# Documentación de configuración y uso para la aplicación “graficadorSencillo”

Aplicación web para graficado de datos meteorológicos filtrados por fecha y campo

Contenido

[Documentación de configuración y uso para la aplicación “graficadorSencillo” 1](#_Toc1051045)

[*Overview* 2](#_Toc1051046)

[Listado y descripción de clases propias 3](#_Toc1051047)

[Listado de clases de terceros y referencias a las fuentes 4](#_Toc1051048)

[Archivos adicionales usados por la aplicación 4](#_Toc1051049)

[Manual de uso 5](#_Toc1051050)

[Configuración y arranque 5](#_Toc1051051)

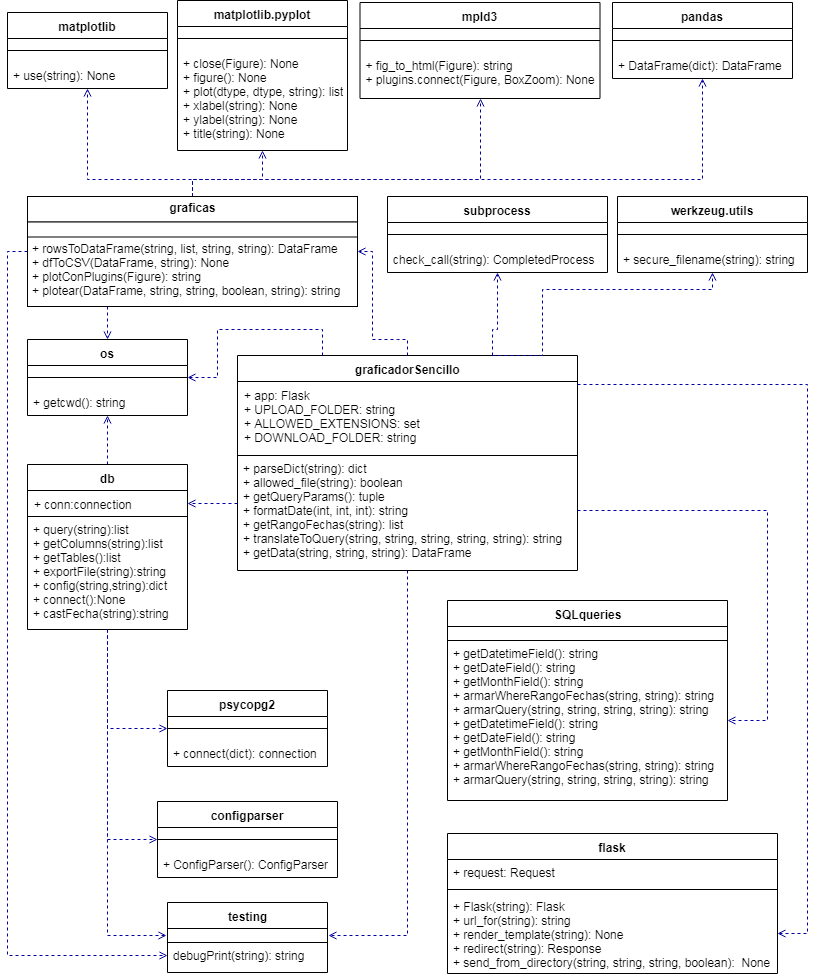
[Uso de la aplicación 8](#_Toc1051052)

[Modo de edición 10](#_Toc1051053)

[Advertencias y recomendaciones 12](#_Toc1051054)

## *Overview*

El graficador es una aplicación web desarrollada en Python con el *framework* Flask, versión 1.0. La aplicación se organiza de la siguiente forma:



Cada recuadro representa una clase, que es un componente de la aplicación. Las flechas indican la relación “depende de”. A continuación se desarrolla una breve explicación de cada clase.

### Listado y descripción de clases propias

* graficadorSencillo.py: la aplicación como tal. Procesa las diferentes direcciones (rutas) reconocidas por la aplicación para realizar operaciones. Presenta una página de identificación y una de operación con opción a modo de edición. En la página de operación permite carga de datos, descarga de datos y gráfica de datos filtrados por fechas y periodicidad.

Depende fundamentalmente de flask para comunicar el *frontend* con el *backend* y procesar las solicitudes realizadas mediante la interfaz. Depende de werkzeug.utils para procesar y cargar archivos con nuevas tablas de datos. Depende de subprocess para ejecutar y *script* de R que permite cargar la nueva información a la base de datos. Depende de graficas.py para dibujar las gráficas, depende de db para comunicarse con la base de datos; y depende de SQLqueries para “armar” las solicitudes que se envían a la base de datos (procurando ser SQL puro, lo más independiente posible del DBMS que, en este caso, es PostgreSQL). Depende de os para obtener el directorio de operación y construir los directorios de carga y descarga de archivos.

* testing.py: contiene un método para habilitar o deshabilitar despliegues de depuración mediante un método primitivo. Alterar el despliegue requiere comentar o “descomentar” directamente la línea de código correspondiente en este método.
* db.py: contiene procedimientos para conectarse a una base de datos PostgreSQL, realizar *queries* a ella y obtener información de ella, como listado de tablas y columnas. Además, contiene el método con el que se exporta una tabla a un archivo separado por comas, para descarga.

**Importante**: esta clase es la encargada de comunicar a la aplicación con una base de datos creada y gestionada mediante PostgreSQL. Para lograrlo depende fundamentalmente de psycopg2. Idealmente, esta clase podría ser reemplazada si se necesita cambiar el manejador de base de datos en el que se apoye la aplicación, sin necesidad de cambiar ningún otro elemento.

Depende de configparser para cargar la información de conexión a la base de datos. Depende de os para armar un directorio de exportación de datos en formato .csv (el método que lo necesita no es usado en la aplicación; quedó como vestigio/método utilitario).

* graficas.py: contiene los procedimientos que dibujan las gráficas según los parámetros recibidos mediante la aplicación. También presenta métodos para transformación de datos, obtenidos mediante consultas a la base de datos, a formatos necesarios para operación interna (como gráficas y descargas de datos).

Depende de matplotlib y matplotlib.pyplot para realizar las gráficas. Depende de mpld3 para convertir las gráficas a código html (presentable mediante el navegador) y para agregar el *plugin* de *zoom in/zoom out*. Depende de pandas para trasladar los datos obtenidos de la base de datos a una estructura graficable mediante pyplot (dataframe). Depende de os para encontrar la ubicación en la que, por defecto, se almacenan los datos en formato .csv (para descarga).

* SQLqueries.py: en cierto modo, intermediario entre db.py y graficadorSencillo.py. Esta clase se encarga de transformar parámetros de consulta a una base de datos al lenguaje SQL de modo que los detalles de la comunicación con la base de datos estén fuera de la aplicación principal, pero tampoco estén casados con el módulo que se comunica directamente con un DBMS específico.

### Listado de clases de terceros y referencias a las fuentes

* matplotlib y matplotlib.pyplot: <https://matplotlib.org/>.
* mpld3: <https://mpld3.github.io/>.
* pandas: <https://pandas.pydata.org/>.
* psycopg2: <https://pypi.org/project/psycopg2/>.
* flask: <http://flask.pocoo.org/>.
* werkzeug.utils (incluido con flask): <http://werkzeug.pocoo.org/>.

Los siguientes módulos son incluidos con la instalación del intérprete de Python 3:

* subprocess: <https://docs.python.org/3/library/subprocess.html>.
* os: <https://docs.python.org/3/library/os.html>.
* configparser: <https://docs.python.org/3/library/configparser.html>.

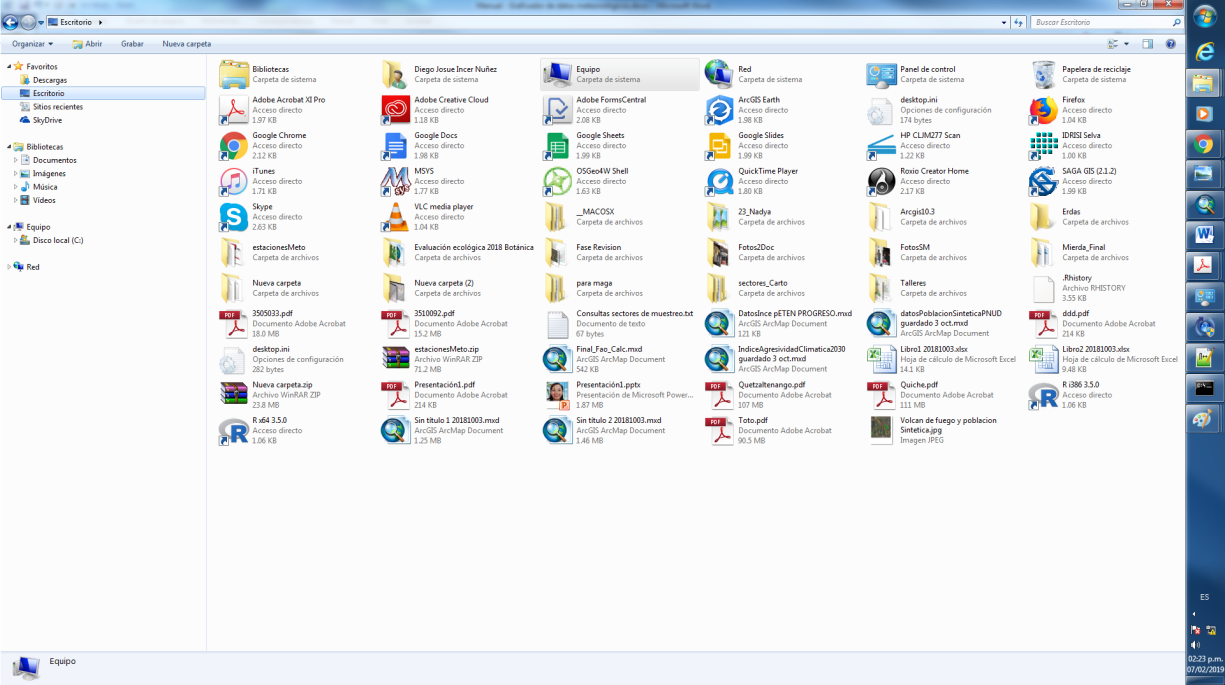
### Archivos adicionales usados por la aplicación

* database.ini: archivo de donde se obtiene la información de conexión a la base de datos.
* migradorDB.R: *script* para carga de datos nuevos a la base de datos de la aplicación.
* prettyVars.txt: archivo de texto que contiene la asociación de nombres reales de campos y su correspondiente reemplazo en la aplicación.
* editVars.html: código HTML + Jinja2 + CSS de la página para cambios de nombres de variables. Dirige a las rutas a las que responde el *backend* escrito en Python.
* index.html: código HTML de la página de inicio, donde se solicita nombre al usuario. Dirige a las rutas a las que responde el *backend* escrito en Python.
* graficador.html: código HTML + Jinja2 + CSS de la página principal, donde se muestran las gráficas y se ingresan los datos. También incluye el código para el modo de edición. Dirige a las rutas a las que responde el *backend* escrito en Python.

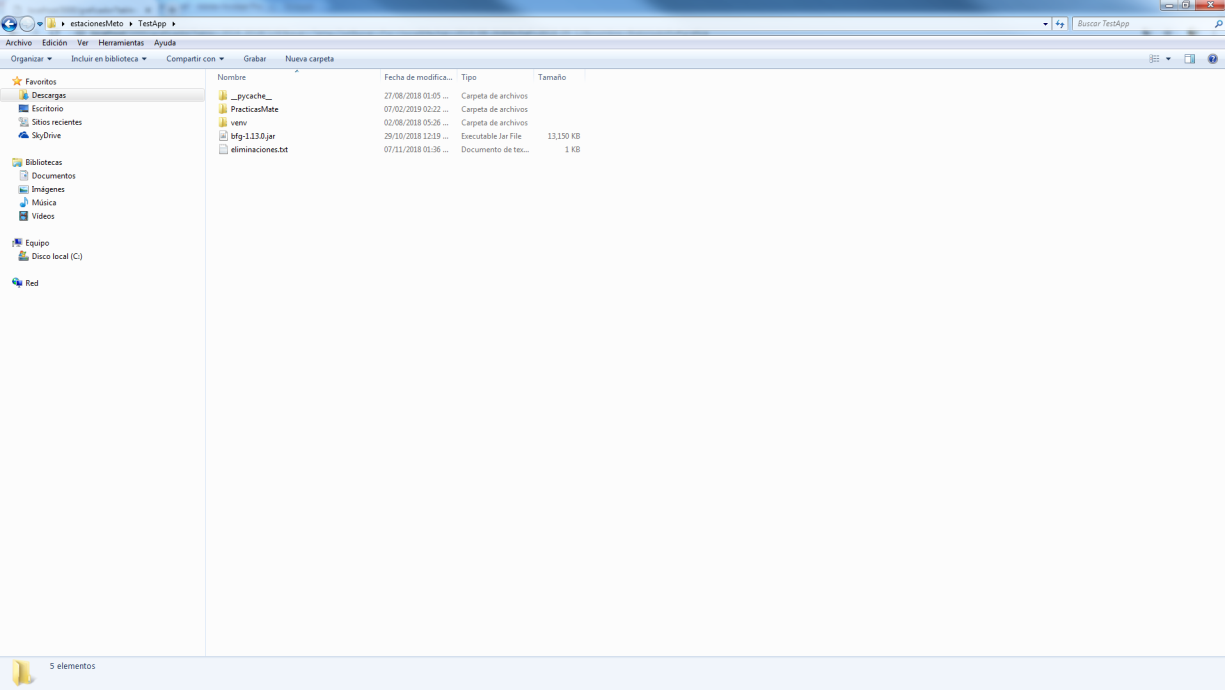
## Manual de uso

### Configuración y arranque

Actualmente, la aplicación se ubica en el siguiente directorio:

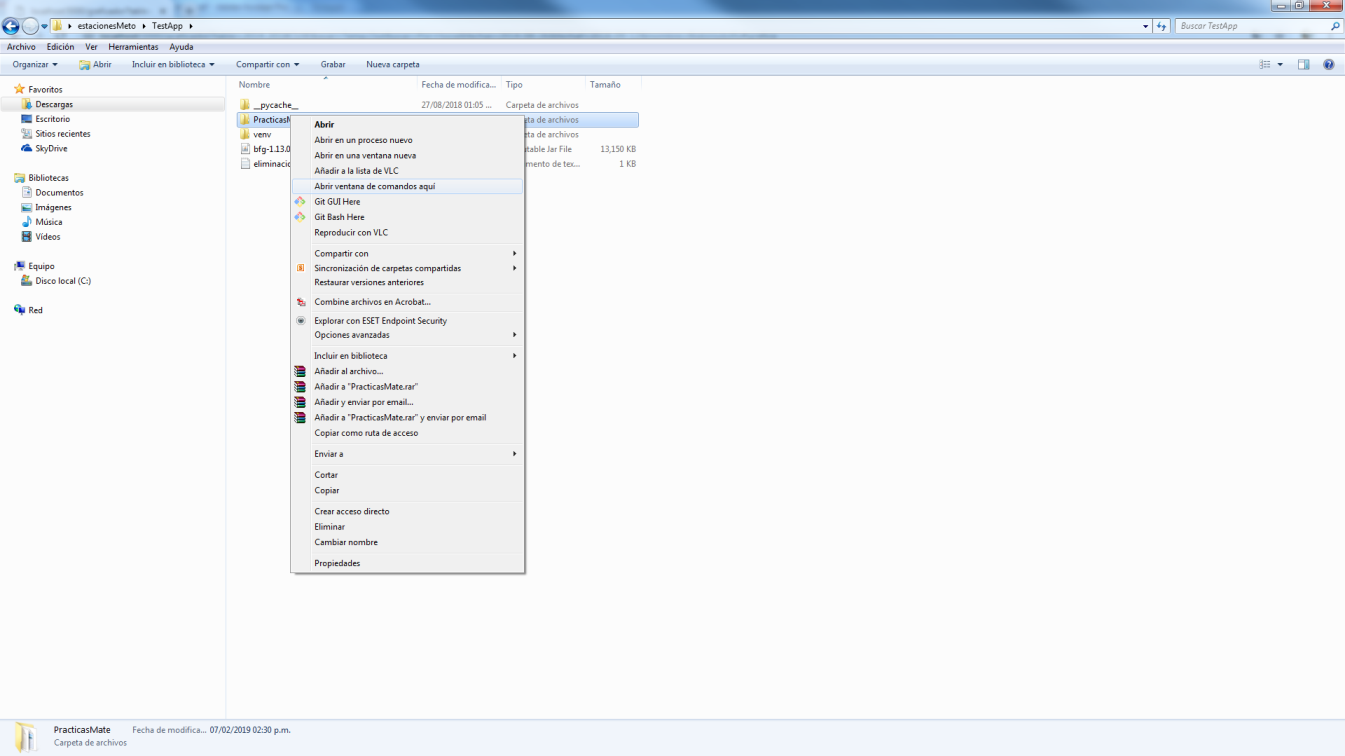


C:\Users\djincer\Desktop\estacionesMeto\TestApp\PracticasMate



Para arrancar la aplicación es necesario seguir los siguientes pasos:

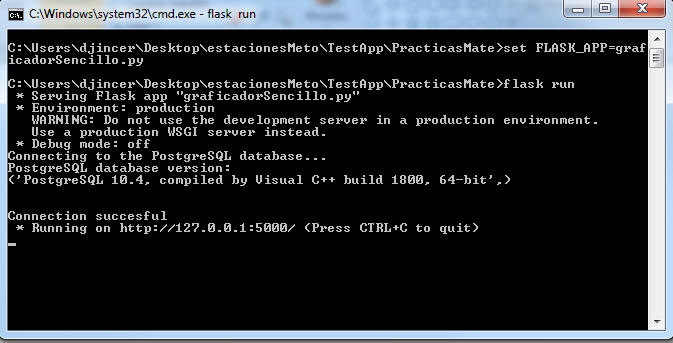
1. Abrir una ventana de comandos en la ubicación de la aplicación: s*hift+click* derecho en la carpeta y luego selección de “Abrir nueva ventana de comandos aquí”.



1. Configurar la plataforma flask para que arranque esta aplicación: ejecutar el siguiente comando en la ventana de comandos abierta:

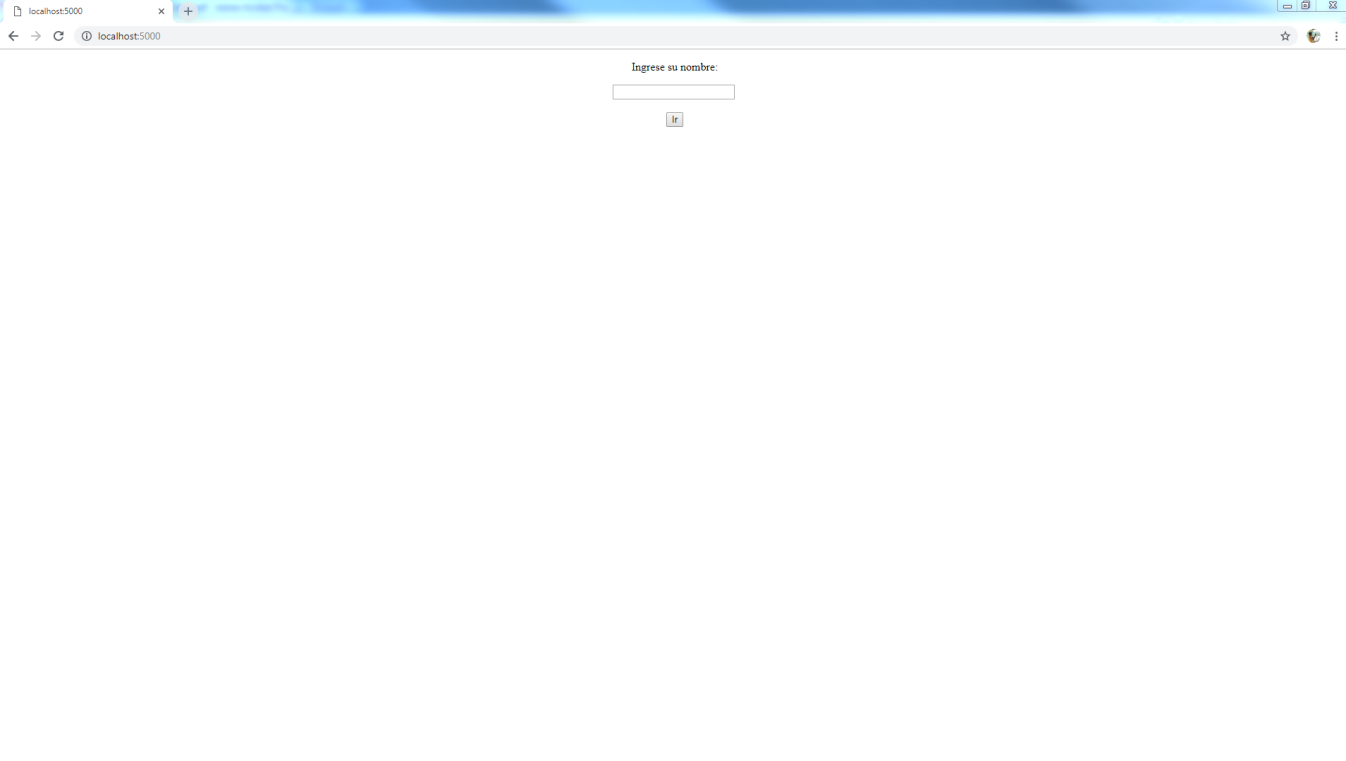
set FLASK\_APP=graficadorSencillo.py

1. Ejecutar la aplicación mediante flask: ejecutar el siguiente comando en la ventana de comandos abierta:

flask run

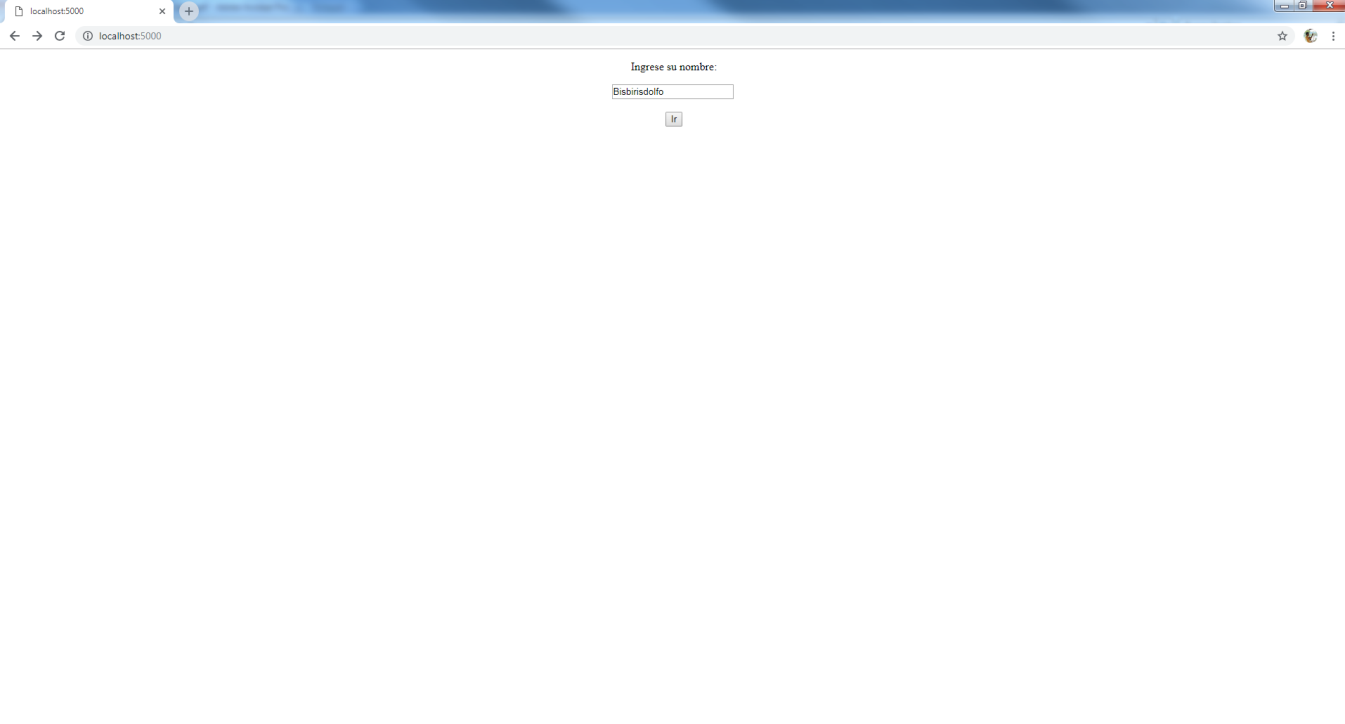
Nótese que la exitosa ejecución del primer comando no produce texto, mientras que la ejecución del segundo comando debería mostrar un texto similar al visto en la ilustración.

1. Abrir un navegador.
2. Dirigir el navegador a la dirección “localhost:5000”.

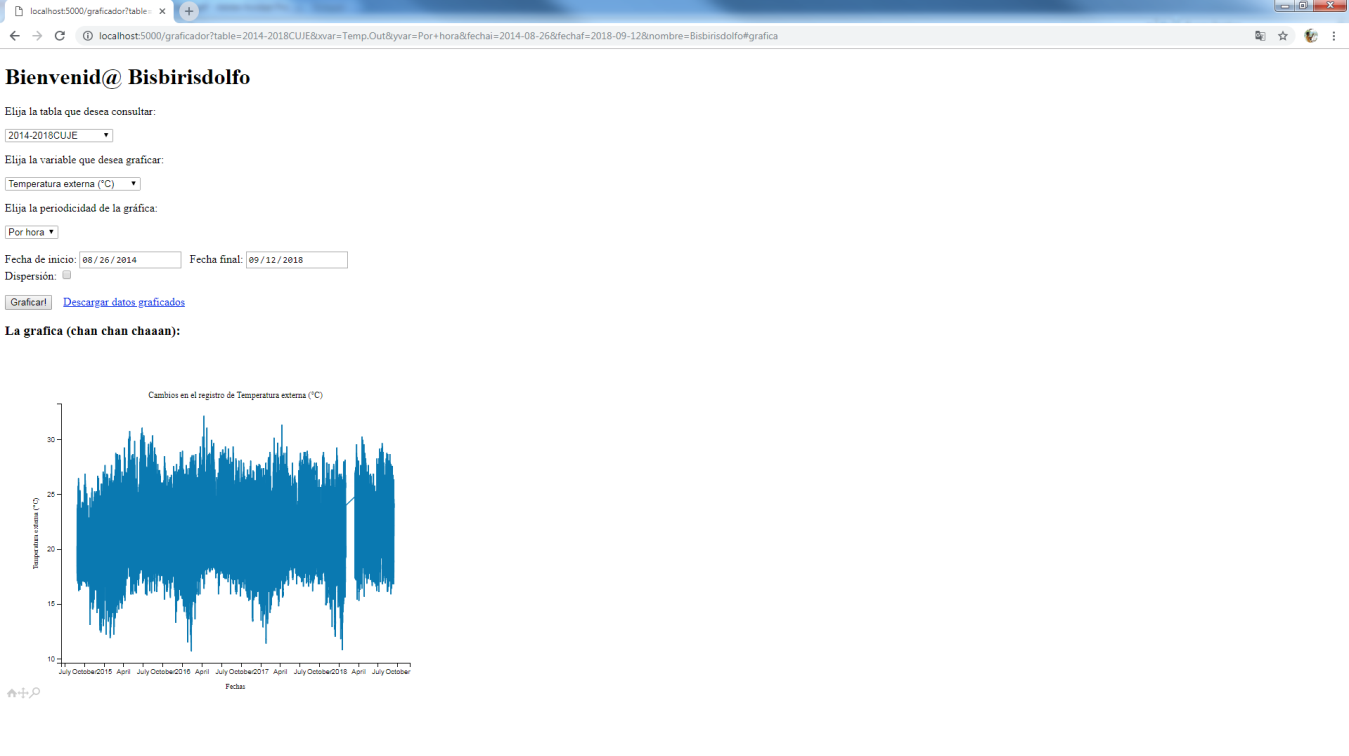


### Uso de la aplicación

La pantalla inicial solicita un nombre al usuario. Dependiendo del nombre ingresado se puede acceder al modo de edición o al modo normal.



Luego de ingresar un nombre y hacer *click* en “Ir” se desplegará una pantalla con un mensaje de bienvenida y los parámetros configurables de la aplicación. A continuación se presenta una imagen etiquetada con descripciones de los componentes de esta pantalla:



**a.**

**b.**

**c.**

**d.**

**e.**

**f.**

**g.**

**h.**

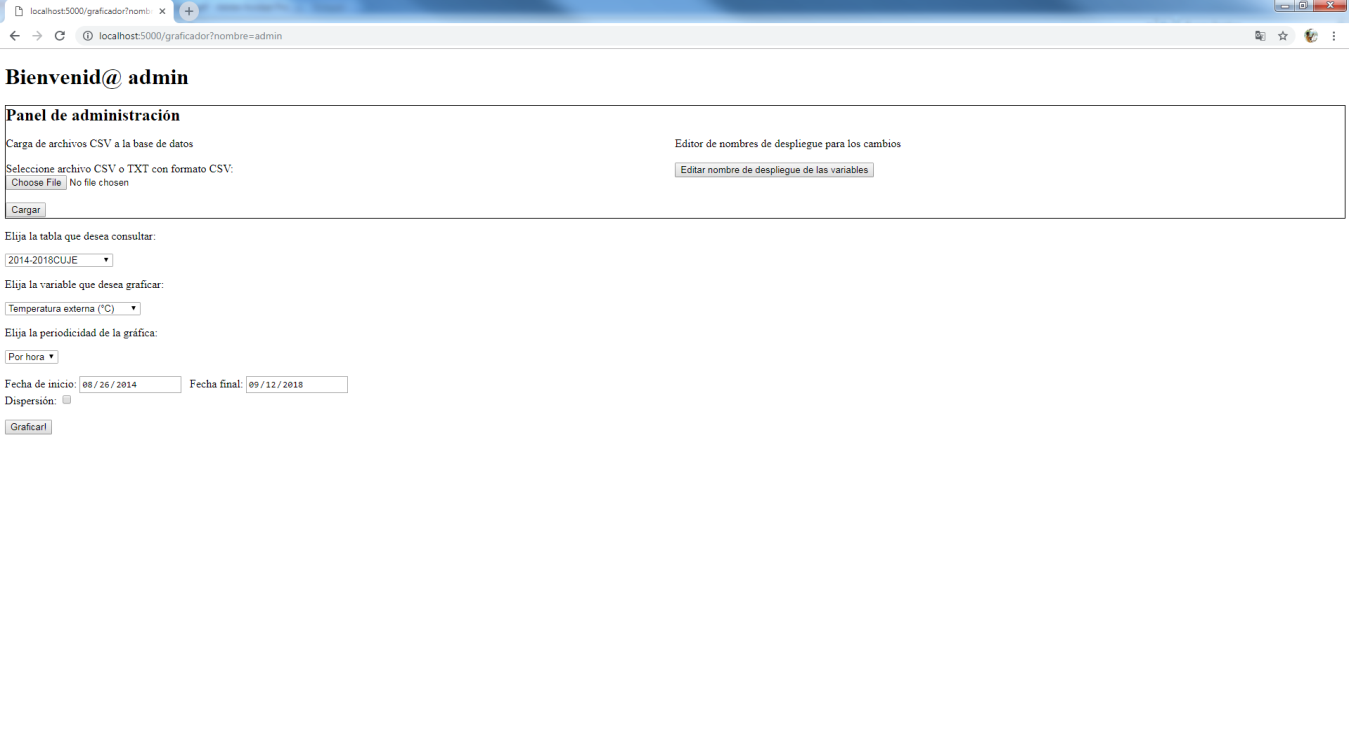
1. Selección de tabla: listado cargado automáticamente con las tablas existentes en la base de datos.
2. Selección de parámetro a graficar: incluye los parámetros numéricos (exclusivamente los numéricos) contenidos en el reporte de las estaciones meteorológicas que alimentan la base de datos.
3. Selección de periodicidad de graficado: la aplicación supone que se cargan datos por hora. Este parámetro permite graficar por hora o graficar resúmenes por día y por mes. Los resúmenes presentan un promedio de los valores por hora del parámetro seleccionado, para el periodo correspondiente.
4. Filtro de fechas: selección mediante calendario de las fechas iniciales y finales de datos a graficar. Los filtros son cargados automáticamente con la fecha más antigua y la más reciente registrada en los datos de la tabla seleccionada. Seleccionar períodos que abarquen fechas fuera del rango máximo en la tabla no produce errores. Simplemente grafica los datos disponibles en ese intervalo de tiempo.
5. Tipo de gráfica: habilitar o deshabilitar esta opción configura la aplicación para dibujar gráficas de dispersión o de líneas, respectivamente.
6. Botón para graficar y enlace de descarga de datos: hacer *click* sobre el botón “Graficar!” recargará la página y mostrará una gráfica siguiendo los parámetros especificados. En caso de error, la gráfica no es mostrada. Reportes de errores se encuentran, al ejecutar la aplicación localmente, en la ventana de comandos abierta para iniciar la aplicación, la cual permanecerá abierta durante todo el tiempo que la aplicación (específicamente, que el servidor de flask) esté funcionando. En general, los errores y mensajes sobre el funcionamiento de la aplicación se despliegan en lo que el servidor, sea cual sea, tenga configurado como “salida estándar”.

El enlace de descarga de datos graficados permitirá descargar un archivo separado por comas (CSV) con los datos de la tabla que han sido graficados. Este enlace aparecerá únicamente cuando se haya dibujado exitosamente una gráfica con el botón “Graficar!”.

1. La gráfica: incluye título de la gráfica y los ejes, escala de valores sobre cada eje (calculada automáticamente) y funcionalidades de acercamiento.
2. Controles de acercamiento: funcionalidad habilitada mediante un *plugin* de mpld3. El botón con forma de lupa permite rectangular un área de la gráfica para luego hacer *zoom in* sobre esa área automáticamente, ajustando la escala sobre cada eje. El botón con forma de casa permite restablecer la gráfica a su estado original. El botón con forma de cruz de flechas permite desplazar el área de acercamiento sobre la gráfica, manteniendo las proporciones.

### Modo de edición

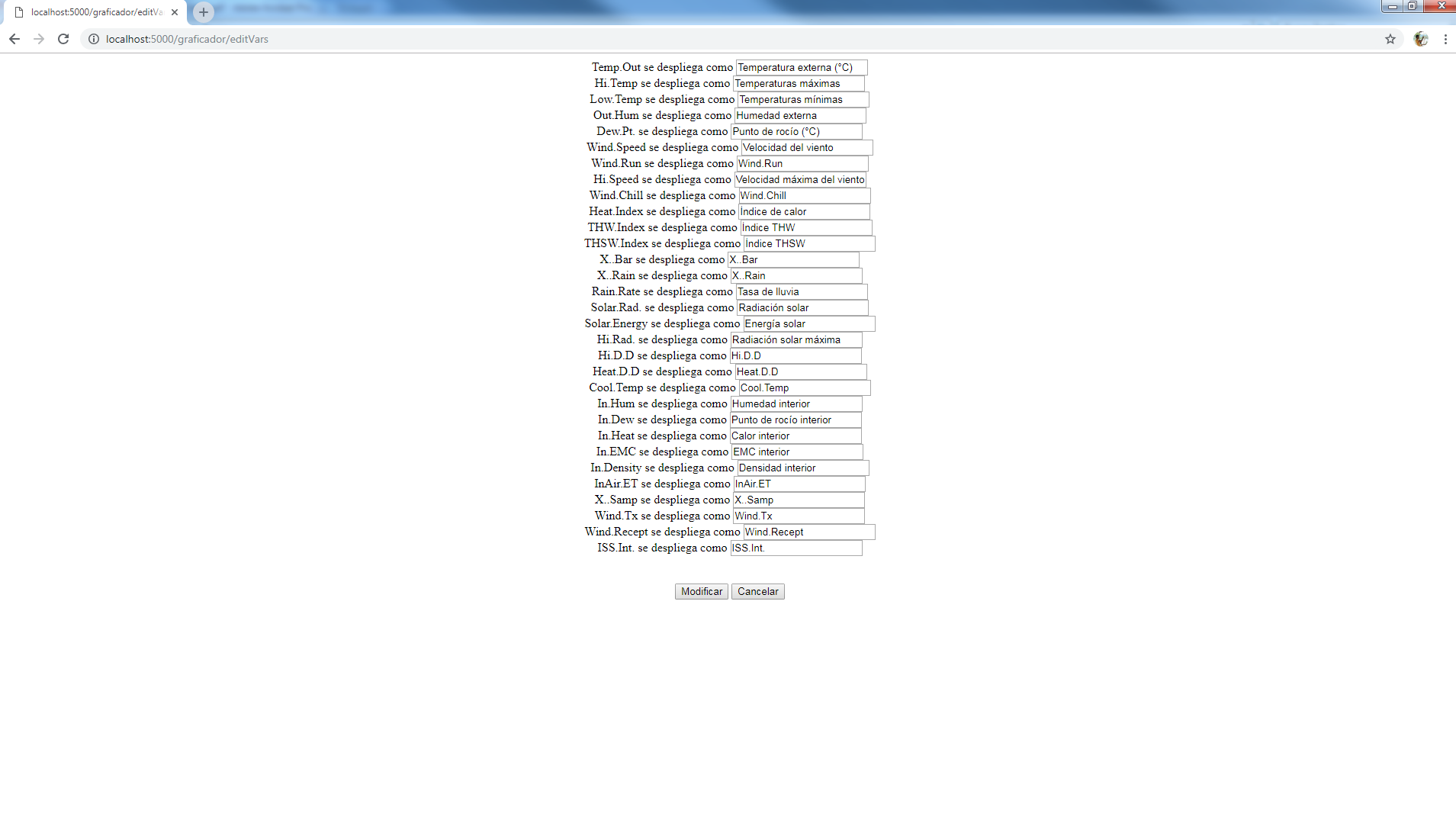
El modo de edición es activado al ingresar al sistema con el nombre “admin”. Permite cargar archivos de texto con nuevos datos y editar los nombres de despliegue de las variables en el listado de datos graficables (esto ajusta automáticamente los títulos de la gráfica y sus ejes).



**i.**

**ii.**

1. Carga de archivos: los archivos deben contener información organizada y estructurada como en el archivo de ejemplo incluido con esta documentación, llamado “SanJacintoJulio\_12\_09\_2018.txt”. La extensión del archivo es poco relevante, mientras el contenido esté en el formato correcto. Los archivos cargados son procesados por un *script* escrito en R, que se encarga de corregir los espacios en el archivo y de crear la tabla en la base de datos con la información contenida en el archivo cargado. Este *script* es muy básico, por lo que presenta requerimientos especiales para funcionar correctamente (ver sección de advertencias y recomendaciones). El botón “Choose File” (cuya etiqueta es determinada automáticamente por el navegador en el que la aplicación se ejecuta) abre una ventana de selección de archivos. Luego de la selección, hacer *click* en el botón “Cargar” para cargar el archivo seleccionado.
2. Edición de nombres: *clickear* este botón redirige el navegador a una página que muestra el nombre “real” de cada campo (*i.e.*, los nombres que los identifican en la base de datos) y los nombres asignados como reemplazo de cada uno. Nombres de reemplazo que se dejen vacíos son ignorados, usándose en cambio el nombre real de cada campo en cuestión. Para guardar los cambios hacer *click* en el botón “Modificar”. Para ignorar los cambios y volver a la última configuración usada, hacer *click* en el botón “Cancelar”.



## Advertencias y recomendaciones

* La aplicación fue desarrollada y probada localmente. No se ha instalado en un ambiente de producción ni se ha interactuado con la aplicación de forma remota, únicamente en el equipo en el que se desarrolló.
* La aplicación se desarrolló usando el servidor web de flask configurado para *debugging*. La documentación de Flask recomienda no usar el servidor de flask en un ambiente de producción, por ofrecer pobre escalamiento. Para trasladar la aplicación a un servidor web en un ambiente de producción, consúltese la siguiente documentación: <http://flask.pocoo.org/docs/1.0/deploying/#deployment>.
* Se dedicó muy poco tiempo y esfuerzo al aspecto visual de la aplicación. Se recomienda, para hacerla más amigable con usuarios, trabajar sobre los *templates* con extensión .html y agregar, directamente o como archivos estáticos, estilos mediante CSS (consúltese la siguiente documentación: <http://flask.pocoo.org/docs/1.0/quickstart/#static-files>).
* No se desarrolló “código seguro”. Se recomienda realizar pruebas y análisis de seguridad sobre el código y la aplicación antes de ponerla en producción, para reducir el riesgo de incidentes.
* El *script* en lenguaje R, desarrollado para trasladar la información meteorológica de un archivo CSV a una tabla en la base de datos, es muy específico en su forma de operar. Espera archivos exportados con el formato presentado en el archivo ejemplar “SanJacintoJulio\_12\_09\_2018.txt”, incluido con esta documentación. Para la correcta lectura y traslado de información a la base de datos, el *script* aplica algunos reemplazos y cambios al texto del archivo con la data, pero dichos cambios podrían alterar de formas inesperadas el contenido de un archivo de texto que no cumpla con el formato esperado. Los datos son importados a una nueva tabla en la base de datos, cuyo nombre dependerá del nombre del archivo cargado.
* El archivo “prettyVars.txt” no debería ser modificado directamente, sólo a través del modo de edición de la aplicación.
* Aunque no es necesario, la documentación de Flask recomienda el uso de ambientes virtuales para la instalación de la aplicación y su *deployment*. Durante el desarrollo se empleó “venv” siguiendo las recomendaciones de la documentación, aunque esto no afecta el comportamiento de la aplicación. Localmente, puede usarse sin acceder al ambiente virtual siguiendo las instrucciones provistas en este documento. Sin embargo, es altamente recomendable usar ambientes virtuales si se traslada la aplicación a producción (ver <http://flask.pocoo.org/docs/1.0/installation/#virtual-environments>).
* Todos los archivos de código se han publicado como *software* gratuito y abierto, disponible bajo los términos de la GNU General Public License (ver <http://www.gnu.org/licenses/>). Por tanto, el *software* se provee sin garantías.
* El repositorio en línea donde se almacena la aplicación se halla en la siguiente dirección: <https://github.com/jin-gitaxias/PracticasMate>. Es un repositorio público, de uso personal del desarrollador, y se ha almacenado allí el código de la aplicación para fines de respaldo y referencia personal.
* La base de datos usada en el equipo donde se desarrolló y probó la aplicación se llama “DataClimatica”. La base de datos puede ser accedida con el usuario “sig” y la contraseña “normalSIG”. El manejador de bases de datos usado es PostgreSQL 10. El usuario administrador de esta instalación de PostgreSQL (llamado “postgres”), tiene por contraseña “superSIG”. No se recomienda conectar aplicaciones a bases de datos mediante cuentas con privilegios, por los riesgos que esto presenta a la seguridad de la información. Por ello, la aplicación se conecta mediante el usuario “sig” al a la base de datos en la computadora de desarrollo. Sin embargo, la información de autenticación se ha excluido del repositorio en la nube donde la aplicación se almacena.
* Para instalar la aplicación en un ambiente de producción será necesario modificar el archivo “database.ini” y proporcionarle la información de conexión necesaria (esto en caso de que no se emplee un sistema más avanzado para lectura de configuraciones). La información de autenticación es también usada por el *script* “migradorDB.R”, aunque no es obtenida del archivo “database.ini”. En su versión local, la aplicación usa un archivo de configuración de ambiente de R para guardar y cargar la información de conexión a la base de datos. Este archivo se ha excluido del repositorio. Para información sobre archivos de configuración de ambiente de R ver:

<https://www.rdocumentation.org/packages/base/versions/3.5.2/topics/Startup>.