**Documentación del Proceso de Instalación de Librerías y Configuración del Entorno**

**1. Introducción**

En este documento, detallaré los pasos realizados para la configuración inicial del entorno de desarrollo y la gestión de dependencias del proyecto. Este proceso garantiza la reproducibilidad del entorno y la consistencia entre equipos, siguiendo las mejores prácticas para el manejo de librerías y dependencias en proyectos Python.

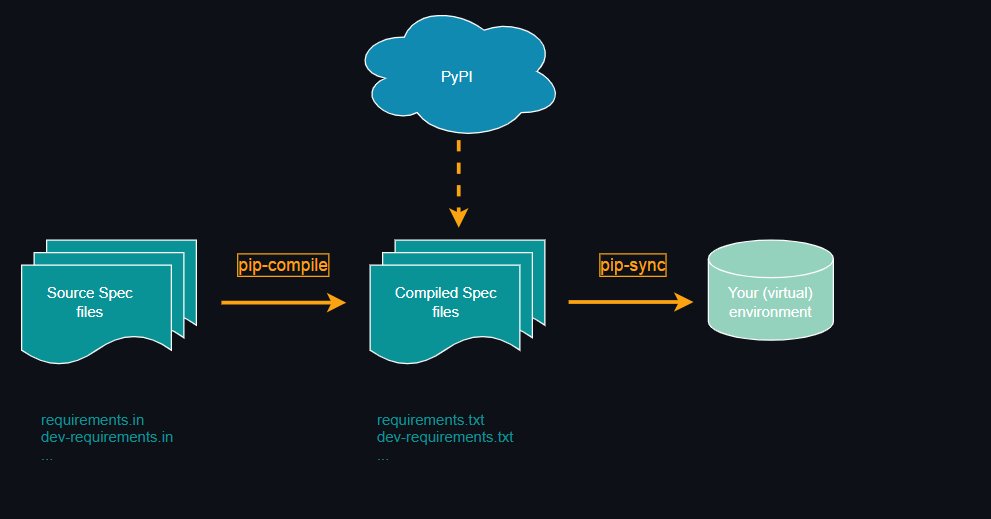
**2. Creación de Archivos de Dependencias**

**Documentación del Proceso de Instalación de Librerías y Configuración del Entorno**

**1. Introducción**

En este documento, detallaré los pasos realizados para la configuración inicial del entorno de desarrollo y la gestión de dependencias del proyecto. Este proceso garantiza la reproducibilidad del entorno y la consistencia entre equipos, siguiendo las mejores prácticas para el manejo de librerías y dependencias en proyectos Python.

**2. Creación de Archivos de Dependencias**

****

**https://pip.pypa.io/en/stable/user\_guide/#requirements-files**

**2.1 Archivo requirements.in**

El archivo requirements.in contiene las dependencias principales necesarias para ejecutar el proyecto. Estas librerías son:

* **pandas**: Para la manipulación y análisis de datos.
* **numpy**: Para cálculos matemáticos y manejo eficiente de arrays.
* **matplotlib**: Para la visualización de datos.
* **seaborn**: Para visualizaciones estadísticas avanzadas.
* **jsonlines**: Para la exportación de datos en formato JSONLines, compatible con bases de datos no relacionales como MongoDB.
* **sqlite3**: Incluido para trabajar con bases de datos SQLite.

Este archivo no incluye versiones de las librerías, ya que se delega a pip-compile la resolución de las versiones compatibles.

**2.2 Archivo dev-requirements.in**

El archivo dev-requirements.in incluye las dependencias necesarias para el desarrollo del proyecto, como pruebas, formateo de código y validación del estilo:

* **pytest**: Para la realización de pruebas automatizadas.
* **black**: Para el formateo automático del código.
* **flake8**: Para la validación del estilo del código y la detección de errores.

Estos archivos separan las dependencias principales de las de desarrollo, lo que sigue el principio de "separación de preocupaciones" y asegura que el entorno de producción no incluya herramientas innecesarias.

**3. Generación de Archivos de Dependencias Compilados**

Se utilizó la herramienta pip-tools para generar los archivos compilados requirements.txt y dev-requirements.txt, que incluyen las versiones exactas de todas las dependencias (directas e indirectas). Esto asegura la reproducibilidad del entorno.

**Pasos realizados:**

1. Ejecutar el siguiente comando para generar el archivo requirements.txt:
2. pip-compile requirements.in
   * Este comando resuelve las versiones compatibles de las dependencias principales y sus sub-dependencias, consultando el repositorio oficial de PyPI.
3. Ejecutar el siguiente comando para generar el archivo dev-requirements.txt:
4. pip-compile dev-requirements.in
   * Este comando sigue el mismo proceso para las dependencias de desarrollo.

**Resultado:**

* Los archivos requirements.txt y dev-requirements.txt contienen las versiones bloqueadas de todas las dependencias, garantizando compatibilidad y estabilidad en el entorno.

**4. Creación y Configuración del Entorno**

El entorno virtual fue creado y configurado utilizando Conda, con los siguientes pasos:

**Pasos realizados:**

1. Crear el entorno virtual llamado "PC1":
2. conda create --name PC1
   * Esto crea un entorno virtual aislado llamado "PC1".
3. Activar el entorno:
4. conda activate PC1
   * Con este comando, el entorno virtual "PC1" pasa a ser el entorno activo.
5. Instalar pip en el entorno:
6. conda install pip
   * Este paso asegura que pip esté disponible para instalar librerías adicionales.
7. Generar los archivos requirements.txt y dev-requirements.txt usando pip-compile (como se describe en la sección 3).
8. Sincronizar el entorno con las dependencias especificadas:
9. pip-sync requirements.txt dev-requirements.txt
   * Este comando instala las dependencias bloqueadas en los archivos generados, asegurando que el entorno contenga exactamente las versiones especificadas.

**Ruta del proyecto:** El entorno está configurado en la ruta:

C:\Users\Oscar\Documents\MBIT Óscar\MBIT\_202501\_Proyecto\_Consolidacion\_1

En esta ruta se encuentran los archivos .in, .txt y otros componentes del proyecto.

**5. Instalación de Dependencias**

Para instalar las dependencias en el entorno virtual, se utilizó el comando pip-sync, que asegura que el entorno contenga exactamente las versiones especificadas en los archivos compilados.

**Pasos realizados:**

1. Instalar dependencias principales:
2. pip-sync requirements.txt
3. Instalar dependencias de desarrollo:
4. pip-sync dev-requirements.txt

**Nota:** Este proceso elimina cualquier librería instalada previamente que no esté en los archivos de dependencias compilados, asegurando un entorno limpio y consistente.

**6. Justificación de Decisiones**

* **Uso de pip-tools:**
  + Automatiza la resolución de dependencias, asegurando compatibilidad entre versiones.
  + Genera archivos con versiones bloqueadas, facilitando la reproducibilidad del entorno.
* **Separación de dependencias (requirements.in y dev-requirements.in):**
  + Evita incluir herramientas de desarrollo en el entorno de producción, reduciendo riesgos y optimizando recursos.
* **Uso de pip-sync:**
  + Asegura que el entorno virtual refleje exactamente las dependencias especificadas, evitando inconsistencias.

**7. Conclusión**

El proceso descrito garantiza un entorno bien configurado, reproducible y compatible, siguiendo las mejores prácticas para la gestión de dependencias en Python. Esta documentación asegura que cualquier miembro del equipo pueda replicar el entorno de manera eficiente y sin errores.

**Próximos pasos:**

* Continuar con la configuración y documentación de las próximas etapas del proyecto, como la manipulación y análisis de datos.