{{ url_for('add_task') }}" class="btn btn-success mt-3">Ajouter une Tâche</a>
  </div>
</body>
</html>
```

Image 6 : Code pour afficher de la vue « Liste des tâches ».

Cette vue est destinée à être utilisée dans mon application Flask pour présenter les informations des tâches en utilisant le moteur de templates Jinja2.

Description de ce que fait ce code :

1. Dans la balise <head>, le lien vers les styles de Bootstrap est inclus. Cela permet de bénéficier des styles prédéfinis de Bootstrap pour rendre la page attrayante et réactive.
2. {{ task.content }}, {{ task.assignment_date }}, {{ task.assigned_to }} : Ces expressions sont du code Jinja2 qui affiche les valeurs correspondantes des attributs de chaque tâche dans la liste. Par exemple, {{ task.content }} affiche le contenu de la tâche, {{ task.assignment_date }} affiche la date d'affectation.
3. Dans la boucle {% for task in tasks %} ... {% endfor %}, chaque tâche de la liste tasks est parcourue. Pour chaque tâche, les détails sont affichés en utilisant les expressions Jinja2.
4. Le bouton **Modifier** est créé pour chaque tâche avec un lien vers la vue d'édition de la tâche. La couleur du badge de statut dépend de l'état de la tâche (Terminée ou En cours).

5. Le bouton **Ajouter une Tâche** crée un lien vers la vue permettant d'ajouter une nouvelle tâche.

```
<!DOCTYPE html>
<html>
<head>
  <title>Ajouter une tâche</title>
</head>
<body>
  # En-tête de la page
  <h1>Ajouter une tâche</h1>
  # Début du formulaire avec méthode POST
  <form method="POST">
    # Balises cachées pour la sécurité CSRF
    {{ form.hidden_tag() }}
    # Champ de saisie pour le contenu de la tâche
    <div class="form-group">
      {{ form.content.label }}
      {{ form.content(class="form-control") }}
    </div>
    # Champ de saisie pour le pôle
    <div class="form-group">
      {{ form.pole.label }}
      {{ form.pole(class="form-control") }}
    </div>
    # Champ de saisie pour la date d'assignation
    <div class="form-group">
      {{ form.assignment_date.label }}
      {{ form.assignment_date(class="form-control") }}
    </div>
    # Champ de saisie pour le destinataire de la tâche
    <div class="form-group">
      {{ form.assigned_to.label }}
      {{ form.assigned_to(class="form-control") }}
    </div>
    # Champ de saisie pour le statut de la tâche
    <div class="form-group">
      {{ form.status.label }}
      {{ form.status(class="form-control") }}
    </div>
    # Bouton de soumission du formulaire
    <button type="submit" class="btn btn-primary">Ajouter</button>
  </form>
</body>
</html>
```

Image 7 : Partie du code de création de la vue « Ajouter une tâches ».

En résumé, j'ai créé cette dernière est pour afficher une liste de tâches, leurs détails et leurs statuts. J'ai utilisé des styles Bootstrap pour une présentation visuelle attrayante et des expressions Jinja2 pour injecter dynamiquement les données de chaque tâche dans la page web.

3.2. Conclusion

Dans cette section, j'ai abordé de manière générale toutes les étapes nécessaires et obligatoires que j'ai mises en œuvre en présentant des exemples de réalisation d'application Python avec Flask, combinés à l'utilisation de HTML et de CSS. Dans la section suivante, je présenterai le développement complet de mon application ainsi que les résultats obtenus en matière de gestion de tâches pour une entreprise.

4. Développement de mon application et résultat obtenu

Le développement est un processus itératif. J'ai commencé par les fonctionnalités essentielles, testé leur fonctionnement et ajouté ensuite des fonctionnalités supplémentaires au fur et à mesure.

4.1. Présentation de l'interface principale

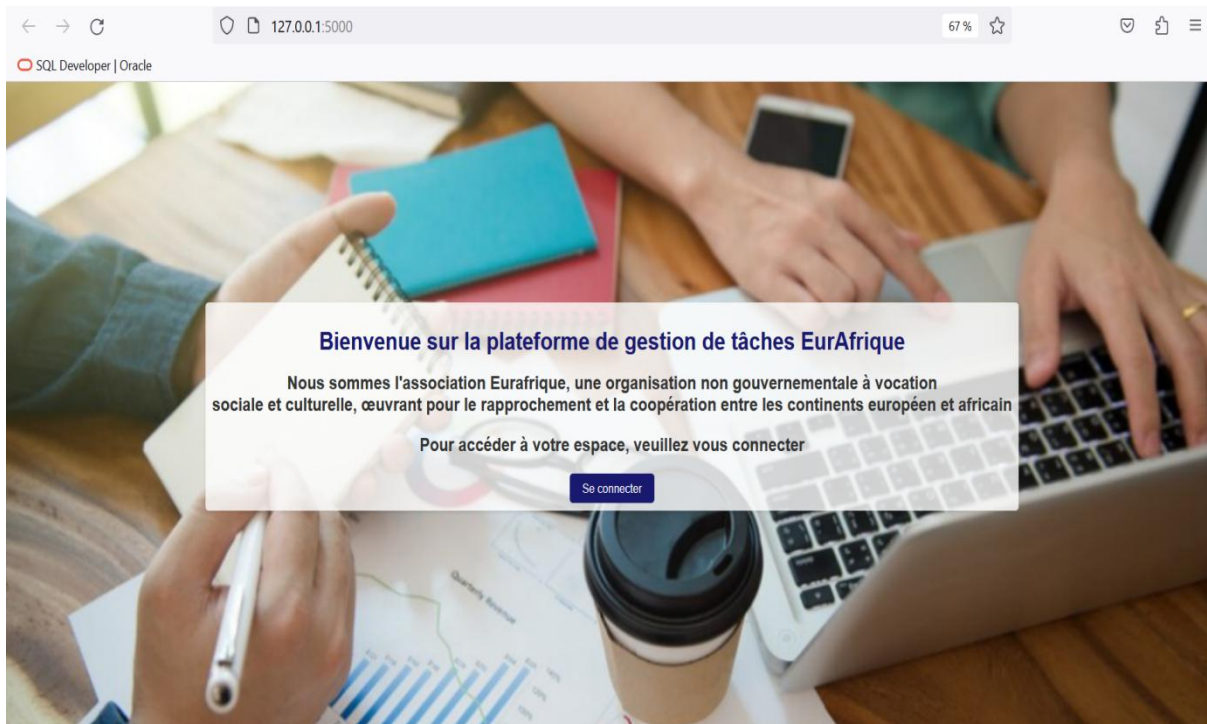


Image 8 : Interface principale de l'application.

4.1.1. Authentification des utilisateurs :



Image 9 : Code de création de la connexion.

Image 10 : Résultat du code de connexion.

Après l'inscription d'un nouveau membre d'équipe, ce dernier sera directement enregistré sur la base de données gestion_tâches que j'ai créé précédemment dans la table user :

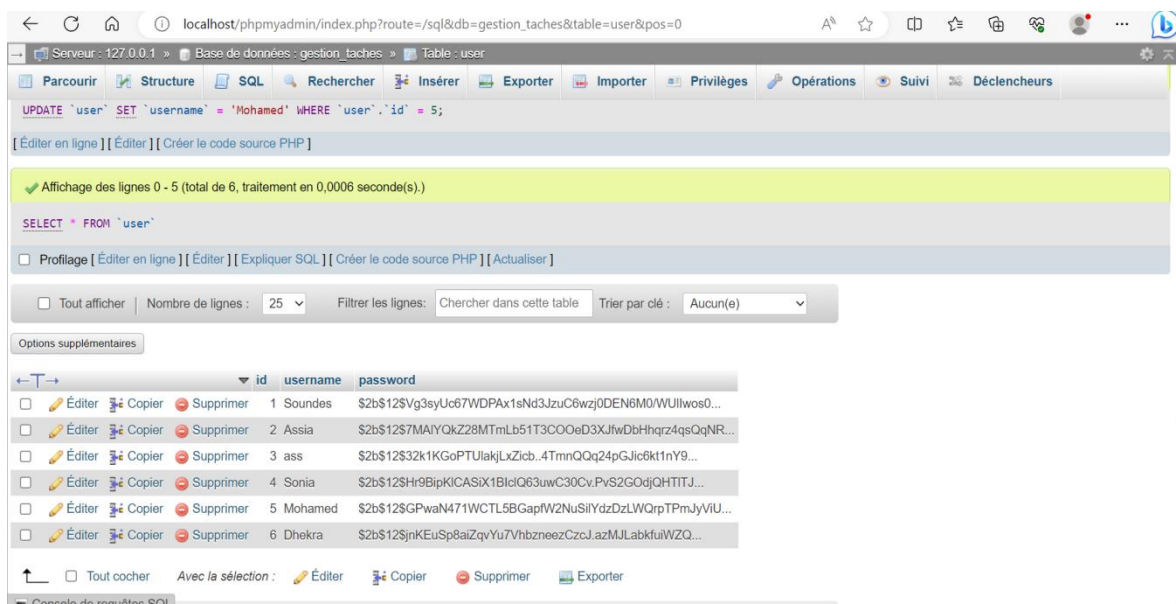


Image 11 : Structure de la table task avec hachage des mots de passe.

Comme vous constatez on remarque bien la mesure de protection que j'ai prise pour protéger le comptes des utilisateurs, l'opération de hachage des mots de passe des utilisateurs afin d'assurer leur sécurité a bien été prise en compte.

4.1.2. Implémentation des opérations ajouter, modifier, supprimer et lister :

Implémentation de ces opérations dans mon application de gestion de tâches avec Flask précédemment présenté a permis :

Création de la vue "Ajouter une tâche" grâce à un formulaire qui permet aux nouveaux utilisateurs de créer de nouvelles tâches en fournissant les détails nécessaires. Lorsque le formulaire est soumis, une nouvelle tâche est créée et ajoutée à la base de données.

Le formulaire de saisie doit renseigner à travers les différents champs les informations suivantes :

- Nom de la tâche
- Description de la tâche
- Personne à laquelle la tâche est attribuée
- Date d'attribution
- Date d'échéance
- Statut de la tâche (En cours, Terminée, En attente)

Une fois qu'un utilisateur de l'équipe de l'association EurAfrique remplit le formulaire, les données sont soumises au serveur pour les traiter. Nous avons utilisé Flask-WTF pour gérer le formulaire, valider les données entrées par l'utilisateur et effectuer des opérations sur la base de données.

Intégration de Bootstrap pour l'Interface Utilisateur

J'ai utilisé Bootstrap pour améliorer l'aspect visuel et la convivialité du formulaire. Cela a permis de rendre l'interface utilisateur plus agréable et réactive, tout en offrant des styles et des composants prêts à l'emploi.

Implémentation

```
# Modèle de tâche
class Task(db.Model):
    # Définition des champs de la table Task dans la base de données
    id = db.Column(db.Integer, primary_key=True)
    nom = db.Column(db.String(255), nullable=False)
    description = db.Column(db.Text)
    attribue_a = db.Column(db.String(255), nullable=False)
    date_attribution = db.Column(db.Date)
    date_echeance = db.Column(db.Date)
    statut = db.Column(db.String(20))
..
```

Image 12 : Partie du Code de la création de la calsse Task.

Ajouter une tâche

Nom:

Développement du site internet de l'association EurAfrique

Description:

Analyse des besoins. Conception, modélisation du système. Développement du site.
Outil utilisé Symphony

Attribué à:

Leon

Date d'attribution:

21 / 06 / 2023

Date d'échéance:

16 / 08 / 2023

Statut:

Terminée

Ajouter

[Retour à la liste des tâches](#)

Image 13 : Formulaire d'enregistrement de nouvelle tâche.

Développement du site internet de l'association EurAfrique

Description : Analyse des besoins. Conception, modélisation du système.
Développement du site. Outil utilisé Symfony

Attribué à : Leon

Date d'attribution : 2023-06-21

Date d'échéance : 2023-08-16

Statut : Terminée

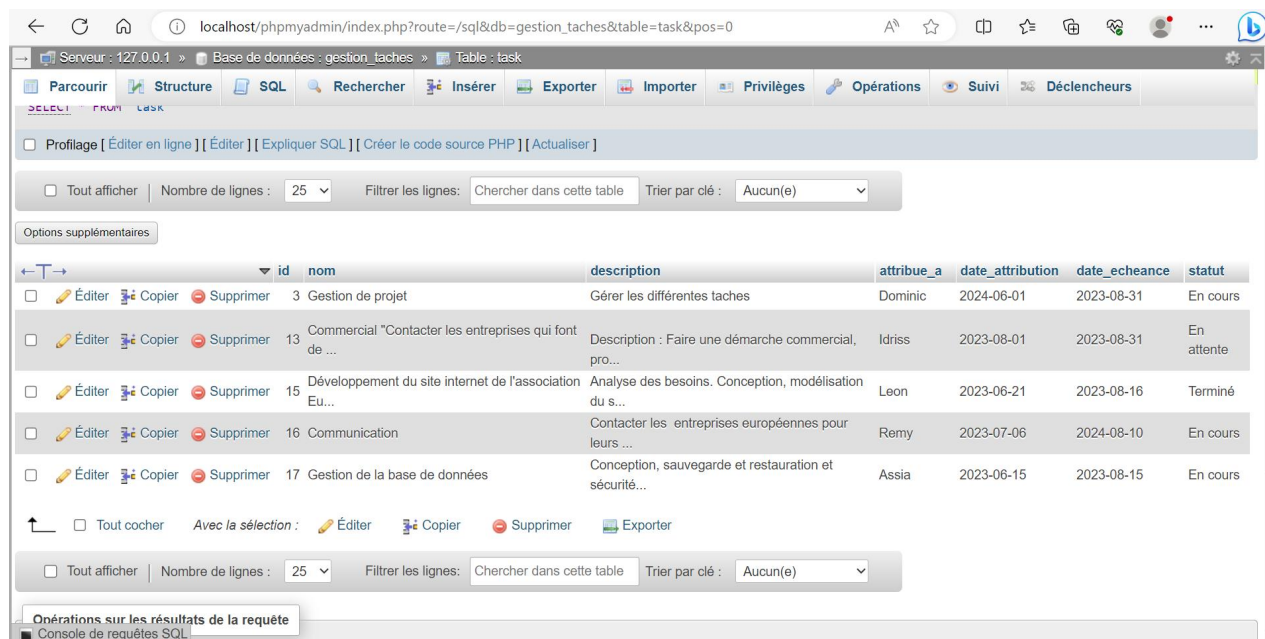
Modifier Supprimer

Image 14 : Résultat de l'ajout d'une nouvelle tâche.

Enregistrement dans la Base de Données

Lorsqu'un utilisateur soumet le formulaire, les données sont traitées côté serveur. Je crée une instance de la classe "Task" correspondant à la nouvelle tâche avec les données fournies par l'utilisateur, puis j'ajoute cette instance à la session SQLAlchemy. Enfin, je valide la session et j'enregistre les modifications dans la base de données.

Cette fonctionnalité a un avantage pour les utilisateurs de pouvoir rapidement et facilement entrer de nouvelles tâches, contribuant ainsi à une meilleure gestion des tâches au sein d'équipe de l'association EurAfrique.



	id	nom	description	attribue_a	date_attribution	date_echeance	statut
<input type="checkbox"/>	3	Gestion de projet	Gérer les différentes tâches	Dominic	2024-06-01	2023-08-31	En cours
<input type="checkbox"/>	13	Commercial "Contacter les entreprises qui font de ...	Description : Faire une démarche commercial, pro...	Idriss	2023-08-01	2023-08-31	En attente
<input type="checkbox"/>	15	Développement du site internet de l'association Eu...	Analyse des besoins. Conception, modélisation du s...	Leon	2023-06-21	2023-08-16	Terminé
<input type="checkbox"/>	16	Communication	Contacteur les entreprises européennes pour leurs ...	Remy	2023-07-06	2024-08-10	En cours
<input type="checkbox"/>	17	Gestion de la base de données	Conception, sauvegarde et restauration et sécurité...	Assia	2023-06-15	2023-08-15	En cours

Image 15 : Vu de la liste des projets dans la base de données gestion_tâches.

Migration de la Base de Données

Pour que notre application puisse fonctionner avec la nouvelle table "Task", nous avons effectué une migration de la base de données à l'aide de Flask-Migrate. Cela nous a permis de créer la structure de la table "task" dans la base de données avec les colonnes appropriées pour stocker les informations des tâches.

Commercial "Contacter les entreprises qui font de la torrification"

Description : Description : Faire une démarche commercial, proposer le produit "café" avec une formule d'intéressement

Attribué à : Idriss

Date d'attribution : 2023-08-01

Date d'échéance : 2023-08-31

Statut : **En cours**

[Modifier](#) [Supprimer](#)

Image 16 : résultat obtenu avec la migration de la base de données gestion_tâches avec la nouvelle table task.

La fonctionnalité d'ajout de tâche de notre application de gestion de tâches offre aux membres d'équipe de l'association EurAfrique, une manière simple et conviviale d'entrer de nouvelles tâches dans le système. En utilisant le formulaire, les utilisateurs peuvent spécifier les détails de la tâche, tels que le nom, la description, l'attribution, les dates et le statut. Grâce à l'intégration de Bootstrap et à la validation des données côté serveur, nous nous assurons que les informations entrées sont correctes et que les tâches sont enregistrées de manière précise dans la base de données.

La vue "Liste des tâches" : affiche toutes les tâches existantes avec leurs détails. Cela correspond à l'opération de lecture, où les utilisateurs peuvent voir les informations des tâches stockées dans la base de données.



Image 17 : Représentation de la Vue liste des tâches.

La vue "Modifier une tâche" : permet aux utilisateurs de mettre à jour les informations d'une tâche existante. Les champs affichés dans le formulaire sont pré-rempli.

Modifier une tâche

Nom Description Attribué à Date d'attribution Date d'échéance Statut

[Retour à la liste des tâches](#)

Modifier une tâche

Nom Description Attribué à Date d'attribution Date d'échéance Statut

[Retour à la liste des tâches](#)

Image 18 : Formulaire de mise à jour d'une tâche.

	id	nom	description	attribue_a	date_attribution	date_echeance	statut
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	3	Gestion de projet	Gérer les différentes tâches	Dominic	2024-06-01	2023-08-31	En cours
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	5	Développement de fonctionnalités :	Créer de nouvelles fonctionnalités pour une applic...	Leon	2023-06-24	2023-08-31	Terminée
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	7	Tests unitaires et d'intégration :	Écrire et exécuter des tests pour vérifier le bon ...	Alexandre	2023-08-23	2023-08-25	En attente
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	8	Developpement du site internet de l'association Sa...	Developpement d'une interface avec plusieurs fon...	Assia	2023-08-01	2023-08-10	Terminée
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	11	Maintenance du système :	Maintenir le site	Djamel	2023-06-16	2023-08-09	En attente

Image 19 : vue du résultat de l'opération de mise à jour au niveau de la table task.

Avec les données actuelles de la tâche, et lorsque le formulaire est soumis, les informations de la tâche sont modifiées dans la base de données.

La vue "Supprimer" : j'ai implémenté cette vue en ajoutant un bouton "Supprimer" à côté de chaque tâche dans la vue "Liste des tâches". Lorsque ce bouton est cliqué, la tâche correspondante est supprimée de la base de données.

En résumé toutes les étapes que j'ai entreprises pour l'implémentation de ces différentes vues permettent aux utilisateurs de créer, lire, mettre à jour et éventuellement supprimer des tâches dans ma base données ont été assurées grâce aux différentes étapes.

1. Configuration initiale : j'ai créé un environnement de développement et installé Flask, ainsi que les autres bibliothèques nécessaires telles que Flask-Bcrypt, Flask-SQLAlchemy et Flask-WTF.
2. Création de la base de données : j'ai configuré une base de données MySQL pour stocker les informations des utilisateurs et des tâches.
3. Modèles de données j'ai créé défini deux modèles SQLAlchemy pour les utilisateurs et les tâches. Le modèle 'User' gère les informations d'authentification et le modèle 'Task' stocke les détails des tâches.
4. Formulaires : j'ai utilisé Flask-WTF pour créer des formulaires de saisie pour l'inscription et la création de tâches. Les formulaires incluent des champs tels que le nom, la description, la date d'attribution.
5. Authentification et autorisation : j'ai implémenté un système d'authentification en utilisant Flask-Login et Flask-Bcrypt. Les utilisateurs peuvent s'inscrire, se connecter et se déconnecter. Les pages sont protégées pour assurer que seuls les utilisateurs authentifiés peuvent y accéder.
6. Pages de routes : j'ai créé des routes pour différentes pages, telles que la page d'accueil, la page de connexion, la page d'inscription, la page de profil et la liste des tâches. Nous avons géré les redirections en fonction de l'état de connexion.
7. Affichage des tâches : j'ai affiché la liste des tâches sur la page d'accueil ('index2.html') en utilisant Jinja2 pour itérer à travers les tâches et les afficher dans un format structuré.

8. Styles et présentation : j'ai amélioré l'apparence de l'application en utilisant du CSS pour styliser les différentes pages. J'ai centré les éléments, ajouté des bordures et des ombres pour un look professionnel.

9. Améliorations de l'affichage : j'ai personnalisé les styles des pages d'inscription et de connexion pour qu'elles soient plus esthétiques, tout en maintenant une mise en page claire et centrée.

Ces étapes résument le processus que j'ai suivi pour développer mon application de gestion de tâches en utilisant Flask.

5. Gestion de la base de données MySQL

MySQL, est un système de gestion de bases de données relationnelles open-source très réputé, il joue un rôle essentiel dans le contexte des bases de données en permet le stockage, la gestion et la récupération efficace des données. Dans ce qui suit, je vous présente un aperçu détaillé de son rôle dans mon projet de stage, où j'ai exploré des aspects essentiels tels que la sauvegarde, la restauration et la sécurisation des droits d'accès des utilisateurs.

5.1. Caractéristiques Clés de MySQL

MySQL présente plusieurs caractéristiques clés qui en font un choix privilégié pour le stockage et la gestion de données :

Sécurité : MySQL met à disposition des mécanismes de sécurité robustes pour protéger les données stockées. La possibilité de définir des permissions spécifiques pour les utilisateurs et les rôles permet de restreindre l'accès, les modifications ou les suppressions de données uniquement aux personnes autorisées.

Performance : La conception optimisée de MySQL lui permet de gérer efficacement d'importantes quantités de données. L'utilisation d'index pour accélérer les opérations de recherche et de récupération, ainsi que d'optimisations internes, améliore considérablement les temps de réponse des requêtes.

Open-Source : L'une des caractéristiques notables de MySQL réside dans sa nature open-source, offrant la possibilité à quiconque d'accéder à son code source et de le modifier selon les besoins.

5.2. Mise en application de quelques performances dans mon application :

5. 1. 2 Droits d'Accès et Limitation de Privilèges

La sécurité étant primordiale, j'ai mis en œuvre des mesures visant à restreindre l'accès à la base de données et à ses fonctionnalités aux seuls utilisateurs autorisés. En accord avec les meilleures pratiques, j'ai attribué des rôles et des permissions spécifiques aux utilisateurs en fonction de leurs responsabilités et de leurs besoins.

Cette approche a permis de garantir que chaque utilisateur a accès uniquement aux fonctionnalités nécessaires pour accomplir ses tâches, tout en limitant les privilèges.

Sauvegarde et Restauration :

Pour garantir l'intégrité de mes données et leur récupération en cas de situations critiques, j'ai mis en place des processus de sauvegarde réguliers pour la base de données. Ces sauvegardes, effectuées à intervalles préétablis, ont été stockées sur des serveurs sécurisés distincts, minimisant ainsi le risque de perte de données en cas d'incidents tels que les pannes matérielles, les erreurs humaines ou les cyberattaques.

```

33699@LAPTOP-HPFNU1TR c:\xampp
# Mysql -u admin -p123
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 967
Server version: 10.4.28-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use gestion_taches
Database changed
MariaDB [gestion_taches]> GRANT GRANT INSERT, UPDATE, DELETE ON ma_table TO mon_role;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'INSERT, UPDATE,
DELETE ON ma_table TO mon_role' at line 1
MariaDB [gestion_taches]> GRANT INSERT, UPDATE, DELETE ON task TO dominic;
Query OK, 0 rows affected (0.028 sec)

MariaDB [gestion_taches]>

```

Image 20 : Partie du code création de la clé_secrète

```

MariaDB [gestion_taches]> GRANT INSERT, UPDATE ON task TO Djamel;
Query OK, 0 rows affected (0.004 sec)

```

Image 21 : Partie du code création de la clé_secrète

```

MariaDB [gestion_taches]> CREATE USER 'Frederic'@'localhost' IDENTIFIED BY 'L31';
Query OK, 0 rows affected (0.009 sec)

```

Image 22 : Partie du code création de la clé_secrète

J'ai créé un nouvel utilisateur nommé "Frederic" qui peut se connecter à la base de données à partir de l'emplacement "localhost" en utilisant le mot de passe "L31".

Les utilisateurs administrateurs ont ainsi bénéficié d'un accès étendu aux fonctionnalités avancées, tandis que les utilisateurs standard ont été restreints aux actions essentielles.

5.2.1. Sauvegarde et Restauration :

Pour garantir l'intégrité de mes données et leur récupération en cas de situations critiques, j'ai mis en place des processus de sauvegarde réguliers pour la base de données. Ces sauvegardes, effectuées à intervalles préétablis, ont été stockées sur des serveurs sécurisés distincts, minimisant ainsi le risque de perte de données en cas d'incidents tels que les pannes matérielles, les erreurs humaines ou les cyberattaques.

```

C:\xampp\mysql\bin>mysqldump -u admin -p123 gestion_taches > "C:\Users\33699\Documents\Projet fin d'etude\gestion_taches_sauvegarde.sql"

```

Ce PC > Documents > Projet fin d'etude				Rechercher dans : Projet fin d'etude
	Nom	Modifié le	Type	
	Base_de_donnees	20/08/2023 17:44	Dossier de fichiers	
	gestion_taches_sauvegarde.sql	20/08/2023 19:39	Fichier source SQL	

De plus, j'ai veillé à tester et documenter les procédures de restauration des données à partir de ces sauvegardes. Ces efforts ont été entrepris dans le but d'assurer une reprise rapide et efficace des données en cas de besoin.

5.3. Conclusion

En somme, MySQL se positionne comme une solution robuste pour le stockage, la gestion et la récupération de données. Mes actions dans le cadre de la gestion de la base de données ont contribué à assurer la stabilité, la sécurité et la disponibilité des données pour l'application développée pendant mon stage. La mise en place de procédures de sauvegarde et de restauration régulières, associées à une gestion rigoureuse des droits d'accès, ont conjugué leurs efforts pour créer une base de données fiable et sécurisée, soutenant ainsi le succès global de mon projet.

Suivi de Progrès et de Performances : L'application permet également de suivre le progrès individuel et de l'équipe en matière de réalisations. Les tâches complétées sont visibles, ce qui peut servir de référence pour évaluer les performances et la progression dans la réalisation des objectifs.

Application de Concepts Techniques : À travers le développement de cette application, j'ai appliqué des concepts techniques clés tels que la création de modèles de données, l'interaction avec une base de données, la manipulation de formulaires et la création de routes pour l'application web. Cette expérience pratique renforce ma compréhension des technologies et leur application dans des projets réels.

En fin, mon application a le potentiel d'améliorer la gestion de tâches et de projets, la productivité, la collaboration et la communication au sein d'une équipe. De plus, elle m'a permis d'acquérir des compétences techniques précieuses en développement web, ce qui est un aspect important de ma formation académique en informatique.

6. Enseignements et Compétences Acquis

6.1. Enseignement

- 6.1.1. Gestion de Tâches et de Projets :** Mon application offre un moyen efficace de gérer les tâches quotidiennes et les projets plus larges. Les utilisateurs peuvent ajouter, modifier et supprimer des tâches, ainsi que les marquer comme complétées ou non complétées. Cela peut être particulièrement utile pour suivre les étapes de progression, organiser le travail d'équipe et prioriser les activités.
- 6.1.2. Productivité Personnelle et d'Équipe :** En permettant aux utilisateurs de suivre leurs tâches et de les marquer comme complétées, l'application favorise une meilleure gestion du temps et de la productivité. Cela peut être très pratique pour les personnes qui ont plusieurs tâches à accomplir et qui veulent garder une vue d'ensemble de leur travail.
- 6.1.3. Collaboration et Communication :** L'application peut être utilisée dans des contextes de travail d'équipe où plusieurs membres doivent collaborer sur des projets. En partageant les tâches et leur état de complétion, les membres de l'équipe peuvent mieux coordonner leurs efforts et communiquer efficacement sur l'état des tâches.

6.2. Compétences Acquis

La réalisation de ce projet de gestion des tâches en utilisant Flask a été une expérience enrichissante, elle m'a permis d'acquérir des enseignements précieux et de développer un éventail diversifié de compétences. J'ai été initié à la configuration d'un environnement de développement solide en utilisant Python et Flask. Les manipulations des routes, la création de vues dynamiques avec le moteur de templates de Flask, ainsi que la gestion des interactions entre les pages m'ont solidement ancré dans les principes fondamentaux du développement web.

L'intégration de la base de données via SQLAlchemy m'a permis de comprendre comment organiser et stocker les données de manière efficace. Les opérations (Ajouter, modifier, supprimer, lister) pour gérer les tâches ont concrétisé ces concepts dans des applications concrètes.

La mise en place des fonctionnalités de migration m'a fait comprendre l'importance de gérer les évolutions structurelles dans la base de données tout en maintenant l'intégrité des données existantes. Cela a renforcé ma capacité à anticiper et à gérer les futurs changements de l'application.

En travaillant sur l'interface utilisateur, j'ai perfectionné mes compétences en conception front-end en utilisant les styles de Bootstrap et les fonctionnalités de formulaire de Flask-WTF. J'ai appris à créer des pages web attrayantes et conviviales, offrant une expérience utilisateur engageante.

Dans l'ensemble, ce projet m'a apporté des enseignements approfondis et des compétences pratiques en matière de programmation web. J'ai acquis une compréhension solide de la structure des applications web, de la gestion des données, de la manipulation des bases de données et de la création d'interfaces interactives. Ces enseignements et compétences auront un impact significatif sur mes projets futurs et ma croissance en tant que développeur web.

7. Conclusion Générale

Mon expérience de stage au sein de l'Association Eurafrique a été incroyablement formatrice, m'offrant l'occasion d'explorer et de maîtriser un éventail de technologies clés pour le développement d'applications web et la gestion de projets. Au cours de ce stage, j'ai eu l'opportunité de mettre en pratique mes compétences en utilisant Flask, Python, HTML, CSS et MySQL pour créer des fonctionnalités innovantes.

L'utilisation de Flask en tant que framework de développement m'a conféré l'agilité nécessaire pour créer des applications web interactives à partir de zéro. Mon expertise en HTML et CSS m'a permis de concevoir des interfaces utilisateur conviviales, optimisant ainsi l'expérience des utilisateurs.

Mon immersion dans MySQL m'a offert une compréhension approfondie de la gestion des bases de données, me permettant de concevoir des schémas de données efficaces et d'exécuter des manipulations précises. Cette compétence a été vitale pour assurer le stockage et la récupération fiables des informations.

L'intégration de cette application a facilité la planification et le suivi des projets, tout en équilibrant judicieusement la répartition des tâches. En somme, mon stage au sein de l'Association Eurafrique m'a doté de compétences précieuses en développement d'applications web et en gestion de projets. Cette expérience a consolidé ma préparation en vue de ma carrière future dans le domaine de l'informatique et du développement.

8. Annexes

Présentation d'une partie du code de mon application

```
# Import des modules et packages nécessaires pour le fonctionnement de l'application

from flask import Flask, render_template, request, redirect, url_for, flash

# Le module Flask gère l'application web, les rendus de templates, les redirections et les messages flash.

from flask_wtf import FlaskForm
from wtforms import StringField, TextAreaField, DateField, SelectField
from wtforms.validators import DataRequired

# Les modules FlaskForm et les classes StringField, TextAreaField, DateField, SelectField, DataRequired
# sont utilisés pour créer des formulaires dans l'application, définissant les champs et les validations.

from flask_bcrypt import Bcrypt

# Le module Flask-Bcrypt facilite le hachage sécurisé des mots de passe.

from flask_sqlalchemy import SQLAlchemy

# Le module SQLAlchemy permet d'interagir avec la base de données de manière orientée objet.

from flask_login import LoginManager, login_user, current_user, login_required, \
logout_user, UserMixin

# Le module Flask-Login gère l'authentification des utilisateurs, la gestion de session et les décorateurs pour restreindre l'accès.
# La classe UserMixin fournit des méthodes utiles pour la gestion des utilisateurs.

import os

# Le module os fournit des fonctions pour interagir avec le système d'exploitation,
# comme la gestion de chemins de fichiers et dossiers.

# Import des modules et packages nécessaires
from flask import Flask, render_template, request, redirect, url_for, flash
from flask_wtf import FlaskForm
from wtforms import StringField, TextAreaField, DateField, SelectField
from wtforms.validators import DataRequired
from flask_bcrypt import Bcrypt
from flask_sqlalchemy import SQLAlchemy
from flask_login import LoginManager, login_user, current_user, login_required, \
logout_user, UserMixin
import os

# Initialisation de l'application Flask
app = Flask(__name__)
app.secret_key = os.urandom(24) # Clé secrète utilisée pour sécuriser les sessions
app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql://admin:123@localhost/gestion_taches' # Configuration de la base de données
db = SQLAlchemy(app) # Initialisation de SQLAlchemy pour interagir avec la base de données
bcrypt = Bcrypt(app) # Initialisation de Bcrypt pour le hachage des mots de passe
login_manager = LoginManager(app) # Initialisation de Flask-Login pour gérer l'authentification
login_manager.login_view = 'login' # Vue vers laquelle rediriger en cas d'utilisateur non connecté

# Modèle d'utilisateur
class User(db.Model, UserMixin):
    # Définition des champs de la table User dans la base de données
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(100), unique=True, nullable=False)
    password = db.Column(db.String(100), nullable=False)

# Modèle de tâche
class Task(db.Model):
    # Définition des champs de la table Task dans la base de données
    id = db.Column(db.Integer, primary_key=True)
    nom = db.Column(db.String(255), nullable=False)
    description = db.Column(db.Text)
    attribue_a = db.Column(db.String(255), nullable=False)
    date_attribution = db.Column(db.Date)
    date_echeance = db.Column(db.Date)
    statut = db.Column(db.String(20))

# ...

# Route pour se déconnecter
@app.route('/logout')
@login_required
def logout():
    logout_user() # Déconnexion de l'utilisateur actuel
    return redirect(url_for('index2')) # Redirection vers la page principale

# ...
```

```

# Route pour ajouter une nouvelle tâche
@app.route('/add_task', methods=['GET', 'POST'])
def add_task():
    form = TaskForm() # Création d'une instance du formulaire
    if form.validate_on_submit():
        # Création d'une nouvelle instance de Task avec les données du formulaire
        new_task = Task(
            nom=form.nom.data,
            description=form.description.data,
            attribue_a=form.attribue_a.data,
            date_attribution=form.date_attribution.data,
            date_echeance=form.date_echeance.data,
            statut=form.statut.data
        )
        db.session.add(new_task)
        db.session.commit()

        return redirect(url_for('index2')) # Redirection vers la liste des tâches après l'ajout

    return render_template('add_task.html', form=form)

# ...

if __name__ == '__main__':
    app.run(debug=True) # Lancement de l'application en mode de débogage

```

```

<!DOCTYPE html>
<html>
<head>
    <title>Liste des tâches</title>
    <style>
        /* Style global pour le corps de la page */
        body {
            font-family: Arial, sans-serif;
            background-color: #f8f9fa;
            margin: 0;
            padding: 0;
        }

        /* Style pour le conteneur principal */
        .container {
            max-width: 800px;
            margin: 0 auto;
            padding: 20px;
            background-color: #ffffff;
            border-radius: 5px;
            box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
        }

        /* Style pour le titre principal */
        h1 {
            color: midnightblue;
            text-align: center;
            margin-bottom: 20px;
        }

        /* Style pour la liste non ordonnée */
        ul {
            list-style-type: none;
            padding: 0;
        }

        /* Style pour chaque élément de la liste */
        li {
            padding: 20px;
            border: 1px solid #ccc;

```

```

        margin-bottom: 10px;
        background-color: #f5f5f5;
    }

    /* Style pour le titre de chaque tâche */
    h2 {
        color: blue;
        margin: 0 0 10px;
    }

    /* Style pour les sous-titres */
    h3 {
        color: #666;
        margin: 0 0 5px;
    }

    /* Style pour les liens */
    a {
        text-decoration: none;
        color: midnightblue;
        margin-right: 10px;
    }

    /* Style pour les liens survolés */
    a:hover {
        text-decoration: underline;
    }

    /* Style pour le lien "Ajouter une tâche" */
    .add-link {
        display: block;
        margin-top: 20px;
        font-weight: bold;
        text-align: center;
    }

    /* Styles pour les différentes classes de statut */
    .en-cours {
        color: orange;
    }

    .termine {
        color: green;
    }

    .en-attente {
        color: red;
    }
}
</style>
</head>
<body>
    <!-- Contenu principal -->
    <div class="container">
        <!-- Titre principal -->
        <h1>Liste des tâches</h1>
        <!-- Liste des tâches -->
        <ul>
            {% for task in tasks %}
            <!-- Élément de liste pour chaque tâche -->
            <li>
                <!-- Titre de la tâche -->
                <h2>{{ task.nom }}</h2>
                <!-- Description -->
                <h3>Description : {{ task.description }}</h3>
                <!-- Attribué à -->
                <h3>Attribué à : {{ task.attribue_a }}</h3>
                <!-- Date d'attribution -->
                <h3>Date d'attribution : {{ task.date_attribution }}</h3>
                <!-- Date d'échéance -->
                <h3>Date d'échéance : {{ task.date_echeance }}</h3>
                <!-- Statut de la tâche avec classe de style en fonction du statut -->
                <h3 class="statut {% if task.statut == 'En cours' %}en-cours{% elif task.statut == 'Terminé' %}termine{% else %}en-attente{% endif %}">
                    Statut : {{ task.statut }}</h3>
                <!-- Boutons d'action -->
                <div class="action-buttons">
                    <!-- Lien pour modifier la tâche -->
                    <a href="{{ url_for('edit_task', task_id=task.id) }}">Modifier</a>
                    <!-- Lien pour supprimer la tâche -->
                    <a href="{{ url_for('delete_task', task_id=task.id) }}">Supprimer</a>
                </div>
            </li>
            {% endfor %}
        </ul>
        <!-- Lien pour ajouter une nouvelle tâche -->
        <a href="{{ url_for('add_task') }}" class="add-link">Ajouter une tâche</a>
    </div>
</body>
</html>

```

```

<!DOCTYPE html>
<html>
<head>
<title>Accueil</title>
<style>
body {
font-family: Arial, sans-serif;
background-image: url('static/eurAf.png');
background-size: cover;
background-position: center;
margin: 0;
padding: 0;
display: flex;
align-items: center;
justify-content: center;
height: 100vh;
}

.container {
text-align: center;
padding: 10px;
border-radius: 5px;
background-color: rgba(255, 255, 255, 0.8);
box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

h1 {
color: midnightblue;
margin-bottom: 20px;
}

p {
color: #333;
font-size: 18px;
margin-bottom: 20px;
}

```

```

.login-button {
display: block;
margin-top: 20px;
}

.login-button a {
display: inline-block;
background-color: midnightblue;
color: white;
padding: 10px 20px;
border-radius: 5px;
text-decoration: none;
}

.login-button a:hover {
background-color: #001f3f;
}

h3 {
color: #333;
font-size: 24px;
margin-top: 20px;
}
</style>
</head>
<body>
<div class="container">
<h1>Bienvenue sur la plateforme de gestion de tâches EurAfrique</h1>
<h3>Nous sommes l'association Eurafrique, une organisation non gouvernementale à vocation<br>
sociale et culturelle, œuvrant pour le rapprochement et la coopération entre les continents
européen et africain</h3>
<h3>Pour accéder à votre espace, veuillez vous connecter</h3>
<div class="login-button">
<a href="{{ url_for('login') }}">Se connecter</a>
</div>
</div>
</body>
</html>

```

```

<!--html
<!DOCTYPE html>
<html>
<head>
<title>Liste des tâches</title>
<style>
/* Style global pour le corps de la page */
body {
font-family: Arial, sans-serif;
background-color: #f8f9fa; /* Couleur de fond */
margin: 0;
padding: 0;
}

/* Style pour le conteneur principal */
.container {
max-width: 800px; /* Largeur maximale du conteneur */
margin: 0 auto; /* Centrer le conteneur horizontalement */
padding: 20px;
background-color: #ffffff; /* Couleur de fond du conteneur */
border-radius: 5px; /* Coins arrondis */
box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1); /* Ombre légère */
}

/* Style pour le titre principal */
h1 {
color: midnightblue; /* Couleur du texte */
text-align: center; /* Alignement centré */
margin-bottom: 20px; /* Marge en bas */
}

/* Style pour la liste non ordonnée */
ul {
list-style-type: none; /* Pas de puces */
padding: 0; /* Pas de marge intérieure */
}

/* Style pour chaque élément de la liste */
li {
padding: 20px; /* Marge intérieure */
border: 1px solid #ccc; /* Bordure */
border-radius: 5px; /* Coins arrondis */
margin-bottom: 10px; /* Marge en bas entre les éléments */
background-color: #f5f5f5; /* Couleur de fond */
}

/* Style pour le titre de chaque tâche */
h2 {
color: blue; /* Couleur du texte */
margin: 0 0 10px; /* Marge en bas */
}

/* Style pour les sous-titres */
h3 {
color: #666; /* Couleur du texte */
margin: 0 0 5px; /* Marge en bas */
}

/* Style pour les liens */
a {
text-decoration: none; /* Pas de soulignement */
color: midnightblue; /* Couleur du texte */
margin-right: 10px; /* Marge à droite */
}

/* Style pour les liens survolés */
a:hover {
text-decoration: underline; /* Soulignement au survol */
}

/* Style pour le lien "Ajouter une tâche" */
.add-link {
display: block; /* Afficher en bloc */
margin-top: 20px; /* Marge en haut */
font-weight: bold; /* Texte en gras */
text-align: center; /* Alignement centré */
}

/* Style pour les liens survolés */
a:hover {
text-decoration: underline; /* Soulignement au survol */
}

/* Style pour le lien "Ajouter une tâche" */
.add-link {
display: block; /* Afficher en bloc */
margin-top: 20px; /* Marge en haut */
font-weight: bold; /* Texte en gras */
text-align: center; /* Alignement centré */
}

/* Styles pour les différentes classes de statut */
.en-cours {
color: orange; /* Couleur du texte */
}

.termine {
color: green; /* Couleur du texte */
}

.en-attente {
color: red; /* Couleur du texte */
}
</style>
</head>

```



```

<body>
  <!-- Contenu principal -->
  <div class="container">
    <!-- Titre principal -->
    <h1>Liste des tâches</h1>
    <!-- Liste des tâches -->
    <ul>
      {% for task in tasks %}
      <!-- Élément de liste pour chaque tâche -->
      <li>
        <!-- Titre de la tâche -->
        <h2>{{ task.nom }}</h2>
        <!-- Description -->
        <h3>Description : {{ task.description }}</h3>
        <!-- Attribué à -->
        <h3>Attribué à : {{ task.attribue_a }}</h3>
        <!-- Date d'attribution -->
        <h3>Date d'attribution : {{ task.date_attribution }}</h3>
        <!-- Date d'échéance -->
        <h3>Date d'échéance : {{ task.date_echeance }}</h3>
        <!-- Statut de la tâche avec classe de style en fonction du statut -->
        <h3 class="statut {% if task.statut == 'En cours' %}en-cours{% elif task.statut == 'Terminé' %}termine{% else %}en-attente{% endif %}">
          Statut : {{ task.statut }}</h3>
        <!-- Boutons d'action -->
        <div class="action-buttons">
          <!-- Lien pour modifier la tâche -->
          <a href="{{ url_for('edit_task', task_id=task.id) }}">Modifier</a>
          <!-- Lien pour supprimer

```

9. Bibliographie

1. "Flask Documentation." Flask. Pallets Projects, <https://flask.palletsprojects.com/>.
2. "Flask-WTF Documentation." Flask-WTF. Read the Docs, <https://flask-wtf.readthedocs.io/>.
3. "SQLAlchemy Documentation." SQLAlchemy. SQLAlchemy, <https://docs.sqlalchemy.org/>.
4. "Bootstrap Documentation." Bootstrap. Bootstrap, <https://getbootstrap.com/>.
5. "WTForms Documentation." WTForms. Read the Docs, <https://wtforms.readthedocs.io/>.
6. "Flask-Migrate Documentation." Flask-Migrate. Read the Docs, <https://flask-migrate.readthedocs.io/>.
7. Real Python. <https://realpython.com/>.
8. Grinberg, Miguel. "Miguel Grinberg's Blog." Miguel Grinberg's Blog. Miguel Grinberg, <https://blog.miguelgrinberg.com/>.
9. Grinberg, Miguel. "Flask Mega-Tutorial." Miguel Grinberg's Blog. Miguel Grinberg, <https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world>.
10. "Python Official Documentation." Python. Python Software Foundation, <https://docs.python.org/>.