Proyecto de Curso

Tema: Python como herramienta multipropósito

Fecha de entrega máxima: 15 Julio 2022, 23:59

Este proyecto es autoguiado, es decir el estudiante deberá tomar ventaja de las diapositivas, el internet, la documentación oficial, o cualquier herramienta disponible para conseguir información correspondiente que le permita descubrir soluciones a los problemas presentados. El proyecto está diseñado para ser completado en un tiempo máximo de **20 horas**, por lo que se cubrirán temas relativamente poco complejos.

Detalles

En este proyecto expondremos una simple API REST para acceder a ciertas funciones de una escuela, adicionalmente un script mediante el cual realizaremos análisis de su base de datos. El proyecto es **individual** y se calificará de acuerdo con la rúbrica especificada al final del presente documento.

REPOSITORIO

1. Crearemos un repositorio en GitHub con el fin de mantener el código del proyecto. Para este propósito dirigirse al servicio y crear una cuenta en caso de no tenerla.
2. Una vez que tengamos una cuenta, crear un repositorio con el nombre **"ista-python-curso-2022".**
3. El estudiante puede decidir si el repositorio es privado o público, sin embargo, será necesario que el capacitador tenga acceso, por lo que, en caso de crear un repositorio privado, agregar como colaborador a la cuenta "**danoc93"**.

ESTRUCTURA DEL REPOSITORIO

4. En el repositorio crearemos un archivo **README.md** que deberá contener el nombre completo del estudiante, así como cualquier detalle relevante sobre el código que deseen compartir.
5. En el repositorio crearemos una carpeta "**escuela_api**" que será donde vivirá todo nuestro código de Python para el servicio.
6. En el repositorio crearemos una carpeta "**analisis**" que será donde escribiremos código que nos permita analizar información de la base de datos de nuestra escuela.
7. Para simplificar el acceso a información y las dependencias, la base de datos será nada más una serie de archivos que leeremos y cambiaremos con la ayuda Python. Creemos la carpeta "**datos"** para este propósito.
8. El repositorio solo debe contener los artefactos esenciales, recomendado usar un ***.gitignore*** que excluya archivos innecesarios durante el desarrollo del proyecto https://www.toptal.com/developers/gitignore/api/python,flask.

Como podemos ver, nuestro repositorio en realidad contiene 2 proyectos Python, uno para el API, y otro para el análisis de datos. Lo único que estos dos proyectos tendrán en común es que ambos leerán la base de datos, pero con propósitos diferentes.

El nombre de esta clase de repositorio es *monorepositorio* (uno solo para varios proyectos).

BASE DE DATOS

Por simpleza, la base de datos es una serie de archivos CSV, un formato de texto plano que representa tablas mediante comas. Las dos tablas, **estudiante**, y **asistencia** están en el repositorio de la clase para que puedan copiarse sin problemas.

estudiante: una tabla que incluye los datos personales de los estudiantes, el identificador del estudiante es su número de cédula.

asistencia: una tabla que incluye los registros de asistencia de los estudiantes, el identificador del estudiante es su número de cédula.

9. Copiar las tablas CSV de ista-2022-junio-python/proyecto en nuestra carpeta "**datos**".
10. Agregar una línea a **estudiante.csv** con tus datos (NO ES NECESARIO USAR LA CÉDULA REAL).
11. Agregar 5 líneas a **asistencia.csv** con 5 días diferentes en la materia "**python**" para el estudiante agregado en el punto anterior.

Importante: Los pasos 9 y 10 son importantes ya que los usaremos en la sección de análisis.

PROYECTO: "escuela_api"

Queremos tener la habilidad de obtener cierta información sobre la asistencia de los estudiantes de nuestra escuela.

12. Agregar **flask, pytest** a un archivo **requirements.txt** dentro de este proyecto, e instalarlo mediante **pip**.
13. Creemos un paquete para un proyecto básico de **flask** dentro de esta carpeta (recuerda el __init__.py), con el módulo principal llamado **api.py**.
14. Creemos un punto de acceso *GET* en nuestra API **"lista_estudiantes**" que lea la tabla **estudiante** de nuestro repositorio y devuelva una lista de objetos JSON, con la lista de nombres y apellidos ordenados en orden ascendente.
15. Creemos un punto de acceso *POST* en nuestra API "**registro_asistencia**" que reciba una cédula, nombre de materia, año, mes y hora, y las registre en la tabla **asistencia**.

Importante: Ya que la base de datos es una serie de archivos, no necesitamos instalar nada complejo para leer los mismos.

PROYECTO: "analisis"

En una institución de cualquier tipo es común hacer uso de herramientas como Python para analizar los datos de múltiples fuentes de datos con el fin de responder a preguntas que permitan tomar decisiones o planificar diferentes aspectos del negocio o institución. Esto se conoce como Inteligencia Empresarial o Business Intelligence (BI).

En este proyecto queremos tener la habilidad de que los directores de la escuela puedan analizar los diferentes patrones de asistencia de los estudiantes por lo que vamos a trabajar con la misma base de datos que la API.

16. Agregar **pandas, numpy, matplotlib, pytest** a un archivo **requirements.txt** dentro de este proyecto, e instalarlo mediante **pip**.
17. Creemos un paquete dentro de la carpeta (recuerda el __init__.py).
18. Creemos un módulo donde vivirá nuestro código de análisis, llamado **reporte_asistencias.py.**
19. En el módulo usemos pandas para crear un DataFrame que lea la tabla **estudiante** directamente desde su ubicación en el disco.
20. En el módulo usemos pandas para crear otro DataFrame que lea la tabla **asistencia** de igual manera.
21. Usemos pandas para combinar ambos dataframes en uno nuevo llamado **asistencias_completas** de manera que cada fila de asistencia también tenga los nombres completos del estudiante *(Tip: Usar merge)*.
22. Imprimamos el dataframe a la pantalla.
23. Usando la información en **asistencias_completas** crear un dataframe que incluya solo los datos del estudiante a cargo del proyecto.
24. Guardar el dataframe en un archivo reporte_xxx.csv donde xxx es la cédula del estudiante.
25. Usando **matplotlib** crea una visualización de asistencias totales por materia para el estudiante.

TESTS

26. Dentro de cualquier proyecto crea una carpeta **tests** (recuerda el __init__.py)**,** con un archivo **test_unidad.py**. Este archivo puede ser tan complejo como desees, lo importante es entender como llamarlos correctamente mediante **pytest**.

EXTRAS

Es importante que se implementen estos elementos SÓLO SI EXISTE TIEMPO SUFICIENTE.

Indicar en el archivo **README.md** si alguno de estos ítems ha sido implementado por el estudiante.

1. En el curso aprendimos los beneficios de los tipos, ajustemos el código para incluirlos.
2. Crear un punto de acceso en la API que genere una tabla HTML de la lista de alumnos en lugar de devolver un objeto JSON.
3. Cambia la base de datos a base de archivos CSV por un mecanismo más eficiente. Por ejemplo, tinydb, o inclusive algo basado en SQL como sqlite/MySQL.
4. Cambia el análisis para crear un reporte para un estudiante pasado como argumento a la consola.
5. Ejecuta chequeos en tu repositorio, por ejemplo para tests, explora Github Actions.

Calificación

Enviar el nombre del repositorio y sus datos personales a la siguiente ficha:

https://forms.gle/Ty8SX8t2PRKckEAc9

El código se calificará en su estado a las 11:59pm del 15 de Julio, es decir que deben asegurarse que todas los cambios se encuentren publicados a esta hora. **Cualquier cambio luego de esta fecha no se considerará.**

Rubro	**Descripción**	**%**
Repositorio	README.md correspondiente y estructura de carpeta de acuerdo con la guía.	3
Base de datos	Cambios a los archivos originales.	2
API escuela	API flask de acuerdo con los requerimientos.	45
Análisis	Código de Análisis de acuerdo con los requerimientos.	45
Tests	Tests en cualquiera de los proyectos.	5
		100
Extras	Cualquier tarea adicional implementada.	5