Projet Python modulaire : système web de classement par vote (dossier deuxième oral du CAPES informatique 2021)

Maverick CHARDET

Le travail présenté dans ce dossier est un projet Python de Terminale NSI, réalisable plutôt en milieu ou en fin de premier trimestre. Il vise à la réalisation d'un site internet qui offre aux utilisateurs la possibilité de voter pour leur personnage préféré d'une œuvre de fiction parmi deux (avec répétitions). Un classement entre les personnages, commun à tous les utilisateurs, est maintenu et mis à jour à chaque vote. Les informations sur les personnages sont récupérées par le code serveur en utilisant une API. Dans ce dossier l'exemple de la série Game of Thrones est utilisé, mais n'importe quelle œuvre peut être utilisée pourvu qu'une API compatible soit disponible publiquement ou créée par l'enseignant. Le projet fonctionnel sera fourni avec le diaporama pour la présentation orale.

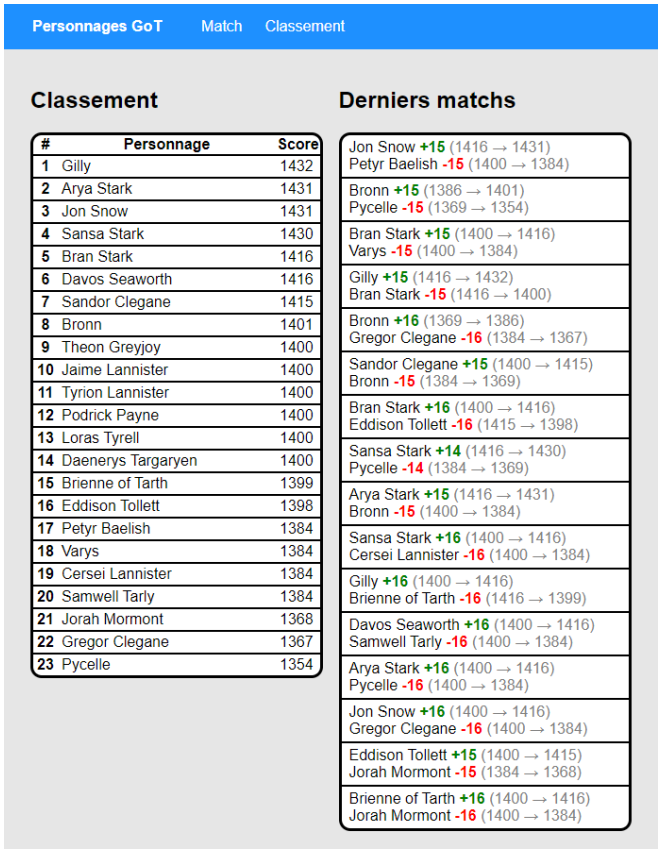
1 Vue générale

Cette section présente le projet fini du point de vue de l'utilisateur (interface), du programmeur (organisation du code) et de l'enseignant (thèmes du programme abordés).

1.1 Projet fini : point de vue utilisateur

Le projet fini se présente comme un site internet avec deux pages : la page de vote et la page de classement. Un exemple de la façon dont ces pages sont présentées à l'utilisateur est présenté dans la figure 1.

(a) Page de vote



(b) Page de classement

FIGURE 1 – Les deux pages présentées à l'utilisateur

La page de match affiche deux personnages de la série et demande à l'utilisateur ou utilisatrice de choisir son préféré en cliquant sur ce dernier. Le choix effectué est pris en compte pour mettre à jour le classement des personnages commun à tous les utilisateurs et un nouveau couple de personnages est affiché.

La page de classement affiche la liste des personnages de la série triée par ordre décroissant de score (le score résultant des votes des utilisateurs). La liste des derniers matchs (choix des utilisateurs) est aussi affichée, listant pour chaque match le gagnant, le perdant, et comment leurs scores ont été affectés.

1.2 Organisation du code

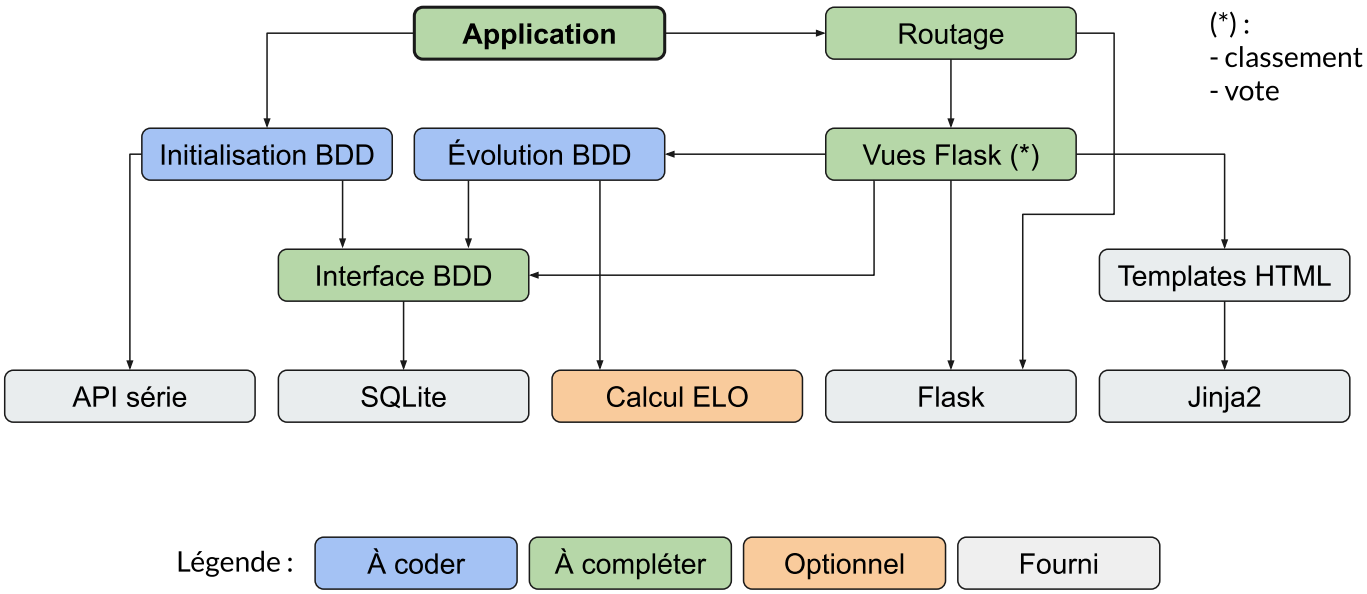


FIGURE 2 – Architecture du code

La structure du code du projet, présentée en Figure 2, est imposée aux élèves (code à trous). La figure inclut aussi les modules Python externes utilisés, notamment Flask (serveur web) et SQLite (base de données). Les différents fichiers liés à chaque module sont présentés dans la Table 1.

Module	Fichiers	Description
Interface BDD	`bdd.py` (incl. SQL)	Classe BDD permettant la gestion de la base de données de personnages (ajout de personnages, mise à jour de leur score, gestion des matchs)
Initialisation BDD	`initialisation_bdd.py`	Fonctions permettant de remplir la base de données au lancement du serveur web si elle est vide, notamment en récupérant la liste des personnages par l'intermédiaire d'une API externe
Évolution BDD	`evolution_bdd.py`	Fonctions permettant de créer un match et d'appliquer le résultat d'un match à la base de données
Application, routage, vues Flask (= points d'accès du serveur web)	`app.py`	Fichier Python principal du projet initialisant le serveur web et listant les points d'accès web
Calcul ELO	`elo.py`	Implantation du système ELO (évolution des scores des personnages en fonction du résultat d'un match)
Templates HTML (répertoire `templates/`)	`layout.html.jinja2` `classement.html.jinja2` `match.html.jinja2`	Fichiers modèles HTML (fournis)

TABLE 1 – Fichiers et description des modules

1.3 Thèmes des programmes abordés

Thèmes du programme de Terminale NSI :
- Modularité
  - Créer des modules simples et les documenter
  - Utiliser des API
  - Exploiter leur documentation
- Système de gestion de bases de données relationnelles
  - Expérimentation
- Langage SQL : requêtes d'interrogation et de mise à jour d'une base de données
  - Construire des requêtes d'interrogation à l'aide des clauses du langage SQL : `SELECT`, `FROM`, `WHERE`, `JOIN` (+ utilisation de `ORDER BY`)
  - Construire des requêtes d'insertion et de mise à jour à l'aide de : `UPDATE`, `INSERT`, `DELETE`

Prérequis de Terminale NSI :
- Modèle relationnel : identification des concepts
- Base de données relationnelle : distinction entre structure et contenu
- Langage SQL : identifcation des composants d'une requête, avoir déjà exécuté des requêtes
- Vocabulaire de la programmation objet : écriture d'une classe, accès aux attributs et méthodes

Prérequis de première NSI :
- Interaction avec l'utilisateur dans une page Web : boutons, formulaires
- Interaction client-serveur / requêtes HTTP, réponse du serveur : distinction de ce qui est exécuté par le client et le serveur et dans quel ordre, requêtes GET (rappels effectués)

2 Plan de séquence

Le tableau Table 2 récapitule l'ensemble des séances liées au projet qui sont détaillées dans la suite de cette section.

Partie	Thème	Type	Durée
Préparation	Modularité	Cours/TD	1h30
Préparation	Modularité	TP	2h
Préparation	Utilisation de Flask	Cours/TP	2h
Projet	Utilisation de SQLite en Python	TP	2h
Projet	Initialisation BDD avec API, création d'un match	TP	2h
Projet	Votes et classement	TP	2h
Projet	Travail à la maison, continu	Travail maison	2h

Table 2 – Ensemble des séances liées au projet

2.1 Travaux préparatoires

Ces travaux visent à donner aux élèves les connaissances pour mener à bien le projet. Deux thèmes sont abordés : la modularité et l'utilisation de Flask (module Python pour développer un serveur web). La modularité est explicite dans le programme de Terminale NSI, tandis que Flask est utilisé pour illustrer le concept d'API et des notions vues en Première NSI (interactions clients/serveurs en particulier).

2.1.1 La modularité

Cette séance est constituée d'un cours/TD de 1h30 et un TP de 2h. Elle a pour objectif de comprendre pourquoi et comment concevoir un système de manière modulaire. Les exercices de TD et de TP ont pour principal objectif de faire différencier aux élèves ce qui relève de l'interface et de l'implantation, et de pouvoir écrire une bonne documentation de module (ou en exploiter une). C'est en effet un aspect qui ne peut pas être développé avant que les élèves connaissent la notion d'interface, tandis qu'ils ont travaillé l'écriture de code durant toute l'année de première. Deux cas sont explorés en détail : les modules Python et les APIs web. Des exemples variés sont utilisés pour illustrer l'universalité de ces notions.

Plan du cours/TD :
- Notion de module (ensemble de classes et fonctions liées, notions d'interface et d'implantation)
- Exemples en Python (modules déjà utilisés par les élèves, système d'import)

— Exercices de TD (déterminer une interface à partir de code et à partir de spécifications)
— Les APIs (définition, spécificités des APIs web, exemples d'APIs web, données ouvertes)

Plan du TP :

— Utilisation d'APIs web (prévisions météo, coordonnées GPS d'une adresse)
— Conception d'une classe Python (annuaire téléphonique avec définition des spécifications)
— Écriture de la classe Python (avec documentation et tests)

Objectifs :

— Comprendre les notions de module et d'interface
— Faire le lien avec les modules utilisés précédemment par les élèves
— Comprendre des particularités d'une API web
— Découvrir les possibilités offertes par les données ouvertes
— Utiliser une API web et exploiter sa documentation
— Concevoir, écrire et documenter un module Python

2.1.2 Utilisation de Flask

Cette séance de cours/TP de 2h permet d'illustrer le concept de serveur web vu en Première, et de l'aborder sous l'angle du concept d'API du programme de Terminale. Les élèves créent un serveur web simple avec Flask, qui est un des modules les plus accessibles en Python pour réaliser un serveur web. La création de points d'accès web par les élèves permet à la fois d'illustrer l'ordre d'exécution des différentes étapes par le client et le serveur, mais aussi de rendre apparente l'interface du serveur web sous forme d'API (rendue explicite par le système d'annotations Python utilisé par Flask).

Plan :

— Rappels sur la communication entre client et serveur
— Principes généraux de Flask
— Principes de modèles (templates)
— Expérimentation de Flask
— Observation du fonctionnement des modèles HTML (Jinja2)
— Création d'une petite application de liste de mémos

Objectifs :

— Rafraîchir les connaissances sur le modèle client-serveur (programme de première)
— Créer un serveur web simple, vu comme une API
— Expérimenter un autre type de modularité sous la forme de modèles (templates)
— Comprendre le besoin d'une base de données dans une application web

2.2 Projet

Le projet consiste en une succession de TPs en groupes, TPs qui idéalement devraient être terminés à la maison si les élèves ont manqué de temps (ce n'est cependant pas nécessaire si les conditions matérielles ne le permettent pas). La taille des groupes est à adapter en fonction des ressources matérielles disponibles dans l'établissement ainsi qu'au domicile des élèves (idéalement des groupes de 2 à 3 élèves). Des rendus intermédiaires (points d'étapes) sont effectués, et les élèves n'ayant pas atteint les objectifs d'un point d'étape récupèrent des fichiers corrigés pour l'étape donnée au début de la séance suivante.

2.2.1 Utilisation de SQLite en Python

Ce TP de 2h est directement en lien avec le programme. On suppose que le vocabulaire des bases de données est déjà acquis et que les élèves ont déjà exécuté des commandes dans un logiciel avec interface graphique tel que DB Browser for SQLite [1]. On se concentre donc sur l'automatisation de l'exécution des commandes en Python avec le module `sqlite3`. Les élèves écrivent le contenu des fonctions pour respecter la documentation fournie. Un fichier de test leur est fourni, ce qui leur permet de s'auto-évaluer pendant le TP. L'évaluation par l'enseignant de ce TP est détaillée dans la dernière section de ce dossier.

Plan :

— Rappels sur le langage SQL
— Présentation des fonctions principales du module Python `sqlite3`
— Écrire des requêtes de création de tables (techniquement hors programme, mais jugé important pour la compréhension)

1. `https://sqlitebrowser.org/`

— Écrire des requêtes d'insertion, d'interrogation, de mise à jour
— Exécuter ces requêtes en Python (en suivant les spécifications)

Objectifs :
— Automatiser l'exécution de requêtes SQL avec le module Python `sqlite3`
— Comprendre comment passer des paramètres (variables) à une requête SQL
— Révisions : construire des requêtes

Avancement du projet :
— Classe Python de gestion de la base de données (ajout, lecture, modification et suppression de personnages et de matchs)

2.2.2 Initialisation de la base de données avec une API, création d'un match

Ce TP de 2h (avec une petite partie préparatoire de TD) a pour objectifs de créer un module d'initialisation d'une base de données contenant une liste de personnages en utilisant le module Python complété lors de la séance précédente, ainsi que d'afficher son contenu sous la forme du classement des personnages. Enfin, les élèves doivent permettre la génération d'un match aléatoire à afficher à l'utilisateur (affichage réalisé en utilisant le modèle HTML fourni). Chaque personnage doit avoir la même probabilité d'apparaître, à gauche et à droite de l'écran de match. Les élèves seront invités à tester leur fonction de choix aléatoire par comptage en important un faible nombre de personnages dans la base de données.

Prérequis (point d'étape) :
— Classe Python fonctionnelle de gestion de la base de données (corrigé fourni le cas échéant)

Plan :
— Découvrir l'API web *got-api*[2]
— Créer un module d'initialisation de la BDD (y-compris sa documentation)
— Rendre fonctionnels la création et l'affichage d'un match aléatoire

Objectifs :
— Utiliser une API web externe (liste de personnages)
— Utiliser une API Python créée par les élèves (gestion de la base de données)
— Créer et documenter un module (initialisation de la base de données)
— Utiliser une API Python externe (templates) en s'inspirant du TP de découverte de Flask

Avancement du projet :
— Initialisation de la base de données et remplissage avec les données des personnages
— Affichage du classement et d'un match aléatoire

2.2.3 Votes et classement

Ce TP de 2h a pour objectifs d'afficher et de faire évoluer le classement des personnages en fonction des votes effectués par les utilisateurs. Les élèves doivent concevoir un module Python pour mettre à jour la base de données en fonction des votes. Le classement peut se faire selon deux niveaux de difficulté. Un premier niveau consiste à faire gagner 1 point au gagnant et faire perdre 1 point au perdant. Un deuxième niveau pour les élèves les plus avancés consiste à utiliser le système ELO[3], qui permet de faire varier le nombre de points gagnés et perdus en fonction du classement relatif des personnages.

Prérequis (point d'étape) :
— Génération d'un match aléatoire fonctionnelle (corrigé fourni le cas échéant)

Plan :
— Créer un module d'évolution de la BDD (avec documentation)
— Récupérer et prendre en compte le vote de l'utilisateur
— (optionnel) Implanter le système ELO

Objectifs :
— Créer et documenter un module (évolution de la base de données, calcul ELO)
— Utiliser une API Python créée par les élèves (BDD)
— Utiliser une API Python externe (templates) en s'inspirant du TP de découverte de Flask
— Comprendre l'architecture complète du projet en utilisant tous les modules conjointement : difficile a priori, mais rendu plus facile par l'approche progressive durant le projet (module par module) et centrée sur l'interface et la documentation de chacun de ces modules

2. `https://api.got.show/`
3. `https://fr.wikipedia.org/wiki/Classement_Elo`

Avancement du projet :
— Prise en compte des votes et évolution de la base de données

2.2.4 Travail à la maison

Ce temps estimé à 2h sur les trois séances de TP du projet lui-même consiste pour les élèves à terminer les TPs s'ils n'ont pas eu le temps, et dans le cas du dernier TP finaliser le projet pour le rendu final. Ils peuvent aussi apporter des améliorations libres au projet s'ils le souhaitent (affichage personnalisé, fonctionnalités supplémentaires, etc.) avec accord préalable de l'enseignant.

3 Évaluation

L'évaluation se fait de manière sommative, mais avec des retours aux élèves entre les TPs, ce qui lui donne une dimension formative. Elle se fait en trois étapes, avec les deux rendus intermédiaire et le rendu final. Ces rendus sont évalués de manière indépendante, grâce aux points d'étapes qui permettent à tous les élèves de repartir sur une base solide à chaque TP (correction des étapes précédentes fournies). L'accent est mis sur le respect des spécifications données (et la syntaxe SQL pour le premier rendu). Les fonctions complétées ou écrites par les élèves sont évaluées séparément, de sorte que l'élève ne soit pas grandement pénalisé si un rendu ne fonctionne pas complètement.

3.1 Notation du rendu 1 (classe de gestion de la base de données)

- Création / ouverture / fermeture de la base de données : 1 point
- Création de tables :
  - Syntaxe SQL correcte : 1 points
  - Spécifications respectées : 1 point
  - Contraintes : 1 point
- Insertion de données :
  - Syntaxe SQL correcte : 1 point
  - Spécifications respectées : 1 point
  - Renvoi correct de l'ID de la dernière entrée ajoutée : 1 point
  - Insertion multiple : 1 point
- Lecture de données (personnages, matchs en cours) :
  - Syntaxe SQL correcte : 1 point
  - Spécifications respectées : 1 point
  - Lecture multiple : 1 point
  - Ordre des personnages par score décroissant : 1 point
- Lecture de données avec jointures de tables (matchs) :
  - Syntaxe SQL correcte : 1 point
  - Spécifications respectées : 1 point
- Suppression de données
  - Syntaxe SQL correcte : 1 point
  - Spécifications respectées : 1 point
- Modification de données
  - Syntaxe SQL correcte : 1 point
  - Spécifications respectées : 1 point
- Syntaxe Python : 2 points

Total : 20 points

3.2 Notation des rendus 2 et 3

Pour les rendus suivants, chaque fonction ou partie de fonction à réaliser ou compléter est évaluée de la manière suivante :
- Les fonctionnalités codées respectent les spécifications : 2 points
- La syntaxe Python est correcte : 1 point
- Si non-fournies par l'enseignant (modules d'initialisation et d'évolution de la base de données), les spécifications et la documentation sont cohérentes : 2 points

À cela s'ajoutent pour chaque rendu complet 2 points de respect de bonnes convention de code : nommage approprié des variables, commentaires si nécessaire.

Conclusion

Le projet présenté dans ce dossier, en plus d'offrir un cas pratique pertinent pour l'utilisation du langage SQL, permet d'enseigner la modularité de manières variées : exploitation d'une API web, exploitation de modules Python existants (SQLite, Flask, Jinja), implantation de modules Python selon de spécifications fournies (Interface BDD, Routage, Vues Flask) et création de deux modules simples complets liés au projet (Initialisation BDD et Évolution BDD) et d'un module avant le projet lui-même (annuaire). Les élèves conçoivent une application modulaire, mettant en évidence l'intérêt d'une approche modulaire et la distinction entre interface et implantation.