BRAIN-TEC

TAREA: *"PHP ADDRESS DATABASE"*

1)Primeros vistazos y decisiones

Tal y como lo hablé con el Sr. José Luis Benito, la primera decisión que tomé fue la de utilizar Python en lugar de PHP ya que creo que, siendo el puesto al que aplico uno de desarrollador Odoo, sería un mejor indicador para el puesto un proyecto escrito utilizando lenguajes y tecnologías propias de Odoo como son Python y la librería Flask.

2)Primera aproximación al problema

El primer paso que di fue el de crear la base de datos así que escribí el siguiente código SQL (escrito aquí para una mejor lectura):

```
CREATE DATABASE IF NO EXISTS exercise CHARACTER SET utf8;
USE exercise;
DROP TABL IFEXISTS contact;
CREATE TABLE contact (
        contact_id        MEDIUMINT NOT NULL AUTO_INCREMENT,
        firstnameVARCHAR(100) NOT NULL,
        lastname VARCHAR(100) NOT NULL,
        address  VARCHAR(200),
        /* Since this is a contact-book, the email can't be repeated... */
        email             VARCHAR(100) NOT NULL UNIQUE,
        /* […] neither the phone number. */
        phone             VARCHAR(30) UNIQUE,
        PRIMARY KEY (contact_id)
);
```

Además de establecer la clave primaria, decidí establecer el mail y el teléfono como campos únicos, para evitar duplicar contactos con los mismos datos.

Para esta aplicación me decidí por una estructura MVC así que cree la siguiente estructura de directorios junto con sus contenidos:

- **Modelo:** Solamente uno, el modelo para el Contacto
- **Vistas:** En este campo tenemos al menos dos vistas distintas, la que se encarga del formulario y la que se encarga de mostrar la lista de contactos.
- **Controladores:** Aquí añadí el código que controla el enrutamiento de la aplicación.

3)Flujo de trabajo y pensamientos

Primero creé el archivo `__init__.py` para inicializar la aplicación Flask. Además de esto escribí también el middleware `bd.py` para facilitar la conexión y manipulación de la base de datos.

Después de esto escribí el modelo del Contacto con todos los métodos necesarios y los tests correspondientes a estos métodos. Escribí los métodos para pasar los tests y luego cree las vistas (llamadas Templates en Flask) así como los controladores.

Después de crear y plantearme las vistas me percaté de que iba a necesitar algunas funcionalidades más fuera del modelo Contacto como por ejemplo la función `get_database_contacts()` que recupera todos los contactos de la base de datos o la función `load_xml_contact(xml_file)` la cual carga todos los contactos provenientes de un archivo XML en forma instancias de la clase Contacto. Una vez planteadas estas dos funciones escribí los tests que debían pasar las susodichas así como la funcionalidad necesaria para pasarlos.

Escribiendo estas funcionalidades me topé en la documentación de Python para la librería ElementTree que existía cierta vulnerabilidad a ataques sobre archivos XML malogrados y que sería necesario utilizar librerías externas para evitar este tipo de ataques. Debido a esto utilicé la librería `defusedxml` recomendada en la documentación para sanear los archivos XML subidos.

Luego de que todo estuviera funcionando escribí las vistas/templates del formulario de contacto así como de la lista de contactos, para crear una experiencia más usable y agradable utilicé el Framework Bootstrap 4, después escribí también los controladores para el formulario de contactos y la lista de contactos además del controlador encargado de gestionar la subida de archivos XML.

Para el código que modifica el DOM, para cambiar algunos colores así como crear una experiencia dinámica de navegación, utilicé Javascript y JQuery. Todo este código se sirve de forma estática utilizando las funcionalidades de Jinja2 incluidas en Flask

Esta fue la ruta principal sin embargo a lo largo del proyecto reescribí los tests para adaptarme a ciertas restricciones que iba descubriendo y por tanto refactoricé el código más de una vez. Además de, por supuesto, crear algo del código boilerplate requerido para cualquier aplicación Flask como es el archivo `config.py`.

4)Pensamientos finales y comentarios

Después de terminar la tarea pienso que encierra más complejidad de la que pensé en un principio como por ejemplo el saneamiento de las declaraciones SQL o de los archivos XML. Fue un corto proyecto pero he aprendido mucho de él.

Para configurar y ejecutar el proyecto por favor lea el archivo README.md que se encuentra dentro del directorio del proyecto.