Manual de Desarrollador Kai Saber 5





Tabla de Contenido

INTRODUCCIÓN	4
REPOSITORIO DEL PROYECTO	
ARQUITECTURA DEL PROYECTO	
La estructura del proyecto es la siguiente:	5
DEPLOY	. 6
DOCUMENTACIÓN	. 6

INTRODUCCIÓN

En el presente documento se presenta la primera versión de Kai Saber 5, un software basado en la web, que permite el reconocimiento y visualización de datos correspondientes a los resultados de las pruebas Saber 5 del 2017. Kai Saber 5 permite realizar un análisis a través de la visualización de datos; estadísticas relevantes y diferentes tipos de gráficos según el rendimiento académico de la selección. Esta herramienta fue desarrollada por el grupo de investigación GRIAS, del departamento de ingeniería de Sistemas de la Universidad de Nariño. Se desarrolló bajo un entorno de programación de Python con la librería Dash Plotly y el framework Flask.

Este manual está enfocado a aquellos usuarios que requieran conocer la estructura y configuración de Kai Saber 5. Para este motivo, el lector que tenga la intención de revisar el código fuente debe tener un conocimiento básico en los siguientes temas:

- Arquitectura MVC
- · Bases de datos relacionales (PostgreSQL)
- Python
- Dash Plotly Library
- Flask Framework
- · Pandas y Numpy
- Control de versiones (GitHub)



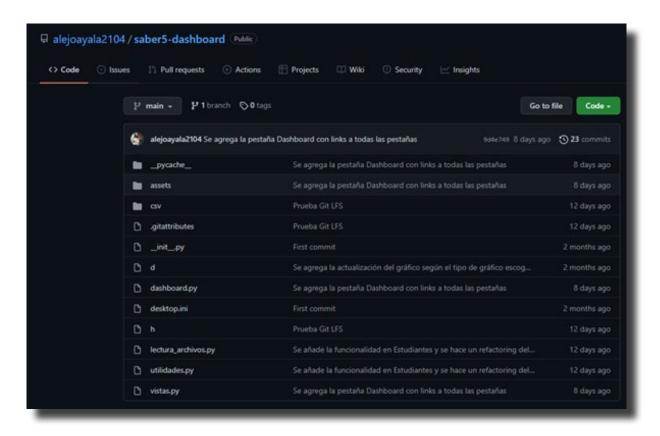
REPOSITORIO DEL PROYECTO

Se utilizó GitHub como plataforma de colaboración y control de versiones en la nube. Dentro de este repositorio se puede revisar la evolución del proyecto, reportar errores dentro del funcionamiento o código del sistema, y también generar ramas para proponer nuevas funcionalidades en Kai Saber 5.

Enlace del repositorio GitHub:

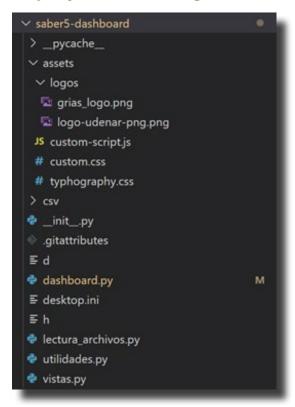
https://github.com/alejoayala2104/saber5-dashboard

ARQUITECTURA DEL PROYECTO

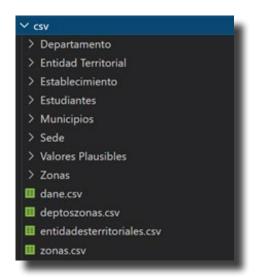


El proyecto del Dashboard Kai Saber 5 está construido bajo una adaptación de la arquitectura MVC, esto debido a que la librería Dash combina la creación de vistas y el manejo de controladores en un mismo archivo, pero las separa en tiempo de ejecución, es decir, el código front-end es generado después de la interacción del controlador con el modelo. Para el manejo de archivos se utilizó la librería Pandas de Python.

La estructura del proyecto es la siguiente:



- El directorio assets contiene los archivos necesarios para interactuar las vistas (HTML, CSS, JS) con la librería Dash y Python interprete los archivos en tales formatos.
- El directorio csv contiene todos los archivos con los datasets limpios y transformados en formato CSV.



- El archivo dashboard.py es el archivo principal de Dash que define la creación de Callbacks y la ejecución de la aplicación junto al despliegue del servidor de producción.
- El archivo lectura_archivos.py contiene el código necesario para leer todos los archivos de la carpeta csv e interpretarlos como DataFrames de la librería Pandas.
- El archivo vistas.py declara las funciones que inicializan y generan contenedores de visualización HTML a través de Dash. Cabe resaltar que, aunque este archivo está separado del archivo principal, en dashboard.py también se generan vistas de aplicación.
- El archivo utilidades.py define funciones de utilidad para asegurar el código limpio y reutilizable.

DEPLOY

Para realizar el despliegue de la herramienta Kai Saber 5 se utilizó la librería Pyinstaller de Python, la cual genera un directorio que contiene todas las dependencias y archivos necesarios para la ejecución de la aplicación, además de un archivo .exe para simplificar su ejecución.

La librería Dash hace uso del framwork Flask para levantar el servicio de servidor web, porque sirve como intérprete de archivos HTML, CSS y JS, entonces el servidor se levanta por defecto en localhost y se accede a la aplicación a través del navegador por el puerto 3000.

DOCUMENTACIÓN

Los archivos de creación del Dashboard tienen sus respectivos comentarios indicando su funcionamiento. En cada sección del directorio csv se encuentra un archivo de documentación sobre las variables del Saber 5, este archivo es utilizado para la generación de tooltips al acercar el cursor sobre el nombre de las variables en las tablas de visualización.

La estructura del Dashboard se constituye de forma similar a la estructura del directorio csv, entonces se generan Callbacks y funciones para cada sección, las cuales son separadas por comentarios con títulos similares a los de cada pestaña del Dashboard.

```
@app.callback(

Output(component_id='deptodepto', component_property='options'),

[Input(component_id='deptozonas', component_property='value')]
)

def update_deptodepto(zona_seleccionada): ...
```

Para la ejecución del programa se debe ejecutar el archivo dashboard.py con el intérprete de Python en cualquier entorno de desarrollo, o en la terminal. Al final de este archivo se encuentra la llamada al método main, donde se despliega el servidor de Flask y se puede configurar el puerto y la opción de modo debug.

```
if __name__ == '__main__':
    app.config.suppress_callback_exceptions = True
    # app.enable_dev_tools(
    # dev_tools_prune_errors=False
    # )
    app.run_server(debug=False,port=3000)
```

Al activar el modo debug con debug=True, se despliegan opciones de desarrollador tales como la alerta de errores, el mapeo de callbacks, el estado del server y reportes en consola.

