

Presentación técnica de la herramienta de diagnóstico.

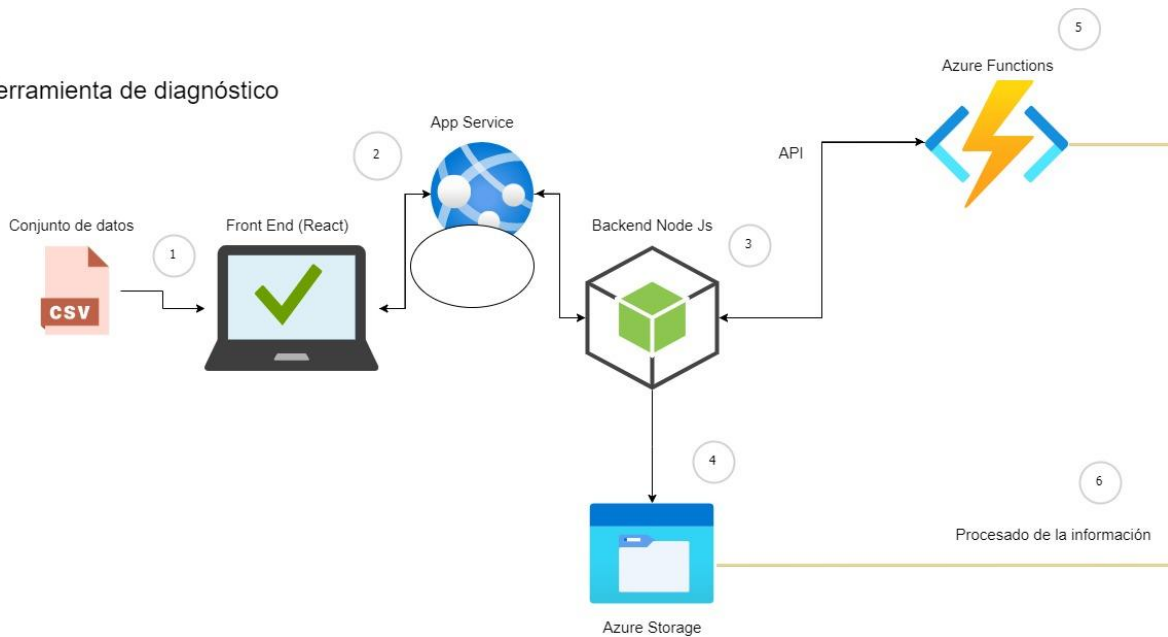
Los conceptos a continuación descritos son única y específicamente para la herramienta de diagnóstico

Descripción general de la Herramienta

La herramienta de diagnóstico se divide en 3 partes fundamentales, el **Front End** que es la parte visual y de interacción del usuario final, El **Back End** se encarga de comunicarse con los servidores de almacenamiento y guardar el archivo apropiadamente para su posterior procesamiento, también se encarga de consumir La respuesta final Y enviarla visualmente a través del Front End al usuario final, y por último se encuentra una **Azure Function** la cual se encarga de leer el archivo en el contenedor de almacenamiento procesarlo(calificarlo) y devolver una respuesta con un formato JSON (*estructura de datos estándar para almacenar e intercambiar información entre sistemas de información*).

Gráfico de conexión entre los 3 sistemas de información

Herramienta de diagnóstico

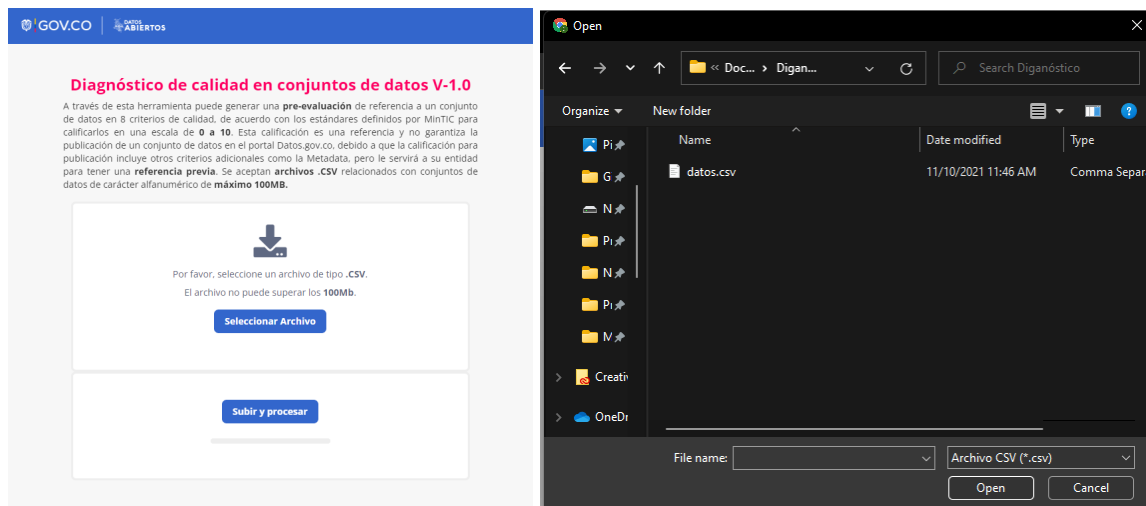


Funcionamiento interno del sistema (explicación del gráfico)

Después de cargado el archivo en el Front se envía por medio de una petición POST (protocolo utilizado para la comunicación entre el front End y el Backend de la aplicación) al Backend realizado en NodeJS, este se comunica con el servidor de azure storage Por medio de un SDK (Kit de desarrollo de software) y guarda el archivo en una carpeta preexistente, después de que la respuesta del servidor de almacenamiento sea satisfactoria se consume una API en la Azure Function para que se ejecute el proceso para leer y calificar el archivo y devuelva los resultados en un formato JSON, el Back End obtiene la respuesta y le responde al Front para mostrarlo en pantalla al usuario final por medio de una tabla que contiene los resultados del diagnóstico

Funcionamiento de cara al usuario

La aplicación se ejecuta en un entorno web, por lo que el usuario debe ingresar desde un navegador ingresando a la url calidad.datosabiertos.com.co en esta se encuentra un botón que dice **seleccionar archivo** el cual al ser presionado abrirá una ventana emergente para seleccionar el archivo a diagnosticar y cuenta con un filtro el cual solo deja seleccionar archivos con la extensión **.CSV** que es uno de los requerimientos indispensables para la carga y procesamiento, también hay un filtro interno para que el archivo no superé los 100 megabytes de peso.





Después de haber realizado este primer paso saldrá un texto que dice **“archivo <nombre del archivo> cargado correctamente”** y un botón para cargar un archivo diferente llamado **“cargar archivo diferente”** en caso de haber seleccionado un archivo equivocado.




El siguiente paso será dar clic en el botón que dice **subir y procesar** el cual mostrará una respuesta de validación del archivo, en caso de ser válido saldrá en color azul con un

mensaje que dice: el archivo se ha cargado correctamente y se está procesando y contiene un botón para cargar otro archivo en caso de que quiera repetir el proceso. También se verá una pantalla de precarga de los resultados que da a entender que se está procesando el archivo y que la respuesta puede tardar unos segundos en llegar



Diagnóstico de calidad en conjuntos de datos V-1.0

A través de esta herramienta puede generar una **pre-evaluación** de referencia a un conjunto de datos en 8 criterios de calidad, de acuerdo con los estándares definidos por MinTIC para calificarlos en una escala de **0 a 10**. Esta calificación es una referencia y no garantiza la publicación de un conjunto de datos en el portal Datos.gov.co, debido a que la calificación para publicación incluye otros criterios adicionales como la Metadata, pero le servirá a su entidad para tener una **referencia previa**. Se aceptan **archivos .CSV** relacionados con conjuntos de datos de carácter alfanumérico de **máximo 100MB**.



Archivo Válido

El archivo se ha cargado correctamente y se está procesando...

Cargar otro Archivo

Resultados:

Por último el sistema arroja los resultados del diagnóstico del archivo clasificándolos de 0-10 dónde la calificación se muestra en la parte derecha y en la parte izquierda se ve el criterio calificado.

calificarlos en una escala de **0 a 10**. Esta calificación es una referencia y no garantiza la publicación de un conjunto de datos en el portal Datos.gov.co, debido a que la calificación para publicación incluye otros criterios adicionales como la Metadata, pero le servirá a su entidad para tener una **referencia previa**. Se aceptan **archivos .CSV** relacionados con conjuntos de datos de carácter alfanumérico de **máximo 100MB**.



Archivo Válido

El archivo se ha cargado correctamente y se está procesando...

[Cargar otro Archivo](#)

Resultados:

Completitud

10/10

Credibilidad

10/10

Comprensibilidad

10/10

Consistencia

10/10

Exactitud

10/10

Unicidad

10/10

Desarrollo de la aplicación a nivel tecnológico y de programación

Frontend:

La aplicación web se desarrolla usando como lenguaje principal React.js, con el componente y/o librería Axios, para la comunicación con el backend de la aplicación encargado de enviar los datos y recibir los resultados del diagnóstico. Tiene tres componentes principales separados en tres archivos index.html, data.js y style.css.



Index.html

está encargado de la maquetación de la página, internamente se encuentra el código que muestra los contenedores, el texto y los botones de la misma. Así como la importación de los archivos para el manejo de los datos y estilos visual (data.js y style.css)

data.js

Este archivo se encarga de la validación antes de la carga al servidor teniendo como parámetros lo anteriormente mencionado que sean archivos de tipo .CSV y que el tamaño no superé los 100 mb, si uno de estos criterios falla, este devuelve como respuesta al usuario un mensaje de error diciendo la causa para su posterior corrección. Después de validado el archivo utiliza la librería de axios para enviar el archivo al backend y quedará a la espera de la respuesta para mostrarla posteriormente en la sección de resultados

style.css

Este archivo cumple la función de darle estilo a la página con los colores, tamaños, padding, margin correspondientes, llevando la imagen corporativa ya establecida de <https://www.mintic.gov.co/>

Backend:

La aplicación se desarrolla en Node.js, con las siguientes dependencias:

```
"azure-storage": "^2.10.5",
```

```
"dotenv": "^10.0.0",  
"express": "4.17.1",  
"into-stream": "3.1.0",  
"multer": "1.4.3",  
"request": "2.88.2"
```

azure-storage

Es el sdk oficial de Azure para El manejo de peticiones hacia el Data storage

dotenv

Tiene como objetivo el manejo de archivos privados para las cadenas de conexión y claves del sistema y evitar la filtración de datos.

Express

Proporciona mecanismos para: Escritura de manejadores de peticiones con diferentes verbos HTTP en diferentes caminos URL (rutas).

Into-Stream

Es un manejador de datos de tipo Stream(*convierte los datos a tipo stream*) para el correcto envío de archivos al BlobStorage

Multer

Es un middleware para Express y Node. js que hace que sea fácil manipular este multipart/form-data cuando tus usuarios suben archivos.

Request

Esta dependencia es usada para consumir el API de la function en Azure y obtener la respuesta de la calificación.

Adicional

Estas dependencias interactúan entre sí para el envío del archivo y el correcto almacenamiento en el azure storage después se consume la respuesta con la dependencia de Request y se envía al Front (en la documentación del código se encuentra el funcionamiento de todos los componentes y Cómo interactúan entre sí)

Azure Functions

Azure Functions es un servicio en la nube disponible a petición que proporciona toda la infraestructura y los recursos, que se actualizan continuamente, necesarios para ejecutar las aplicaciones

Functions proporciona proceso sin servidor para Azure, también se puede usar para crear API web, responder a los cambios en las bases de datos, procesar secuencias de IoT, administrar colas de mensajes, etc.

Para el caso del proyecto la Functions se utiliza como API web, donde mediante una petición POST a la url https://func-bdguidance-score.azurewebsites.net/api/score_dataset y con parámetros en el body: el nombre del archivo csv a procesar; da como respuesta con json con la siguiente estructura:

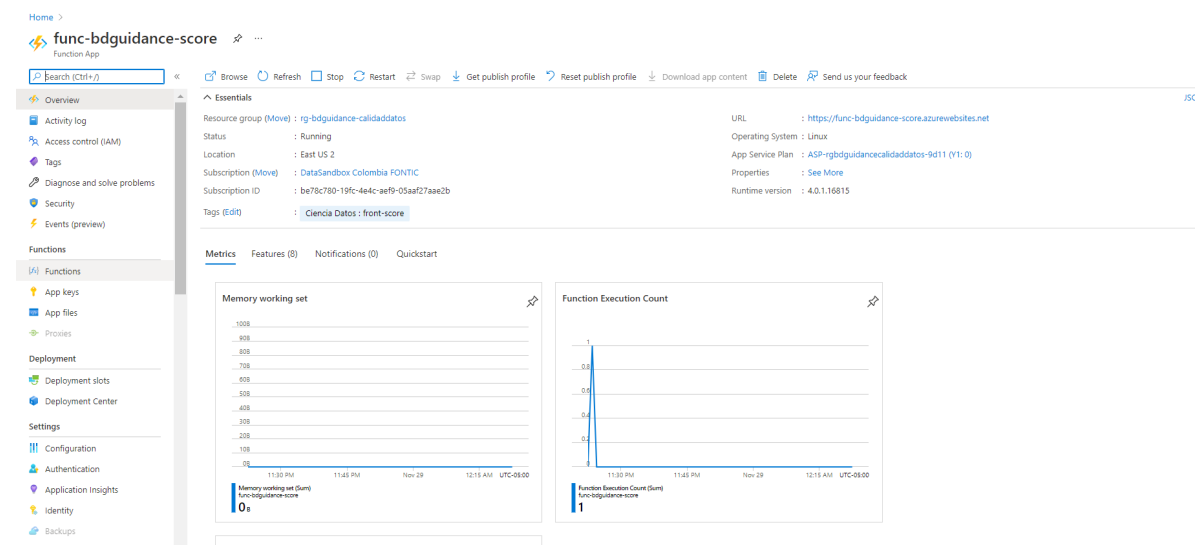
```
result = {  
    'tamaño' : 1,  
    'Conjunto de datos': 2,  
    'email' : 3,  
    'Compleitud': 4,  
    'Credibilidad': 5,  
    'Actualidad' : 6,  
    'Trazabilidad': 7,  
    'Disponibilidad': 8,  
    'Conformidad': 9,  
    'Comprensibilidad':10,  
    'Portabilidad':11,  
    'Consistencia': 12,  
    'Exactitud': 13,  
    'Unicidad': 14}
```

1. Tamaño: Describe la cantidad de filas del conjunto de datos procesado
2. Conjunto de datos: Nombre del conjunto de datos procesado, parámetro que se envía en el body
3. Los argumentos de 4 a 14 describen la puntuación de los criterios de calidad de datos, esta puntuación va de 1 a 10, siendo el 10 la mejor calificación del criterio indicando que el conjunto de datos cumple satisfactoriamente con las reglas definidas. Para los criterios de: trazabilidad, disponibilidad y actualidad siempre tendrán el valor de 'N/A' esto se debe a que estos criterios dependen de la metadata, y la functions está diseñada para calificar los criterios que dependan de los datos.

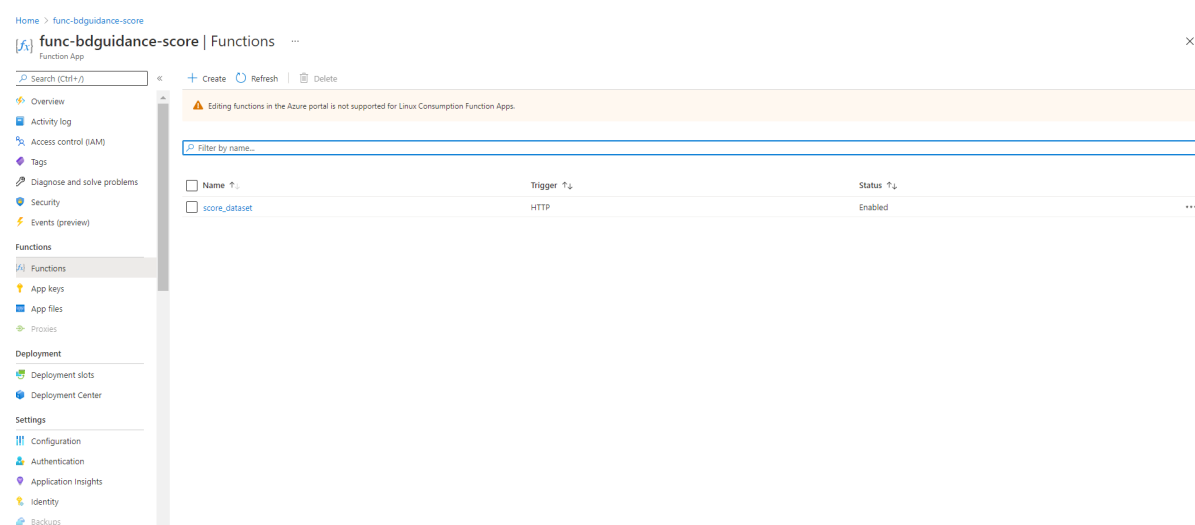
Para el desarrollo de la functions se utilizó el IDE Visual Studio donde se definieron las funciones necesarias para la calificación de cada uno de los criterios, en total se cuentan con 11 funciones.

El despliegue de la función se realizó en el portal de Azure con la cuenta de Data Sandbox, grupo recurso **rg-bdguidance-calidaddatos** con el nombre de **func-bdguidance-score**; el proceso del despliegue de la función se describe en la documentación de Microsoft <https://docs.microsoft.com/es-es/azure/azure-functions/functions-develop-vs?tabs=in-process> donde se describe el paso a paso de la creación y despliegue de las funciones.

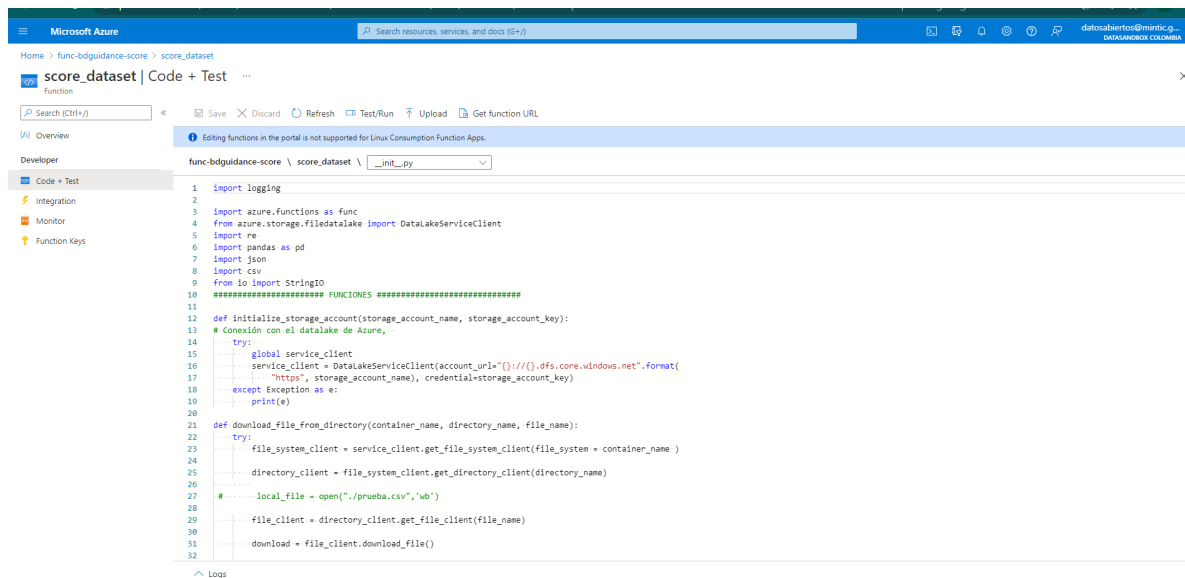
Para visualizar y descargar el código fuente de la función se debe ingresar al portal de Azure donde se tiene definido la función, seleccionarla y dar clic en la de la izquierda llamada Functions :



Luego dar clic en el nombre de la función que se definió como score_dataset:



En la opción de “Code + Test” se encuentra el código fuente y además la url donde se puede hacer las peticiones de la function.



```
1 import logging
2
3 import azure.functions as func
4 from azure.storage.filedatalake import DataLakeServiceClient
5 import re
6 import pandas as pd
7 import json
8 import csv
9 from io import StringIO
10
11 ##### FUNCIONES #####
12 def initialize_storage_account(storage_account_name, storage_account_key):
13     # Conexión con el data lake de Azure.
14     try:
15         global service_client
16         service_client = DataLakeServiceClient(account_url="{}://{}dfs.core.windows.net".format(
17             "https", storage_account_name), credential=storage_account_key)
18     except Exception as e:
19         print(e)
20
21 def download_file_from_directory(container_name, directory_name, file_name):
22     try:
23         file_system_client = service_client.get_file_system_client(file_system = container_name )
24         directory_client = file_system_client.get_directory_client(directory_name)
25
26         # local_file = open("./prueba.csv", "wb")
27         file_client = directory_client.get_file_client(file_name)
28         download = file_client.download_file()
```

En el código se comenta cada una de las funciones trabajadas.

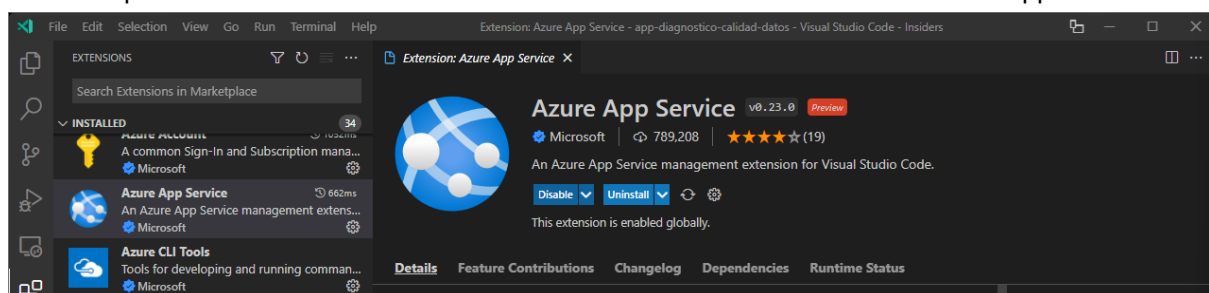
Infraestructura

Descripción general

Se creó dentro del grupo de recursos asignados un **AppService** tipo B1 El cual utiliza Node Js 14 y una dirección web temporal <https://calidadatos.azurewebsites.net/> el cual tiene como objetivo alojar el contenido del Backend y FrontEnd El versionamiento y actualización está conectado a una cuenta de github dónde se encuentra el código fuente y las respectivas ramas de desarrollo.

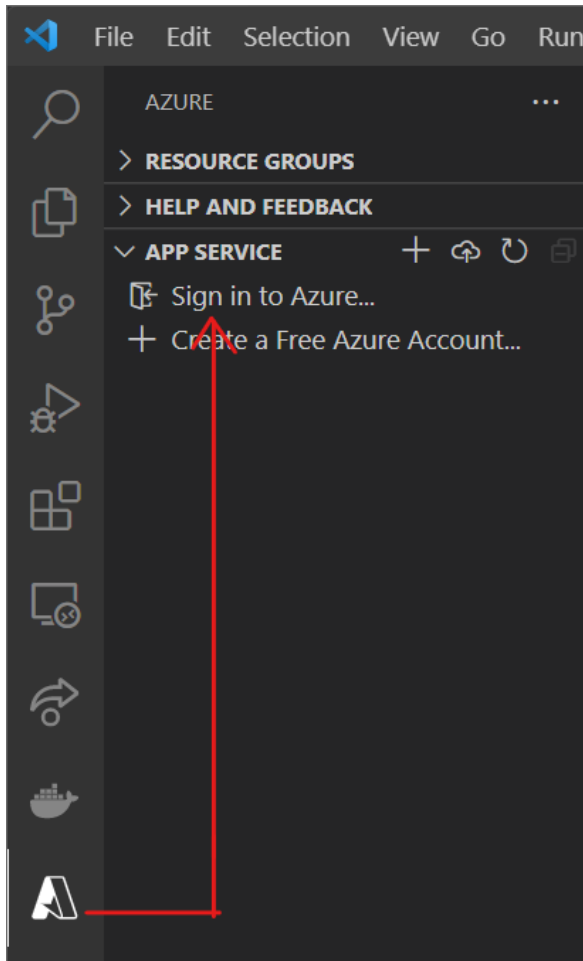
Implementación de la herramienta de diagnóstico en el AppService

Para la implementación se usó **visual Studio code** con la **extensión** oficial llamada AppService

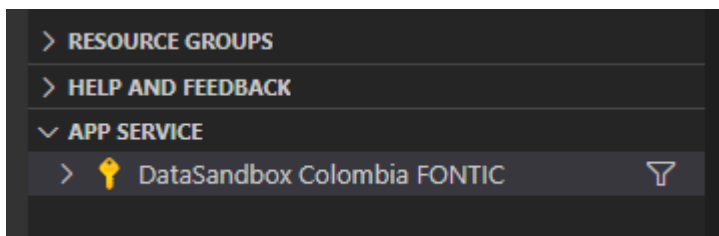


Despliegue de la aplicación en un entorno de producción

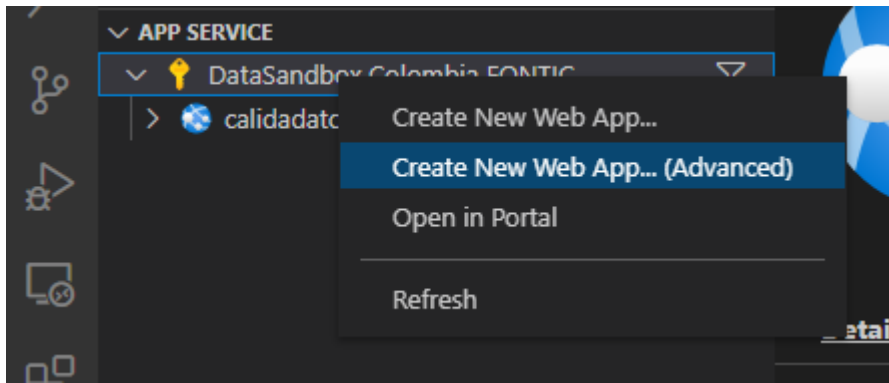
Lo primero para comenzar la implementación es acceder al portal de Azure con los datos de inicio de sesión y verificar la cuenta desde VS code



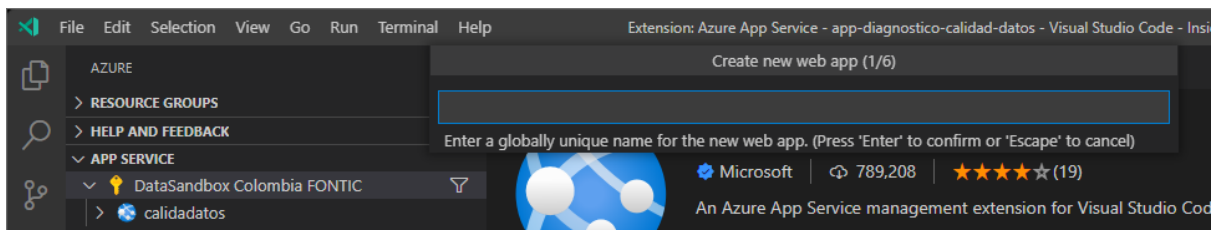
Si todo salio correcto aparecerá una llave con el nombre de usuario



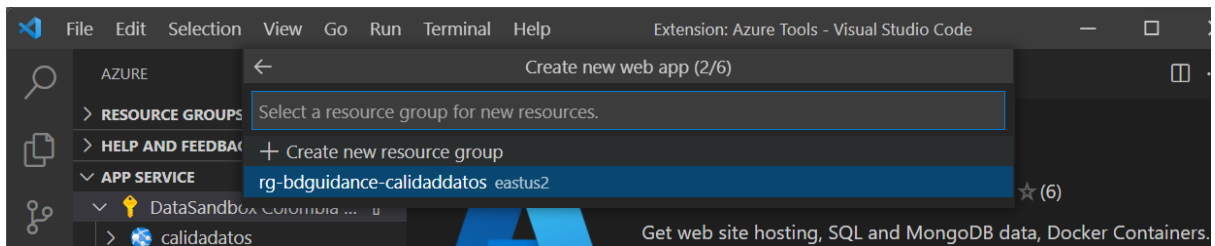
Seguido damos clic derecho y en la opción crear nuevo AppService (Avanzado)



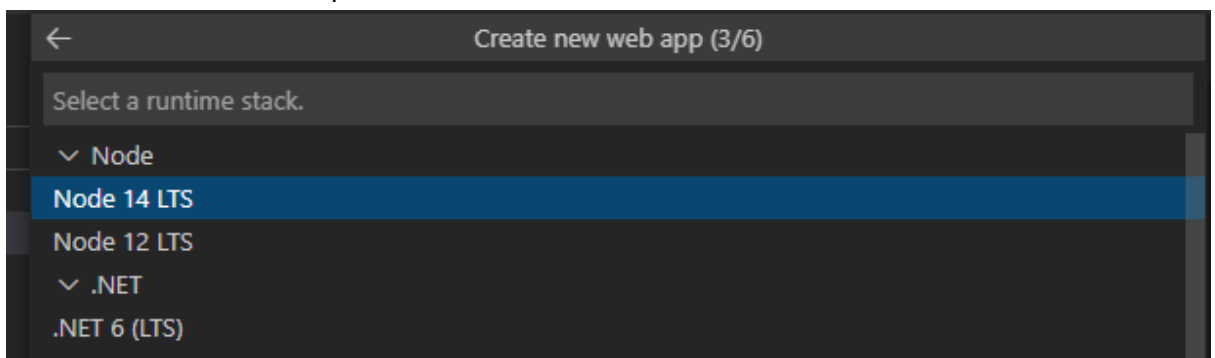
Nos pedirá un nombre para la aplicación, este debe ser unico y estará ligado al subdominio final



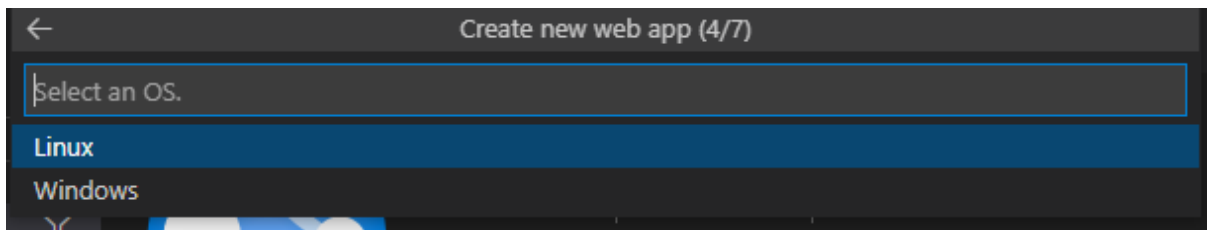
seleccionaremos el grupo de recursos correspondientes, en este caso ya está creado y configurado



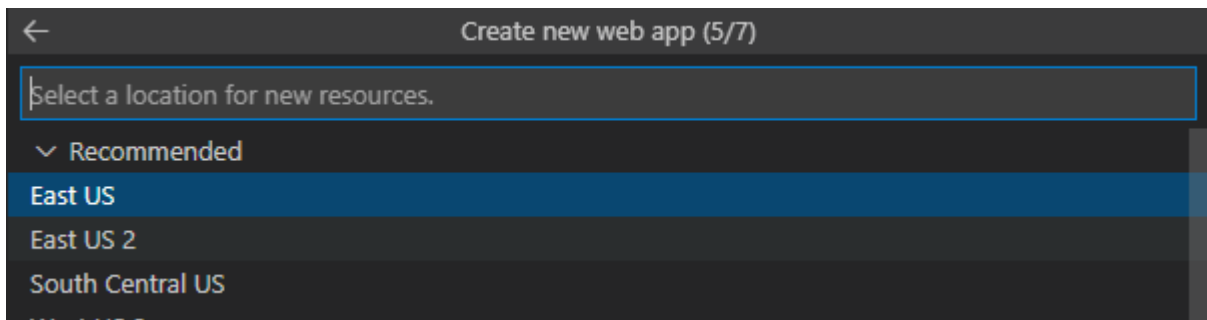
Seleccionamos el Stack que necesitamos en este caso Node Js en su versión 14



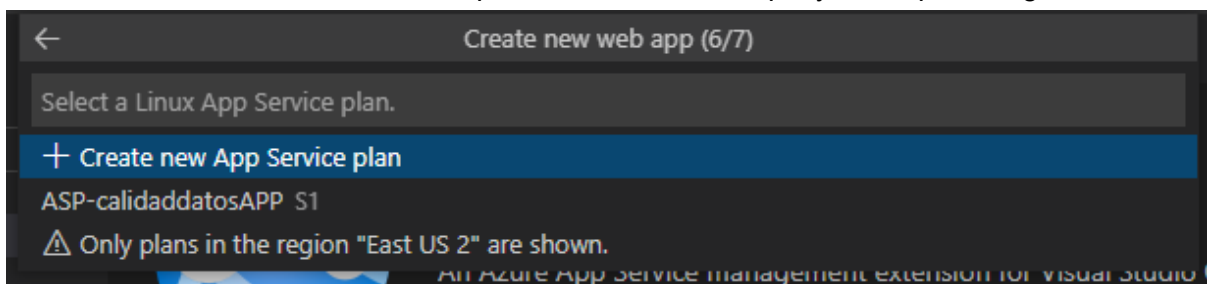
Ahora seleccionamos el sistema operativo de preferencia, en este caso Linux



Seleccionamos la locación dónde se va encontrar el “servidor”

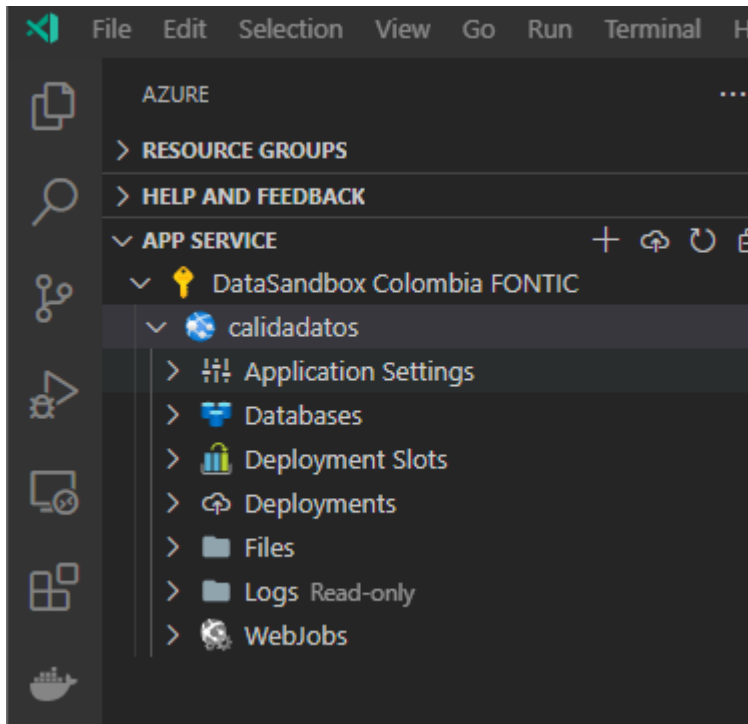


Por último seleccionamos el service plan en este caso S1 que ya está preconfigurado

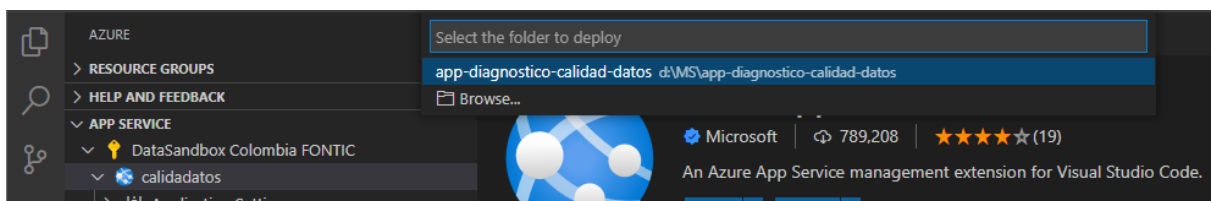
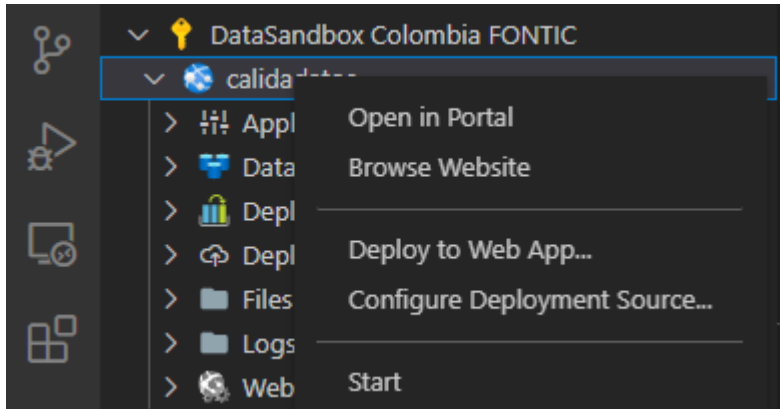


En este caso es lo más óptimo para el tipo de aplicación que se está utilizando, Pero puede variar basado en los estudios y cuando se vaya a implementar la solución.

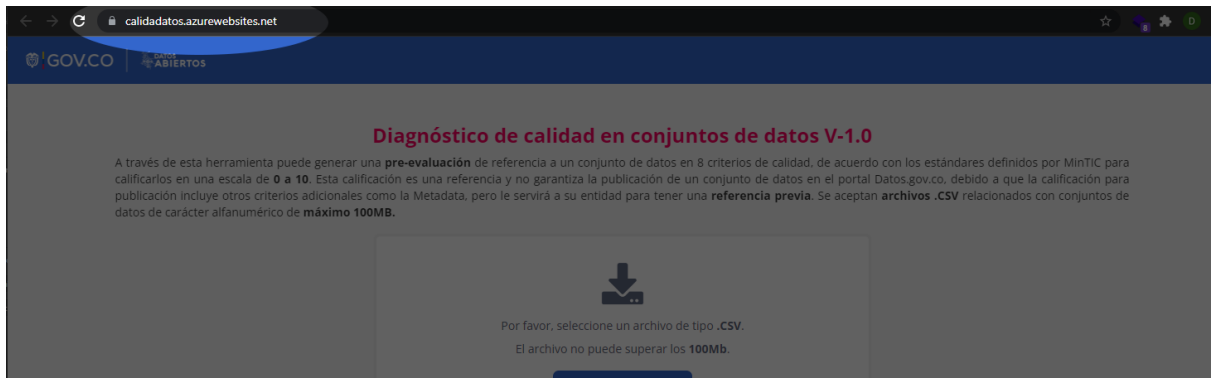
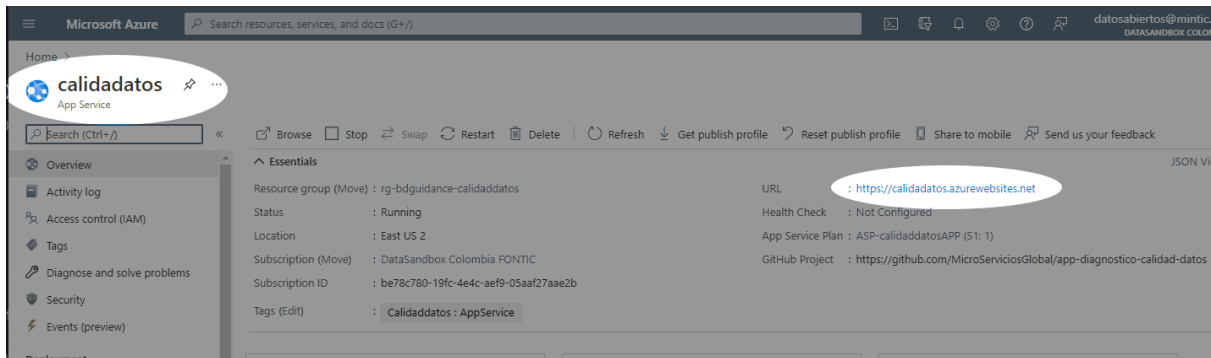
Eso es todo para la configuración, después de aceptar nos quedará un despliegue de la siguiente forma



Ahora simplemente seleccionamos **Deploy Web App** y seleccionamos la carpeta a sincronizar, en este caso debe ser nuestra carpeta con el código ya previamente descargada de GitHub



y listo, podremos ingresar al subdominio creado en el portal para verificar que todo se haya implementado correctamente



Pruebas de Software

Funcionales

Las pruebas funcionales se dividen en 2 partes, Pruebas unitarias y prueba de humo.

Las pruebas unitarias se realizan con el evento del botón subir y procesar del paso 2, y este ejecuta una serie de funciones conectadas entre sí para validar la precarga del archivo.

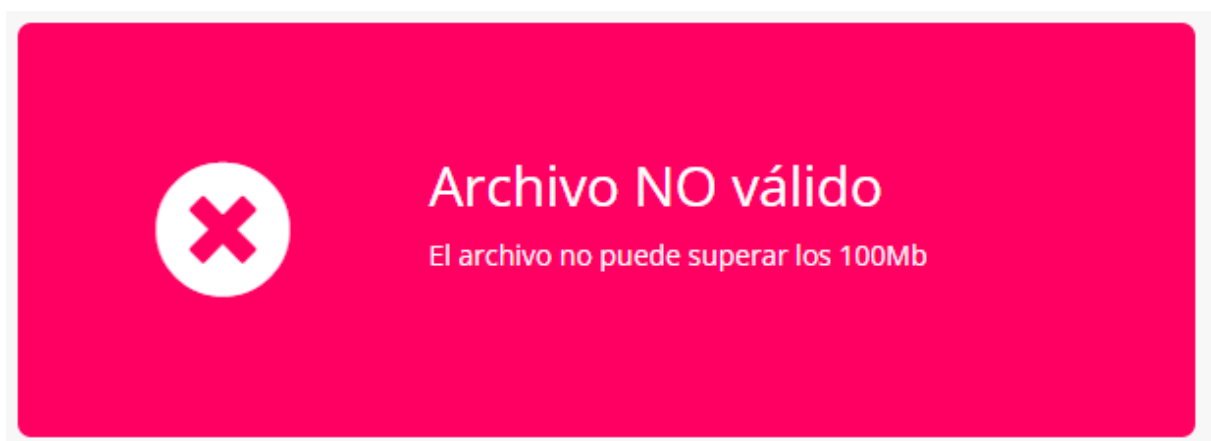
Pruebas unitarias.

Las salidas se mostrarán gráficamente para cada una de las validaciones para una mejor comprensión ya que las pruebas arrojan códigos de error y de este se muestra una alerta de color magenta cuando hay alguna inconsistencia, de lo contrario el color será azul

1. Validación de archivo no cargado



2. Si el archivo fue cargado correctamente se valida que el tamaño no sea superior a 100 MB



3. Si el archivo es muy pesado se mostrará la carga del mismo en una barra de progreso del 0% al 100% para que el usuario sepa en qué estado se encuentra el sistema



Archivo **C_DIGO_NICO_DE_MEDICAMENTOS_VIGENTES.csv** cargado correctamente


Cargar Archivo diferente

Paso 2

Subir y procesar

Cargando archivo...12%

4. Si ninguna de estas excepciones se cumple el código arrojará un mensaje de carga exitosa



Archivo Válido

El archivo se ha cargado correctamente y se está procesando...

Cargar otro Archivo

Si el código se rompe en el backend (**status 500**) se envía el siguiente mensaje de error



Si no arroja ninguna respuesta ni calificación esto significa que hubo algún fallo a nivel de código pero no devuelve nada (**status 200, data={}**)



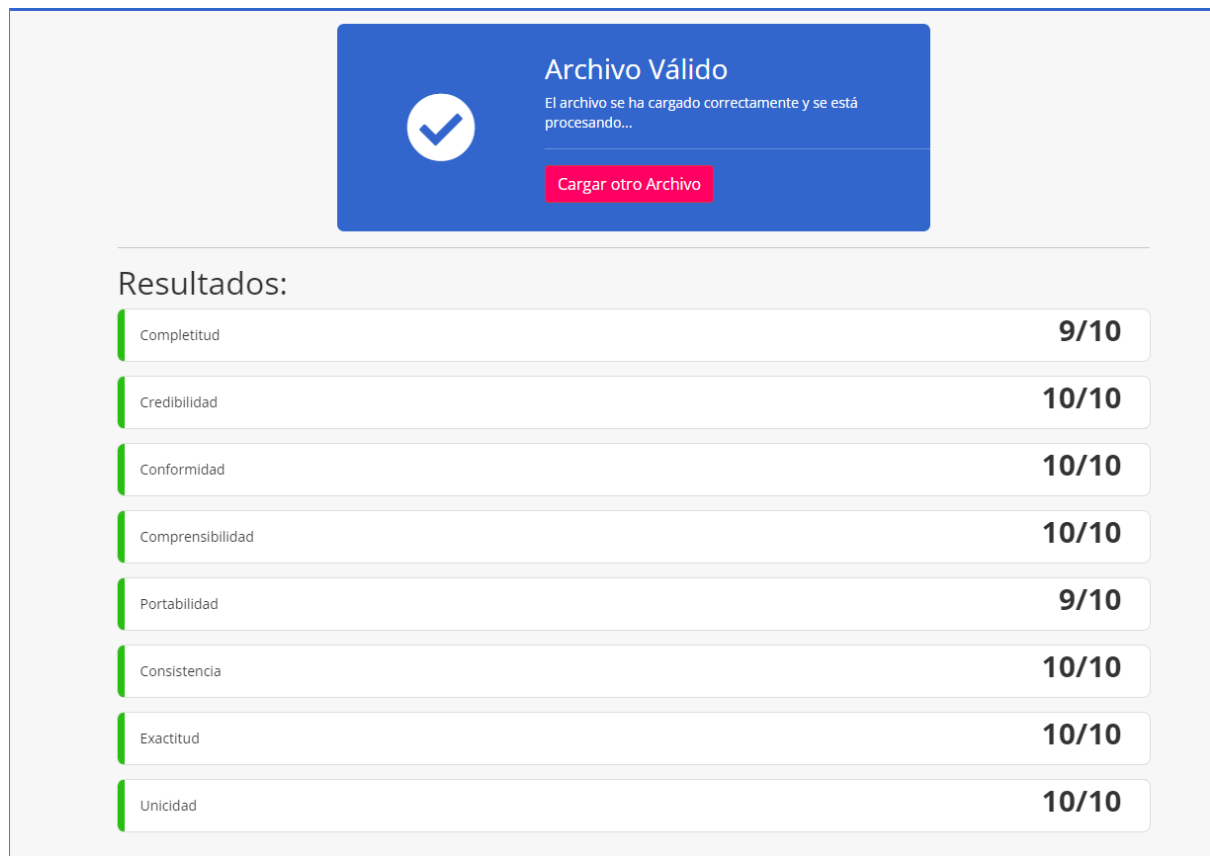
Si se tiene una respuesta con datos, pero el frontEnd no los puede interpretar, significa igualmente que el archivo tiene alguna inconsistencia, se arroja el error de lectura



Error al leer el archivo

Existe un problema al leer este archivo, verifique que sea un archivo CSV y que tenga el formato de datos correcto

Si ninguna de estas excepciones se cumple el código de status es válido y la data es correcta, el sistema arrojará un listado con las calificaciones correspondientes para una ventana final como la siguiente

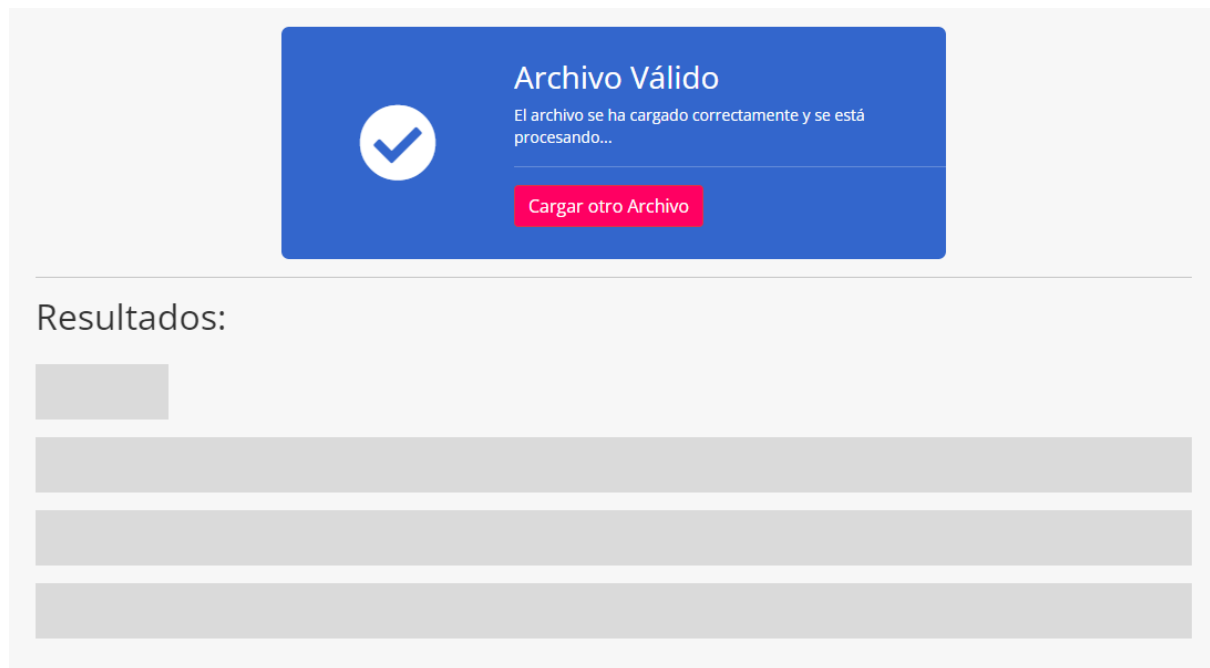


Como última acción el usuario podrá volver a comenzar pulsando el botón **Cargar otro archivo**

Pruebas de humo

Las pruebas de humo (smoke test) son una revisión rápida de un producto de software para comprobar que funciona y no tiene defectos evidentes que interrumpan la operación básica del mismo.

1. Se identificaron algunos problemas que ocasionaron que la página se quedara “cargando” pero nunca se obtenía una respuesta.



Esto se soluciona colocando TimeOuts (*Término que se refiere al momento en que un usuario hace uso de la Red por un período determinado hasta que se agota*) de esta forma se puede arrojar el error para el Status 500 que devuelve un mensaje con el evento que ha ocurrido

2. Complejidad de la página

En un inicio se lanzó una versión en la cual solo se resaltan las validaciones básicas de la aplicación sin ningún contexto para el usuario final

The screenshot shows the initial version of the 'Diagnóstico de calidad en conjuntos de datos' (Data Quality Diagnostic) interface. It features a blue header with the 'GOV.CO' logo and 'DATOS ABIERTOS' text. The main content area has a light gray background with a white box containing a download icon and instructions: 'Por favor, seleccione un archivo de tipo .CSV. El archivo no puede superar los 100Mb.' Below the instructions are two blue buttons: 'Seleccionar Archivo' and 'Subir y procesar'.

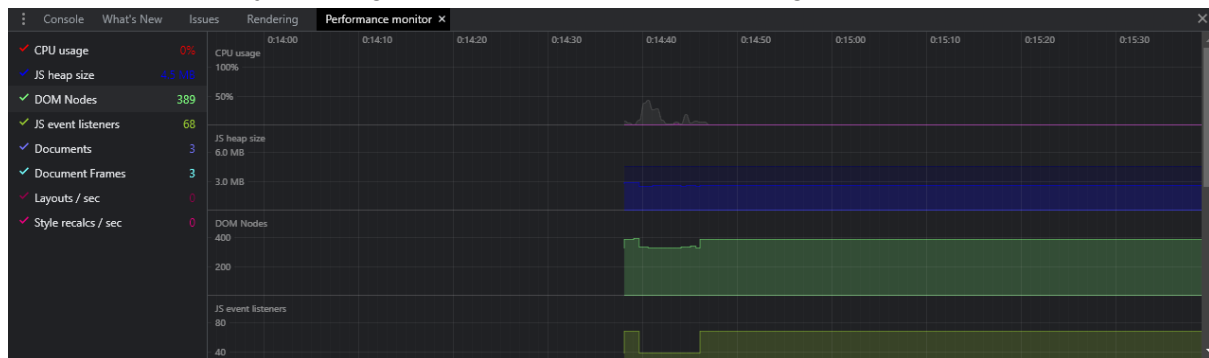
Después de las pruebas se realizaron algunos ajustes y se dejó contexto e instrucciones precisas de cómo usar la herramienta de diagnóstico

The screenshot shows the updated version of the 'Diagnóstico de calidad en conjuntos de datos' interface. It includes a detailed introduction paragraph explaining the tool's purpose and the evaluation scale (0 to 10). Below the text, there is a section titled 'Tenga en cuenta lo siguiente antes de cargar el archivo:' followed by three bullet points: 'Se aceptan únicamente archivos .CSV', 'Tamaño máximo de archivo 100Mb (Como referencia aproximadamente son: 200.000 registros)', and 'El archivo debe encontrarse en codificación UTF8'. The interface is divided into two steps: 'Paso 1' with a download icon and 'Seleccionar Archivo' button, and 'Paso 2' with a 'Subir y procesar' button.

No funcionales

Pruebas de rendimiento

las pruebas no arrojaron ninguna inconsistencia de sobrecarga



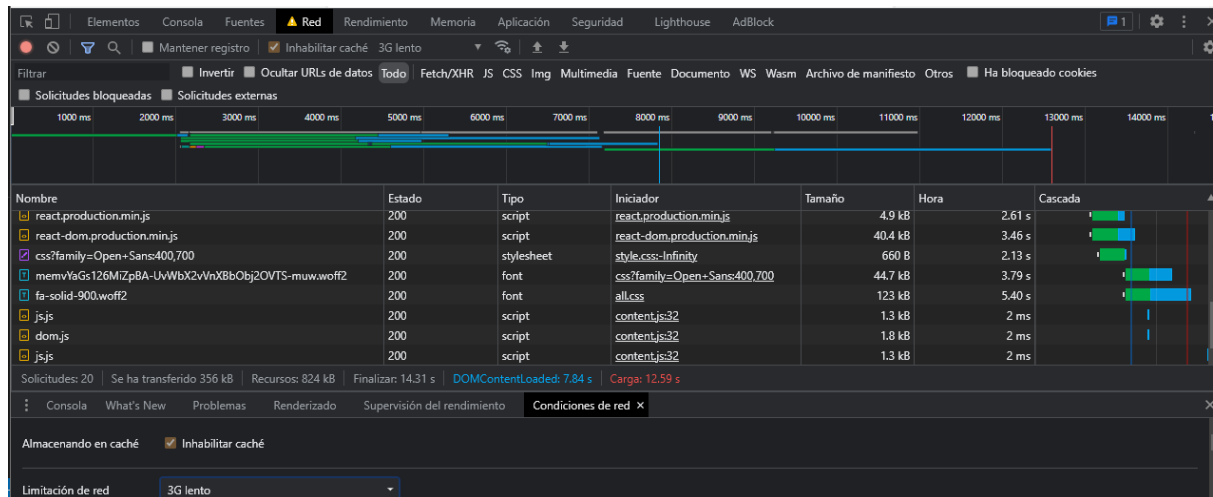
las pruebas no arrojaron ninguna inconsistencia de TimeOut y todos los recursos se cargan de manera adecuada

The screenshot shows the Network tab in Chrome DevTools. The top bar indicates the filter is set to 'All'. The table below lists the resources loaded during the test.

Name	Status	Type	Initiator	Size	Time	Waterfall
calidadatos.azurewebsites.net	200	document	Other	6.5 kB	108 ms	
normalize.min.css	200	stylesheet	(index)	1.5 kB	62 ms	
bootstrap.min.css	200	stylesheet	(index)	24.8 kB	67 ms	
all.css	200	stylesheet	(index)	29.7 kB	117 ms	
animate.min.css	200	stylesheet	(index)	4.3 kB	56 ms	
style.css?v3	200	stylesheet	(index)	5.7 kB	120 ms	
react.production.min.js	302	script / Redirect	(index)	150 B	44 ms	
react-dom.production.min.js	302	script / Redirect	(index)	131 B	44 ms	
bootstrap.bundle.min.js	200	script	(index)	23.8 kB	60 ms	
axios.min.js	200	script	(index)	6.7 kB	62 ms	
data.js?v3.0	200	script	(index)	7.0 kB	98 ms	
logo_pda.png	200	png	(index)	27.1 kB	198 ms	
react.production.min.js	200	script	react.production.min.js	4.8 kB	25 ms	
react-dom.production.min.js	200	script	react-dom.production.min.js	40.3 kB	44 ms	
css?family=Open+Sans:400,700	200	stylesheet	style.css-Infinity	660 B	100 ms	
fa-solid-900.woff2	200	font	all.css	123 kB	75 ms	
memvYaGs126MiZpBA-UvWbX2vVnXBbObj2OVTS-muw.woff2	200	font	css?family=Open+Sans:400,700	44.7 kB	38 ms	
js.js	200	script	content.js:32	1.3 kB	3 ms	
dom.js	200	script	content.js:32	1.8 kB	3 ms	
js.js	200	script	content.js:32	1.3 kB	3 ms	

En condiciones de red **3G (lento)** no se tuvieron inconvenientes, aunque la carga principal y la espera de subida del archivo puede tardar hasta **12s** mientras con que en condiciones normales suele estar sobre los **3.5s** la primera carga, y con el caché automático suele

rondar los **450ms**



Pruebas de usabilidad

Una de las mejores Herramientas para comprobar la usabilidad de una plataforma es [PageSpeed Insights](#) donde se analiza la web con mucho detenimiento del lado de la usabilidad y problemas relacionados con el rendimiento.



Diagnostica problemas de rendimiento

Obtén análisis detallados y recomendaciones al cargar tu sitio en un entorno simulado.

Esta URL

<https://calidadatos.azurewebsites.net/>

99

Rendimiento

Los valores son estimaciones y pueden variar. La [puntuación del rendimiento se calcula](#) directamente a partir de estas métricas. [Ver calculadora.](#)

▲ 0-49 ■ 50-89 ● 90-100



MÉTRICAS

[Ampliar vista](#)

● First Contentful Paint

0,8 s

● Speed Index

0,8 s

● Renderizado del mayor elemento con contenido

0,9 s

● Time to Interactive

0,8 s

● Total Blocking Time

0 ms

● Cumulative Layout Shift

0

📅 Captured at 14 dic 2021 0:31 GMT-5

🕒 Carga inicial de la página

🖥️ Escritorio emulado with Lighthouse 9.0.0

🚫 Limitación personalizada

🔗 Carga de una única página

🔗 Using HeadlessChromium 95.0.4638.69 with lr

Pruebas de estrés y volumen