Quentin GAVILAN
Remy VINCENT
Laureline JEUNEMAITRE
Luigi SOUSA MARTINS

RENDU FINAL
MUGIWARA NO KISTUNE

2022-2023

Encadré par :
Franck POMMEREAU

Table des matières

Aucune entrée de table des matières n'a été trouvée.

I - Reprise du dépôt

Le texte est colorié en fonction de ce qui a été fait entièrement:

bleu **signifie** fait comme prévu

- rouge: enlevé car inutile ou non fait
- orange: l'option a été changé partiellement/entièrement

SITE DE DÉPÔT DE PROJET

Réaliser un site permettant aux étudiants de rendre un projet/exercice donné par l'enseignant. Chaque projet est déposé dans une matière en particulier et rattaché à un groupe d'étudiants identifiés, à la suite de ce dépôt les étudiants reçoivent une preuve de dépôt signé confirmant le bon dépôt.

1. NOYAUX

- STOCKAGE DE FICHIER

- Création d'un système de dépôt fichier
  - grâce à la balise html <form> et <input> on peut déposer facilement un fichier il sera stocké dans le dossier principal de l'emplacement du code
- Emplacement stockage des archives/fichiers
  - lien avec le code: l'emplacement du fichier est défini dans la fonction "rangement". l'emplacement du fichier est défini avec ces 2 lignes types:

```
case "info": #Cas pour le dossier Info
    app.config['UPLOAD_FOLDER'] = grp_info + grp
    fichier.save(os.path.join(app.config['UPLOAD_FOLDER'],filename))
```

grp_info renvoi le chemin "rendu/info/groupe" qui est ensuite lié au numéro du groupe sélectionnée.

- REMISE PREUVE DEPOT

- Fichier contenant une Signature Numérique (hachage SHA512)
  - un fichier est donné en tant que preuve de dépôt, le fichier rendu est créé dans la fonction "formulaire" dans la variable rendu. On y ajoute les informations nécessaires puis on stocke le nom de ce fichier dans une liste (seulement le nom du fichier).
    La signature est faite, la suite de la création de ce fichier en .txt
    La liste est nécessaire pour stocker le nom du dernier fichier de rendu pour ensuite le récupérer dans la fonction "download" qui renvoie un "send_file". Cette méthode de flask permet de télécharger un fichier en cliquant sur un lien via un "url_for()" inséré dans un html.

```
@app.route('/download')
def download():
    indice=len(tabRendu)
    path=tabRendu[indice-1]
    return send_file(path, as_attachment=True)
```

- Fichier contenant la taille du fichier déposé, la date du dépôt, le nom du fichier
    - Dans la preuve de rendu, les informations nécessaires sont ajoutées comme ceci. lien avec le code: on utilise open(...) pour créer un fichier .txt puis on écrit dedans grâce à la méthode ".write"

```
#ecriture de la preuve de dépôt
rendu= open("rendu_"+parcour+"_"+groupe+".txt", 'a')
rendu.write("Date de rendu: "+dateRendu+"\nParcours:"+ parcour+"\nGroupe: "+groupe)
rendu.write("\nFichier rendu: "+ filename+ "\nTaille du fichier rendu: "+tailleFichier +" octets")
nomrendu= "rendu_"+parcour+"_"+groupe+".txt"
```

- SITE INTERFACE

- Création d'un site web accessible localement
    - Création de l'interface en utilisant les langages HTML, CSS et JS, nous nous sommes basés sur un modèle libre de droit sur internet que nous avons modifié en fonction de notre direction artistique et de nos besoins
- Possibilité de dépôt de fichier
    - lien avec le code: dans la fonction "formulaire", la variable "fichier" stocke le fichier puis est sécurisée avec la méthode "secure_filename". Ce fichier est envoyé dans la fonction "rangement" pour être stocké.
- Texte indiquant les informations à suivre pour l'étudiant (non modifiable)
    - lien avec le code: toutes les lignes "@app.route()" avec la définition qui suit en retournant la fonction render_template() de flask lié à une page html.

2. FONCTIONNALITÉS

- OPTION DEPOT- (Ordre de priorité 1)

- Sélection du format de fichiers autorisés
    - Choix du professeur de modifier le format autorisé pour sa filière (voir la fonction change).
- Date limite et impossibilité de rendre le projet après la date
- Modification de projet après dépôt grâce à la signature de preuve
    - Date de dépôt, heure et preuve
    - vérification de la preuve de dépôt
    - Écrase un projet au bout d'un nombre d'envois maximum(FIFO)

- Après plusieurs essais nous avons abandonné cette idée pour laisser place au remplacement des fichiers ayant le même nom

- LIMITE DE DEPOT-4

- Minuteur en format jour: heure: min▪
  - Dépend de la date de dépôt
  - Les variables “limite_phy” “limite_bio” et “limite_info” determinent la date limite de dépôt de chaque matière en fonction des variables du jour mois et année. Elles sont reliées à l’affichage de l’html, l’administrateur ayant le mot de passe peut donc modifier cette date et faire en sorte que le dépôt soit repoussé.
- Historique de dépôt pour le professeur
  - Pas la possibilité ni le temps de remplir cette fonction

- SERVEUR –2

- Ajout de description texte donnant les consignes modifiables par le professeur
  - Texte modifiable par l’administrateur ayant accès grâce à un code donné

```python
#Fonction sécurisant au minimum l'acces au changement des modalités de rendu pour les e tudiants (Quentin)
def check_code(code):
    match code :
        case "1234":
            return render_template("modif_info.html",date = limite_info, type = type_info)
        case "1111":
            return render_template("modif_bio.html",date = limite_bio,type = type_bio)
        case "2222":
            return render_template("modif_phy.html",date = limite_phy,type = type_phy)
        case _:
            return render_template("malin.html")
```

- Menu déroulant option à choisir pour le dépôt selon les fonctionnalités
  - Dépend des options dépôt: Format, Date limite
    - Grâce à la fonction change(parcours) on peut modifier la date le type ainsi que rajouter un texte pour les étudiants, après modification on remarque que la date change et s’actualise tout en l’affichant sur les différentes filières

Code de la fonction change() :

```python
#Fonction permettant de recuperer le formulaire de changement de date, du type et du text des f
#Toujours en separant chaque filiere avec un switch
def change(parcour):
    resultat=request.form
    a=resultat['anne']
    m=resultat['mois']
    j=resultat['jour']
    type1 = resultat['type_1']
    type2 = resultat['type_2']
    type3 = resultat['type_3']
    scribe = resultat['scribe']
    if type3 =="NULL":
        type3 =""
    if type2 =="NULL":
        type2 =""
    if type1 =="NULL":
        type1 =""
    match parcour :
        case "info":
            global anne_info
            global jour_info
            global mois_info
            global limite_info
            global type_info
            global scribe_info
            #Gestion d'erreur en cas de date non remplis et modification des variables
            if j!="NULL" and m !="NULL" and a!="NULL":
                jour = int(j)
                mois = int(m)
                annee = int(a)
                anne_info =annee
                jour_info = jour
                mois_info =mois
                limite_info=datetime(anne_info, mois_info, jour_info, 23, 59, 59)
            if type1 =="" and type2 =="" and type3 == "":
                type_info = type_info
            else:
                type_info = type1+","+type2+","+type3
            if scribe != "":
                scribe_info = scribe
            return render_template("index.html")
        case "bio":
            global anne_bio
            global jour_bio
            global mois_bio
            global limite_bio
            global type_bio
            global scribe_bio
            if j!="NULL" and m !="NULL" and a!="NULL":
                jour = int(j)
                mois = int(m)
                annee = int(a)
                anne_bio =annee
                jour_bio = jour
                mois_bio =mois
                scri
                limite_bio=datetime(anne_bio, mois_bio, jour_bio, 23, 59, 59)
            if type1 =="" and type2 =="" and type3 == "":
                type_bio = type_bio
            else:
                type_bio = type1+","+type2+","+type3
            if scribe != "":
                scribe_bio = scribe
            return render_template("index.html")
        case "phy":
            global anne_phy
            global jour_phy
            global mois_phy
            global limite_phy
            global type_phy
            if j!="NULL" and m !="NULL" and a!="NULL":
                jour = int(j)
                mois = int(m)
                annee = int(a)
                anne_phy =annee
                jour_phy = jour
                mois_phy =mois
                limite_phy=datetime(anne_phy, mois_phy, jour_phy, 23, 59, 59)
            if type1 =="" and type2 =="" and type3 == "":
                type_phy = type_phy
            else :
                type_phy = type1+","+type2+","+type3
            if scribe != "":
                scribe_phy = scribe
            return render_template("index.html")
        case _:
            return "NULL"
```

- INTERFACE CLIENT–3

- Afficher la description ajoutée par le professeur
- Menue déroulant des matières, exercice
    - Ajout et modification du système d'interface
        - Le menu déroulant concerne seulement le groupe, la matière est choisie au préalable en demandant dans quelle matière il veut déposer son projet. Il est codé directement sur la page html. Il renvoie la bonne valeur lorsque sélectionné grâce à l'option "value=...".

```html
<p>Groupe :<select id="Groupe" name="Groupe">
    <option value=NULL>Faire un choix</option>
    <option value="1">1</option>
    <option value="2">2</option>
    <option value="3">3</option>
    <option value="4">4</option>
```

II - Bibliothèques utilisées:

Flask: c'est la bibliothèque principale de notre code. Celle-ci nous a permis de faire une interface web ou nous pouvons naviguer en toute tranquillité. Celle-ci s'accompagne de pages html qui sont nécessaires à l'affichage souhaité.
Dans notre projet, cette bibliothèque nous a permis de créer toutes les pages web locales.

werkzeug.utils as osy: cette bibliothèque nous a permis de chiffrer et de pouvoir sauvegarder le fichier reçu.

Oudjirasign: Pour pouvoir hacher un fichier puis le signer avec une clé privée nous avons utilisé la bibliothèque "oudjirasign as osy". Celle-ci est utile pour générer des clés publiques et privées et les raccrocher à un fichier. Dans notre cas nous pourrons ensuite stocker ces clés en fonction du groupe auxquelles elles ont été assignées, pour ensuite permettre la vérification de clé pour modifier le dépôt.
Dans notre projet, cette bibliothèque nous a permis de signer avec des clés privées différentes les fichiers de preuve de rendu qui sont donnés après dépôts du projet.

datetime: Cette bibliothèque permet de manipuler la date du jour et des dates définies.
Dans notre projet, cette bibliothèque nous a permis de dater le remise de projet pour l'ajouter dans notre preuve de rendu. Elle nous a aussi permis de définir les dates limites de dépôt et de bloquer le dépôt après la date limite pour chacune des matières.

III- Répartition du travail dans le groupe :

Laureline :
Mise en place d'une date limite de dépôt et fonction permettant de savoir si la date est dépassée ou non afin de laisser la possibilité ou non de dépôt. Mise en place d'une impossibilité de dépôt lorsque les conditions ne sont pas remplies. Recherche et compréhension d'oudjirasign afin de mettre en place la signature/hash. Aide à la création de preuve dépôt. Test pour mettre en place une vérification de signature des preuves de dépôt.

Rémy :
Compréhension de flask et sa liaison avec html. Liaison des valeurs entre python et html grâce à flask. Création de la preuve de rendu et de l'envoie en téléchargement de celui-ci après signature. Aide sur la signature de la preuve de rendu et l'implémentation de la date. Essais de mise en place d'une page de modification de dépôt avec vérification de signature de la preuve donnée avec les signatures connues.

Quentin :
Création de l'Interface et de la direction artistique du site. Dépôt des fichiers dans des dossiers personnalisés en fonction de la réponse du formulaire. Assemblage des différents code et fonctionnalités. Mise en forme du code pour éviter les doublons. Mise en place d'une modification possible de la date du type et d'un ajout de texte pour les différentes modalités de rendu par un administrateur n'ayant pas accès au code.

Luigi :
Réunification des différents codes partagés et aide à la création de fonctions permettant d'éviter les doublons. Création des différentes pages HTML des filières info.html, bio.html, phy.html afin de séparer la gestion des filières. Aide à la signature des fichiers, à la recherche des différentes librairies ainsi qu'au rangement des dépôts d'étudiants dans différents dossiers.

Merci à Monsieur Franck Pommereau pour toute l'aide apportée à la réalisation de ce projet.