Projet Pendu

Le projet Pendu est basé sur l'un des jeux les plus simples qui soit et qui se joue traditionnellement à deux.
L'objectif du projet est d'associer la partie "interface joueur" à un serveur qui sera le seul à décider et à connaître le mot à deviner. Le client (*qui est pratiquement réduit à un IHM « joueur »*) n'aura ici que des indications.

Pour que le joueur puisse se connecter au serveur, il doit être authentifié en tant qu'invité avec une clé chiffrée.

Le serveur lui donne comme seuls renseignements : Un numéro (*jeton échangé à chaque requête*), la taille du mot et le nombre maxi d'essais infructueux, puis les lettres bien placées, le nombre d'essais infructueux et la fin de partie.

Comment jouer ?

Un serveur est installé en local sur la machine (*en Python ou en Javascript/node.js*). Il est accessible en mode localhost http://localhost:port/chemin ou http://127.0.0.1:port/chemin
Dès que le serveur est lancé, on peut soit lancer le client à partir de sa localisation sur le disque (*cas du client écrit en Python ou en Javascript avec le protocole file:///*), soit accéder à la racine du serveur pour obtenir automatiquement le client sous la forme d'une page web (*possible à partir de tout serveur, qu'il soit écrit en Python ou en Javascript/node.js*)
Le respect de l'architecture API Web REST permet alors d'autoriser tout type de client (*Python ou page web/Javascript ou tout autre langage*), et ceci, quelque soit la technologie utilisée pour le serveur (*valable, donc, pour les 2 versions*).

Organisation du projet

Le projet se compose :

1. d'un serveur Web (*Python + Flask ou Javascript/node.js + express*) qui peut :
   - fournir une page web client avec ses fichiers associés (*.css, .js, .jpg…*),
   - communiquer avec un client (*architecture API WEB REST*).
2. d'un client (*Tkinter/Python ou page web/Javascript*) qui peut communiquer avec le serveur pour récupérer un numéro (*le jeton*) associé au mot choisi par le serveur.

Le serveur fournira la longueur du mot associé et le nombre maximum d'essais infructueux autorisés.
A chaque clic sur une lettre (*et/ou, en option dans le client, suite à un appui sur une touche du clavier*), le client envoie une requête au serveur avec toutes les lettres déjà jouées.
Le serveur répondra sous la forme suivante : status, positions des lettres correctes dans le mot, nombre de lettres bien placées, nombre d'essais infructueux, fin de la partie

Les fichiers utiles au serveur sont :

- version Python : server.py, mots.txt,
- version Javascript/node.js : serveur.js, lesMots.js (*en option : un fichier pour la base de données : pendu.db*).

Les fichiers utiles au client sont :

- image pendu4x4.gif de 720x720 pixels (*contenant 4x4 images de 180x180 pixels montrant l'évolution du jeu*),
- version Python : pendu.py,
- version Javascript : index.html, pendu.js, pendu.css et jquery.js.

Parties indépendantes : Travail à faire

Il est possible de travailler indépendamment sur plusieurs parties du projet. Les parties les plus intéressantes à faire sont :

- développement d'un client html/Javascript (*fichiers html, css, image et bibliothèque jquery déjà fournis*) ou Python (*on utilisera alors l'un des serveurs fournis en le lançant en local*),
- développement du serveur Javascript/node.js ou Python,
  - partie Web statique chargée de fournir le client dans sa version page web,
  - partie API WEB REST pour échanger avec tout type client,
  - développement de la partie BDD.

Il est aussi possible d'améliorer ce qui est fourni.
Par exemple :

- Le client pourrait permettre à l'utilisateur de choisir le niveau de jeu,
- Le serveur pourrait ne fournir, selon le choix du client, que des mots de 6 lettres ou 7 ou 8 lettres.

Communication Client / Serveur

La communication est réalisée au travers d'une API simple. Le client peut interroger le serveur ou lui envoyer des informations au travers de requêtes Web ordinaires (*types GET, POST ou PUT*). Le serveur répond en utilisant le format Json, facile à traiter par le client.
Toutes les requêtes ont la même forme :

- http://adresse_serveur:port => récupération de la page web
- http://adresse_serveur:port/ commande/login/cle => requêtes API REST

où :

- login est le login utilisé par le client
- cle est une clé au format uuid que l'utilisateur a obtenu auparavant
- il n'y a, ici, qu'une seule commande : jeu
- 2 requêtes sont nécessaires et sont définies par les verbes READ (*Get*) et UPDATE (*Post ou Put*)

Requête READ (*Get*) :

Get à l'URI http://127.0.0.1:port/jeu/invite/06064a1f-e01e-4e0b-b0c0-e5541b74f45e

Réponse : { "etat": 1, "jeton": 9875578754, "mot": "_ _ _ _ _ _ _ _ ", "max": 14 }

Requête UPDATE (*Post ou Put*) :

Post ou Put à l'URI http://127.0.0.1:port/jeu/invite/06064a1f-e01e-4e0b-b0c0-e5541b74f45e
Post : { "jeton": 9875578754, "lettres": "abcde" }

Réponse : { "status": 1, "jeton": 9875578754, "mot": "_ _ B _ _ _ E _ ", "lettres": "abcde",
"echecs": 3, "max": 14, "gagne": false, "perdu": false }

En Python, la conversion du Json en dictionnaire est immédiate.
En Javascript, la structure Json est immédiate car c'est celle qui définit une structure objet de base.

Base de données (*proposée en option avec un serveur node.js*)
Une base de données, déjà créée (*modifiable*), peut être utilisée pour fournir un mot au client et comparer la clé fournie à celle correspondant au login.

CREATE TABLE mots (id INT PRIMARY KEY, taille INT, mot TEXT)
CREATE TABLE keys (login TEXT PRIMARY KEY, key TEXT)

Il y a 2 tables : TABLE mots et TABLE keys
La table mots contient 3 colonnes : id (*clé primaire*), taille et mot
La table keys contient 2 colonnes : login (*clé primaire contenant le login de l'utilisateur*), et key, contenant un uuid. Cet uuid est la clé qu'il faut rappeler dans les requêtes API.

Exemples de syntaxes de requêtes pour interroger la base de données :
"SELECT Count(*) FROM mots" pour connaître le nombre de mots
"SELECT Count(*) FROM mots WHERE taille=6" pour connaître le nombre de mots de 6 lettres
"SELECT * FROM mots WHERE id=40" pour récupérer le mot n° 40
"SELECT * FROM keys WHERE login=invite" pour récupérer la clé associée

Note : Ces requêtes ne peuvent être faites que par le serveur