2021-10-22

Audacix code exercise

The goal of this exercise is to get an overview of your coding & problem solving abilities. Your submission will be judged on:

- Correctness. Does the code fulfill the requirements?
- Cleanliness. Is the code written in a clean manner and follows the accepted Python coding conventions? Is it easily understandable to a developer looking at it for the first time?
- Performance. Does the code make use of appropriate data structures? Did you have performance in mind when developing the algorithms?

What is required?

A Django Rest Framework based API that can be used to power the frontend for a hangman game. You are not expected to integrate it with any frontend.

You can use any database backend that you are comfortable with. However, `sqlite` is advised as it makes testing easier for our engineers.

Your app should expose an API with the following endpoints. All requests and responses should use the JSON data type.

New game
The `/game/new` endpoint should start a new game. It should assign a random word from the list of following words for the player to guess:

```
["Hangman", "Python", "Audacix", "Bottle", "Pen"]
```

The player should be allowed to guess incorrectly $1/2$ the number of characters in the word. So for example if the selected word in "Pen", the player can make 2 incorrect guesses before loosing the game.

Your application should have database models to store a game and it's complete state. The state you need to store will need to be derived from the functionality that is required as defined in this document.

This endpoint should return the id of the newly created game object.

Game state
The `/game/<:id>` endpoint should accept a game id and return a state which contains:

- The current state of the game. `InProgress`, `Lost`, or `Won`.

- The current state of the word. For example, if the word to guess in "Pen" and the player has correctly guessed the letter "P", the state of the word should be "P__". The letters that are still to be guessed are replaced by underscores.
- The number of incorrect guesses the player has already made.
- The number of incorrect guesses the player can still make.

Guess

The `/game/<:id>/guess` endpoint should accept a single character and return both the game state as defined in the **Game state** endpoint and if the guess was correct or not.

This endpoint should update the game state before returning it.