Manual Apache Airflow

Apache Airflow, PostgreSQL, Next JS

Vicente Alves O.

16 de enero 2025

Resumen

Este manual tiene como objetivo proporcionar una guía detallada para la instalación, configuración y puesta en marcha de las herramientas necesarias para el desarrollo de un proyecto web utilizando tecnologías clave como **Apache Airflow**, **Next.js** y **PostgreSQL**.

El documento aborda desde los aspectos más básicos, como la instalación de cada herramienta, hasta la integración de las mismas para el correcto funcionamiento de un sistema de gestión de prácticas.

El manual comienza con la instalación de **Apache Airflow**, una plataforma de orquestación de flujos de trabajo, para gestionar tareas automatizadas relacionadas con la carga, transformación y análisis de datos.

Se detallan los pasos para configurar y ejecutar Airflow, incluyendo la creación de flujos de trabajo (DAGs) y la configuración de variables y conexiones necesarias para interactuar con la base de datos.

En la segunda parte, se aborda la instalación de **Next.js**, un framework de React, para desarrollar una interfaz web interactiva y dinámica que permita a los estudiantes visualizar sus prácticas y realizar un seguimiento de su progreso. Se incluyen los pasos para configurar el proyecto Next.js, así como la integración con **PostgreSQL** como sistema de gestión de base de datos.

La tercera sección se centra en la instalación y configuración de **PostgreSQL**, una base de datos relacional que se utilizará para almacenar la información sobre las prácticas de los estudiantes. Se incluyen instrucciones detalladas para la instalación en sistemas basados en Linux, así como la configuración de usuarios y contraseñas.

Finalmente, se guía al usuario en la configuración del proyecto de prácticas en **Apache Airflow** y **Next.js**, detallando cómo migrar y gestionar los flujos de trabajo, establecer las variables necesarias y realizar las conexiones a la base de datos. Con todas estas configuraciones en su lugar, el sistema está listo para ejecutarse, proporcionando una solución integral para la gestión de prácticas y el seguimiento de actividades de los estudiantes.

Índice

1 Introducción **5**

2 Tecnologías **6**
2.1 Diagramas del Proyecto . . . 8
2.1.1 Diagrama de componente . . . 8
2.1.2 Diagrama de secuencia (Practicante) . . . 9
2.1.3 Diagrama de secuencia (Oferente de la práctica) . . . 10

3 Instalación y Configuración de Airflow (Máquina Local) **11**
3.1 Prerrequisitos . . . 11
3.2 Instalación de Airflow . . . 11
3.2.1 Instalar Airflow . . . 11
3.2.2 Agregar dependencias adicionales (Base de Datos) . . . 11
3.2.3 Inicialización de Airflow . . . 11
3.3 Estructura Generada por Airflow . . . 12
3.4 Cómo Lanzar Apache Airflow . . . 13
3.4.1 Iniciar el Servidor Web . . . 13
3.4.2 Ejecutar el Programador de Tareas (Scheduler) . . . 13
3.5 Diagrama de Arquitectura de Airflow . . . 13

4 Instalación y Configuración de Airflow (Contenedor Docker) **14**
4.1 Prerrequisitos . . . 14
4.2 Pasos para la Instalación . . . 14
4.2.1 Crear un archivo Docker Compose . . . 14
4.2.2 Preparar el Entorno . . . 14
4.2.3 Lanzar los Servicios . . . 14
4.2.4 Acceso a la Interfaz de Airflow . . . 14
4.2.5 Añadir DAGs . . . 14
4.2.6 Detener los Servicios . . . 14

5 Instalación y Configuración de Next.js **15**
5.1 Requisitos Previos . . . 15
5.2 Pasos para la Instalación . . . 15
5.2.1 Configura tu Entorno . . . 15
5.2.2 Clonar Proyecto de Git . . . 15
5.2.3 Navega al Directorio del Proyecto . . . 15
5.2.4 Inicia el Servidor de Desarrollo . . . 15

6 Instalación y Configuración de PostgreSQL **16**
6.1 Pasos para la Instalación . . . 16
6.1.1 Actualizar el Sistema . . . 16
6.1.2 Instalar PostgreSQL . . . 16
6.1.3 Configurar la Contraseña del Usuario . . . 16
6.1.4 Verificar la Instalación . . . 16
6.1.5 Iniciar el Servicio de PostgreSQL . . . 16
6.1.6 Habilitar PostgreSQL al Inicio . . . 16
6.2 Notas Extra . . . 16

7 Instalación de DBeaver (Opcional) **17**
7.1 Pasos para la Instalación . . . 17
7.1.1 1. Descargar DBeaver . . . 17
7.1.2 2. Instalar DBeaver . . . 17
7.1.3 3. Lanzar DBeaver . . . 17

8 Configuración del Proyecto de Prácticas **18**
8.1 Airflow . . . 18

8.2 Next.js 21

9 Proyecto de Prácticas En Funcionamiento **22**

10 Creación de Dags **24**
10.1 ¿Qué es un DAG? 24
10.2 Estructura de un DAG en Airflow 25
10.3 Variables de Airflow 25
10.4 Parámetros de un DAG 26
10.5 Manejo de Errores 26

11 Interfaz de Airflow **27**
11.1 Visión General de la Interfaz 27
11.2 Gestión de DAGs 28
11.3 Visualización del Grafo de Dependencias 28
11.4 Historial de Ejecuciones 29
11.5 Exploración de Logs 30
11.6 Administración del Sistema 31
11.7 Conclusión 32

1. Introducción

Este manual te guiará en la instalación y configuración de las herramientas necesarias para crear un entorno de desarrollo web moderno. Utilizando **Apache Airflow**, **Next.js** y **PostgreSQL**, aprenderás cómo implementar un sistema que permita gestionar flujos de trabajo automatizados y desplegarlo en un proyecto web .

Durante el proceso, instalarás y configurarás **Apache Airflow** para la orquestación de tareas, **Next.js** para el desarrollo de la interfaz web, y **PostgreSQL** como base de datos para almacenar la información de las prácticas. Todo esto se integrará para formar un sistema funcional que gestione de manera eficiente las prácticas de los estudiantes.

Requisitos del Sistema

Para seguir este manual, asegúrate de contar con los siguientes requisitos:

- Sistema operativo: Linux o WSL (Windows Subsystem for Linux).
- Node.js (versión 16 o superior) para trabajar con Next.js.
- PostgreSQL para gestionar las bases de datos.
- Python 3.x y `pip` para instalar Apache Airflow.
- Un editor de código como Visual Studio Code.

Con estos requisitos, estarás listo para empezar con la instalación y configuración de las tecnologías necesarias.

2. Tecnologías

Para este manual, se va a tener en consideración tres principales tecnologías que serán las directrices fundamentales para el desarrollo y despliegue de las prácticas destinadas a los practicantes. Estas tecnologías son:

1. **Apache Airflow**: Es una plataforma de orquestación de flujos de trabajo que permite programar, automatizar y monitorear tareas complejas. Se utilizará para gestionar procesos como la automatización de tareas relacionadas con la carga, transformación y análisis de datos, necesarias para la lectura y evaluación de prácticas realizadas por los practicantes.

2. **Next.js**: Es un framework de React que facilita la creación de aplicaciones web modernas con renderizado del lado del servidor (SSR) y generación de sitios estáticos (SSG). Será utilizado para desarrollar una interfaz interactiva que permita a los practicantes visualizar sus prácticas, recibir retroalimentación y gestionar su progreso de manera dinámica y eficiente.

3. **PostgreSQL**: Es un sistema de gestión de bases de datos relacional de código abierto, conocido por su capacidad para manejar datos estructurados y no estructurados. Servirá como el almacén central para gestionar y organizar los datos de las prácticas de los estudiantes, garantizando integridad, seguridad y fácil acceso para consultas y análisis posteriores.

Contexto del Proyecto

Para este proyecto, se ha diseñado una interfaz dirigida a estudiantes que buscan realizar su práctica inicial o profesional. Como parte de esta iniciativa, se creó un formulario de Google que permite a las empresas y organizaciones interesadas en ofrecer prácticas completarlo de manera sencilla, funcionando como un mural virtual de oportunidades laborales.

Este formulario está vinculado a la cuenta institucional `informacionesinf@inf.uach.cl`, la cual fue gestionada con el apoyo de la Escuela de Informática para garantizar acceso adecuado a las herramientas necesarias. Gracias a esta cuenta, la información recopilada a través del formulario se almacena automáticamente en un archivo Excel.

Diariamente, este archivo es revisado por Airflow y la información que es extraída se utiliza para alimentar una base de datos, asegurando que los estudiantes cuenten con un flujo constante de oportunidades.

Estas bases podrán realizar la lectura, gestión y despliegue de prácticas, ofreciendo una solución integral y eficiente que facilite la interacción de los practicantes con las empresas.

2.1. Diagramas del Proyecto

A continuación se presentarán los diferentes **Diagramas** del proyecto, los cuales muestran la interacción entre los diferentes usuarios de la aplicación y cómo se comunican con los diversos componentes con el sistema.

2.1.1. Diagrama de componente

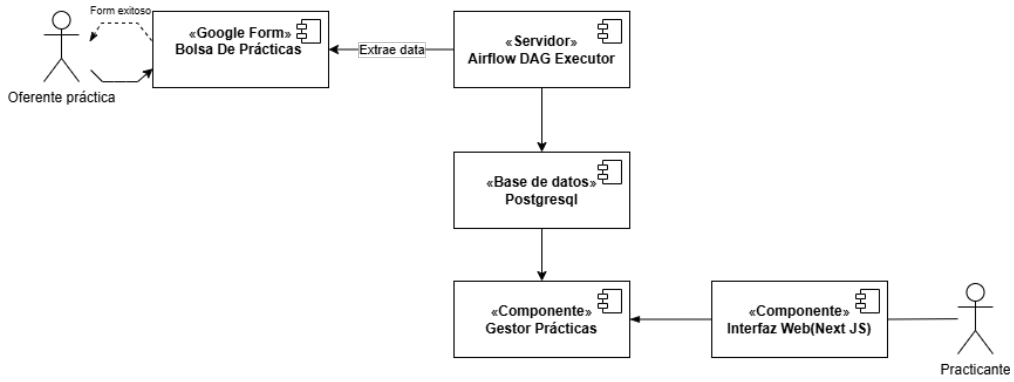

Figura 1: Diagrama de componentes del proyecto

Este diagrama detalla cómo los usuarios, como estudiantes y empresas, interactúan con el sistema a través de sus respectivas interfaces, y cómo estas interactúan con el servicio web y la base de datos. A través de estas conexiones, el sistema gestiona las oportunidades de prácticas, los datos de los estudiantes y las empresas.

2.1.2. Diagrama de secuencia (Practicante)

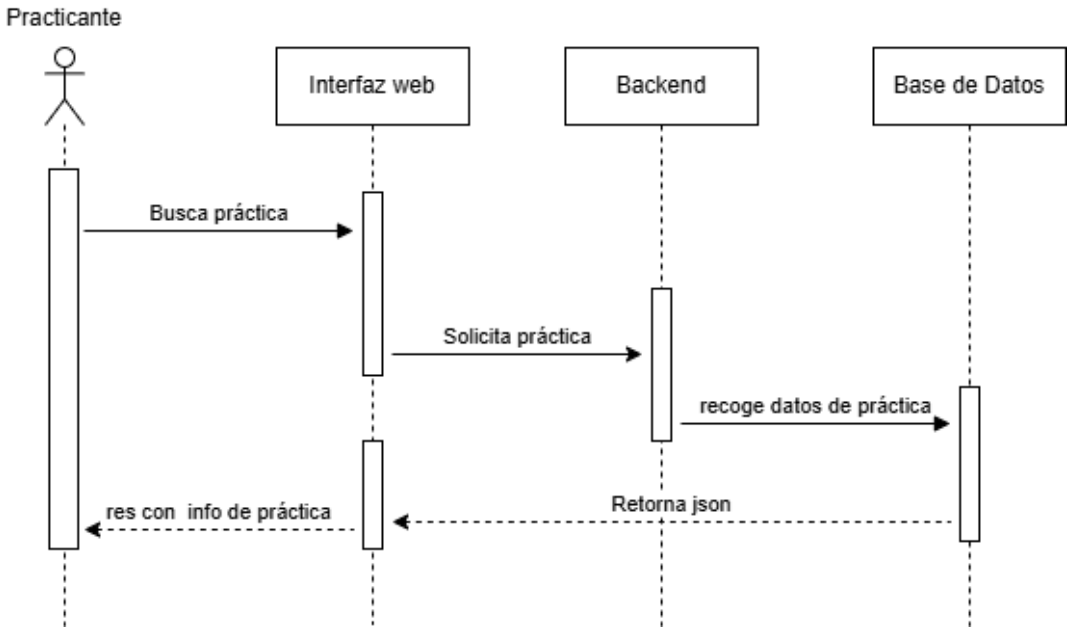

Figura 2: Diagrama de secuencias del practicante

El **Diagrama de Secuencias** presentado ilustra detalladamente el flujo de interacción entre los usuarios, en este caso los **Practicantes**, y el sistema. En este flujo lineal, el **Practicante** inicia la búsqueda de oportunidades de prácticas a través de la interfaz. A continuación, la interfaz envía una solicitud al **Backend**, el cual se encarga de recuperar la información pertinente desde la **Base de Datos**. Finalmente, el backend procesa los datos y devuelve la información solicitada en formato JSON, completando así la interacción y proporcionando al practicante las oportunidades de prácticas disponibles.

2.1.3. Diagrama de secuencia (Oferente de la práctica)

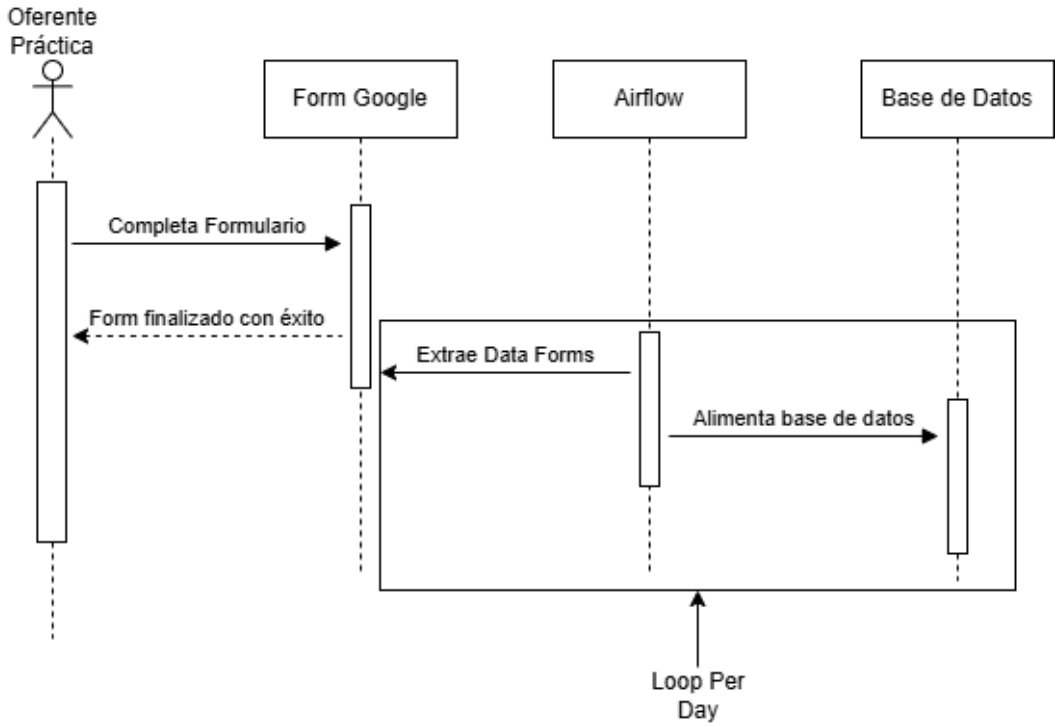

Figura 3: Diagrama de secuencias del Oferente de práctica

En este diagrama se puede observar cómo el **Oferente de la práctica** interactúa con el sistema al completar un formulario, el cual es enviado a través de Google Forms. Una vez enviado, el **Oferente** recibe una confirmación de que el formulario se ha procesado con éxito.

No obstante, más allá de este flujo inicial, el sistema está diseñado para funcionar de manera autónoma y eficiente. En intervalos regulares, el servicio, gestionado por **Apache Airflow**, ejecuta un proceso automatizado que extrae de manera periódica la información ingresada en el formulario y la almacena en la **Base de Datos**. Este proceso asegura que la información esté siempre actualizada y accesible para el análisis y uso posterior, garantizando la continuidad y la precisión del flujo de datos en el sistema.

3. Instalación y Configuración de Airflow (Máquina Local)

La instalación de Airflow que haremos será desarrollada en un entorno de una máquina local. Para ello deberán seguirse los siguientes pasos.

3.1. Prerrequisitos

Antes de comenzar, asegúrate de cumplir con un Sistema Operativo Linux o en su defecto un Windows con WSL2.

- Sistema operativo: Linux o WSL (Windows Subsystem for Linux).
- Python 3.x y `pip` para instalar Apache Airflow.

3.2. Instalación de Airflow

3.2.1. Instalar Airflow

El paso inicial para la instalación y configuración de **Apache Airflow** es instalar el paquete principal utilizando el siguiente comando en un terminal de Linux:

```
pip install apache-airflow
```

3.2.2. Agregar dependencias adicionales (Base de Datos)

Este paso es necesario para proporcionar soporte a bases de datos específicas. Para el caso de **PostgreSQL**, se añaden las dependencias correspondientes. Si se desean incluir otras tecnologías en el futuro, simplemente agréguelas separadas por comas. Utiliza el siguiente comando:

```
pip install apache-airflow[Postgres]
```

3.2.3. Inicialización de Airflow

Antes de utilizar Airflow, es imprescindible configurar su entorno base y crear un usuario administrador. Este proceso se realiza ejecutando los siguientes comandos, en el orden indicado:

1. Inicializar la base de datos de Airflow:

   ```
   airflow db init
   ```

2. Crear un usuario administrador:

   ```
   airflow users create \
       --username admin \
       --firstname Admin \
       --lastname User \
       --role Admin \
       --email admin@example.com \
       --password adminpassword
   ```

Nota: Cada uno de los datos utilizados para crear el usuario (nombre de usuario, nombre, apellido, correo electrónico y contraseña) puede ser personalizado según las necesidades del usuario.

3.3. Estructura Generada por Airflow

Al completar los pasos de instalación y configuración, se generará automáticamente en el directorio `/home` una carpeta llamada **airflow**. Esta carpeta contendrá todas las configuraciones necesarias para ejecutar el programa y gestionar sus componentes.

Dentro de esta estructura, encontrarás una subcarpeta clave llamada **dags**. Este directorio es donde se crearán y almacenarán los flujos de trabajo (DAGs) que luego podrás desplegar y gestionar desde la interfaz web de Apache Airflow.

Esta organización estructurada facilita el desarrollo, mantenimiento y despliegue de flujos de trabajo, garantizando un entorno eficiente y ordenado para tus proyectos.

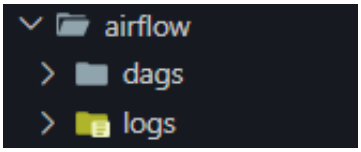

Figura 4: Estructura generada por Airflow en el directorio `/home`

3.4. Cómo Lanzar Apache Airflow

3.4.1. Iniciar el Servidor Web

Para acceder a la interfaz gráfica de Airflow, inicia el servidor web ejecutando en `bash`:

```
airflow webserver --port 8080
```

Propósito: Este comando lanza la interfaz web de Apache Airflow, que permite visualizar, gestionar y monitorear los DAGs. La opción `-port 8080` especifica el puerto en el que se ejecutará el servidor.

3.4.2. Ejecutar el Programador de Tareas (Scheduler)

Además del servidor web, es necesario ejecutar el programador de tareas con:

```
airflow scheduler
```

Propósito: El *scheduler* es el componente que asigna las tareas a los trabajadores y asegura que los flujos de trabajo (DAGs) se ejecuten según lo programado.

3.5. Diagrama de Arquitectura de Airflow

Apache Airflow tiene una arquitectura modular diseñada para gestionar flujos de trabajo complejos. A continuación, se describen sus principales componentes, como se representa en la Figura 5:

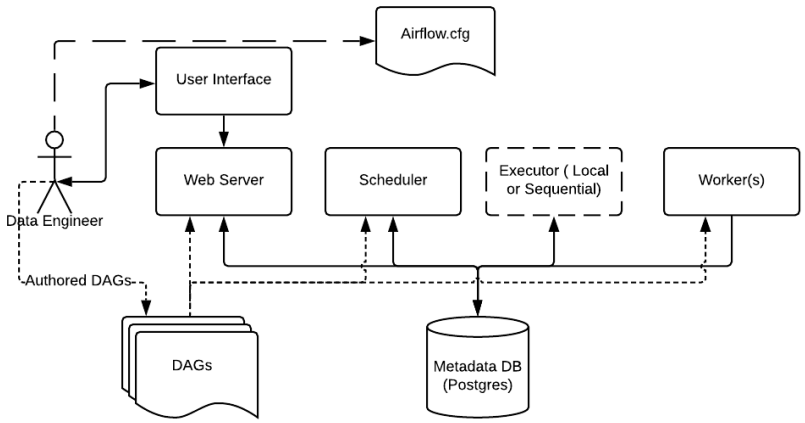

Figura 5: Diagrama de arquitectura y comunicación de Airflow.

- **Web Server:** Es la interfaz gráfica de usuario (GUI) que permite a los usuarios visualizar, gestionar y monitorear los flujos de trabajo (DAGs). Está construido sobre Flask y sirve como punto de interacción para los administradores y desarrolladores.
- **Scheduler:** Es el componente central que organiza y planifica la ejecución de las tareas definidas en los DAGs. Se encarga de identificar las tareas pendientes y encolarlas para su ejecución, según las dependencias y los tiempos definidos.
- **Airflow Configuration (airflow.cfg):** Es el archivo principal de configuración donde se definen las propiedades clave del sistema, como la base de datos a utilizar, el tipo de ejecutor, y otros parámetros esenciales.
- **Executor:** Define cómo se ejecutan las tareas. Los ejecutores pueden ser locales (para pequeñas implementaciones) o distribuidos, como CeleryExecutor, que permite gestionar tareas en múltiples nodos.
- **Worker:** Los trabajadores son los encargados de ejecutar las tareas asignadas por el *scheduler*. En configuraciones distribuidas, se pueden implementar múltiples trabajadores para aumentar la capacidad de procesamiento.

4. Instalación y Configuración de Airflow (Contenedor Docker)

En este apartado podrá ver de manera rápida y detalla los pasos a seguir para instalar Airflow como contenedor de Docker

---

4.1. Prerrequisitos

Antes de comenzar, asegúrate de cumplir con los siguientes requisitos:

- Docker instalado en tu sistema.
- Docker Compose instalado (versión 1.29 o superior).
- Sistema operativo: Linux, macOS o Windows con WSL2.

4.2. Pasos para la Instalación

4.2.1. Crear un archivo Docker Compose

Crea un archivo llamado `docker-compose.yaml` con la configuración básica para Apache Airflow. Este archivo se puede encontrar en la carpeta `Docker` del repositorio.

4.2.2. Preparar el Entorno

Crea los directorios necesarios para los DAGs, logs y plugins con el siguiente comando:

```
mkdir -p dags logs plugins
```

Inicializa la base de datos de Airflow ejecutando:

```
docker-compose up airflow-db -d
docker-compose run airflow-webserver airflow db init
```

Crea un usuario administrador para Airflow:

```
docker-compose run airflow-webserver airflow users create \
    --username admin \
    --firstname Admin \
    --lastname User \
    --role Admin \
    --email admin@example.com \
    --password adminpassword
```

4.2.3. Lanzar los Servicios

Para iniciar todos los servicios de Airflow, ejecuta el siguiente comando:

```
docker-compose up -d
```

4.2.4. Acceso a la Interfaz de Airflow

La interfaz de usuario de Apache Airflow estará disponible en `http://localhost:8080`. Inicia sesión con las credenciales del usuario creado en el paso anterior.

4.2.5. Añadir DAGs

Trae tus archivos de DAGs en el directorio `dags` que creaste anteriormente.

4.2.6. Detener los Servicios

Para detener todos los servicios de Airflow, utiliza el siguiente comando:

```
docker-compose down
```

5. Instalación y Configuración de Next.js

Next.js es un framework de React que permite crear aplicaciones modernas con renderizado del lado del servidor (SSR), generación estática y otras funcionalidades avanzadas. Esta guía te llevará paso a paso para configurar Next.js en tu entorno de desarrollo.

5.1. Requisitos Previos

- Tener Node.js instalado (versión 16 o superior). Puedes verificar la versión instalada con:

```
node -v
npm -v
```

- Tener un gestor de paquetes como npm (incluido con Node.js) o yarn.
- Un editor de código, preferiblemente Visual Studio Code (VS Code).

5.2. Pasos para la Instalación

5.2.1. Configura tu Entorno

Navega a la ruta de tu proyecto con el siguiente comando:

```
cd ruta/proyecto
```

5.2.2. Clonar Proyecto de Git

Usa el siguiente comando para inicializar un nuevo proyecto con Next.js:

```
git clone https://github.com/robleaustral/Lectura-Forms-Apach.Airflow.git
```

5.2.3. Navega al Directorio del Proyecto

Accede al directorio del proyecto clonado:

```
cd /web
```

5.2.4. Inicia el Servidor de Desarrollo

Para iniciar el servidor de desarrollo:

```
npm run dev
```

> **Nota Importante:** Este comando será necesario más adelante en el desarrollo del proyecto, por lo que es importante guardarlo para futuros pasos.

6. Instalación y Configuración de PostgreSQL

En este apartado verás como instalar y configurar adecuadamente tu PostgreSQL, para ello sigue los pasos en orden y de forma adecuada

---

6.1. Pasos para la Instalación

6.1.1. Actualizar el Sistema

Antes de comenzar, asegúrate de que tu sistema esté actualizado ejecutando:

```
sudo apt update && sudo apt upgrade -y
```

6.1.2. Instalar PostgreSQL

Para instalar PostgreSQL en un sistema basado en Debian/Ubuntu:

```
sudo apt install postgresql postgresql-contrib -y
```

6.1.3. Configurar la Contraseña del Usuario

Configura una contraseña para el usuario `postgres` con:

```
sudo -u postgres psql -c "ALTER USER postgres WITH PASSWORD 'password';"
```

Reemplaza `password` con la contraseña deseada.

6.1.4. Verificar la Instalación

Para verificar que PostgreSQL esté correctamente instalado:

```
psql --version
```

6.1.5. Iniciar el Servicio de PostgreSQL

Asegúrate de que el servicio de PostgreSQL esté en ejecución:

```
sudo systemctl start postgresql
```

6.1.6. Habilitar PostgreSQL al Inicio

Para que PostgreSQL se inicie automáticamente al arrancar el sistema:

```
sudo systemctl enable postgresql
```

6.2. Notas Extra

Ahora PostgreSQL está instalado y el usuario `postgres` tiene una contraseña configurada. Para detener el servicio de PostgreSQL, utiliza:

```
sudo systemctl stop postgresql
```

7. Instalación de DBeaver (Opcional)

A continuación se presentaras una serie de pasos para instalar dbeaver en nuestro sistema, los pasos deberán ser ejecutados en terminal de bash, por favor siga en orden y adecuadamante los pasos para obtener el resultado esperado.

7.1. Pasos para la Instalación

7.1.1. 1. Descargar DBeaver

Descarga el archivo de instalación más reciente para sistemas de 64 bits:

```
wget https://dbeaver.io/files/dbeaver-ce_latest_amd64.deb
```

7.1.2. 2. Instalar DBeaver

Instala el paquete descargado con:

```
sudo apt install ./dbeaver-ce_latest_amd64.deb
```

7.1.3. 3. Lanzar DBeaver

Ejecuta DBeaver con:

```
/dbeaver
```

8. Configuración del Proyecto de Prácticas

8.1. Airflow

Ahora que hemos configurado todas las tecnologías necesarias, estamos listos para ejecutar correctamente el gestor de prácticas de la UACh. El primer paso será organizar los DAGS correspondientes a nuestro proyecto. Estos se encuentran almacenados en el repositorio, dentro de la carpeta /dags. Lo único que necesitamos hacer es copiar todo el contenido de esta carpeta y pegarlo en el directorio /home/user/airflow/dags; en caso de ser el entorno virtual, se deberá llevar a la carpeta dags del mismo.

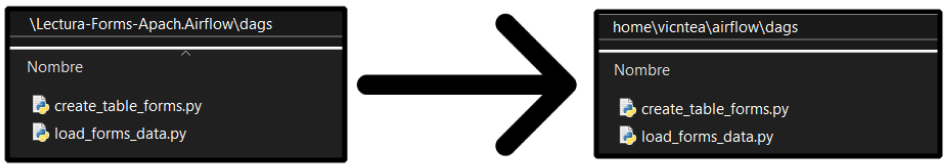

Figura 6: Ejemplo de Migración DAGS

Nota: La Figura muestra cómo debería ser la migración de las carpetas.

Nota Importante: Antes de lanzar, recordar abrir el archivo `txt` alojado en la carpeta raíz del repositorio llamado `requirements_python.txt` para descargar las librerías que usarán los DAGS.
Luego de ello, ya podremos lanzar Airflow, tal como se detalla en la sección Resumen de la ejecución conjunta.[en] la página 7.

Una vez lanzado Apache Airflow, debería desplegarse un login el cual nos pedirá nuestras credenciales que definimos en Ïnstalación y Configuración de Airflow.[en] la página 6.

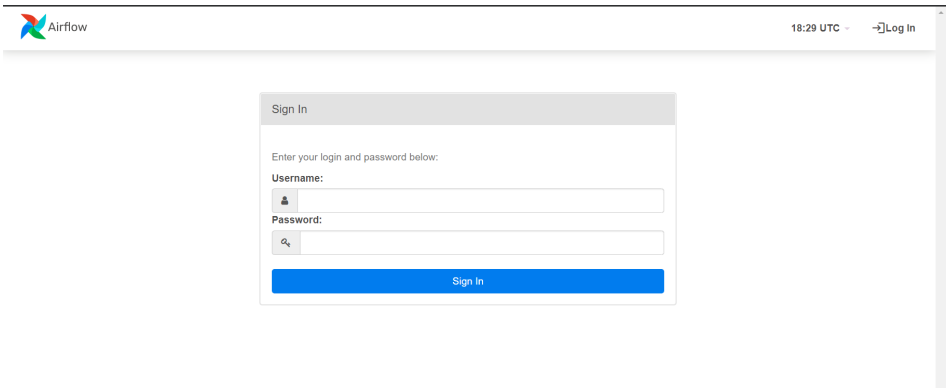

Figura 7: Página de Login

Una vez dentro de Airflow, podremos visualizar todos los DAGS existentes. Nosotros buscaremos los DAGS llamados: `update_info_DB` y `create_table_forms`. Y los marcaremos como activos.

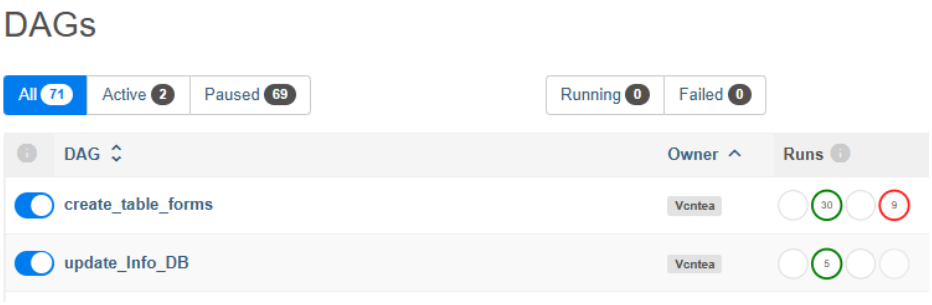

Figura 8: Menú de gestión de DAGS Apache

En caso de querer añadir algún DAG, estos se tendrán que crear dentro de la carpeta `airflow/dags`. Para ello, basta con crear un archivo Python con el flujo de trabajo (DAG) correspondiente, utilizando la estructura y las tareas que deseas automatizar.

Antes de ejecutar los DAGS, primero deberán establecerse unas variables de Airflow, las cuales se pueden añadir desde el menú `admin/variables`. O puedes hacerlo pegando estos comandos en el bash.

```
airflow variables set FRIENDLY_NAMES '["marca_temporal", "tipo_practica", "nombre_contacto",
    "cargo_contacto", "correo_contacto", "telefono_contacto", "nombre_empresa", "
    sitio_web_empresa", "unidad_empresa", "fechas_practica", "modalidad", "sede_practica", "
    regimen_trabajo", "labores", "beneficios", "requisitos_especiales"]'
```

Siguiente comando:

```
airflow variables set CSV_URL "https://docs.google.com/spreadsheets/d/1eVkZ-
    zrQc7oRAXO1hizLi8F4hcX9bGO31oDFqFv7QTU/export?format=csv"
```

Estas variables se pueden modificar en `menu/variables` en el navBar con la finalidad de agregar en caso de que se añadan columnas en el forms o cambiar el URL de ser necesario.

Y por último, establecer la base de datos, se puede realizar mediante el comando bash que está a continuación o manualmente desde `admin/connections` en el navBar.

```
airflow connections add 'postgres_default' \
    --conn-type 'postgres' \
    --conn-host 'localhost' \
    --conn-schema 'Nombre_base_datos' \
    --conn-login 'postgres' \
    --conn-password 'password' \
    --conn-port '5432'
```

Ahora podemos ejecutar los DAGS. Primero ejecutaremos el DAG llamado:

`create_table_forms`: Este inicializará la base de datos con una tabla llamada `forms`, la cual tendrá la estructura del formulario de Google.

Luego ejecutaremos:

`update_info_DB`: El cual se irá ejecutando diariamente para que pueda alimentar la base de datos con la información correspondiente.

Con todo esto, ya tenemos finalizado todo de momento con Airflow y nuestra base de datos funcionando. Ahora nos resta configurar nuestro entorno en Next.js.

8.2. Next.js

Primero navegaremos a la ruta de donde se encuentra la página web.

```
cd ruta/web
```

Ahora instalamos dependencias:

```
npm i
```

Ahora inicializamos nuestro ORM llamado Prisma:

```
npx prisma init
```

De esto se nos generará un archivo `dotenv`, en donde deberemos poner nuestra base de datos con el siguiente formato.

```
DATABASE_URL="postgresql://<postgres_usuario>:<contrasea>@host:puerto/nombre_base_datos?
    schema=public"
```

Ejemplo:

```
DATABASE_URL="postgresql://postgres:password@localhost:8059/postgres?schema=public"
```

Seguido de esto, ingresamos el siguiente comando:

```
npx prisma db pull
```

Para traer la información de la estructura de la base de datos a nuestro ORM Prisma. Con ello podremos ahora lanzar el programa y se realizarán correctamente las consultas de backend.

Luego, podremos lanzar el servicio web, tal como se detalla en la sección Ïnstalación y Configuración de Next.js.[en] la página 15.

9. Proyecto de Prácticas En Funcionamiento

Figura 9: Menú de Inicio para Prácticas

En la página principal con dirección: `http://localhost:3000/`, se podrá ver el botón de inicio para dar pie al menú que muestra cada opción relacionada con las prácticas.

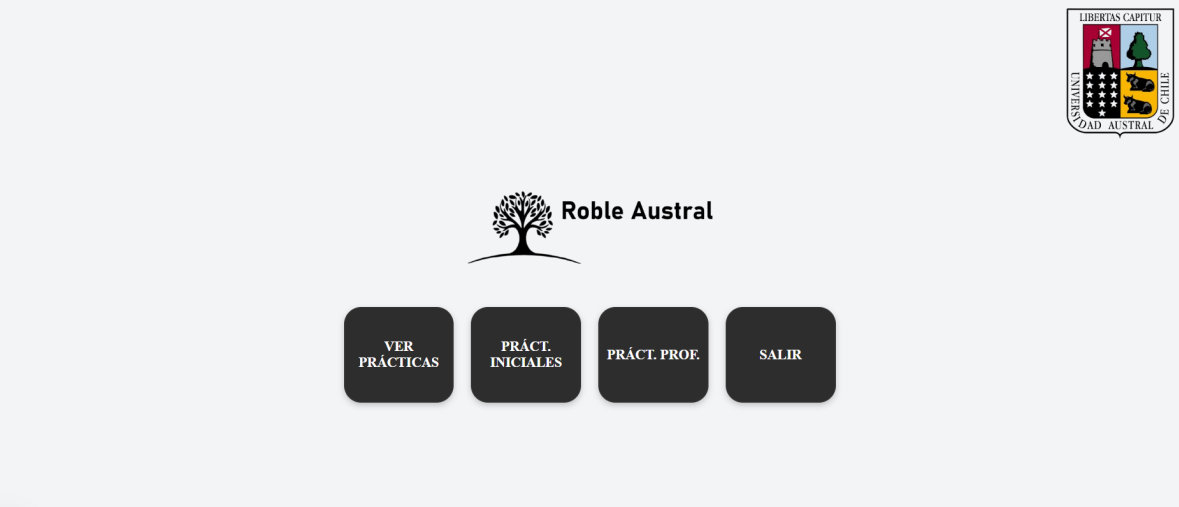

Figura 10: Menú de Prácticas

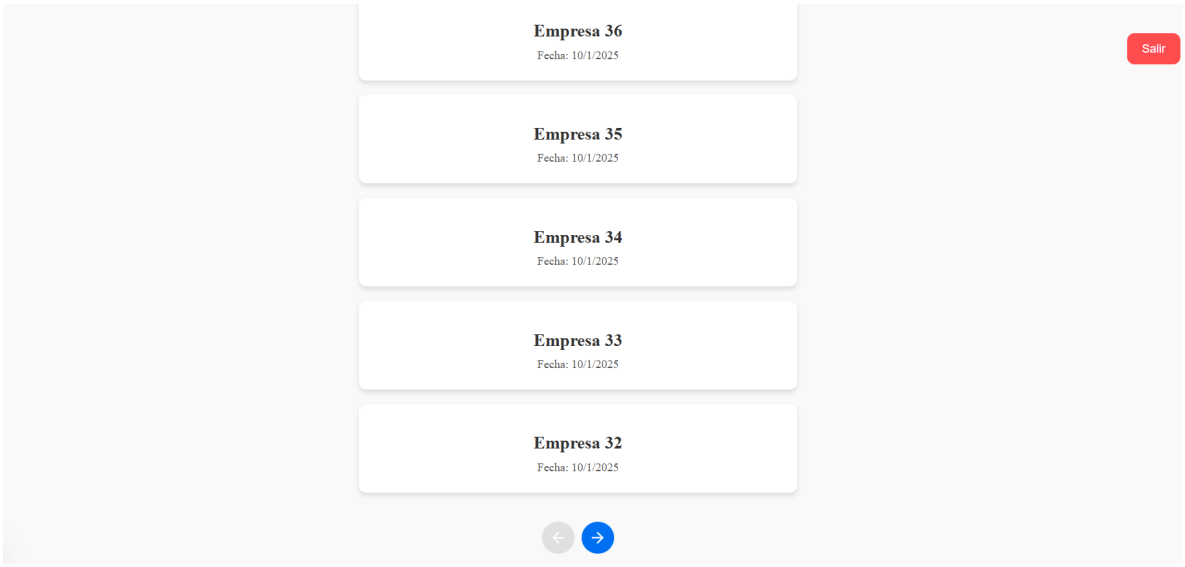

Figura 11: Ejemplo Vista de Prácticas

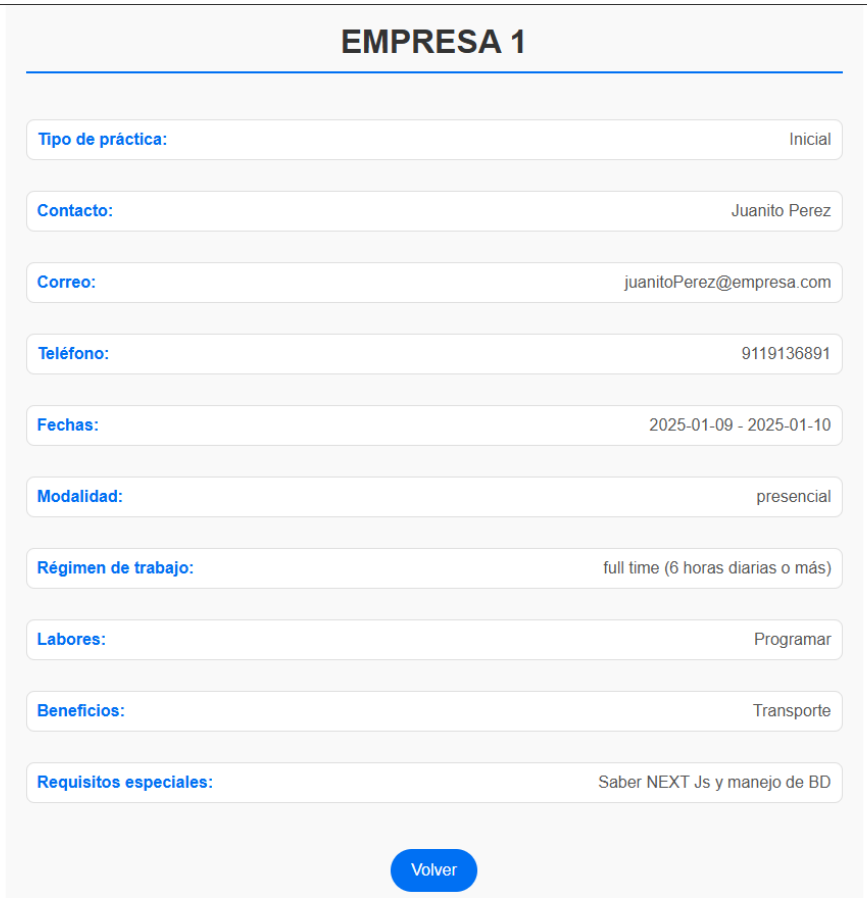

Figura 12: Ejemplo Vista de Práctica Individual

En la Figura 10, se puede apreciar el detalle del menú con las diferentes prácticas. La Figura 11 muestra ejemplos de vistas de prácticas, mientras que la Figura 12 Muestra el detalle de la práctica.

10. Creación de Dags

Apache Airflow es una plataforma de orquestación de flujos de trabajo que permite programar, monitorear y gestionar tareas de manera eficiente y flexible. Un DAG (Directed Acyclic Graph) es el concepto central en Airflow, y se utiliza para definir un conjunto de tareas que deben ejecutarse en una secuencia específica. A continuación, se presentan los conceptos clave que se deben tener en cuenta al crear un DAG en Airflow.

10.1. ¿Qué es un DAG?

Un DAG (Directed Acyclic Graph) es un grafo dirigido acíclico que representa un conjunto de tareas y sus dependencias. Cada nodo del grafo es una tarea (o task), y las aristas representan las dependencias entre estas tareas. En Airflow, el DAG se utiliza para especificar cómo y cuándo deben ejecutarse las tareas.

Características de un DAG:

- **Dirigido:** Las tareas tienen un orden de ejecución definido, lo que implica que el flujo de trabajo tiene un comienzo y un final.
- **Acíclico:** No se permiten ciclos dentro del grafo. Es decir, no puede haber una tarea que dependa indirectamente de sí misma.
- **Orientado:** El flujo de tareas se ejecuta en una secuencia definida, según las dependencias entre ellas.

El DAG define la estructura general de un flujo de trabajo, incluyendo la programación de ejecución y las dependencias entre las tareas.

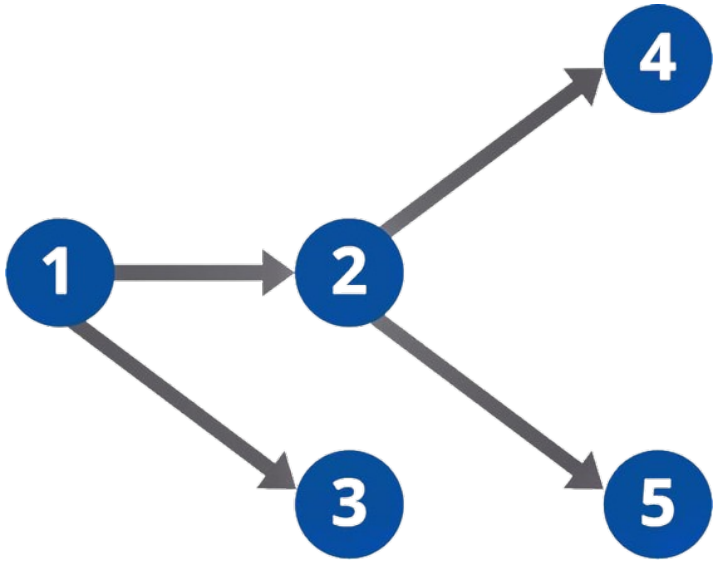

Figura 13: Diagrama de Flujo de Tareas

Nota: Podemos ver cómo hay una dependencia entre tareas, que es lo que busca representar al explicar qué es un DAG.

10.2. Estructura de un DAG en Airflow

En Airflow, un DAG se define mediante un archivo Python, normalmente ubicado en el directorio `dags/` del proyecto. A continuación, se muestra un ejemplo básico de un DAG:

```
from airflow import DAG
from airflow.operators.dummy import DummyOperator
from airflow.operators.python import PythonOperator
from datetime import datetime, timedelta

def my_python_function():
    print("Hello, Airflow!")

default_args = {
    'owner': 'airflow',
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
}

with DAG(
    dag_id='example_dag',
    default_args=default_args,
    start_date=datetime(2023, 1, 1),
    schedule_interval='@daily',
    catchup=False,
) as dag:
    task_1 = DummyOperator(task_id='task_1')
    task_2 = PythonOperator(task_id='task_2', python_callable=my_python_function)
    task_3 = DummyOperator(task_id='task_3')

    task_1 >> task_2 >> task_3
```

Listing 1: Ejemplo DAG Inicial

En este ejemplo:

- `DummyOperator` se usa como una tarea de marcador de posición.
- `PythonOperator` ejecuta una función de Python como tarea.
- El flujo de trabajo está definido por las dependencias entre las tareas: `task_1` debe completarse antes de `task_2`, y `task_2` debe completarse antes de `task_3`.

10.3. Variables de Airflow

Las variables en Airflow son valores clave que pueden ser utilizados para parametrizar tareas o DAGs. Estas variables se almacenan en la base de datos de Airflow y pueden definirse a través de la interfaz web, línea de comandos o código.

Características principales:

- Accesibles globalmente en el entorno de Airflow.
- Almacenadas en la base de datos para garantizar persistencia.
- Útiles para evitar el uso de valores codificados (*hardcoded*).

Ejemplo de uso:

```
from airflow.models import Variable

# Obtener una variable
my_variable = Variable.get("my_variable_name", default_var="valor_por_defecto")

# Usar la variable en un DAG
with DAG(...) as dag:
```

```
    task = BashOperator(
        task_id="example_task",
        bash_command=f"echo {my_variable}"
    )
```

Listing 2: Uso de Variables en Airflow

10.4. Parámetros de un DAG

Los DAGs pueden recibir parámetros que permiten personalizar su ejecución. Estos parámetros se especifican mediante la clase `DagRun` o directamente al definir el DAG.

Ejemplo:

```
with DAG(
    dag_id='parameterized_dag',
    default_args=default_args,
    params={"param1": "valor1", "param2": 42},
    ...
) as dag:
    task = PythonOperator(
        task_id='use_params',
        python_callable=lambda **kwargs: print(kwargs['params']['param1']),
    )
```

Listing 3: Definición de Parámetros en un DAG

10.5. Manejo de Errores

El manejo de errores es crucial para garantizar que el flujo de trabajo sea resiliente ante fallos. Airflow proporciona varios mecanismos para manejar errores:

Retries: Permite reintentar una tarea fallida un número determinado de veces.

```
default_args = {
    'retries': 3,
    'retry_delay': timedelta(minutes=5),
}
```

Listing 4: Definición de Retries

Callbacks: Definen funciones que se ejecutan cuando una tarea falla o finaliza.

```
def on_failure_callback(context):
    print(f"Tarea {context['task_instance'].task_id} fall")

with DAG(
    dag_id='dag_with_callbacks',
    default_args={'on_failure_callback': on_failure_callback},
    ...
) as dag:
    ...
```

Listing 5: Uso de Callbacks

Alertas: Integra notificaciones (por ejemplo, por correo electrónico o Slack) para alertar sobre errores.

11. Interfaz de Airflow

Airflow proporciona una interfaz web poderosa y fácil de usar que permite a los usuarios gestionar, monitorear y depurar flujos de trabajo (DAGs) de manera eficiente. Esta interfaz está diseñada para ofrecer una visión clara del estado de los flujos de trabajo y proporcionar herramientas intuitivas para administrar las tareas y sus dependencias.

11.1. Visión General de la Interfaz

La interfaz de Airflow está dividida en varias secciones principales que permiten interactuar con los DAGs y sus componentes. A continuación, se describen cada una de estas secciones y sus funcionalidades:

- **Barra de Navegación:** La barra superior proporciona acceso rápido a las diferentes vistas y funcionalidades. Incluye opciones como *Home*, *DAGs*, *Browse*, *Admin*, *Docs*, y un botón para iniciar sesión o cerrar sesión.
- **Lista de DAGs:** Muestra todos los DAGs disponibles en el sistema junto con información relevante, como el estado de las ejecuciones, el número de tareas exitosas, en ejecución o fallidas, y botones de acción para iniciar, pausar o modificar el DAG.
- **Detalles del DAG:** Cada DAG tiene una página dedicada que muestra información detallada, como su grafo de dependencias, historial de ejecuciones, y logs.
- **Herramientas de Navegación:** Permiten explorar ejecuciones históricas, tareas individuales y estadísticas generales del sistema.

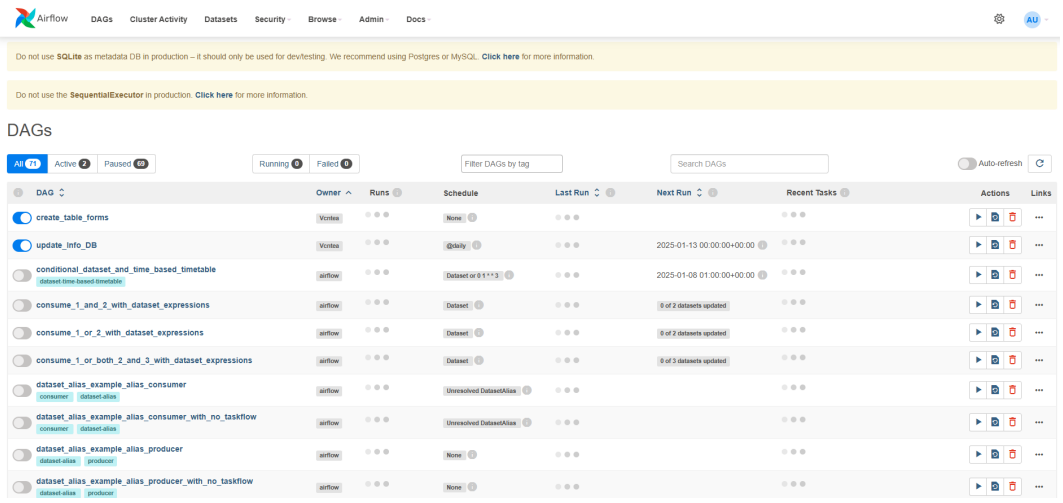

Figura 14: Barra de Navegación de Airflow

11.2. Gestión de DAGs

La interfaz permite una gestión completa de los DAGs. Las funcionalidades principales incluyen:

Visualización del Estado del DAG: La vista principal muestra el estado de cada DAG, indicando si está activo, pausado o si se ha producido un error en su ejecución.

Ejecución Manual: Los usuarios pueden iniciar manualmente un DAG desde la interfaz, lo que es útil para pruebas o ejecuciones fuera de la programación habitual.

Pausa y Reanudación: Un DAG puede pausarse para detener su ejecución programada y reanudarse cuando sea necesario.

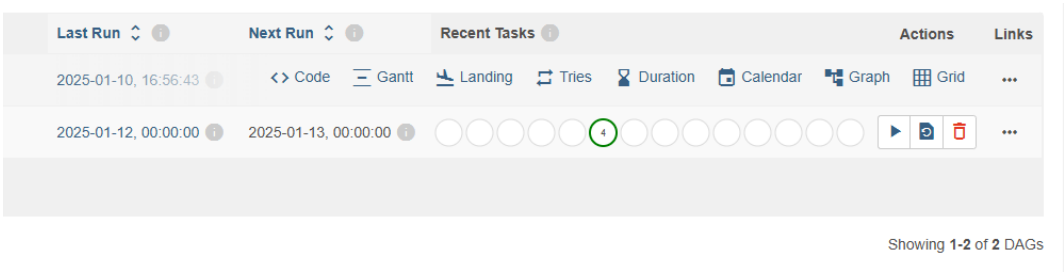

Figura 15: Captura de la lista de DAGs mostrando los estados y opciones.

11.3. Visualización del Grafo de Dependencias

Una de las características más destacadas de Airflow es la representación gráfica de los flujos de trabajo. Esta vista permite a los usuarios comprender de manera visual las dependencias entre las tareas de un DAG.

- **Nodos:** Representan las tareas individuales dentro del DAG.
- **Aristas:** Muestran las dependencias entre las tareas.
- **Colores:** Indican el estado de las tareas (ejecutado, fallido, en espera, etc.).

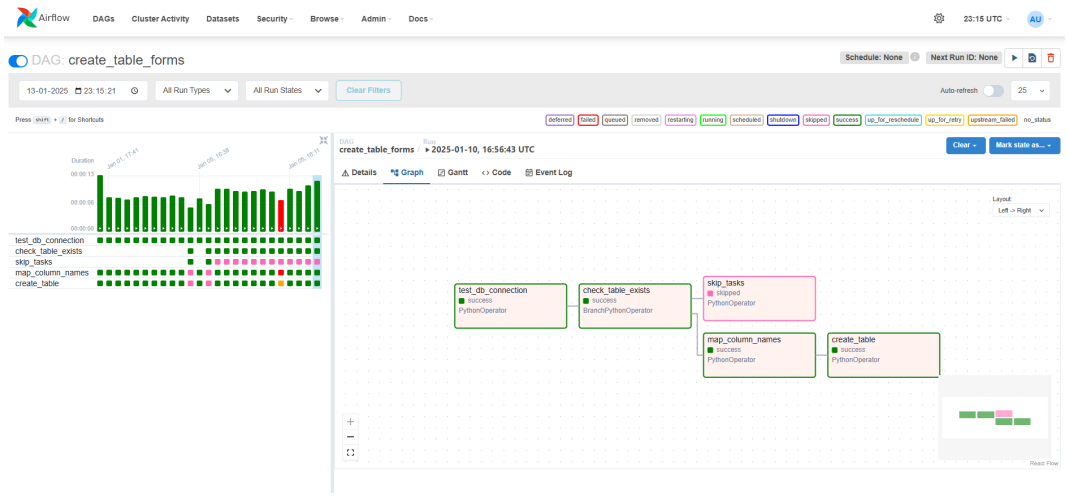

Figura 16: Grafo de dependencias

11.4. Historial de Ejecuciones

Airflow almacena un registro detallado de todas las ejecuciones de un DAG. La interfaz permite a los usuarios explorar el historial y analizar el rendimiento de las tareas. Las funcionalidades incluyen:

- **Vista de Calendario:** Permite identificar rápidamente los días en los que un DAG se ejecutó con éxito o presentó errores.
- **Detalles de Ejecución:** Proporciona información granular sobre cada ejecución, incluyendo tiempos de inicio y fin, y errores encontrados.
- **Filtros:** Los usuarios pueden filtrar por estado de la ejecución para enfocarse en fallos o ejecuciones específicas.

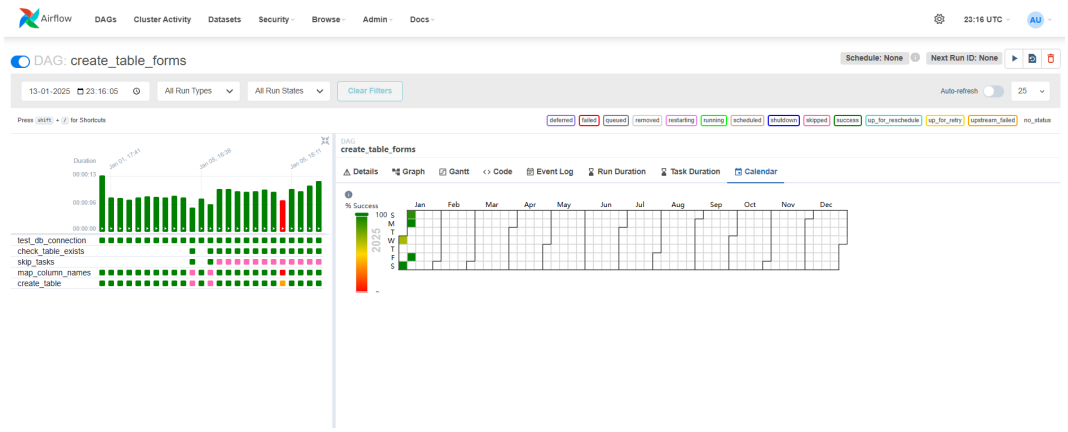

Figura 17: Historial de ejecuciones.

11.5. Exploración de Logs

Airflow facilita la depuración mediante la visualización de logs detallados para cada tarea. Estas funcionalidades son clave para identificar y resolver problemas en los flujos de trabajo.

- **Acceso a Logs:** Los usuarios pueden acceder a los logs de cada tarea directamente desde la interfaz.
- **Descarga:** Los logs pueden descargarse para su análisis fuera de línea.
- **Colores y Formato:** Los logs están organizados de manera que sean fácilmente legibles, utilizando colores para destacar errores o advertencias.

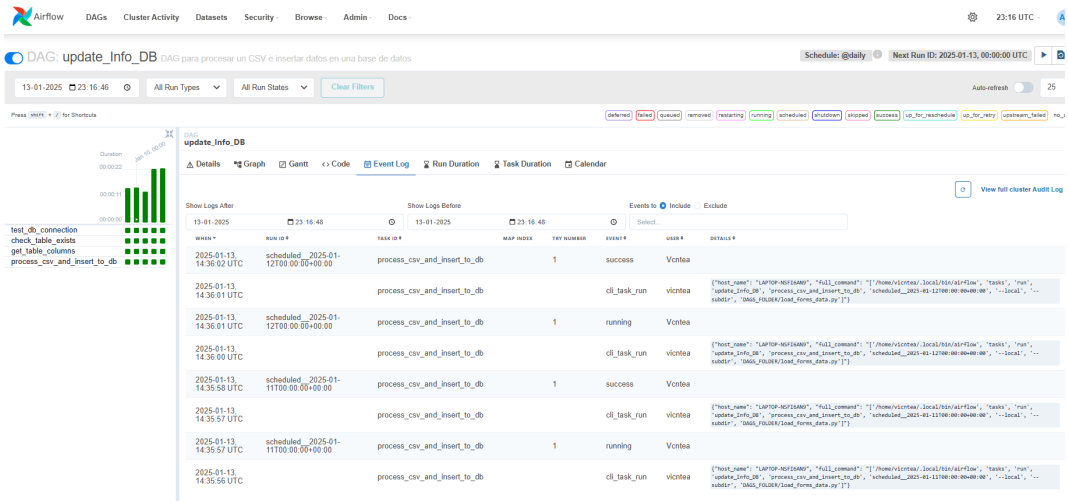

Figura 18: Logs de una tarea

11.6. Administración del Sistema

La sección de administración de la interfaz de Airflow permite a los usuarios configurar y supervisar diversos aspectos del sistema.

- **Usuarios:** Gestión de cuentas, roles y permisos.
- **Conexiones:** Configuración de conexiones a bases de datos, APIs y otros sistemas externos.
- **Variables:** Configuración de variables globales para los DAGs.
- **Pools:** Configuración de pools de recursos para limitar la concurrencia.

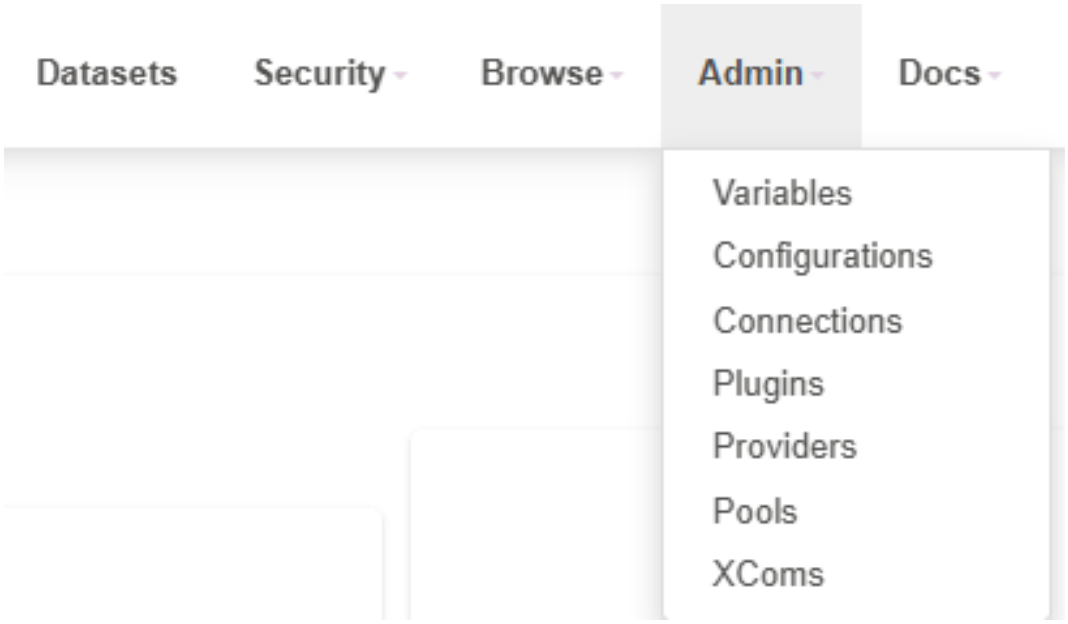

Figura 19: Sección administración del sistema

11.7. Conclusión

Con esto se da por finalizado el manual. Se espera que la información presentada haya sido útil y clara, proporcionando al lector los conocimientos necesarios para comprender y aplicar los conceptos tratados. A lo largo del manual, hemos cubierto los aspectos esenciales sobre *Apache Airflow*, lo que permite un entendimiento integral del contenido.

Es importante destacar que, aunque este manual cubre los puntos fundamentales, siempre es recomendable continuar explorando el tema de manera más profunda a medida que se ganan más experiencias prácticas. En caso de dudas adicionales, se recomienda consultar recursos adicionales.

Agradecemos su tiempo y esperamos que este manual haya sido de ayuda.