Análisis de Bitácoras

Reporte Técnico

Humberto Mondragón García A01711912
Gabriela Marissa Mosquera A01666191
Felipe de Jesús Damián Rodríguez A01707246

Preparación de los datos:

El proceso comenzó con unificar múltiples archivos de bitácora en formato Parquet, almacenados en una carpeta denominada data. Cada uno de estos archivos correspondía a un día específico de registros de la aplicación. Para lograr esta consolidación, se desarrolló un script en Python que, mediante el uso de la librería os para interactuar con el sistema de archivos y pandas para la manipulación de datos, leía cada archivo Parquet y lo cargaba en un DataFrame. Posteriormente, todos los DataFrames individuales fueron concatenados en uno solo, resultando en un archivo único llamado bitacora_completa.parquet. Esta acción inicial nos permitió tener una visión completa de los datos en un solo lugar.

Una vez consolidados los datos, se procedió a una exploración inicial del DataFrame resultante. Durante esta fase, se identificó una columna de particular interés denominada Properties. Esta columna contenía información valiosa, anidada como una cadena de texto. Para poder utilizar estos datos de manera efectiva en análisis posteriores, era indispensable extraer cada uno de los campos y convertirlos en columnas independientes dentro de nuestro DataFrame.

Para llevar a cabo esta tarea, se implementó una función en Python que utilizaba expresiones regulares (re) para analizar la cadena de texto de la columna Properties y extraer los pares clave-valor. El resultado de esta extracción fue un diccionario que luego se expandió, creando nuevas columnas con el prefijo Prop_ para identificarlas claramente. Este paso fue crucial para desanidar y estructurar la información, haciendo que campos como SessionId, UserId, MethodName, entre otros, estuvieran directamente accesibles para su análisis.
Con los datos ya estructurados, el siguiente paso fue realizar una limpieza para optimizar el conjunto de datos. Se observó que existían numerosas columnas que no aportaban información relevante para el análisis, ya que contenían un único valor en todos los registros (por ejemplo, TenantId, SourceSystem, ClientType, etc.) o se encontraban completamente vacías (como Measurements, SyntheticSource, etc.).

Estas columnas "no informativas" fueron identificadas mediante programación y eliminadas del DataFrame. Esta optimización redujo significativamente el tamaño del conjunto de datos, lo que a su vez mejoró el rendimiento y la eficiencia de los análisis, permitiéndonos enfocar en las variables que realmente importaban.

Durante la fase de análisis, se hizo evidente una omisión importante: los registros individuales de la bitácora no contenían un campo de fecha explícito que indicara el día en que ocurrió el evento. Si bien existía el campo TimeGenerated con la marca de tiempo exacta, esta se trataba de la fecha de la creación de las bitácoras. La ausencia de una columna de fecha dificultaba la agregación y el análisis por día.

La solución a este problema provino de los nombres de los archivos Parquet originales, los cuales seguían una nomenclatura que incluía la fecha de los registros que contenían (AnalyticsRequestsXTracesLogs_2025-02-20.parquet). Se modificó el script de carga inicial para que, al procesar cada archivo, extrajera la fecha del nombre del mismo y la añadiera como una nueva columna llamada fecha a cada uno de los registros correspondientes a ese archivo. Para asegurar la correcta asignación, los archivos fueron ordenados cronológicamente antes del procesamiento.

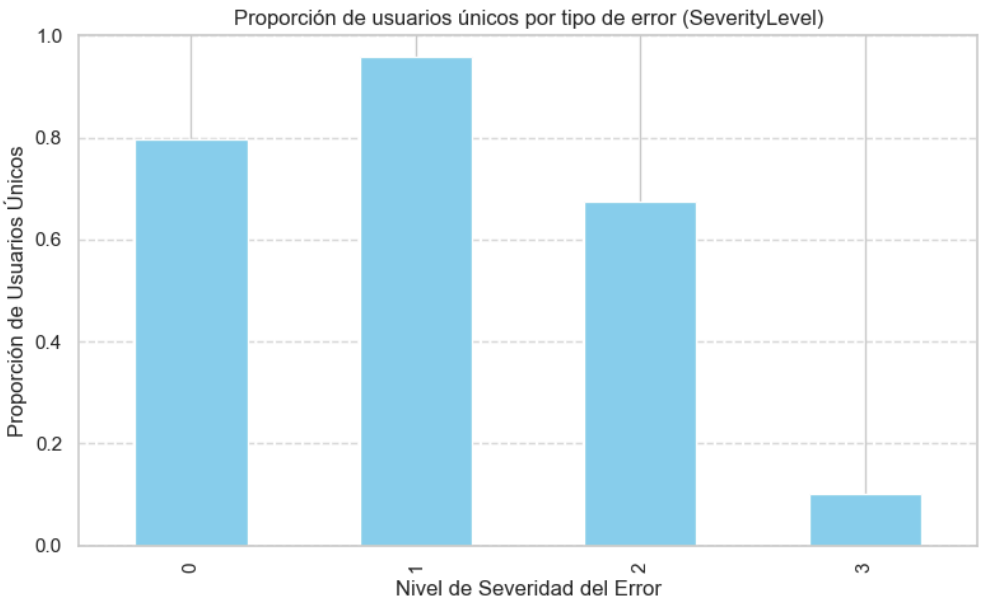
Esto nos permitió realizar análisis temporales, agrupar eventos por día y obtener una perspectiva mucho más clara del comportamiento de la aplicación a lo largo del tiempo. Finalmente, el DataFrame, ya limpio y enriquecido, se guardó en un nuevo archivo Parquet, bitacora_completa3.parquet, listo para su análisis final.

Query 1: Concentración de errores críticos (Severity 3) en un grupo reducido de usuarios

Para esta consulta, nosotros utilizamos las columnas SeverityLevel, UserId y Message. Primero filtramos el DataFrame para conservar únicamente los registros con SeverityLevel == 3, ya que nos interesaba analizar los errores más críticos del sistema. Después, agrupamos por la columna UserId para contar cuántos errores críticos generó cada usuario. Esto nos permitió identificar si los errores graves estaban distribuidos entre muchos usuarios o concentrados en unos pocos. Además, a partir de la columna Message, extraímos el tipo de error (ErrorTipo) para observar cuáles se repetían más y si estaban relacionados con usuarios específicos. De esta forma, pudimos detectar casos donde un solo usuario generaba miles de errores del mismo tipo, lo que indicaría un problema individual y no general del sistema.

Durante el análisis exploratorio de la bitácora, identificamos cuatro niveles de severidad (0 a 3). Se analizó qué proporción de usuarios únicos incurrió en cada uno de los tipos de error, donde encontramos los siguientes resultados:

- El 95% de los usuarios únicos generan errores de severidad 1 (leves)
- El 80% en errores de severidad 0 (informativos)
- Cerca del 70% presentó errores de severidad 2 (moderados)
- Y el 10% de los usuarios provocó errores de severidad 3 (críticos)



Gráfica 1: Proporción de usuarios únicos por tipo de error

Este comportamiento nos revela que, mientras los errores leves y moderados están ampliamente distribuidos entre la mayoría de los usuarios, los errores críticos se encuentran fuertemente concentrados en un grupo muy reducido.

```
ErrorTipo
Control Type Mismatch. Tag 50000 is TKToolBar not a TCAToolBarDesigner    16349
Control Not Found!                                                         2755
Invalid Tag: 0. Command: DisableFields                                     2187
Control Type Mismatch. Tag 1111 is TCAButton not a TKButton                1056
GetProductUsersByDBId                                                        84
GetProductProfilesByDBId                                                     84
Name: count, dtype: int64
```

Tabla 1: Errores más comunes de tipo 3

Al profundizar en los errores de severidad 3, el patrón se confirma:

- “Control Type Mismatch. Tag 50000 is TKToolBar not a TCAToolBarDesigner” fue generado por un solo usuario
- “Control Not Found!” fue generado por un usuario

- "Invalid Tag: 0. Command: DisableFields" fue generado por un usuario
- "GetProductProfilesByDBId" fue generado por un usuario

En particular, uno de estos usuario generó el solo más de 16,000 errores críticos del mismo tipo. Además, otros errores tipo 3 muestran el mismo patrón, con todas las ocurrencias vinculadas al mismo usuario.

	ErrorTipo	Total ocurrencias	Usuarios únicos
0	Control Type Mismatch. Tag 50000 is TKToolBar ...	16349	1
1	Control Not Found!	2755	1
2	Invalid Tag: 0. Command: DisableFields	2187	1
3	Control Type Mismatch. Tag 1111 is TCAButton n...	1056	1
4	GetProductUsersByDBId	84	1
5	GetProductProfilesByDBId	84	1

Tabla 2: Errores más comunes tipo 3 provocados por un solo usuario único diferente

Al revisar los errores tipo 3, encontramos que un solo usuario generó más de **16,000 errores críticos del mismo tipo**, lo cual es un dato bastante llamativo. En la tabla se observa que no solo se repite un mismo error, sino que hay otros errores críticos como "Control Not Found!" o "Invalid Tag" que también están **asociados unicamente a ese usuario**. Esto indica que el problema no parece venir del sistema en general, sino de algo específico con ese usuario: podría tratarse de una mala configuración, un mal uso del sistema o un error técnico muy puntual. Por eso, sería recomendable revisar el entorno de ese usuario en particular ya que su caso está teniendo un impacto muy alto en los registros de fallos.

	UserId	ErrorTipo	Ocurrencias
4	q3jl0z2ceDGReWSeifyTrj	Control Type Mismatch. Tag 50000 is TKToolBar ...	16349
3	nm4lPB3QyfwUdNMqpBms22	Control Not Found!	2755
0	7HudNhG1J8x5OP0Ri4eRwd	Invalid Tag: 0. Command: DisableFields	2187
5	tnr5w5YKWmBCjUJIXYx1BK	Control Type Mismatch. Tag 1111 is TCAButton n...	1056
1	jNrNGCGQ8Uj6jkq9wRsCfn	GetProductProfilesByDBId	84
2	jNrNGCGQ8Uj6jkq9wRsCfn	GetProductUsersByDBId	84

Tabla 3: Usuarios que provocan los errores más comunes tipo 3

Esto nos refuerza la idea de que las fallas más graves no se deben a un mal funcionamiento general del sistema sino a problemas muy localizados.

Se identificaron 22,525 errores tipo 3 en el sistema. Sorprendentemente, estos errores fueron generados únicamente por 5 usuarios distintos, lo que representa aproximadamente.

El análisis revela que los errores del sistema no se distribuyen de forma uniforme entre los usuarios. Mientras que los errores informativos (nivel 0) y leves (nivel 1) son comunes entre la mayoría de los usuarios, los errores críticos (nivel 3) están altamente concentrados en un pequeño subconjunto. Solo 5 usuarios son responsables de los 22,515 errores de nivel 3, y algunos errores específicos fueron causados exclusivamente por un único usuario. Este patrón sugiere que los errores críticos no reflejan fallos sistemáticos generalizados, sino problemas muy puntuales que podrían estar ligados a condiciones específicas de configuración o uso del sistema por parte de ciertos usuarios. Este hallazgo es valioso para priorizar acciones correctivas enfocadas y no generalizadas.

Query 2: Análisis de errores por país y su distribución proporcional

Para esta consulta, utilizamos principalmente las columnas ClientCountryOrRegion, Message y SeverityLevel. Primero, a partir de la columna Message, extraímos el tipo de error más representativo utilizando una expresión regular que nos permitió aislar la porción inicial del mensaje como ErrorTipo. Luego, agrupamos los datos por país (ClientCountryOrRegion) y por tipo de error (ErrorTipo), y contamos cuántas veces ocurría cada combinación. A partir de esto, seleccionamos los **cinco errores más frecuentes en cada país**, lo que nos permitió

detectar patrones técnicos diferenciados por región. Posteriormente, para complementar el análisis, calculamos el **total de bitácoras por país** y, con ello, estimamos la **proporción de errores graves (SeverityLevel ≥ 2)** con respecto al volumen total de registros. Esto nos permitió no sólo ver cuáles errores eran más comunes por país, sino también evaluar **qué tan crítica era la situación en cada región** al comparar su proporción de fallos severos frente al total de bitácoras registradas.

Como parte del análisis de los errores contenidos en las bitácoras, se realizó una consulta para identificar los **errores más comunes por país**, así como su **distribución relativa** en función del volumen total de registros (bitácoras) reportados en cada región.

En primer lugar, se extrajeron los cinco errores con mayor número de ocurrencias para cada país, obteniendo el siguiente resultado:

	ClientCountryOrRegion	ErrorTipo
309	Mexico	WebMethod=ProcessBRRequest
304	Mexico	WebMethod=GetKendoCustomDataSourceInstance
223	Mexico	ProcessBRRequest.CheckForReturn
132	Mexico	FastRequest.CheckForReturn
174	Mexico	Params ? Id: ? Values:
320	United Kingdom	WebMethod=ProcessBRRequest
316	United Kingdom	ParseTagValuePairToClientTransferPackage
317	United Kingdom	ProcessBRRequest.CheckForReturn
318	United Kingdom	Requesting
319	United Kingdom	Unpack
338	United States	FastRequest.CheckForReturn
322	United States	Control Not Found!
321	United States	- UpdateSession
352	United States	UpdateSession
353	United States	WebMethod=GetKendoGridInstance

Tabla 4: Top 5 errores más comunes por país

En el análisis de los errores más frecuentes por país, se observan patrones diferenciados por región. En **México**, predominan fallas en los procesos del backend como ProcessBRRequest y CheckForReturn, lo cual sugiere errores vinculados a la lógica de negocio, como validaciones fallidas, reglas mal definidas o problemas en el flujo interno de las operaciones. En el **Reino Unido**, los errores más comunes están relacionados con la **estructura de datos y la interoperabilidad entre sistemas**, destacando fallas como Unpack y

ParseTagValuePairToClientTransferPackage, típicas de problemas en el intercambio o transformación de información entre componentes. Por otro lado, en **Estados Unidos**, los errores están mayormente asociados al **uso del sistema por parte del usuario**, presentando incidencias en la interfaz como UpdateSession y Control Not Found!, lo que apunta a errores en sesiones activas, botones faltantes o componentes de UI mal cargados. Esta segmentación por país permite enfocar las acciones correctivas según el origen técnico de los fallos: lógica de negocio, integración de sistemas o experiencia del usuario.

	Ocurrencias
309	671811
304	568233
223	329637
132	263716
174	235631
320	288
316	144
...	
322	16452
321	9468
352	9439
353	8535

Tabla 5: Ocurrencias de los errores más comunes de países

En este primer hallazgo indica que el país con mayor volumen de errores absolutos es México, lo cual es esperable dado que también es el país con mayor número de bitácoras registradas (~ 5.6 millones), seguido por Estados Unidos y Reino Unido.

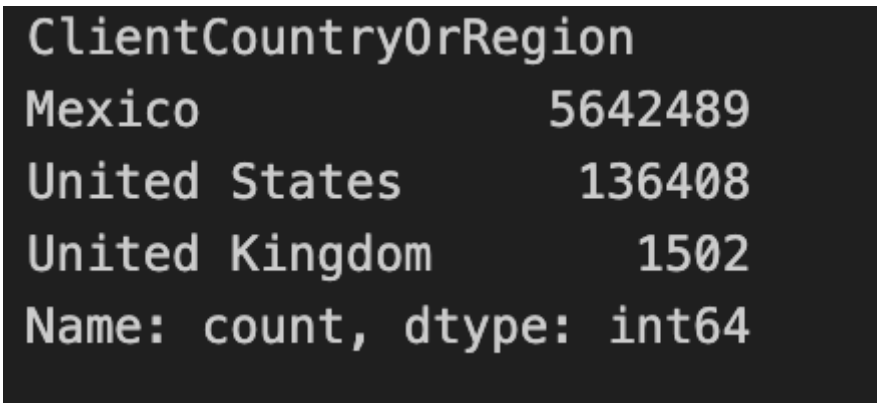


Tabla 6: Total de Bitácoras por país

No obstante, para obtener una interpretación más precisa y **evitar sesgos por el volumen total de bitácoras**, se calculó la **proporción de errores graves** (SeverityLevel ≥ 2) respecto al total de bitácoras por país. El resultado se presenta a continuación:

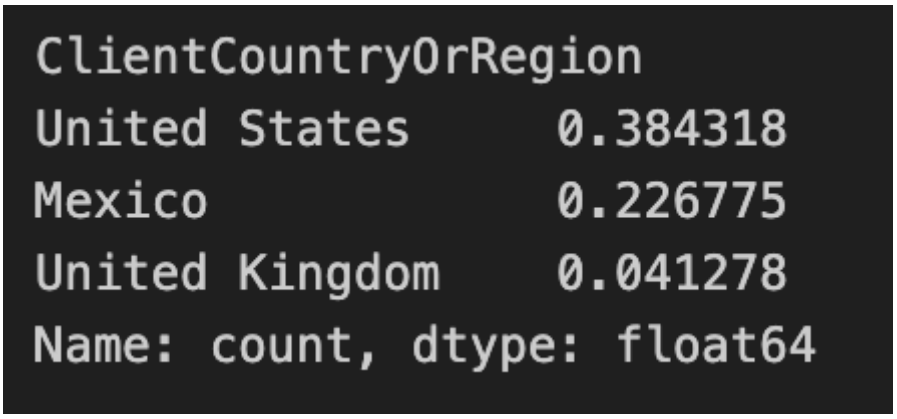


Tabla 7: Proporción de errores graves

Este análisis revela un patrón interesante: **aunque México reporta más errores en términos absolutos, Estados Unidos tiene la proporción más alta de errores graves en relación con su volumen total de bitácoras.** Esto sugiere que, en términos relativos, los usuarios en Estados Unidos enfrentan mayores dificultades o fallas críticas dentro del sistema.

La consulta permite comprender no sólo **cuáles errores son más comunes en cada región**, sino también **qué tan propensas son esas regiones a experimentar fallos graves**, considerando su tamaño. Esta información puede orientar decisiones sobre soporte técnico, prioridades de mantenimiento o auditorías focalizadas en ciertas regiones geográficas.

Query 3: Usuarios con Errores Diversos en un Mismo Rol

Para esta consulta utilizamos las columnas UserId, AppRoleName, AppRoleInstance y Message. Primero confirmamos que todos los usuarios compartían el mismo rol (AppRoleName), pero estaban distribuidos en diferentes instancias técnicas (AppRoleInstance). Luego, a partir de la columna Message, extraímos el tipo de error (ErrorTipo) y agrupamos por usuario (UserId) para contar cuántos tipos distintos de errores había generado cada uno. Esto nos permitió identificar usuarios con una alta diversidad de fallos, lo cual sugiere que, además del entorno técnico, también existen factores individuales que afectan el comportamiento del sistema.

Esta consulta identifica los usuarios que han generado la mayor cantidad de tipos distintos de errores (ErrorTipo) en el sistema. No mide la cantidad total de errores sino cuántas clases diferentes de errores ha producido cada usuario.

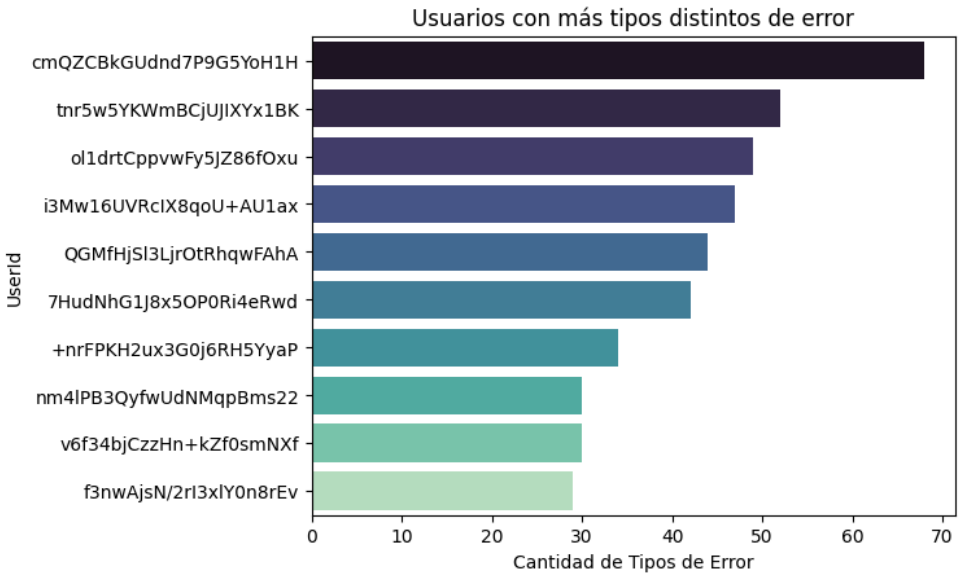
- El usuario **cmQZCBkGUdnd7P9G5YoH1H** ha generado **68 tipos distintos** de errores, lo que lo convierte en el más diverso en fallas.

- Otros usuarios como **tnr5w5YKWmBCjUJIXYx1BK** y **ol1drtCppvwFy5JZ86fOxu** han presentado entre **49 y 52 tipos de errores** diferentes.

- **Usuarios como 7HudNhG1J8x5OP0Ri4eRwd y nm4lPB3QyfwUdNMqpBms22**, que ya habían aparecido como responsables de errores críticos tipo 3, también aparecen en este ranking de errores diversos.

	UserId	TiposDeErrorDistintos
0	cmQZCBkGUdnd7P9G5YoH1H	68
1	tnr5w5YKWmBCjUJIXYx1BK	52
2	ol1drtCppvwFy5JZ86fOxu	49
3	i3Mw16UVRcIX8qoU+AU1ax	47
4	QGMfHjSl3LjrOtRhqwFAhA	44
5	7HudNhG1J8x5OP0Ri4eRwd	42
6	+nrFPKH2ux3G0j6RH5YyaP	34
7	nm4lPB3QyfwUdNMqpBms22	30
8	v6f34bjCzzHn+kZf0smNXf	30
9	f3nwAjsN/2rl3xlY0n8rEv	29

Tabla 8: Usuarios con más tipos de errores distintos



Gráfica 2: Usuarios con más tipos de error distintos

Este resultado sugiere que hay usuarios que no solo generan errores críticos, sino también una gran variedad de fallos en distintas funciones del sistema. Esto se puede deber a mal uso de la herramienta, mala configuración del entorno y fallas sistemáticas en flujos no probados.

Todos los usuarios con errores provienen de un único rol (innv103-wapp), por lo que la diversidad no se debe a diferencias de permisos. Lo que indica que el comportamiento se debe a diferencias en el uso individual del sistema.

AppRoleName	count	mean	max	std
innv103-wapp	49	14.755102	68	16.692726

Tabla 9: Existencia de un solo rol (innv103-wapp)

El análisis de la diversidad de errores por usuario revela que algunos individuos no solo generan errores críticos, sino que también concentran una gran variedad de fallos en diferentes componentes del sistema. En particular, se identificó que el usuario cmQZCBkGUdnd7P9G5YoH1H generó 68 tipos distintos de errores, seguido por tnr5w5YKWmBCjUJIXYx1BK con 52 y otros usuarios con rangos entre 30 y 50 errores distintos.

Este patrón indica una posible correlación entre la diversidad de errores y el mal uso, mal entrenamiento o incluso posibles fallos en entornos específicos de estos usuarios. Además, al cruzar esta información con los datos de errores críticos (tipo 3), se encontró que varios de los usuarios con alta diversidad también son responsables de errores graves, lo cual refuerza la hipótesis de fallas concentradas en un grupo reducido de usuarios.

	AppRoleInstance	Usuarios Únicos
0	wn1mdwk00055L	3
1	wn1mdwk000630	13
2	wn1mdwk0006AR	26
3	wn1mdwk0006E1	7

Tabla 10: Diferentes instancias del único rol

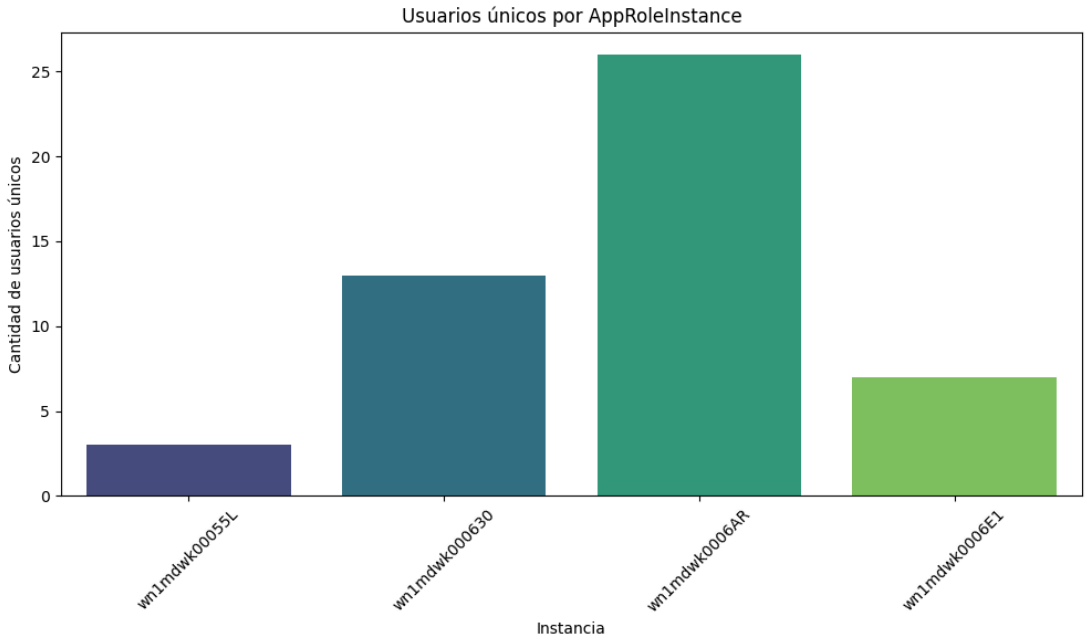
Durante el análisis detectamos que **todos los usuarios que generan errores pertenecen al mismo rol: innv103-wapp**. A pesar de eso, dentro de ese mismo rol se están utilizando **cuatro instancias distintas del sistema**, que son:

- wn1mdwk00055L
- wn1mdwk000630
- wn1mdwk0006AR
- wn1mdwk0006E1

Estas instancias pueden entenderse como diferentes entornos dentro del mismo sistema, donde se ejecutan las operaciones según el rol, pero separadas física o lógicamente.

	AppRoleInstance	Errores Críticos	% del total
0	wn1mdwk0006AR	22347	99.253831
1	wn1mdwk00055L	168	0.746169

Tabla 11: Errores críticos por instancia



Gráfica 3: Usuarios únicos por instancia

La instancia wn1mdwk0006AR es la que más usuarios concentra, lo cual ya nos empieza a dar una idea del porqué podría haber más errores ahí.

Cuando analizamos solo los errores críticos (nivel de severidad 3), el resultado fue muy claro: **la mayoría de los errores críticos están ocurriendo en la instancia wn1mdwk0006AR**, con más del 99 % del total.Esto sugiere que esta instancia en particular podría tener problemas de configuración, más usuarios haciendo pruebas o simplemente estar más expuesta por el uso diario.

Aunque todos los usuarios tienen el mismo rol (innv103-wapp), hay diferencias importantes en cómo usan el sistema y en qué instancia lo hacen. La mayoría de los errores críticos están concentrados en la instancia wn1mdwk0006AR, y además ahí también están los usuarios que generan más tipos de errores.

Esto **descarta que los errores se deban al rol** o a los permisos, y apunta más bien a **problemas individuales**, como mal uso del sistema, falta de entrenamiento o configuraciones incorrectas en la instancia.

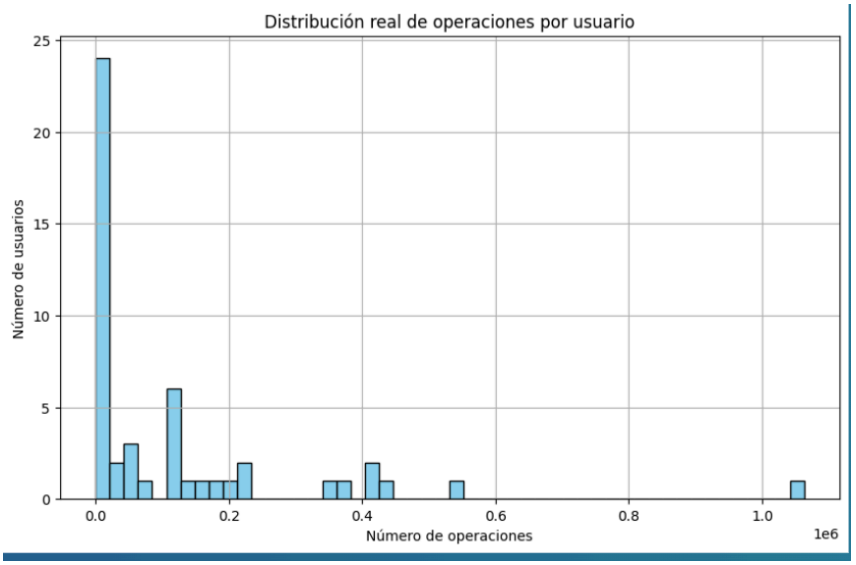
Insight 4

Frecuencias por usuario, región, tiempo

Se seleccionó la columna User_Id que contiene los identificadores de usuario.
Cuenta las apariciones de cada Usuario único. El Top 5 de usuarios se puede obtener simplemente tomando los primeros 5 elementos de esta serie.

Este análisis es útil para identificar los usuarios con mayor actividad dentro del sistema, lo que puede ser relevante tanto para auditoría como para entender el comportamiento general del uso del sistema.

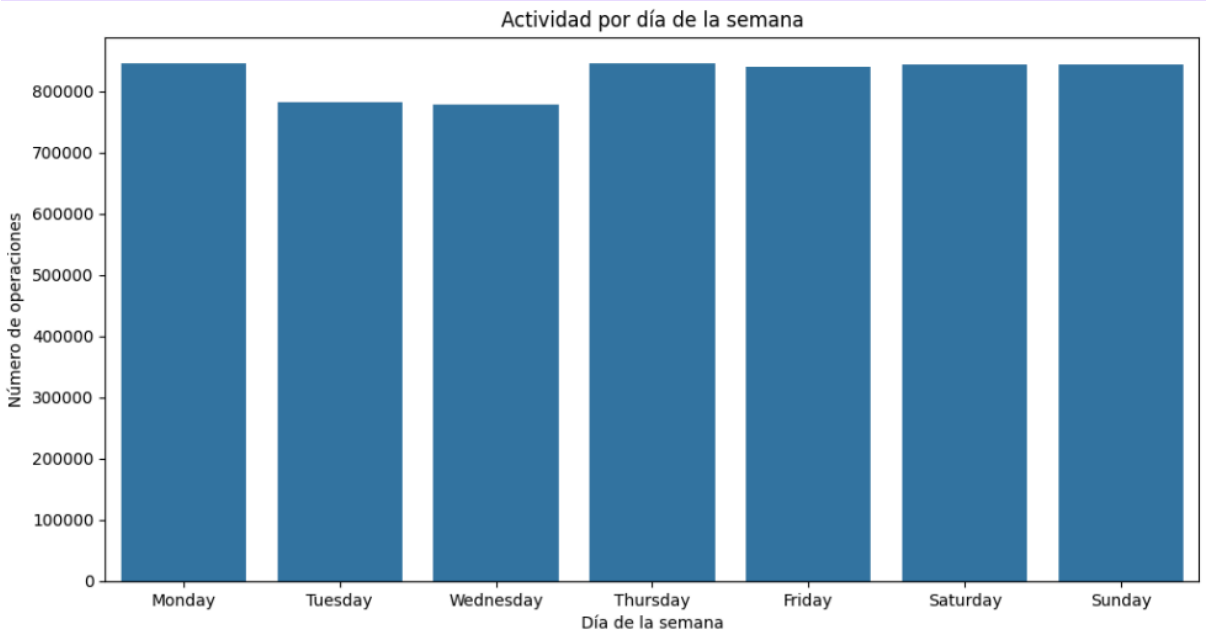
Podemos cruzar esta información con otras columnas como Region o TimeGenerated para hacer un análisis más completo.



Gráfica 4: Distribución de operaciones por usuario (en millones)

Se realizó un conteo de las apariciones de cada identificador único en la columna User_Id, correspondiente a los registros de actividad por usuario. Al ordenar los resultados en orden descendente, se identificó a los cinco usuarios con mayor volumen de operaciones registradas en las bitácoras del sistema.

Se observó que el usuario con más actividad acumula más de un millón de registros, lo cual representa una frecuencia de uso inusualmente alta. Esta cifra supera ampliamente el comportamiento esperado para un usuario estándar, haremos un análisis posterior más específico para determinar si esta actividad corresponde a automatizaciones autorizadas, scripts de sistema, pruebas de carga, o bien comportamientos anómalos.

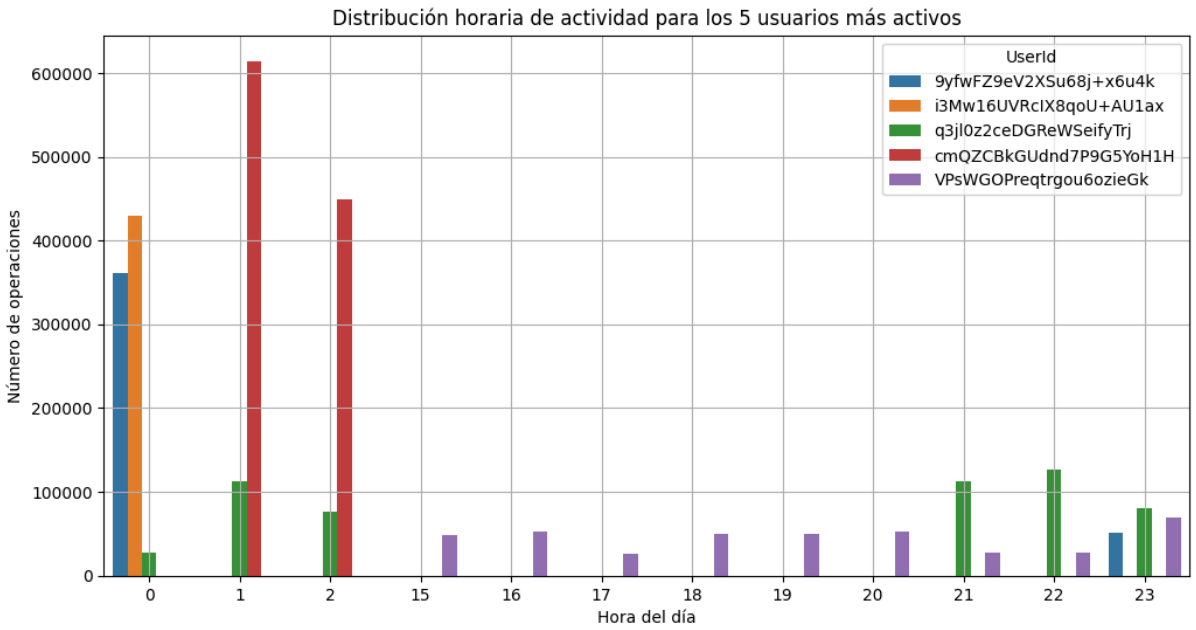


Gráfica 5: Distribución de operaciones por día de la semana

Se analizó la frecuencia de operaciones distribuidas según el día de la semana, utilizando los registros de fecha y hora asociados a cada evento. A diferencia de una expectativa centrada en días hábiles, el análisis reveló que el comportamiento es homogéneo durante los siete días de la semana, incluyendo fines de semana.

Este patrón sugiere que el sistema opera de forma continua sin diferencias notables entre días laborables y no laborables, lo cual es característico de procesos automatizados o plataformas que permanecen activas para servicios en línea.

Desde una perspectiva operativa, no se detectan picos anómalos ni ausencias abruptas de actividad, por lo que se concluye que el sistema presenta un comportamiento estable y predecible a lo largo del calendario semanal. Este tipo de estabilidad facilita la futura detección de desviaciones o incidentes de seguridad.



Gráfica 6: Distribución horaria para los 5 usuarios más activos

En este gráfico podemos observar que los usuarios con mayor frecuencia en las bitácoras realizan operaciones en horarios específicos y repetitivos, particularmente alrededor de la 1 y 2 de la mañana. (Para el usuario que más movimientos tiene). Este comportamiento sugiere que dichas actividades podrían estar automatizadas, ya que no es común que usuarios reales interactúen con el sistema de forma consistente en horarios nocturnos.

Este patrón puede ser indicativo de procesos programados, scripts o tareas de mantenimiento que se ejecutan sin intervención humana directa. Si bien esto no implica necesariamente una anomalía, sí es recomendable validar con el equipo si estas operaciones están documentadas y autorizadas, para descartar actividades no supervisadas o potencialmente riesgosas.

UserId	
i3Mw16UVRcIX8qoU+AU1ax	46
9yfwFZ9eV2XSu68j+x6u4k	21
cmQZCBkGUdnd7P9G5YoH1H	17
q3jl0z2ceDGReWSeifyTrj	14
VPsWGOPreqtrgou6ozieGk	12

Tabla 12: Cantidad de días que tienen actividad los usuarios sospechosos.

Podemos ver en la Tabla 12 que los usuarios sospechosos no tienen actividades aisladas o esporádicas, sino que se distribuyen a lo largo de varios días. Esto podría indicar que su comportamiento está bien establecido y no responde a eventos puntuales.

Esto puede ser por automatización de procesos.

Insight 5. Métodos usados por pocas personas

Prop_MethodName	
CloseBRSession	0
Session_End	0
CloseTabForm	1
<RecreateFavorites>b__47_0	1
FreeBRLibrary	1
GetKendoGridCatalogs	1
GetProductProfilesByDBId	1
Environment	1
KendoComboBoxData	1
GetProductUsersByDBId	1

Tabla 13: Métodos usados por muy pocos usuarios

En la Tabla 13 se presentan los métodos que han sido ejecutados por pocos usuarios.

Esto puede indicar que son funciones poco comunes, o de uso especializado. Recordemos que un tipo de ataque es hacer la mayor cantidad de logs posibles, a veces usando funciones que casi nadie haría, por eso los métodos raramente utilizados deben ser revisados para asegurarse de que su activación no corresponde a accesos indebidos, pruebas no autorizadas o intentos de explotación de funciones ocultas.

También podrían ser métodos que podrían estar en desuso o ser parte de funcionalidades que ya no se emplean activamente, por lo que podrían considerarse para revisión y posible eliminación si no tienen una función actual válida.

Prop_MethodName	Invocaciones	UsuariosUnicos
Session_End	58	0
CloseBRSession	41	0
ConfigureControls	819697	10
ChangePropierties	223750	3
LogTime	1181845	26
GetKendoCustomDataSourceInstance	568233	14
ProcessBRRequest	1210627	33
ExecuteClientProtocol	191291	9
CheckForReturn	631494	37
ToolBarDesigner	16349	1

Tabla 14: Métodos con mayor número de invocaciones pero utilizados por pocos usuarios

En esta tabla se presentan los métodos que, a pesar de haber sido invocados miles de veces, han sido utilizados por un número muy reducido de usuarios. Este comportamiento es relevante desde una perspectiva de seguridad y auditoría.

Destaca ToolBarDesigner, que fue invocado más de 16,000 veces por un solo usuario, lo que puede ser señal de una automatización, un bug o un posible abuso.

Se recomienda analizar quiénes son los usuarios asociados a estos métodos, validar si las ejecuciones están documentadas y justificadas, y establecer umbrales de uso para detectar comportamientos anómalos en el futuro.

```
Series([], Name: count, dtype: int64)

Usuario cmQZCBkGUdnd7P9G5YoH1H usó los métodos raros: []

Usuario q3jl0z2ceDGReWSeifyTrj usó los métodos raros: []

Usuario i3Mw16UVRcIX8qoU+AU1ax usó los métodos raros: []

Usuario 9yfwFZ9eV2XSu68j+x6u4k usó los métodos raros: []

Usuario VPsWGOPreqtrgou6ozieGk usó los métodos raros: []
```

Tabla 15. Comparación de los métodos raros con los usuarios sospechosos

Para esta tabla, comparamos los usuarios que habíamos identificado como sospechosos con estos métodos que también consideramos como raros porque pocos los habían usado, no tuvimos ninguna coincidencia lo que quiere decir que no fueron parte de estos métodos raros.