

1 Architecture du projet

```
fifo_and_lifo
├── app/.....Dossier contenant les modules tiers
├── doc/
│   └── UtilisationFlask .....Ce document
├── static/
│   ├── assets/ ..... Dossier contenant les fichiers tiers
│   ├── css/ .....Dossier des fichiers CSS du projet
│   ├── db/ .....Dossier contenant la(es) base(s) de données
│   ├── images/
│   ├── pdfs/
│   └── videos/
├── template/..... Dossier contenant les fichiers HTML du projet
├── main.py..... Programme principal
├── README.md
└── fifoandlifo.ini ..... Fichier de configuration uWSGI
```

2 Utilisation de Flask

2.1 Code minimal fonctionnel d'une application

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

@app.route('/')
def hello_world():
    return 'Hello, World!'
```

2.2 Créer et afficher une page HTML

2.2.1 Code HTML

Les fichiers statiques doivent être déposés dans le dossier static

Ainsi pour créer une page `bonjour.html`, qui contient l'image `bonjour.html` placée dans `static/images`

```
<html>
  <head>
    <meta charset="utf-8">
    <title>Bonjour</title>
  </head>
  <body style="
    background-image: url('/static/images/bonjour.jpg');
    background-size: cover;">

    <h1>Bonjour l'équipe</h1>
  </body>
</html>
```

2.2.2 Attribuer une fonction Python à une URL

Maintenant, il faut que le navigateur puisse avoir accès à cette page.

On lie donc cette page HTML à une fonction python puis on lie la fonction à l'URL souhaitée.

```
return render_template('/template_de_la_page')
```

Pour lier la fonction à l'URL, on utilise un décorateur.

```
app.route('/url')
```

```
@app.route('/bonjour')
def bonjour():
    return render_template('/bonjour.html')
```

```
@app.route('/bonjour')
def fct_bonjour():
    return render_template('bonjour.html')
```

3 Communication Python -> HTML

1. On peut utiliser des variables Python dans le code en les encadrant par des doubles accolades `{{mavariable}}`.
2. On peut utiliser des boucles et des instructions conditionnelles en les encadrant par `{%moninstruction%}`.

Dans le fichier `main.py`

Dans le fichier `bonjour.html`

```
@app.route('/bonjour')
def bonjour():
    equipe = ['denis',\
              'alan',\
              'joris',\
              'eric',\
              'etienne']
    return render_template(
        '/bonjour.html',
        equipe=equipe)
```

```

<html>
  <head>
    <meta charset="utf-8">
    <title>Bonjour</title>
  </head>
  <body style="
    background-image: url('/static/images/bonjour.jpg');
    background-size: cover;">

    <h1>Bonjour l'équipe</h1>
    <ul>
      {% for nom in equipe %}
        <li>{{nom}}</li>
      {% endfor %}
    </ul>
  </body>
</html>

```

4 Communication HTML -> Python

Un des moyens de communication d'une page HTML vers le serveur est de passer par des formulaires.

On utilisera la methode **POST** pour faire facilement la distinction entre la demande du formulaire et l'envoi de la réponse.

TP : création d'un formulaire

Dans le fichier que l'on nommera **form.html**, créer un formulaire HTML utilisant la méthode POST, ayant aucune action et demandant à l'utilisateur d'entrer son nom.

Note :

La balise form aura donc :

- Un attribut **method** dont la valeur est **POST**
- Un attribut **action** dont la valeur est ""

Exemple : Dans le fichier **form.html**

```

<html>
  <head>
    <meta charset="utf-8">

```

```

    <title>Bonjour</title>
  </head>
  <body>

<h1>Un splendide formulaire</h1>

<form method="POST" action="">
  <input type="text" name="nom">
  <input type="submit" value="Valider">
</form>

  </body>
</html>

```

On a un splendide formulaire, maintenant il faut le faire traiter par le serveur.

Pour differencier les méthodes GET et POST on utilise la methode `request` de la bibliothèque `flask`.

Il faut donc verifier qu'elle soit bien importée au départ.

```
from flask import Flask, request
```

Ensuite comme d'habitude, on affecte notre url à la fonction souhaitées.

```

@app.route('/formulaire/')
def formulaire():
    """ici on traite le formulaire"""
    return render_template("form.html")

```

Attention, on doit prevenir notre serveur que sur cette url, il peut recevoir aussi des requetes "POST", on ajoute donc à la route la methode POST

```

@app.route('/formulaire/', methods=["GET", "POST"])
def formulaire():
    """ici on traite le formulaire"""
    return render_template("form.html")

```

Il ne reste plus qu'à récupérer les valeurs passées par le formulaire et les traiter :

```
@app.route('/formulaire/', methods=["GET", "POST"])
def formulaire():
    """ici on traite le formulaire"""
    if request.method == 'POST':
        nom = request.form.get("reponse")
        print(nom) # On affiche le résultat dans la console
        return nom # On affiche le résultat dans une page WEB
    return render_template("form.html")
```

0269 61 92 05