

Artículo

[Alberto Fuentes](#) · 15 jun, 2022 Lectura de 2 min

[Open Exchange](#)

## Una interfaz gráfica y API de ejemplo para la utilidad SystemPerformance

¡Hola desarrolladores! Quería compartir hoy un ejemplo muy interesante por parte de [Tani Frankel](#). Se trata de una aplicación sencilla sobre la utilidad SystemPerformance.

Repasando nuestra documentación sobre la rutina de monitorización ^SystemPerformance (conocida como ^pButtons en versiones anteriores a IRIS), un cliente me dijo «Entiendo todo esto pero ojalá fuese más simple, más sencillo para definir perfiles y gestionarlos, etc.».

Entonces pensé que sería interesante como ejercicio facilitar una pequeña interfaz para hacer esas tareas más sencillas.

El primer paso era envolver en una API basada en clases la rutina actual de ^SystemPerformance.

Además, aproveché para añadir algunas otras funcionalidades como mostrar qué perfiles están ejecutándose actualmente, el tiempo que les falta, procesos que han estado en ejecución con anterioridad, etc.

El siguiente paso era añadir sobre esta API, una API REST.

Con este artefacto listo, cualquier puede ya lanzarse y construir una pequeña interfaz de usuario moderna.

Por ejemplo:

The screenshot displays a web application interface for managing pButtons profiles. At the top, there is a navigation bar with four tabs: "PROFILES MANAGEMENT", "CURRENT RUNS", "PREVIOUS RUNS", and "GENERAL". Below the navigation bar, a status bar shows "Last Updated: 16:00:12" and a "Refresh" button. The main content area is divided into two sections. The first section, titled "pButtons Profile Edit Form", contains four input fields: "Name", "Description", "Interval", and "Samples Num". To the right of these fields are "Add" and "Reset Form" buttons. The second section, titled "List of pButtons Profiles", contains a table with the following data:

| Name    | Description                           | Interval | Samples Num |                                                                                          |
|---------|---------------------------------------|----------|-------------|------------------------------------------------------------------------------------------|
| 12hours | 12 hour run sampling every 10 seconds | 10       | 4320        | <button>Edit</button> <button>Copy</button> <button>Remove</button> <button>Run</button> |
| 24hours | 24 hour run sampling every 10 seconds | 10       | 8640        | <button>Edit</button> <button>Copy</button> <button>Remove</button> <button>Run</button> |

Así que aquí os comparto algunos de los pasos necesarios:

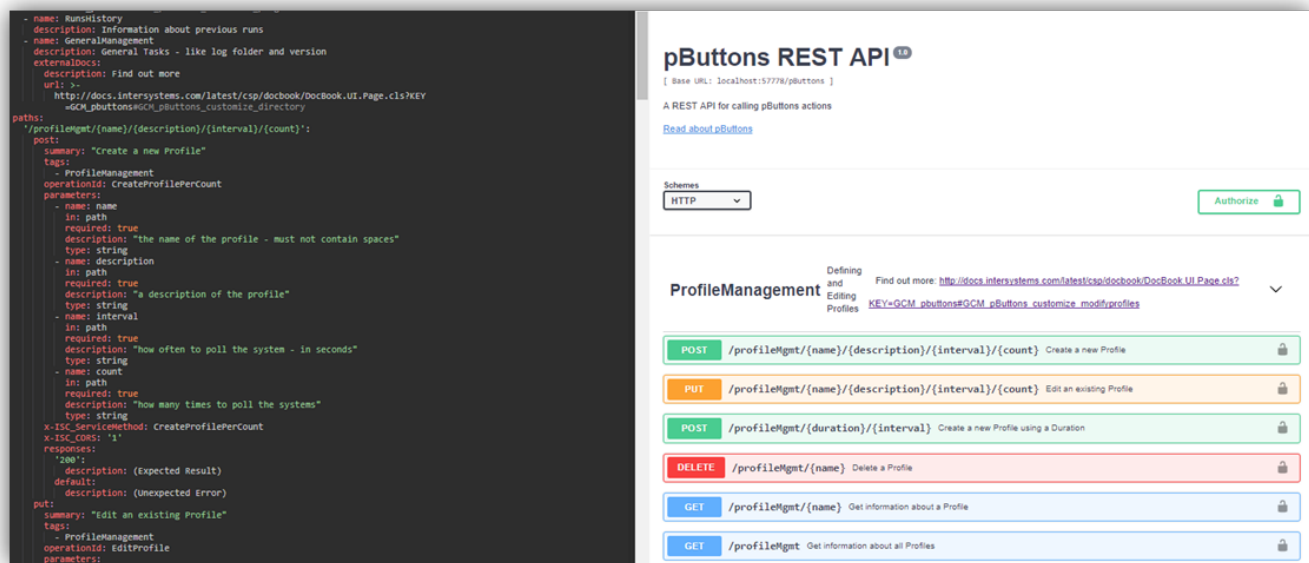
Dos clases, que son la API básica:

✓  zpButtons.API.Basic.api Class

-  CopyProfile ClassMethod
-  CreatePreviewReport ClassMethod
-  CreateProfilePerCount ClassMethod
-  CreateProfilePerDuration ClassMethod
-  DeleteProfile ClassMethod
-  EditProfile ClassMethod
-  GetLogFolder ClassMethod
-  GetPreviousRuns ClassMethod
-  GetProfile ClassMethod
-  GetProfiles ClassMethod
-  GetResultObjFromString ClassMethod
-  GetVersion ClassMethod
-  GetWaitTimeForCurrentRuns ClassMethod
-  GetWaitTimeForRunId ClassMethod
-  ResetLogFolder ClassMethod
-  RunProfile ClassMethod
-  SetLogFolder ClassMethod
-  StopRun ClassMethod

así como la clase que contiene la API REST (incluyendo algunos tests unitarios).

Un JSON con la especificación Swagger para la API REST.



## ✓ PATHS

- ✓ /generalMgmt/logFolder
  - > get
  - > post
  - > put
- > /generalMgmt/version
- > /profileMgmt
- ✓ /profileMgmt/{duration}/{interval}
  - > post
- ✓ /profileMgmt/{name}
  - > delete
  - > get
- ✓ /profileMgmt/{name}/{description}/{interval}/{count}
  - > post
  - > put
- > /profileMgmtCopy/{name}/{newName}
- > /runMgmt
- > /runMgmt/{profile}/{liteRun}
- ✓ /runMgmt/{runId}
  - > get
  - > put
- ✓ /runMgmt/{runId}/{delete}
  - > delete
- > /runMgmtPrevious

Y una interfaz gráfica sencilla en Angular (basada en <http://websystique.com/angularjs/angularjs-crud-application-using-ngresource/>)

PROFILES MANAGEMENTCURRENT RUNSPREVIOUS RUNSGENERAL

Last Updated: 11:18:51Refresh

pButtons Profile Edit Form

Name2mins

Description2 minute run sampling every 40 seconds

Interval40

Samples Num3

UpdateReset Form

List of pButtons Profiles

| Name    | Description                            | Interval | Samples Num |                   |
|---------|----------------------------------------|----------|-------------|-------------------|
| 12hours | 12 hour run sampling every 10 seconds  | 10       | 4320        | EditCopyRemoveRun |
| 1hour   | 1 hour run sampling every 13 seconds   | 13       | 276         | EditCopyRemoveRun |
| 24hours | 24 hour run sampling every 10 seconds  | 10       | 8640        | EditCopyRemoveRun |
| 2hours  | 2 hour run sampling every 10 seconds   | 10       | 720         | EditCopyRemoveRun |
| 2mins   | 2 minute run sampling every 40 seconds | 40       | 3           | EditCopyRemoveRun |

List of pButtons Current Runs

| Run ID               | Wait Time            |                    |
|----------------------|----------------------|--------------------|
| 20200504_134459_test | 6 minutes 48 seconds | Create PreviewStop |

PROFILES MANAGEMENTCURRENT RUNSPREVIOUS RUNSGENERAL

Last Updated: 12:22:09Refresh

List of pButtons Previous Runs

| Run ID               | Report Time         | Report File Name                                             | Report File Exists |
|----------------------|---------------------|--------------------------------------------------------------|--------------------|
| 20200504_090525_test | 2020-05-04 09:16:55 | /usr/irissys/mgr/c7032c24b616_IRIS_20200504_090525_test.html | 1                  |

Algunas notas importantes:

- La mayor parte de los métodos de la API básica utilizan puntos de entrada que no están documentados o soportados en ^SystemPerformance o ^pButtons. Algunos de esos métodos manejan estructuras internas. Estos métodos que no están soportados ni documentados pueden dejar de funcionar según próximas versiones sin previo aviso.
- El código no intenta de ninguna manera servir como un ejemplo de «mejor práctica» a la hora de crear

aplicaciones Angular basadas en APIs REST sobre IRIS. La parte de la interfaz gráfica es un simple ejemplo como punto de partida (de hecho, hay partes sin implementar como el directorio para logs, el refresco de contenidos, etc.)

- El código inicial fue escrito hace tiempo, así que:
  - (a) Se ha modificado para que funcione sobre IRIS (se han tenido que hacer algunos cambios en los nombres).
  - (b) Inicialmente no se utilizaba el planteamiento spec-first. Ahora sí que lo utiliza.
  - (c) Es posible que no utilice todas las últimas funcionalidades disponibles a día de hoy.
  - (d) Se ha añadido también la posibilidad de ejecutarlo en un contenedor Docker.
  - (e) Se ha añadido también soporte para ser instalado como un paquete ZPM.

[#Angular](#) [#API](#) [#API REST](#) [#Mejores prácticas](#) [#Rendimiento](#) [#Caché](#) [#Ensemble](#) [#HealthShare](#) [#InterSystems](#)  
[IRIS](#) [#InterSystems IRIS for Health](#)  
[Ir a la aplicación en InterSystems Open Exchange](#)

---

URL de  
fuente: <https://es.community.intersystems.com/post/una-interfaz-gr%C3%A1fica-y-api-de-ejemplo-para-la-utilidad-systemperformance>