Rapport du Projet

Thème : Développement d'un Système de Gestion des Notes Scolaires

Nom : Kamtchueng Kouokam Daline Anastasie

Introduction

Ce rapport décrit la réalisation d'un projet sur python qui consistait à créer une application permettant d'enregistrer et suivre les notes des apprenants d'établissements d'enseignements. En plus d'être dotée d'une interface développée avec Flask et comportant deux parties dont l'une est consacrée à la gestion des notes de lycéens et l'autre à celles d'étudiants d'université, notre application permet également de générer automatiquement des bulletins de notes.

1. Langages utilisés pour la conception de pages web et de formulaires

1.1. Le langage HTML

HTML (HyperText Markup Langage) a été utilisé pour structurer le contenu de nos pages web. Nous avons appris à utiliser les balises HTML pour organiser le texte, insérer des images, créer des listes, définir des liens hypertextes et structurer des formulaires. Il nous à permit d'organiser et de rendre le contenu compréhensible par les navigateurs.

Dans le cadre de notre projet, l'application permet tout d'abord la **gestion des élèves**. Pour cela, plusieurs fichiers et routes ont été mis en place :

- **index1.html** : Ce fichier permet de recueillir les informations liées à une classe, telles que le nombre de trimestres, le nombre d'élèves, etc.
- **recuperation1.html** : Il permet de saisir les notes du premier élève. Les noms sont saisis et intégrés via du code Python.
- **tableau1.html** : Ce fichier affiche les informations recueillies, afin que l'utilisateur puisse les vérifier grâce aux fichiers :
    - **ajouttrimestres.html** : pour ajouter les trimestres.

    - **ajouteleves1.html** : pour ajouter des élèves.
    - **ajoutnoteseleves1.html** : pour ajouter leurs notes.
    - **modification1.html** : pour modifier les informations précédemment enregistrées.
- Un dossier **eleves.html** est disponible, permettant de visualiser les classes enregistrées et d'accéder à leurs tableaux respectifs.
- Bulletin1.html et bulletin1_pdf.html : pour la gestion voir et télécharger les différents relèvé des étudiant.

Ensuite, l'application permet également la **gestion des étudiants** (niveau universitaire). Elle repose sur un ensemble similaire de fichiers :

- **index.html** : Permet de récupérer les informations liées à une classe d'étudiants (nombre de trimestres, nombre d'étudiants, etc.).
- **recuperation1.html** : Utilisé pour saisir les notes du premier étudiant, dont le nom est intégré via du code Python.
- **tableau.html** : Affiche les informations de la classe. L'utilisateur peut les vérifier à travers :
    - **ajoutsemestres.html** : pour ajouter les semestres.
    - **ajoutnotes.html** : pour ajouter les notes.
    - **ajouteleves.html** : pour ajouter les étudiants.
    - **ajoutnoteseleves.html** : pour saisir les notes des étudiants.
    - **modification.html** : pour modifier les données existantes.
- Une **route étudiant** permet d'afficher les niveaux enregistrés et d'accéder aux tableaux associés.
- Bulletin.html et bulletin_pdf.html : pour la gestion voir et télécharger les différents relevés des étudiants

Par exemple, voici un extrait du fichier **modification1.html** :

```
    <form method="POST" action="{{ url_for('modification1') }}">
        <h3>Modification des informations et notes </h3>

        <input type="text" id="recherche" placeholder="Rechercher un élève par
nom ou prénom..." onkeyup="filtrerEleves()">

        {% for eleve in eleves1 %}
        <fieldset class="eleve" data-nom="{{ eleve.nom | lower }}" data-
prenom="{{ eleve.prenom | lower }}">
        <legend>Élève{{loop.index0 +1}}: {{ eleve.nom }} {{ eleve.prenom
}}</legend>
        <div class="input-box">
            <input type="text"placeholder="Nom :" name="nom_{{ eleve.id }}"
value="{{ eleve.nom }}">
            <i class="fa-solid fa-user-tie"></i></div>
            <div class="input-box">
```

```
            <input type="text"placeholder="Prénom : " name="prenom_{{ eleve.id
}}" value="{{ eleve.prenom }}">
            <i class="fa-solid fa-user-tie"></i></div>
            <div class="input-box">
             <input type="number"placeholder="Âge :" name="age_{{ eleve.id }}"
value="{{ eleve.age }}">
             <i class="fa-solid fa-input-numeric"></i></div>

        {% for trimestre in range(1, num_trimestres + 1) %}
            <h4>Trimestre {{ trimestre }}</h4>
            {% for matiere in matieres1 %}
            <p>Matiére{{loop.index0 +1}}: {{ matiere }} (Coefficient : {{
coefficient }})</p>
                {% for seq in sequences1 %}
                    {% if seq.matiere == matiere %}
                        {% for seq_num in range(1, seq.num_sequences + 1) %}
                        <div class="input-box">
                            <input type="text" placeholder="Séquence {{
seq_num }}:" min="0" max="20"
                                    name="note_{{ eleve.id }}_{{ matiere
}}_trim{{ trimestre }}_seq{{ seq_num }}"
                                    value="{{
notes1[eleve.id][trimestre][matiere]['Seq' ~ seq_num] if notes1.get(eleve.id)
and notes1[eleve.id].get(trimestre) and
notes1[eleve.id][trimestre].get(matiere) and ('Seq' ~ seq_num) in
notes1[eleve.id][trimestre][matiere] else '' }}">
                                    <i class="fa-solid fa-input-numeric"></i>
                                    </div>
                        {% endfor %}
                    {% endif %}
                {% endfor %}
            {% endfor %}
        {% endfor %}
        </fieldset>
    {% endfor %}
    <input type="submit" class="login-btn" value="Enregistrer les
modifications">
    <script src="{{ url_for('static', filename='js/modification.js')
}}"></script>
    </form>
```

1.2. Le langage CSS

Le **CSS (Cascading Style Sheets)** a été utilisé pour définir la présentation visuelle de nos pages HTML. Nous avons exploré les **sélecteurs**, les **propriétés** (couleurs, polices, marges, paddings, etc.) ainsi que le **modèle de boîte** (*box model*). Cela nous a permis de **styliser** et de **mettre en forme le contenu** des pages web en définissant l'apparence des éléments HTML.

Dans le cadre de notre devoir, nous avons appliqué différents fichiers CSS selon le type de page :

- Nous avons utilisé un **style transparent** avec une **image de fond** pour les formulaires, notamment sur les routes où l'utilisateur devait saisir des informations.
- Pour les **tableaux** et les **bulletins**, nous avons appliqué des styles spécifiques adaptés à ces contenus, via d'autres fichiers CSS dédiés.
- Nous avons également un fichier CSS dédié à la **barre de navigation (navbar)**, ainsi qu'un autre pour la **route "À propos"**. Cette page permet de présenter notre école et de décrire brièvement le fonctionnement de l'application aux utilisateurs.

Par exemple, le fichier **bulletin.css** gère la mise en forme des pages conserant uniquement les bulletin (ainsi que ses variantes).
Enfin, pour chaque fichier CSS, nous avons intégré des **media queries** afin d'assurer une bonne visualisation sur téléphone et autres petits écrans.

```css
.titre-eleve {
    text-align: center;
    color: #2c3e50;
    font-size: 1.5em;
    margin-bottom: 10px;
}

.infos-eleve {
    background-color: #f9f9f9;
    border: 1px solid #ddd;
    border-radius: 8px;
    padding: 15px 20px;
    width: fit-content;
    margin: 0 auto 20px auto;
}

.infos-eleve ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
}

.infos-eleve li {
    display: flex;
    justify-content: space-between;
    margin: 5px 0;
    min-width: 250px;
}

.label {
    font-weight: bold;
    margin-right: 10px;
```

```
    width: 100px;
    text-align: right;
}

.valeur {
    text-align: left;
    flex: 1;
}
table {
    width: 100%;
    border-collapse: collapse;
    margin-bottom: 20px;
    font-family: Arial, sans-serif;
    font-size: 14px;
}

th, td {
    border: 1px solid #aaa;
    padding: 8px;
    text-align: center;
}

/* Couleur des en-têtes */
thead th {
    background-color: #3c0af1; /* vert doux */
    color: white;
}

/* Lignes alternées */
tr:nth-child(even) {
    background-color: #f9f9f9;
}

h1, h3 {
    font-family: Arial, sans-serif;
    color: #333;
}

fieldset {
    border: 1px solid #ccc;
    padding: 15px;
    margin-bottom: 30px;
    border-radius: 8px;
    background-color: #fafafa;
}

ul {
    list-style: none;
    padding-left: 0;
```

```css
}

ul li {
    padding: 5px 0;
}
/* Style du bouton de téléchargement PDF */
.btn-download {
    display: inline-block;
    padding: 8px 16px;
    background-color: #007bff;
    color: white;
    text-decoration: none;
    border-radius: 6px;
    margin-top: 10px;
}

.btn-download:hover {
    background-color: #0056b3;
}
.signature {
    display: flex;
    justify-content: space-between;
    margin-top: 60px;
}

.signature div {
    width: 30%;
    text-align: center;
}
/* Écrans ≤ 1024px */
@media (max-width: 1024px) {
    .infos-eleve li {
        flex-direction: row;
        flex-wrap: wrap;
        min-width: auto;
    }

    table {
        font-size: 13px;
    }

    .btn-download {
        font-size: 14px;
        padding: 7px 14px;
    }
}
```

1.3. Le langage JavaScript

Le **JavaScript** a été utilisé pour ajouter de l'interactivité à nos pages web. Nous avons étudié les bases du langage, notamment les **variables**, les **fonctions**, les **événements**, ainsi que la **manipulation du DOM**.

Dans le cadre de notre devoir, JavaScript nous a permis de mieux gérer les **effets dynamiques** sur certaines routes, comme index.html, index1.html, et d'autres encore, notamment en **intégrant du HTML dynamiquement** dans ces pages.

JavaScript a également été utilisé pour mettre en place un **système de recherche par nom et prénom**, afin de faciliter la tâche des utilisateurs, surtout lorsqu'il y a un grand nombre d'élèves ou d'étudiants à traiter.

Par exemple, le fichier **tableau.js** gère le **système de recherche dynamique dans les tableaux** d'élèves et d'étudiants.

```
        function generateElevesFields() {
            const numEleves = document.getElementById('num_eleves').value;
            const elevesFieldsDiv = document.getElementById('elevesFields');
            elevesFieldsDiv.innerHTML = '';  // Réinitialiser les champs
dynamiques

            for (let i = 0; i < numEleves; i++) {
                elevesFieldsDiv.innerHTML += `
                        <h4>Élève ${i + 1}</h4>
                        <div  class="input-box">
                        <input type="text" name="nom_${i}" id="nom_${i}"
placeholder="Nom" required>
                         <i class="fa-solid fa-user-tie"></i>
                        </div>
                         <div  class="input-box">
                        <input type="text" name="prenom_${i}" id="prenom_${i}"
placeholder="Prénom" required>
                         <i class="fa-solid fa-user-tie"></i>
                        </div>
                         <div  class="input-box">
                        <input type="number" name="age_${i}" id="age_${i}"
placeholder="Âge" required>
                        <i class="fa-solid fa-input-numeric"></i>
                        </div>
                `;
            }
        }
```

```
        // Fonction pour générer les champs de matières
        function generateMatieresFields() {
            const numMatieres = document.getElementById('num_matieres').value;
            const matieresFieldsDiv =
document.getElementById('matieresFields');
            matieresFieldsDiv.innerHTML = '';  // Réinitialiser les champs
dynamiques

            for (let i = 0; i < numMatieres; i++) {
                matieresFieldsDiv.innerHTML += `
                        <h4>Matière ${i + 1}</h4>
                        <div  class="input-box">
                        <input type="text" name="matiere_${i}"
id="matiere_${i}" placeholder="Matière" required>
                        <i class="fa-solid fa-pencil-mechanical"></i>
                        </div>
                        <div  class="input-box">
                        <input type="number" name="coefficient_${i}"
id="coefficient_${i}" placeholder="Coefficient" required step="0.1" min="0">
                        <i class="fa-solid fa-input-numeric"></i>
                        </div>
                `;
            }
        }
```

2. Framework Flask

Le Framework Flaks permet de crée rapidement des applications web simples grâce à sa flexibilité et sa légèreté, tout en offrant des fonctionnalités spécifiques via des extensions. Il est idéal pour les projets de petite ou moyenne envergure. Mais, il présente deux principaux inconvénients : il ne propose pas de fonctionnalités complètes intégrées -et-est difficile à gérer pour des applications complexes à cause de l'absence de structure rigide.

Dans la conception de notre application, le Framework Flask nous a permis de mettre en place les différentes routes associées à chaque Template créé. Nous avons défini des fonctions permettant de sauvegarder les classes et les niveaux pour chaque utilisateur connecté, afin que celui-ci puisse revenir plus tard sur son tableau de bord et continuer la gestion des notes. Voici un exemple de code illustrant cette logique pour la gestion des élèves :

```
@app.route('/index1', methods=['GET', 'POST'])
def index1():
    global eleves1, matieres1, coefficients1, num_trimestres, sequences1
```

```
    if request.method == 'POST':
        # Récupérer la classe des élèves
        classe1 = request.form.get('classe')
        # Récupérer le nombre de trimestres
        num_trimestres = int(request.form.get('num_trimestres', 0))
        eleves1=[]
        # Récupérer le nombre d'élèves et leurs informations
        num_eleves1 = int(request.form.get('num_eleves', 0))
        for i in range(num_eleves1):
            eleve1_id = i
            nom1 = request.form[f'nom_{i}']
            prenom1 = request.form[f'prenom_{i}']
            age1 = int(request.form[f'age_{i}'])
            eleves1.append({'id': eleve1_id, 'nom': nom1, 'prenom': prenom1,
'age': age1, 'classe': classe1})

        # Récupérer les matières et coefficients (une seule fois)
        num_matieres1 = int(request.form['num_matieres'])
        matieres1 = []
        coefficients1 = []
        for i in range(num_matieres1):
            matiere1 = request.form[f'matiere_{i}']
            coefficient1 = float(request.form[f'coefficient_{i}'])
            matieres1.append(matiere1)
            coefficients1.append(coefficient1)

        # Récupérer le nombre de séquences (valeur unique pour toutes les
matières)
        num_sequences = int(request.form.get('num_sequences', 0))

        # Créer les séquences pour chaque trimestre, appliquées à toutes les
matières
        sequences1 = []  # Réinitialiser les séquences pour chaque soumission
        for matiere in matieres1:
                # Ajout des séquences pour chaque matière et trimestre
                sequences1.append({ 'matiere': matiere, 'num_sequences':
num_sequences})

        return redirect(url_for('recuperation1'))

    return render_template('index1.html', num_eleves=len(eleves1))

@app.route('/eleve')
def eleve():
    sauvegardes = []
    user_id = session.get('user_id')
    if not user_id:
        return redirect(url_for('connection'))  # Rediriger si l'utilisateur
n'est pas connecté
```

```
    dossier = 'projetpython1/sauvegardes1'
    if os.path.exists(dossier):
        for fichier in os.listdir(dossier):
            if fichier.endswith(f'_user{user_id}.json'):
                # Extraire le nom de la classe sans le suffixe _user<ID>.json
                classe = fichier.replace(f'_user{user_id}.json', '')
                sauvegardes.append(classe)

    return render_template('eleve.html', classes=sauvegardes)

@app.route('/supprimer_tableau1/<classe>', methods=['POST'])
def supprimer_tableau1(classe):
    chemin = os.path.join('projetpython1/sauvegardes1', f'{classe}.json')
    if os.path.exists(chemin):
        os.remove(chemin)
    return redirect(url_for('eleve'))

@app.route('/charger_tableau/<classe>')
def charger_tableau(classe):
    global eleves1, notes1, matieres1, sequences1, num_trimestres,
coefficients1
    user_id = session.get('user_id')
    if not user_id:
        return
    chemin = f'projetpython1/sauvegardes1/{classe}_user{user_id}.json'

    if os.path.exists(chemin):
        with open(chemin, 'r', encoding='utf-8') as f:
            donnees = json.load(f)
            eleves1 = donnees['eleves1']
            matieres1 = donnees['matieres1']
            coefficients1 = donnees['coefficients1']
            sequences1 = donnees['sequences1']
            num_trimestres = donnees['num_trimestres']
            notes1 = {
                int(eleve_id): {
                    int(trimestre): {
                        matiere: seqs for matiere, seqs in matieres.items()
                    } for trimestre, matieres in trimestres.items()
                } for eleve_id, trimestres in donnees['notes1'].items()
            }

    return redirect(url_for('tableau1'))


@app.route('/recuperation1', methods=['GET', 'POST'])
def recuperation1():
    global notes1, matieres1, coefficients1, num_trimestres, sequences1
```

```
    matieres_coefficients = {matieres1[i]: coefficients1[i] for i in
range(len(matieres1))}

    if request.method == 'POST':
        # Initialisation de notes1 comme une liste vide au début de chaque
POST
        notes1 = {}

        for eleve1 in eleves1:
```

Conclusion

Ce projet nous a permis d'acquérir une **compréhension approfondie des concepts fondamentaux** de la conception d'applications web à l'aide de **HTML, CSS, JavaScript** et du **Framework Flask** en Python.

La phase de réalisation, marquée par la création concrète de l'application, a renforcé ces acquis et mis en évidence la **portée pratique** de ces outils.

Ce rapport reflète l'**atteinte des objectifs initiaux** du projet et constitue une **base solide pour de futurs apprentissages** et développements dans le domaine du web avec Python.