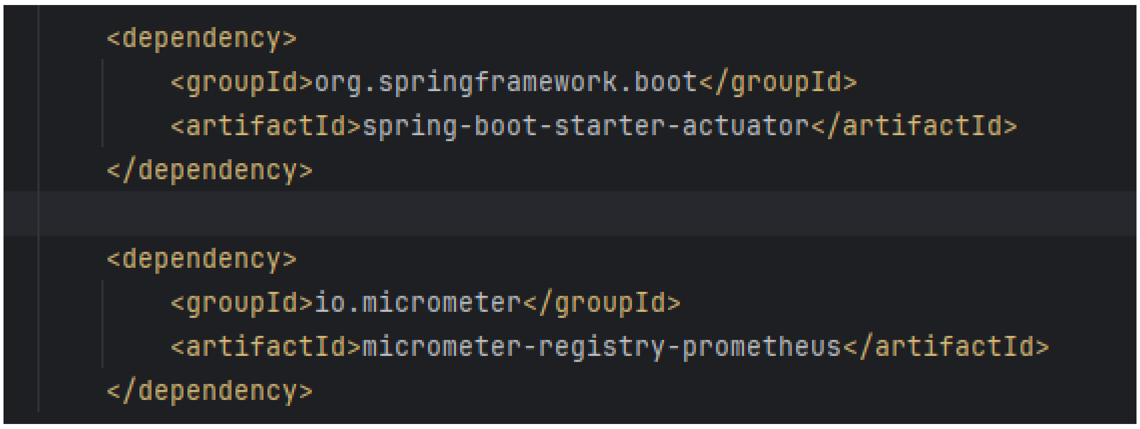
Monitorización API Productos

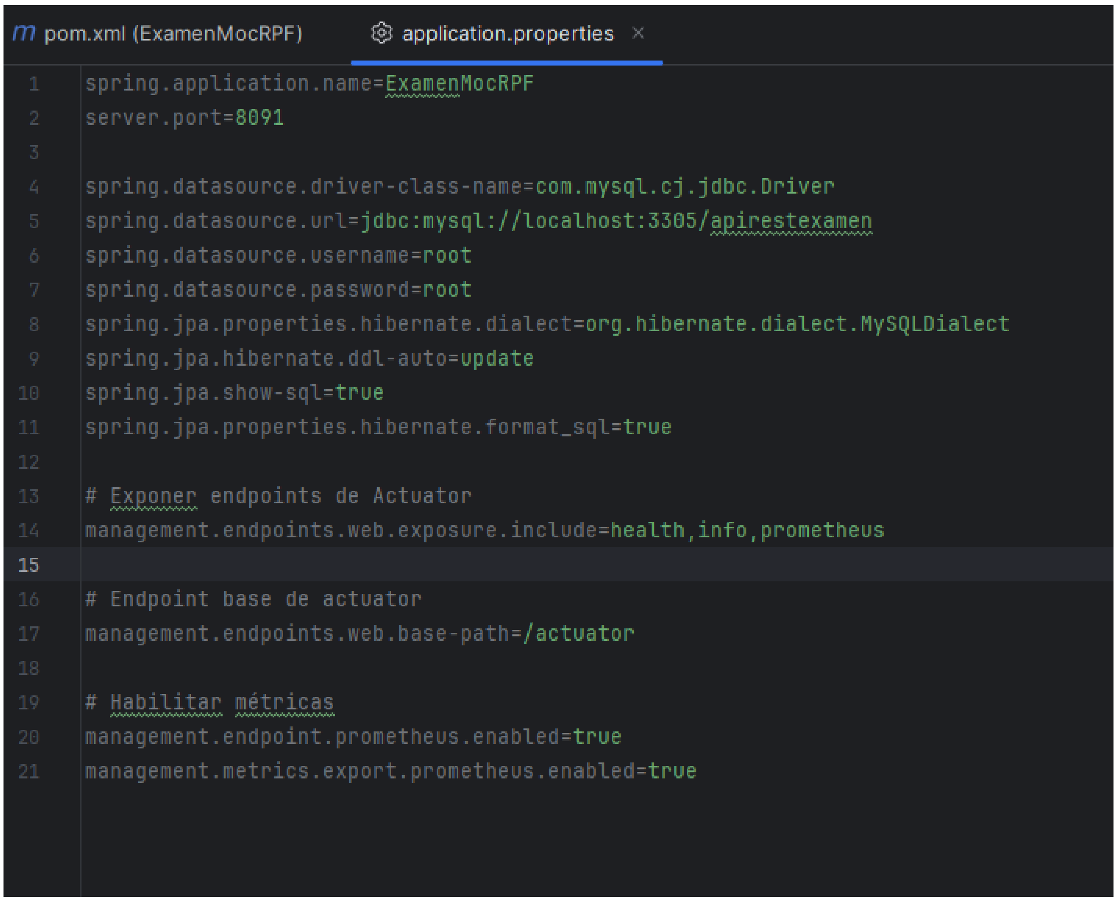
1.1 Añadir dependencias en pom.xml



```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>

<dependency>
    <groupId>io.micrometer</groupId>
    <artifactId>micrometer-registry-prometheus</artifactId>
</dependency>
```

En el archivo pom.xml se añaden las dependencias necesarias para habilitar la monitorización de la aplicación.
La dependencia spring-boot-starter-actuator permite exponer información interna de la aplicación, como el estado de salud y métricas.
La dependencia micrometer-registry-prometheus permite que Prometheus pueda recolectar dichas métricas a través de un endpoint compatible.

1.2 Configurar application.properties



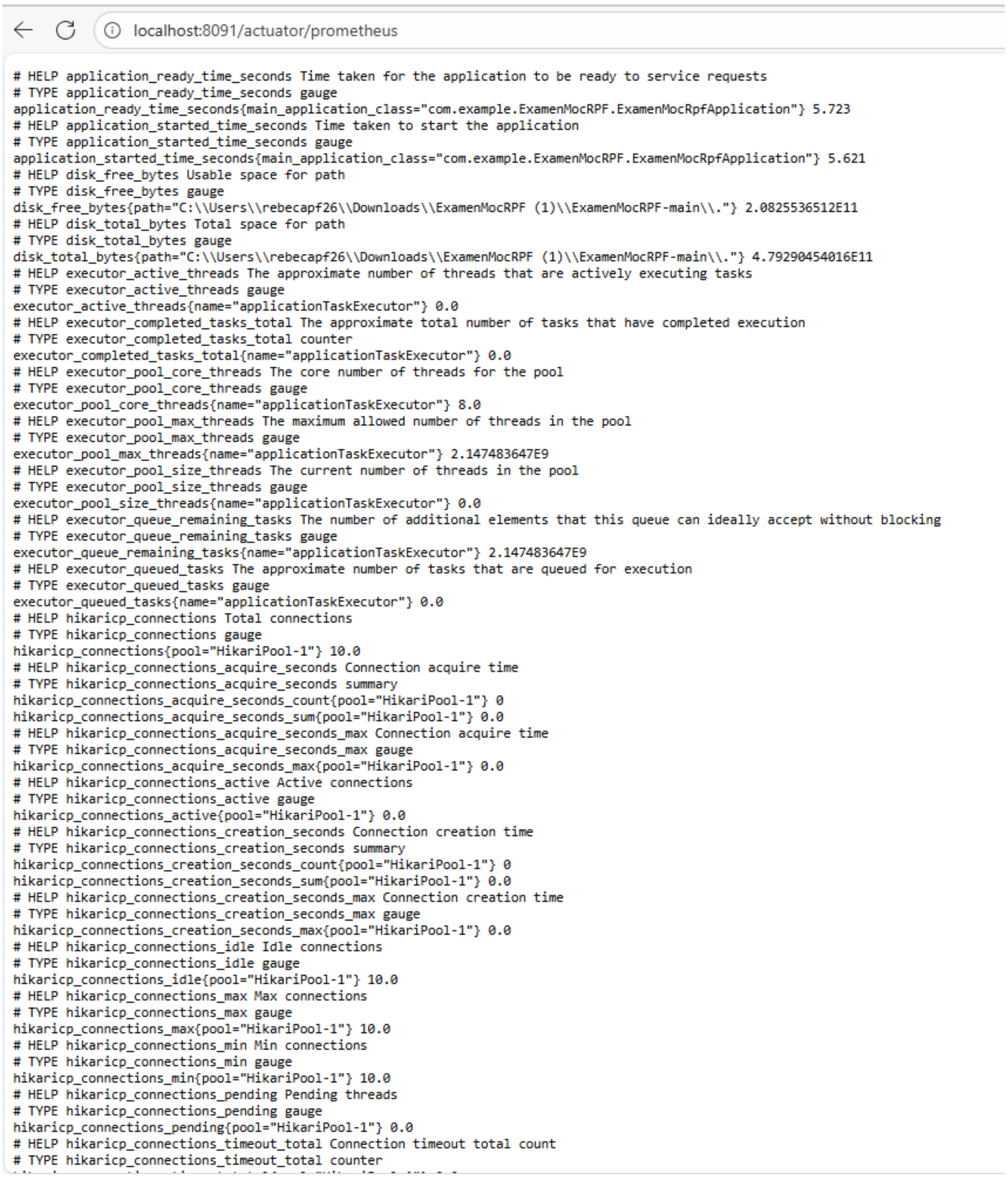
```properties
spring.application.name=ExamenMocRPF
server.port=8091

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3305/apirestexamen
spring.datasource.username=root
spring.datasource.password=root
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true

# Exponer endpoints de Actuator
management.endpoints.web.exposure.include=health,info,prometheus

# Endpoint base de actuator
management.endpoints.web.base-path=/actuator

# Habilitar métricas
management.endpoint.prometheus.enabled=true
management.metrics.export.prometheus.enabled=true
```

En el archivo application.properties se configura la exposición de los endpoints de Actuator.

Se habilita el endpoint /actuator/prometheus, que será utilizado por Prometheus para obtener las métricas de la API.

También se especifica el puerto en el que se ejecuta la aplicación y se habilita la exportación de métricas en formato Prometheus.

1.3 Comprobar que la API expone métricas



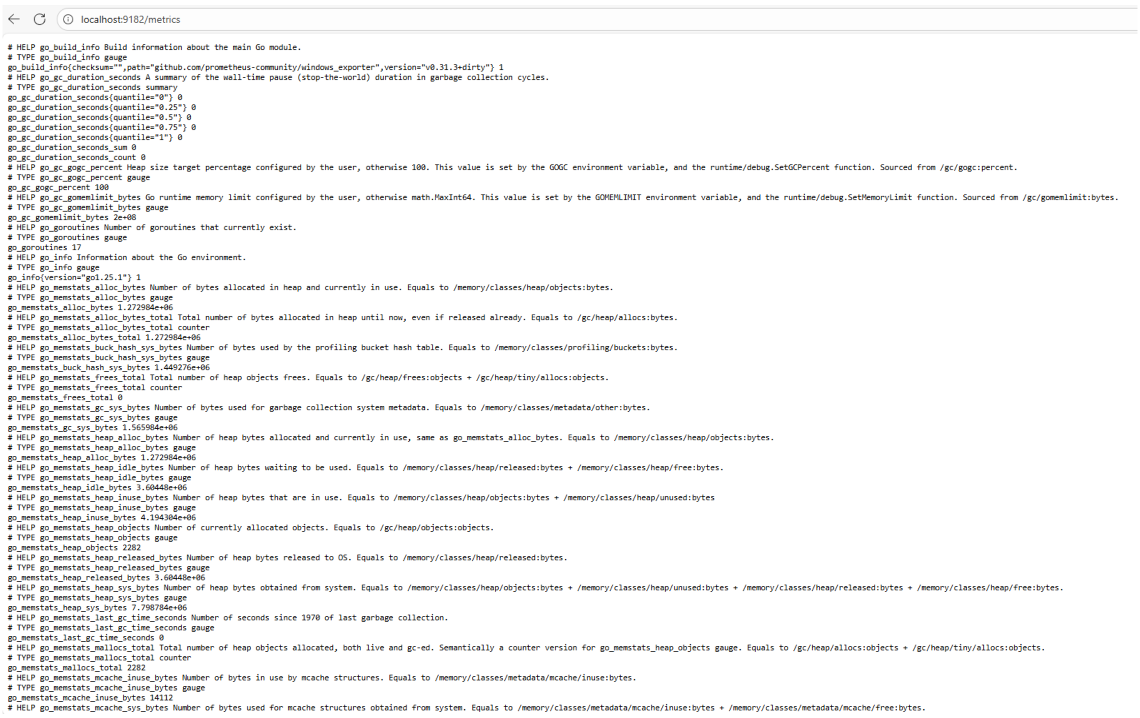
localhost:8091/actuator/prometheus

```
# HELP application_ready_time_seconds Time taken for the application to be ready to service requests
# TYPE application_ready_time_seconds gauge
application_ready_time_seconds{main_application_class="com.example.ExamenMocRPF.ExamenMocRpfApplication"} 5.723
# HELP application_started_time_seconds Time taken to start the application
# TYPE application_started_time_seconds gauge
application_started_time_seconds{main_application_class="com.example.ExamenMocRPF.ExamenMocRpfApplication"} 5.621
# HELP disk_free_bytes Usable space for path
# TYPE disk_free_bytes gauge
disk_free_bytes{path="C:\\Users\\rebecapf26\\Downloads\\ExamenMocRPF (1)\\ExamenMocRPF-main\\."} 2.0825536512E11
# HELP disk_total_bytes Total space for path
# TYPE disk_total_bytes gauge
disk_total_bytes{path="C:\\Users\\rebecapf26\\Downloads\\ExamenMocRPF (1)\\ExamenMocRPF-main\\."} 4.79290454016E11
# HELP executor_active_threads The approximate number of threads that are actively executing tasks
# TYPE executor_active_threads gauge
executor_active_threads{name="applicationTaskExecutor"} 0.0
# HELP executor_completed_tasks_total The approximate total number of tasks that have completed execution
# TYPE executor_completed_tasks_total counter
executor_completed_tasks_total{name="applicationTaskExecutor"} 0.0
# HELP executor_pool_core_threads The core number of threads for the pool
# TYPE executor_pool_core_threads gauge
executor_pool_core_threads{name="applicationTaskExecutor"} 8.0
# HELP executor_pool_max_threads The maximum allowed number of threads in the pool
# TYPE executor_pool_max_threads gauge
executor_pool_max_threads{name="applicationTaskExecutor"} 2.147483647E9
# HELP executor_pool_size_threads The current number of threads in the pool
# TYPE executor_pool_size_threads gauge
executor_pool_size_threads{name="applicationTaskExecutor"} 0.0
# HELP executor_queue_remaining_tasks The number of additional elements that this queue can ideally accept without blocking
# TYPE executor_queue_remaining_tasks gauge
executor_queue_remaining_tasks{name="applicationTaskExecutor"} 2.147483647E9
# HELP executor_queued_tasks The approximate number of tasks that are queued for execution
# TYPE executor_queued_tasks gauge
executor_queued_tasks{name="applicationTaskExecutor"} 0.0
# HELP hikaricp_connections Total connections
# TYPE hikaricp_connections gauge
hikaricp_connections{pool="HikariPool-1"} 10.0
# HELP hikaricp_connections_acquire_seconds Connection acquire time
# TYPE hikaricp_connections_acquire_seconds summary
hikaricp_connections_acquire_seconds_count{pool="HikariPool-1"} 0
hikaricp_connections_acquire_seconds_sum{pool="HikariPool-1"} 0.0
# HELP hikaricp_connections_acquire_seconds_max Connection acquire time
# TYPE hikaricp_connections_acquire_seconds_max gauge
hikaricp_connections_acquire_seconds_max{pool="HikariPool-1"} 0.0
# HELP hikaricp_connections_active Active connections
# TYPE hikaricp_connections_active gauge
hikaricp_connections_active{pool="HikariPool-1"} 0.0
# HELP hikaricp_connections_creation_seconds Connection creation time
# TYPE hikaricp_connections_creation_seconds summary
hikaricp_connections_creation_seconds_count{pool="HikariPool-1"} 0
hikaricp_connections_creation_seconds_sum{pool="HikariPool-1"} 0.0
# HELP hikaricp_connections_creation_seconds_max Connection creation time
# TYPE hikaricp_connections_creation_seconds_max gauge
hikaricp_connections_creation_seconds_max{pool="HikariPool-1"} 0.0
# HELP hikaricp_connections_idle Idle connections
# TYPE hikaricp_connections_idle gauge
hikaricp_connections_idle{pool="HikariPool-1"} 10.0
# HELP hikaricp_connections_max Max connections
# TYPE hikaricp_connections_max gauge
hikaricp_connections_max{pool="HikariPool-1"} 10.0
# HELP hikaricp_connections_min Min connections
# TYPE hikaricp_connections_min gauge
hikaricp_connections_min{pool="HikariPool-1"} 10.0
# HELP hikaricp_connections_pending Pending threads
# TYPE hikaricp_connections_pending gauge
hikaricp_connections_pending{pool="HikariPool-1"} 0.0
# HELP hikaricp_connections_timeout_total Connection timeout total count
# TYPE hikaricp_connections_timeout_total counter
```

Una vez arrancada la aplicación, se comprueba que las métricas están disponibles accediendo desde el navegador al endpoint http://localhost:8080/actuator/prometheus.

En este endpoint se muestran métricas relacionadas con la JVM, memoria, CPU y peticiones HTTP, confirmando que la API expone correctamente la información necesaria para Prometheus.

2.1 Descargar Windows Exporter



```
# HELP go_build_info Build information about the main Go module.
# TYPE go_build_info gauge
go_build_info{checksum="",path="github.com/prometheus-community/windows_exporter",version="v0.31.3+dirty"} 1
# HELP go_gc_duration_seconds A summary of the wall-time pause (stop-the-world) duration in garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 0
go_gc_duration_seconds{quantile="0.25"} 0
go_gc_duration_seconds{quantile="0.5"} 0
go_gc_duration_seconds{quantile="0.75"} 0
go_gc_duration_seconds{quantile="1"} 0
go_gc_duration_seconds_sum 0
go_gc_duration_seconds_count 0
# HELP go_gc_gogc_percent Heap size target percentage configured by the user, otherwise 100. This value is set by the GOGC environment variable, and the runtime/debug.SetGCPercent function. Sourced from /gc/gogc:percent.
# TYPE go_gc_gogc_percent gauge
go_gc_gogc_percent 100
# HELP go_gc_gomemlimit_bytes Go runtime memory limit configured by the user, otherwise math.MaxInt64. This value is set by the GOMEMLIMIT environment variable, and the runtime/debug.SetMemoryLimit function. Sourced from /gc/gomemlimit:bytes.
# TYPE go_gc_gomemlimit_bytes gauge
go_gc_gomemlimit_bytes 2e+08
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 17
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.25.1"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated in heap and currently in use. Equals to /memory/classes/heap/objects:bytes.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 1.272984e+06
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated in heap until now, even if released already. Equals to /gc/heap/allocs:bytes.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 1.272984e+06
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table. Equals to /memory/classes/profiling/buckets:bytes.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.449276e+06
# HELP go_memstats_frees_total Total number of heap objects frees. Equals to /gc/heap/frees:objects + /gc/heap/tiny/allocs:objects.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 0
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata. Equals to /memory/classes/metadata/other:bytes.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 1.565984e+06
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and currently in use, same as go_memstats_alloc_bytes. Equals to /memory/classes/heap/objects:bytes.
# TYPE go_memstats_heap_alloc_bytes gauge
go_memstats_heap_alloc_bytes 1.272984e+06
# HELP go_memstats_heap_idle_bytes Number of heap bytes waiting to be used. Equals to /memory/classes/heap/released:bytes + /memory/classes/heap/free:bytes.
# TYPE go_memstats_heap_idle_bytes gauge
go_memstats_heap_idle_bytes 3.60448e+06
# HELP go_memstats_heap_inuse_bytes Number of heap bytes that are in use. Equals to /memory/classes/heap/objects:bytes + /memory/classes/heap/unused:bytes
# TYPE go_memstats_heap_inuse_bytes gauge
go_memstats_heap_inuse_bytes 4.194304e+06
# HELP go_memstats_heap_objects Number of currently allocated objects. Equals to /gc/heap/objects:objects.
# TYPE go_memstats_heap_objects gauge
go_memstats_heap_objects 2282
# HELP go_memstats_heap_released_bytes Number of heap bytes released to OS. Equals to /memory/classes/heap/released:bytes.
# TYPE go_memstats_heap_released_bytes gauge
go_memstats_heap_released_bytes 3.60448e+06
# HELP go_memstats_heap_sys_bytes Number of heap bytes obtained from system. Equals to /memory/classes/heap/objects:bytes + /memory/classes/heap/unused:bytes + /memory/classes/heap/released:bytes + /memory/classes/heap/free:bytes.
# TYPE go_memstats_heap_sys_bytes gauge
go_memstats_heap_sys_bytes 7.798784e+06
# HELP go_memstats_last_gc_time_seconds Number of seconds since 1970 of last garbage collection.
# TYPE go_memstats_last_gc_time_seconds gauge
go_memstats_last_gc_time_seconds 0
# HELP go_memstats_mallocs_total Total number of heap objects allocated, both live and gc-ed. Semantically a counter version for go_memstats_heap_objects gauge. Equals to /gc/heap/allocs:objects + /gc/heap/tiny/allocs:objects.
# TYPE go_memstats_mallocs_total counter
go_memstats_mallocs_total 2282
# HELP go_memstats_mcache_inuse_bytes Number of bytes in use by mcache structures. Equals to /memory/classes/metadata/mcache/inuse:bytes.
# TYPE go_memstats_mcache_inuse_bytes gauge
go_memstats_mcache_inuse_bytes 14112
# HELP go_memstats_mcache_sys_bytes Number of bytes used for mcache structures obtained from system. Equals to /memory/classes/metadata/mcache/inuse:bytes + /memory/classes/metadata/mcache/free:bytes.
```

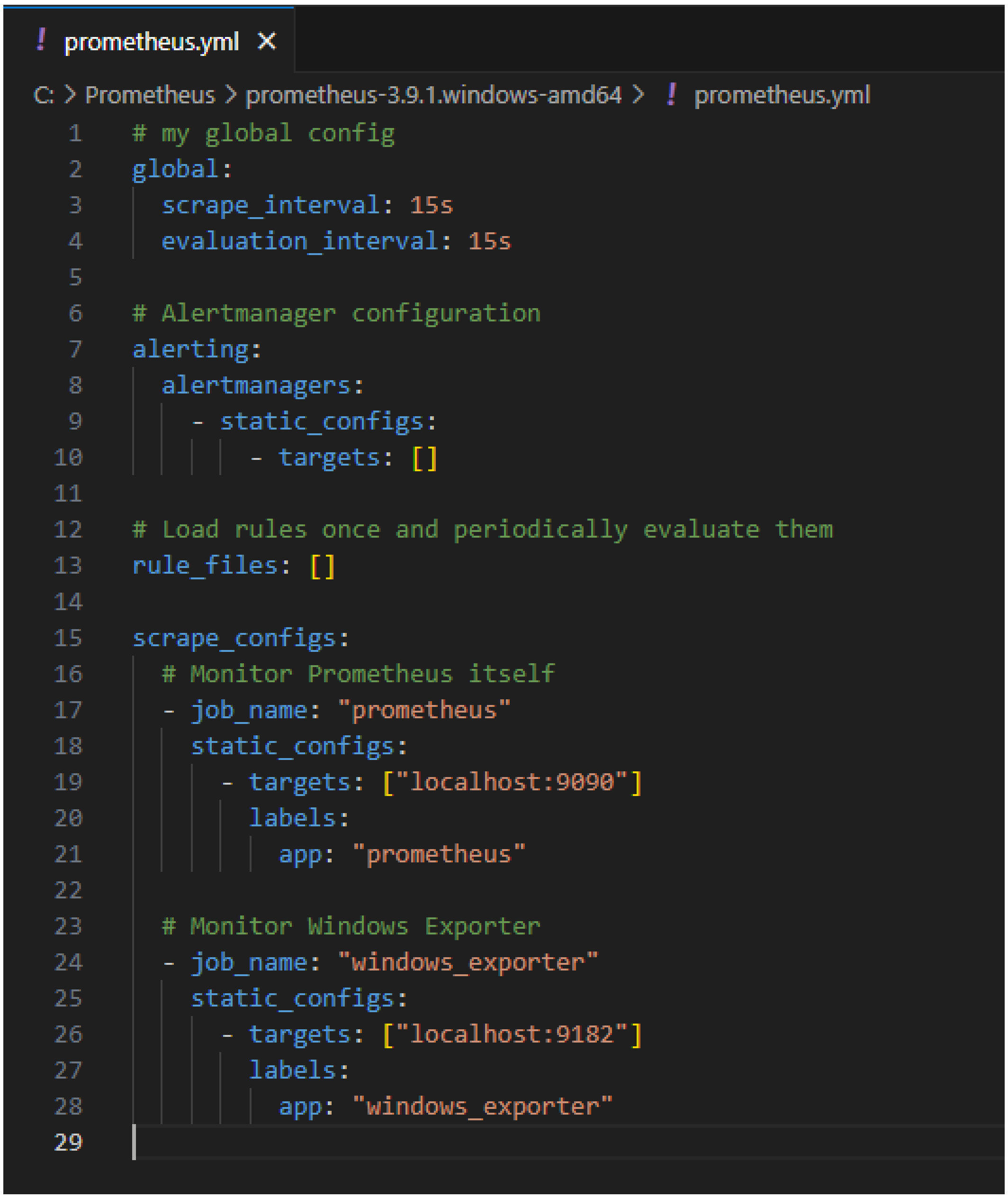
Windows Exporter se utiliza para recopilar métricas del sistema operativo Windows, como el uso de CPU, memoria y disco.

Tras su instalación, las métricas quedan disponibles en el endpoint http://localhost:9182/metrics, el cual será consultado por Prometheus.

3.1 Instalar Prometheus

Prometheus se instala como sistema de monitorización encargado de recolectar y almacenar las métricas expuestas tanto por la API de Spring Boot como por Windows Exporter.

3.2 Configurar prometheus.yml


! prometheus.yml ×
C: > Prometheus > prometheus-3.9.1.windows-amd64 > ! prometheus.yml

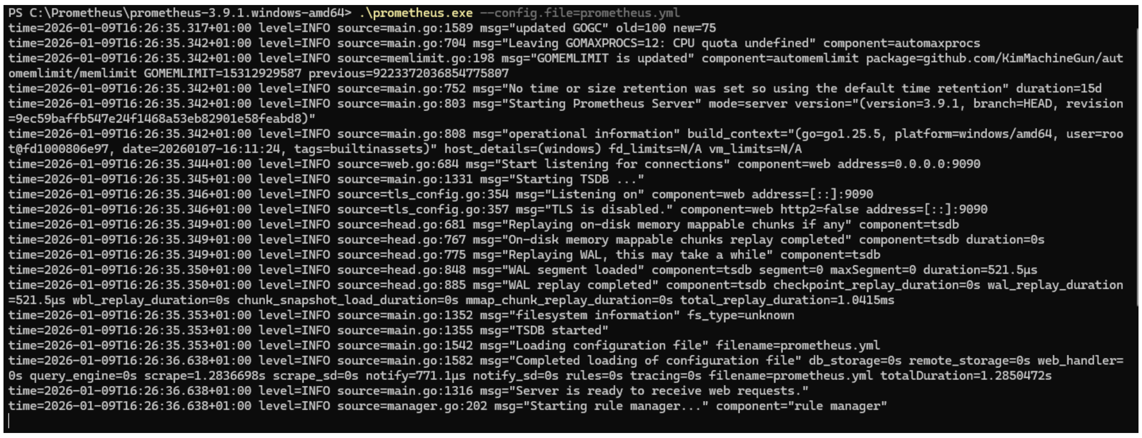
```yaml
# my global config
global:
  scrape_interval: 15s
  evaluation_interval: 15s

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets: []

# Load rules once and periodically evaluate them
rule_files: []

scrape_configs:
  # Monitor Prometheus itself
  - job_name: "prometheus"
    static_configs:
      - targets: ["localhost:9090"]
        labels:
          app: "prometheus"

  # Monitor Windows Exporter
  - job_name: "windows_exporter"
    static_configs:
      - targets: ["localhost:9182"]
        labels:
          app: "windows_exporter"

```

En el archivo prometheus.yml se definen los *jobs* que indican a Prometheus desde qué endpoints debe recoger las métricas.

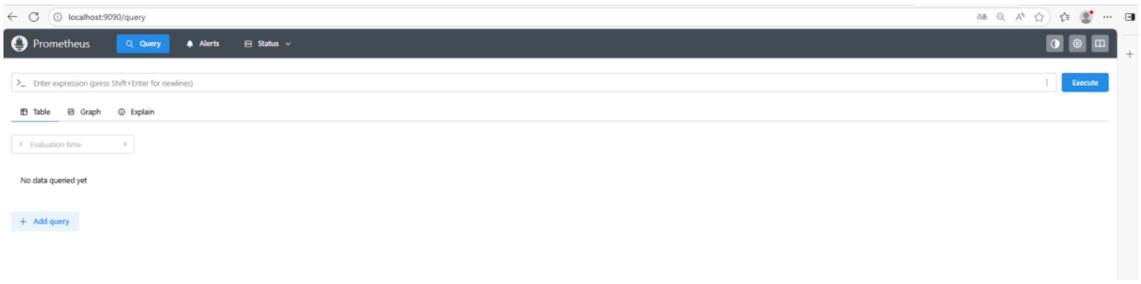
Se configura un job para la API de Spring Boot utilizando el endpoint /actuator/prometheus, y otro job para Windows Exporter en el puerto 9182.

De esta forma, Prometheus centraliza las métricas de la aplicación y del sistema.

3.3 Arrancar Prometheus



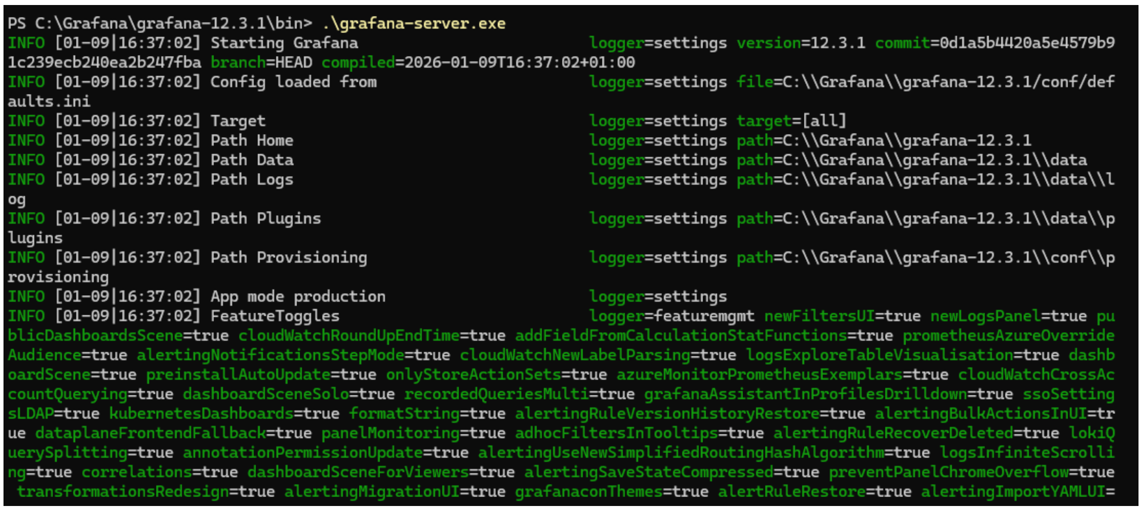
```
PS C:\Prometheus\prometheus-3.9.1.windows-amd64> .\prometheus.exe --config.file=prometheus.yml
time=2026-01-09T16:26:35.317+01:00 level=INFO source=main.go:1589 msg="updated GOGC" old=100 new=75
time=2026-01-09T16:26:35.342+01:00 level=INFO source=main.go:704 msg="Leaving GOMAXPROCS=12: CPU quota undefined" component=automaxprocs
time=2026-01-09T16:26:35.342+01:00 level=INFO source=memlimit.go:198 msg="GOMEMLIMIT is updated" component=automemlimit package=github.com/KimMachineGun/aut
omemlimit/memlimit GOMEMLIMIT=15312929587 previous=9223372036854775807
time=2026-01-09T16:26:35.342+01:00 level=INFO source=main.go:752 msg="No time or size retention was set so using the default time retention" duration=15d
time=2026-01-09T16:26:35.342+01:00 level=INFO source=main.go:803 msg="Starting Prometheus Server" mode=server version="(version=3.9.1, branch=HEAD, revision
=9ec59baffb547e24f1468a53eb82901e58feabd8)"
time=2026-01-09T16:26:35.342+01:00 level=INFO source=main.go:808 msg="operational information" build_context="(go=go1.25.5, platform=windows/amd64, user=roo
t@fd1000806e97, date=20260107-16:11:24, tags=builtinassets)" host_details=(windows) fd_limits=N/A vm_limits=N/A
time=2026-01-09T16:26:35.344+01:00 level=INFO source=web.go:684 msg="Start listening for connections" component=web address=0.0.0.0:9090
time=2026-01-09T16:26:35.345+01:00 level=INFO source=main.go:1331 msg="Starting TSDB ..."
time=2026-01-09T16:26:35.346+01:00 level=INFO source=tls_config.go:354 msg="Listening on" component=web address=[::]:9090
time=2026-01-09T16:26:35.346+01:00 level=INFO source=tls_config.go:357 msg="TLS is disabled." component=web http2=false address=[::]:9090
time=2026-01-09T16:26:35.349+01:00 level=INFO source=head.go:681 msg="Replaying on-disk memory mappable chunks if any" component=tsdb
time=2026-01-09T16:26:35.349+01:00 level=INFO source=head.go:767 msg="On-disk memory mappable chunks replay completed" component=tsdb duration=0s
time=2026-01-09T16:26:35.349+01:00 level=INFO source=head.go:775 msg="Replaying WAL, this may take a while" component=tsdb
time=2026-01-09T16:26:35.350+01:00 level=INFO source=head.go:848 msg="WAL segment loaded" component=tsdb segment=0 maxSegment=0 duration=521.5µs
time=2026-01-09T16:26:35.350+01:00 level=INFO source=head.go:885 msg="WAL replay completed" component=tsdb checkpoint_replay_duration=0s wal_replay_duration
=521.5µs wbl_replay_duration=0s chunk_snapshot_load_duration=0s mmap_chunk_replay_duration=0s total_replay_duration=1.0415ms
time=2026-01-09T16:26:35.353+01:00 level=INFO source=main.go:1352 msg="filesystem information" fs_type=unknown
time=2026-01-09T16:26:35.353+01:00 level=INFO source=main.go:1355 msg="TSDB started"
time=2026-01-09T16:26:35.353+01:00 level=INFO source=main.go:1542 msg="Loading configuration file" filename=prometheus.yml
time=2026-01-09T16:26:36.638+01:00 level=INFO source=main.go:1582 msg="Completed loading of configuration file" db_storage=0s remote_storage=0s web_handler=
0s query_engine=0s scrape=1.2836698s scrape_sd=0s notify=771.1µs notify_sd=0s rules=0s tracing=0s filename=prometheus.yml totalDuration=1.2850472s
time=2026-01-09T16:26:36.638+01:00 level=INFO source=main.go:1316 msg="Server is ready to receive web requests."
time=2026-01-09T16:26:36.638+01:00 level=INFO source=manager.go:202 msg="Starting rule manager..." component="rule manager"
```



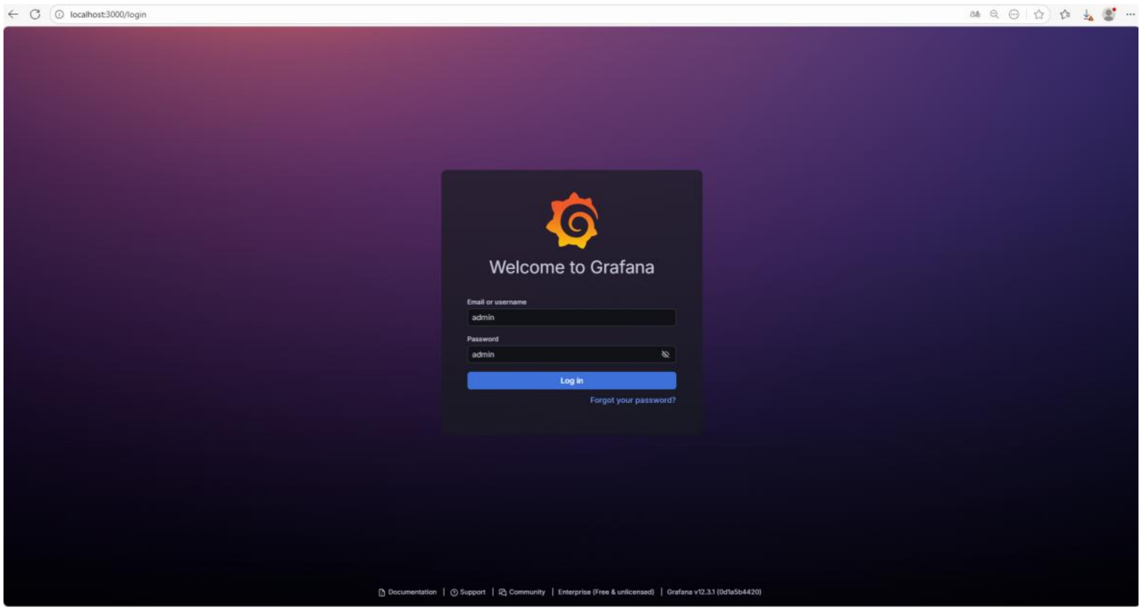
Una vez configurado Prometheus, se arranca el servicio y se accede a la interfaz web en http://localhost:9090.

Desde el apartado *Status* → *Targets* se verifica que tanto la API de Spring Boot como Windows Exporter se encuentran en estado *UP*, confirmando que Prometheus está recolectando correctamente las métricas.

4.1 Instalar Grafana



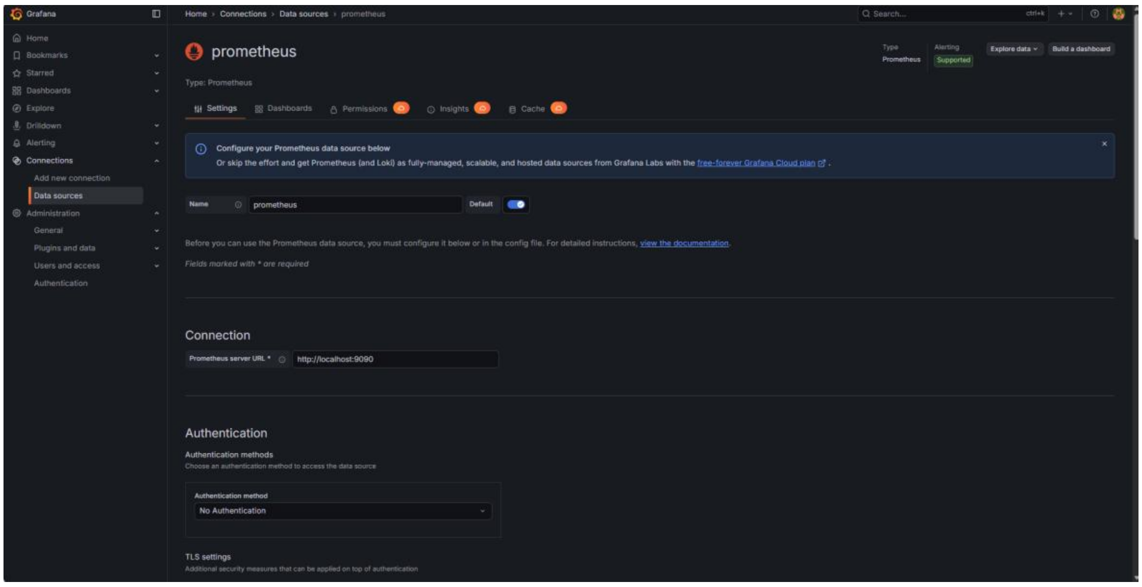
```
PS C:\Grafana\grafana-12.3.1\bin> .\grafana-server.exe
INFO [01-09|16:37:02] Starting Grafana                         logger=settings version=12.3.1 commit=0d1a5b4420a5e4579b9
1c239ecb240ea2b247fba branch=HEAD compiled=2026-01-09T16:37:02+01:00
INFO [01-09|16:37:02] Config loaded from                       logger=settings file=C:\\Grafana\\grafana-12.3.1/conf/def
aults.ini
INFO [01-09|16:37:02] Target                                   logger=settings target=[all]
INFO [01-09|16:37:02] Path Home                                logger=settings path=C:\\Grafana\\grafana-12.3.1
INFO [01-09|16:37:02] Path Data                                logger=settings path=C:\\Grafana\\grafana-12.3.1\\data
INFO [01-09|16:37:02] Path Logs                                logger=settings path=C:\\Grafana\\grafana-12.3.1\\data\\l
og
INFO [01-09|16:37:02] Path Plugins                             logger=settings path=C:\\Grafana\\grafana-12.3.1\\data\\p
lugins
INFO [01-09|16:37:02] Path Provisioning                        logger=settings path=C:\\Grafana\\grafana-12.3.1\\conf\\p
rovisioning
INFO [01-09|16:37:02] App mode production                      logger=settings
INFO [01-09|16:37:02] FeatureToggles                           logger=featuremgmt newFiltersUI=true newLogsPanel=true pu
blicDashboardsScene=true cloudWatchRoundUpEndTime=true addFieldFromCalculationStatFunctions=true prometheusAzureOverride
Audience=true alertingNotificationsStepMode=true cloudWatchNewLabelParsing=true logsExploreTableVisualisation=true dashb
oardScene=true preinstallAutoUpdate=true onlyStoreActionSets=true azureMonitorPrometheusExemplars=true cloudWatchCrossAc
countQuerying=true dashboardSceneSolo=true recordedQueriesMulti=true grafanaAssistantInProfilesDrilldown=true ssoSetting
sLDAP=true kubernetesDashboards=true formatString=true alertingRuleVersionHistoryRestore=true alertingBulkActionsInUI=tr
ue dataplaneFrontendFallback=true panelMonitoring=true adhocFiltersInTooltips=true alertingRuleRecoverDeleted=true lokiQ
uerySplitting=true annotationPermissionUpdate=true alertingUseNewSimplifiedRoutingHashAlgorithm=true logsInfiniteScrolli
ng=true correlations=true dashboardSceneForViewers=true alertingSaveStateCompressed=true preventPanelChromeOverflow=true
 transformationsRedesign=true alertingMigrationUI=true grafanaconThemes=true alertRuleRestore=true alertingImportYAMLUI=
```



Grafana se instala como herramienta de visualización de métricas.

Permite representar gráficamente la información recolectada por Prometheus mediante dashboards personalizables.

4.2 Configurar Data Source Prometheus



Configure your Prometheus data source below

Or skip the effort and get Prometheus (and Loki) as fully-managed, scalable, and hosted data sources from Grafana Labs with the free-forever Grafana Cloud plan.

Name: prometheus   Default

Before you can use the Prometheus data source, you must configure it below or in the config file. For detailed instructions, view the documentation.

*Fields marked with * are required*

Connection

Prometheus server URL *: http://localhost:9090

Authentication

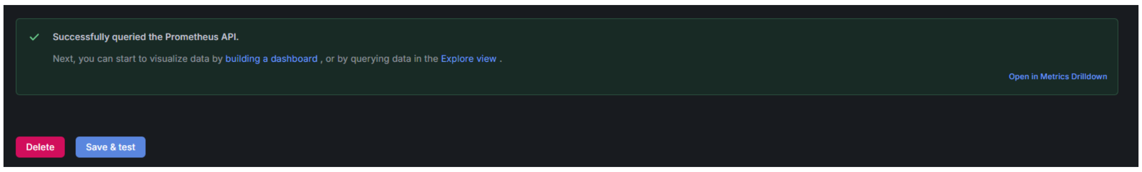
Authentication methods

Choose an authentication method to access the data source

Authentication method: No Authentication

TLS settings

Additional security measures that can be applied on top of authentication



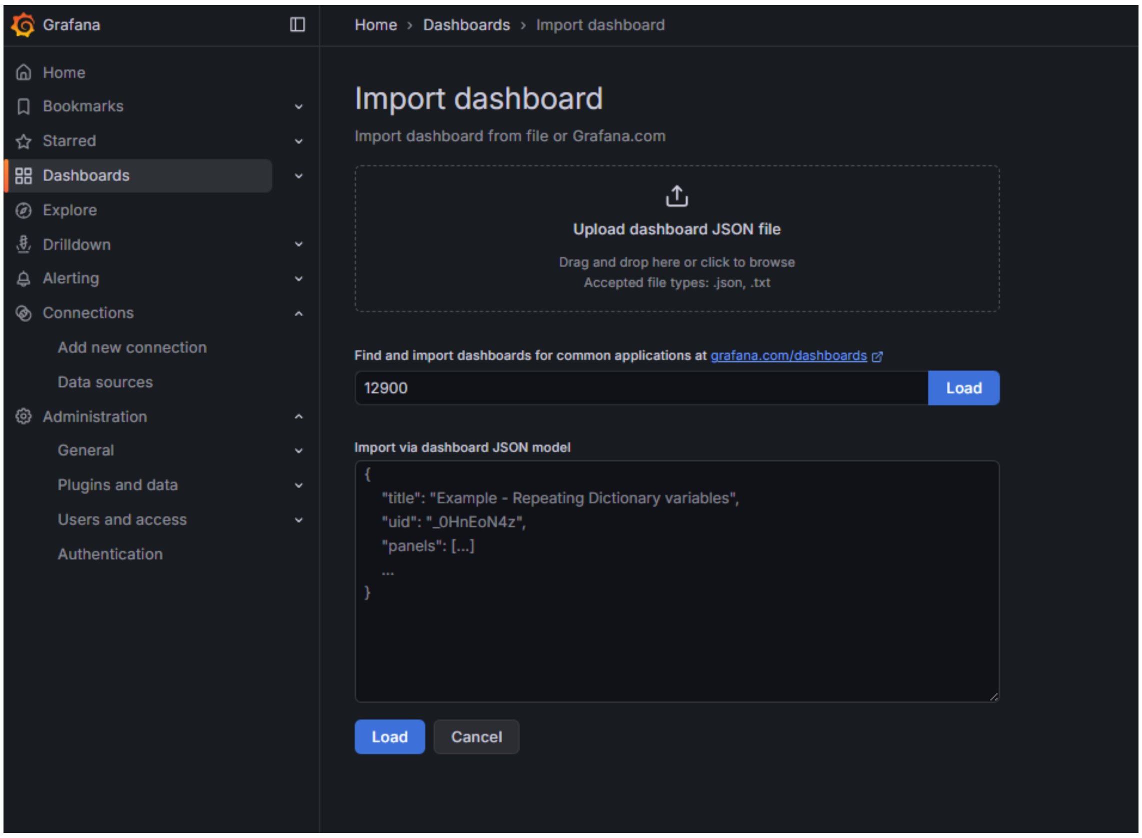
Successfully queried the Prometheus API.

Next, you can start to visualize data by building a dashboard, or by querying data in the Explore view.

En Grafana se configura Prometheus como *Data Source*, indicando la URL http://localhost:9090.

Esto permite que Grafana pueda consultar las métricas almacenadas en Prometheus y utilizarlas en los dashboards.

5.1 Elegir dashboard



Import dashboard

Import dashboard from file or Grafana.com

Find and import dashboards for common applications at grafana.com/dashboards

12900

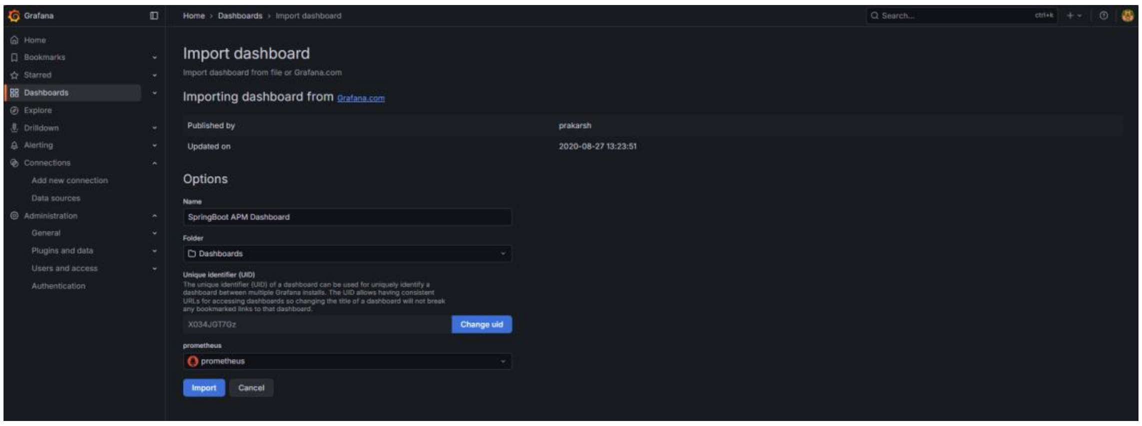
Import via dashboard JSON model

```
{
   "title": "Example - Repeating Dictionary variables",
   "uid": "_0HnEoN4z",
   "panels": [...]
   ...
}
```

Para la visualización de métricas se selecciona un dashboard orientado a aplicaciones Spring Boot y Micrometer, como el *Spring Boot APM Dashboard*.

Este dashboard permite visualizar métricas de la JVM, uso de memoria, CPU y peticiones HTTP.

5.2 Importar dashboard en Grafana



Import dashboard

Import dashboard from file or Grafana.com

Importing dashboard from Grafana.com

Published by	prakarsh
Updated on	2020-08-27 13:23:51

Options

Name: SpringBoot APM Dashboard

Folder: Dashboards

Unique identifier (UID)
The unique identifier (UID) of a dashboard can be used for uniquely identify a dashboard between multiple Grafana installs. The UID allows having consistent URLs for accessing dashboards so changing the title of a dashboard will not break any bookmarked links to that dashboard.

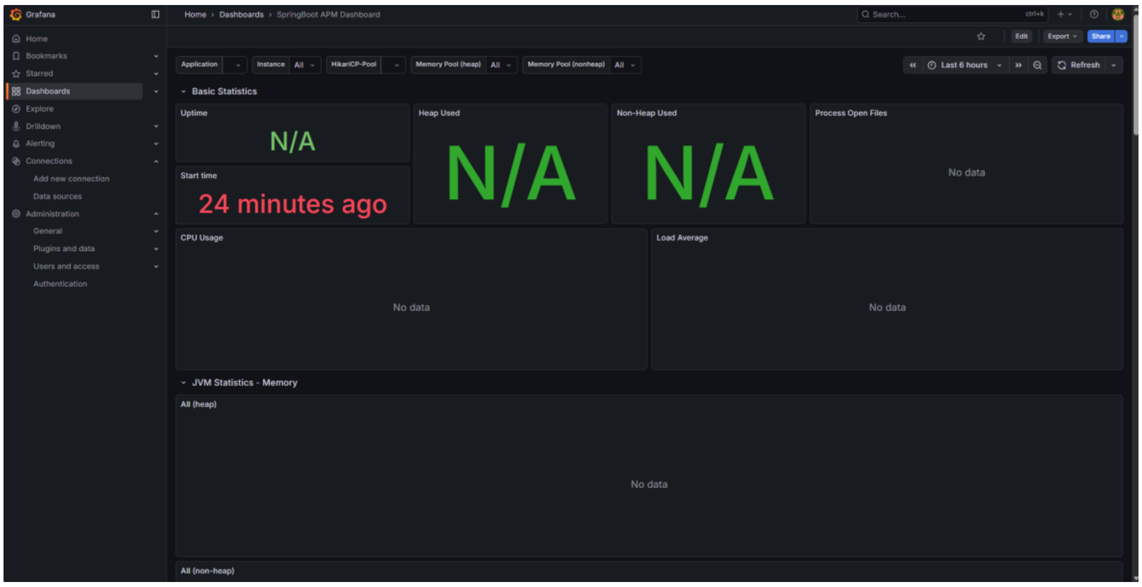
X034JGT7Gz

prometheus: prometheus

El dashboard se importa en Grafana mediante un archivo JSON.

Durante la importación se selecciona Prometheus como *Data Source*, asegurando que las gráficas utilizan las métricas recolectadas previamente.

5.3 Resultado final en Grafana



Application | Instance: All | HikariCP-Pool | Memory Pool (heap): All | Memory Pool (nonheap): All

Basic Statistics

Uptime: N/A

Start time: 24 minutes ago

Heap Used: N/A

Non-Heap Used: N/A

Process Open Files: No data

CPU Usage: No data

Load Average: No data

JVM Statistics - Memory

All (heap): No data

All (non-heap)

Como resultado final, se visualizan correctamente las métricas de la API de productos en Grafana.

El dashboard muestra información en tiempo real sobre el rendimiento de la aplicación y del sistema, confirmando que la monitorización ha sido configurada correctamente.