Prueba BackEnd Software Engineer - LQN

Parte 1 - Ejercicios de lógica.

a. Desarrolla un algoritmo que imprima los números del 0 al 100. Adicionalmente debe imprimirse en la misma línea la palabra buzz en caso de que el número sea par. Sí el número es múltiplo de 5 debe imprimirse en la misma línea la palabra bazz.

b. (Inglés) Your task in this exercise is as follows: Take the following selection of 70 English Pokemon names (extracted from Wikipedia's list of Pokemon), and generate the/a sequence with the highest possible number of Pokemon names where the subsequent name starts with the final letter of the preceding name. No Pokemon name is to be repeated.

   Pokemon list:
   audino bagon baltoy banette bidoof braviary bronzor carracosta charmeleon cresselia croagunk darmanitan deino emboar emolga exeggcute gabite girafarig gulpin haxorus heatmor heatran ivysaur jellicent jumpluff kangaskhan kricketune landorus ledyba loudred lumineon lunatone machamp magnezone mamoswine nosepass petilil pidgeotto pikachu pinsir poliwrath poochyena porygon2 porygonz registeel relicanth remoraid rufflet sableye scolipede scrafty seaking sealeo silcoon simisear snivy snorlax spoink starly tirtouga trapinch treecko tyrogue vigoroth vulpix wailord wartortle whismur wingull yamask

Parte 2 - Mini proyecto

Objetivo

Agregar nuevos features a un proyecto que contiene un API GraphQL para un sitio web para los fanáticos de Star Wars.

Descripción

Se debe modifica runa aplicación existente. La aplicación está escrita en Python y Django, el desasrrollador original ya no está disponible y se nos hace nesario agregar funcionalidad nueva al sitio.

Repositorio

https://github.com/LQNTech/swapi-back

Consideraciones

1. En el modelo **People** deben modificarse los campos: **hair_color** y **eye_color**. Toma en cuenta que el cambio que se aplique en el modelo debe reflejarse en el GraphQL. Se requiere que los 2 campos tengan los siguientes choices:

    a. hair_color: BLACK, BROWN, BLONDE, RED, WHITE, BALD.
    b. eye_color: BLACK, BROWN, YELLOW, RED, GREEN, PURPLE, UNKNOWN.
2. Documentar con un docstring la función ubicada en swapi/app/utils llamada **generic_model_mutation_process.** Es importante que quede claro que hace la función y cuando debe utilizarse.
3. La única mutación que está funcionando en el proyecto lleva el nombre de: **AddPlanetMutation**, sin embargo su nombre no es claro con respecto a lo que realiza la mutación. Es necesario cambiar el nombre de la mutación por un nuevo nombre que refleje lo que hace la mutación.
4. Es necesario crear una nueva mutación que nos permita crear personajes. Debes tomar en cuenta que debe ser posible asignarle las películas en las cuales ha participado.
5. Es necesario crear una nueva mutación que nos permita actualizar toda la información de personajes.
6. Al agregar nuevo código a la aplicación se hace necesario crear pruebas unitarias que nos garanticen que tanto el código antiguo cómo el nuevo funcionen sin inconvenientes. Es necesario crear pruebas unitarias para la(s) mutación(es) de creación y edición de personajes.
7. Es necesario poder filtrar en el ***query allPeople*** por los valores de género. Es indispensable que el explorador de graphene (GraphiQL) nos sugiera cuáles son las opciones válidas para ese filtro. Es decir, al filtrar por género desde la URL "**/graphql**" utilizando el ***query allPeople***, si enviamos el parámetro/variable de **gender** debe mostrarse la sugerencia de cuáles son las opciones válidas para el género. En este caso, las opciones válidas son: *male, female, hermaphrodite y n/a*.

Plus

1. Crear/actualizar un README con instrucciones para correr el proyecto.
2. Aplicar formateo **Flake8** en todo el código que agregues.
3. Crear un Dockerfile que me permita generar una imagen de **Docker** del proyecto.
4. Implementación de **PyTest** para pruebas unitarias.
5. Generar un collection en **Postman** y adjuntarlo en el repositorio para verificar las nuevas mutaciones y el cambio en el *query* de *allPeople*.
6. Resolver los TODOs dentro del proyecto.
7. Desplegar el proyecto en algún un servicio online gratuito u hosting personal.

Tecnologías

1. Python / Django.
2. GraphQL.
3. Graphene.
4. Relay.
5. Docker.

Cómo debes presentar el proyecto

Debes crear un repositorio público en **Github** y subir tu prueba a él. Recuerda que debes incluir los ejercicios de lógica también.

Debes compartir el enlace del proyecto respondiendo el correo electrónico por el cual se te hizo llegar la prueba.

Si decides realizar el plus de desplegar tu proyecto a un sitio online también debes compartirnos la dirección a la cual podemos acceder para revisarlo.

Considera que también evaluamos la forma en que utilizas el git.

Recursos

1. https://docs.graphene-python.org/projects/django/en/latest/
2. https://docs.graphene-python.org/projects/django/en/latest/tutorial-relay/
3. https://graphql.org/

Tienes 3 días para el desarrollo de la prueba, una vez la termines te pedimos nos compartas el repositorio de tu prueba al email desde el cual recibiste la prueba para que nuestra área técnica la revise.

¡Esperamos tenerte pronto en nuestro equipo!