Caso Data Funtastics

En una compañía creativa llamada **DataFuntastics**, quieren implementar un proyecto Django que integre tres aplicaciones separadas, cada una con la capacidad de generar y descargar un archivo único. La idea es mostrar las posibilidades de Django para manejar diversos tipos de contenido y formatos en aplicaciones especializadas, de una forma accesible y divertida.

Descripción: DataFuntastics desea contar con un sistema donde cada aplicación cumpla una función específica relacionada con contenido descargable. Los equipos quieren tener ejemplos de generación de archivos PDF, Excel y gráficos PNG, usando librerías como **ReportLab**, **OpenPyXL** y **Matplotlib** respectivamente.

Criterios de Aceptación

1. **Estructura de las Aplicaciones**:
    - El proyecto Django deberá contener tres aplicaciones, cada una con un `view index` que devuelva un `HttpResponse` con el texto `"nombre de la aplicación index"`.
    - Las aplicaciones tendrán los siguientes nombres y funciones:
        - **App 1 - Librarium**: Genera y devuelve un archivo PDF.
        - **App 2 - SheetMaker**: Genera y devuelve un archivo XLS.
        - **App 3 - Graphify**: Genera y devuelve un archivo PNG.

2. **Funciones de Cada Aplicación**:
    - **Librarium (App 1)**:
        - Debe contener una vista `index` que retorne un `HttpResponse("Librarium index")`.
        - Además, debe contar con una vista `download_pdf` que genere un archivo PDF simple usando la librería **ReportLab** y permita descargarlo.
    - **SheetMaker (App 2)**:
        - Debe contener una vista `index` que retorne un `HttpResponse("SheetMaker index")`.
        - Además, debe contar con una vista `download_xls` que genere un archivo XLS simple usando **OpenPyXL** y permita descargarlo.
    - **Graphify (App 3)**:
        - Debe contener una vista `index` que retorne un `HttpResponse("Graphify index")`.
        - Además, debe contar con una vista `download_png` que genere un gráfico PNG simple usando **Matplotlib** y permita descargarlo.

3. **Publicación y Repositorio Git**:
    - El proyecto debe ser publicado en un repositorio Git público.
    - Incluir un archivo `requirements.txt` con las dependencias necesarias (`django`, `reportlab`, `openpyxl`, `matplotlib`).
    - El README.md debe contener instrucciones claras para clonar el proyecto y ejecutarlo en un entorno local.

4. **Divertido Reto Extra**:
    - Cada archivo descargable debe contener algún elemento humorístico:
        - El **PDF** en **Librarium** incluirá un mensaje con "Bienvenido al mundo de los libros digitales... ¡Disfruta este PDF!"
        - El **XLS** en **SheetMaker** incluirá una hoja con la celda A1 que diga "Bienvenido a la magia de los datos en Excel... ¡Vamos a analizar!"
        - El **PNG** en **Graphify** debe tener un gráfico de barras con leyendas como "Datos divertidos del día".

5. **No olvidar:**
    - Crear el entorno venv.
    - Activar el entorno.
    - Instalar las librerías usando pip.
    - Incluir el archivo .gitignore, ignorando el entorno venv.
    - Usar git con la terminal o github desktop.
    - Crear el proyecto en github.com

Ejemplo de la estructura del proyecto.

```
datafuntastics/
├── datafuntastics/
│   ├── __init__.py
│   ├── settings.py            # Configuración del proyecto Django
│   ├── urls.py                # URLs principales del proyecto
│   ├── asgi.py
│   └── wsgi.py
│
├── librarium/                 # Primera aplicación
│   ├── admin.py
│   ├── models.py
│   ├── tests.py
│   ├── urls.py                # URLs específicas de Librarium
│   └── views.py  .....        # Vistas para Librarium
│
├── sheetMaker/                # Segunda aplicación
│   ├── admin.py
│   ├── apps.py
│   ├── migrations/
│   ├── models.py
│   ├── urls.py                # URLs específicas de SheetMaker
│   └── views.py ...           # Vistas para SheetMaker
│
├── graphify/                  # Tercera aplicación
│   ├── apps.py
│   ├── migrations/
│   ├── models.py
│   ├── tests.py
│   ├── urls.py                # URLs específicas de Graphify
│   └── views.py   ....        # Vistas para Graphify
├── manage.py                  # Archivo de gestión principal de Django
├── requirements.txt           # Dependencias del proyecto
```