Carlos Andres

Veremos los escenarios donde desarrollamos los escenarios de calidad

Modelos Calidad

1906630

Tabla de contenido

[INTRODUCCION 3](#_Toc89629073)

[ESCENARIO SEGURIDAD 4](#_Toc89629074)

[PRUEBA OWASP 4](#_Toc89629075)

[PRUEBA SONARQUBE 4](#_Toc89629076)

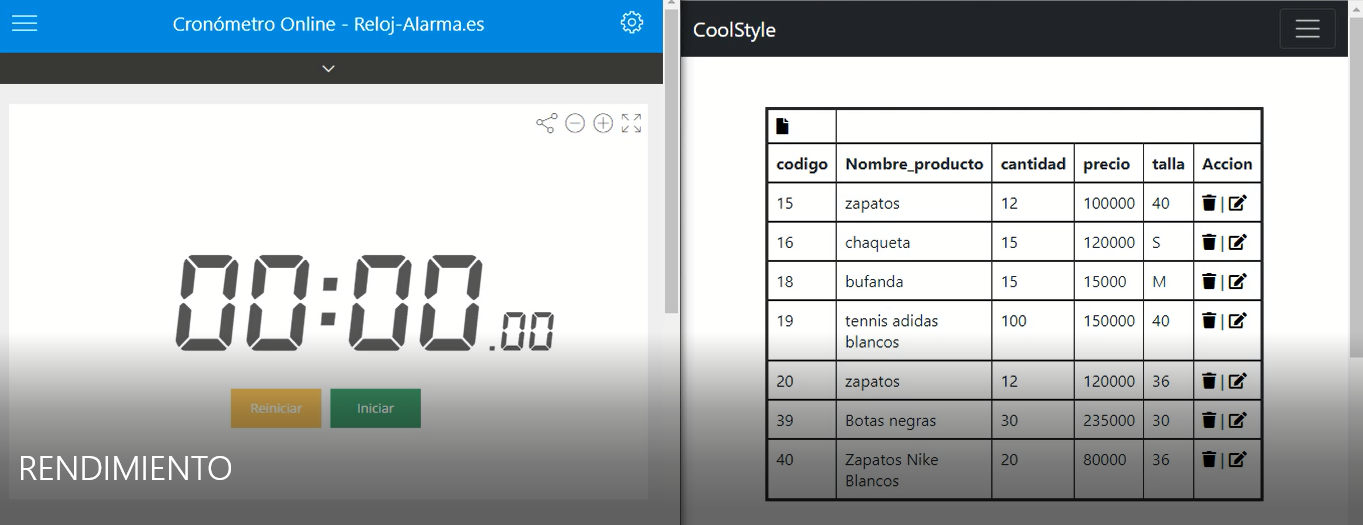
[METRICAS 5](#_Toc89629077)

[SONARQUBE 5](#_Toc89629078)

[OWASP 6](#_Toc89629079)

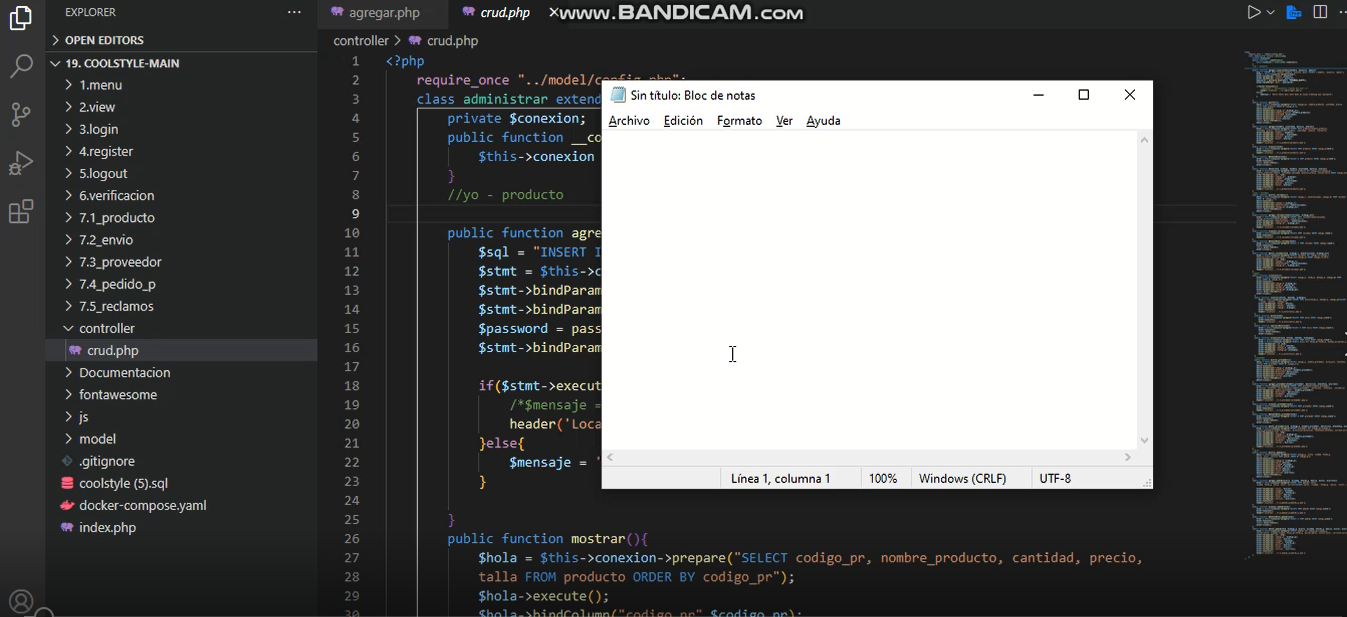
[ESCENARIO RENDIMIENTO 6](#_Toc89629080)

[METRICAS 7](#_Toc89629081)

[[](RENDIMIENTO.mp4) 7](#_Toc89629082)

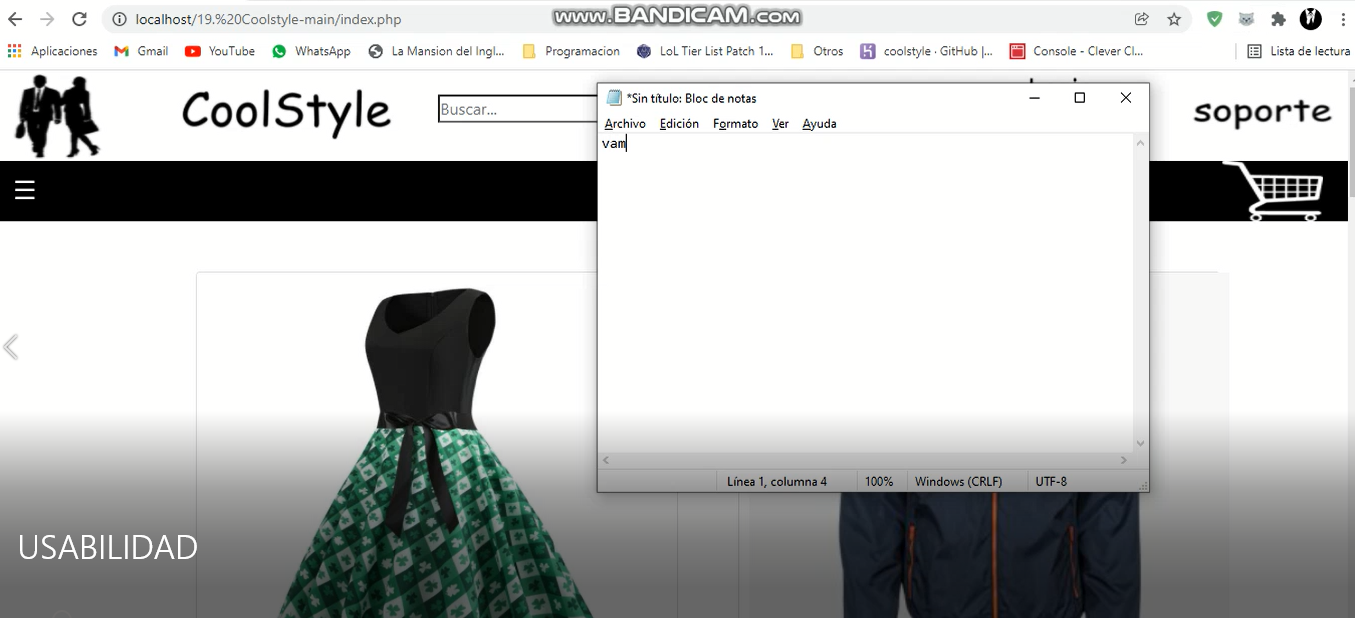
[ESCENARIO MODIFICABILIDAD 7](#_Toc89629083)

[METRICAS 8](#_Toc89629084)

[[](MODIFICABILIDAD.mp4) 8](#_Toc89629085)

[ESCENARIO DE USABILIDAD 8](#_Toc89629086)

[METRICAS 9](#_Toc89629087)

[[](USABILIDAD.mp4) 9](#_Toc89629088)

[ESCENARIO INTEROPERABILIDAD 11](#_Toc89629089)

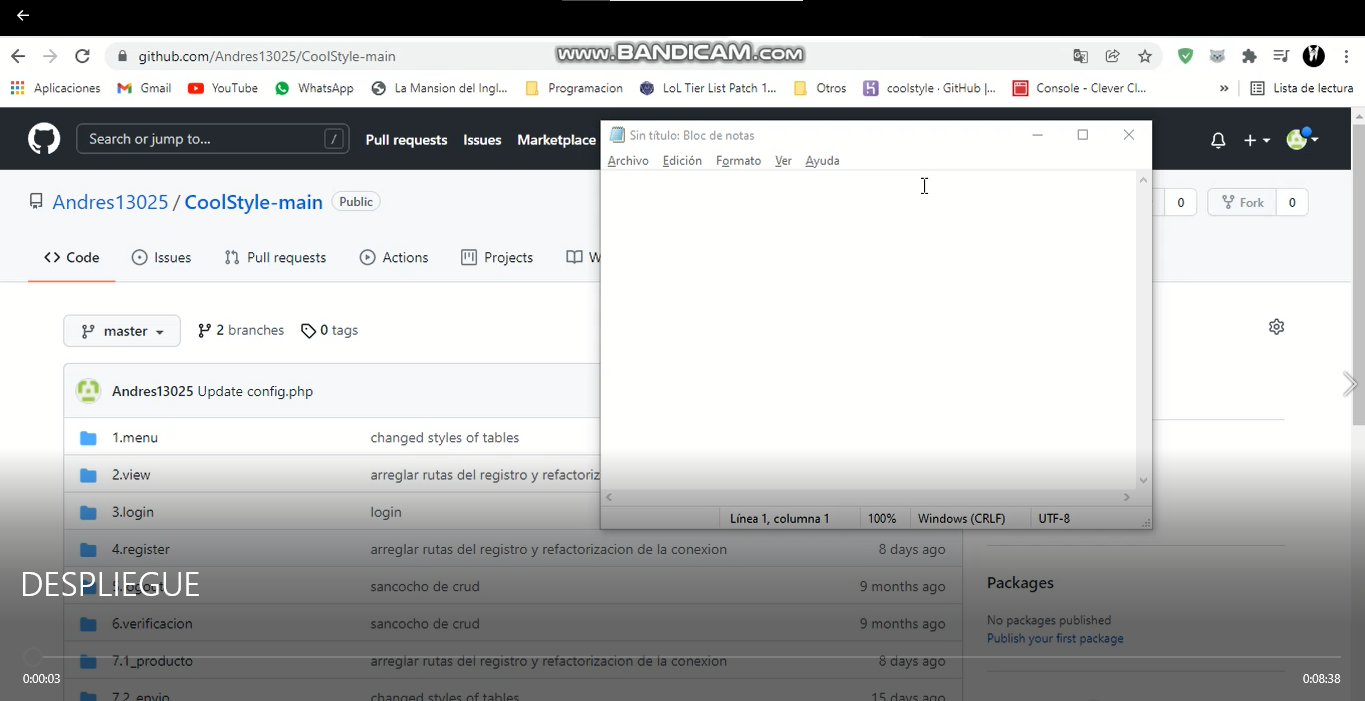
[METRICAS 11](#_Toc89629090)

[ESCENARIO ESCALABILIDAD 13](#_Toc89629091)

[METRICAS 13](#_Toc89629092)

[ESCENARIO DESPLEGABILIDAD 15](#_Toc89629093)

[METRICAS 15](#_Toc89629094)

[[](DESPLIEGUE.mp4) 15](#_Toc89629095)

# INTRODUCCION

Este informe pretende mostrar los resultados de las pruebas de calidad realizadas en la página de CoolStyle donde se evidenciarán si la página cumple con los estándares mínimos de calidad requeridos, las pruebas o escaneos se desarrollarán en el ambiente de desarrollo de la página, por lo cual se necesitarán conocimientos mínimos técnicos para entender completamente las pruebas

# ESCENARIO 1 y 2 SEGURIDAD

En este se escenario se plantea la situación de hacer el sondeo o escaneo del código fuente del desarrollo del proyecto, donde se analizan las vulnerabilidades más conocidas y posibles mejoras a implementar

Diagram

Description automatically generated

## PRUEBA OWASP

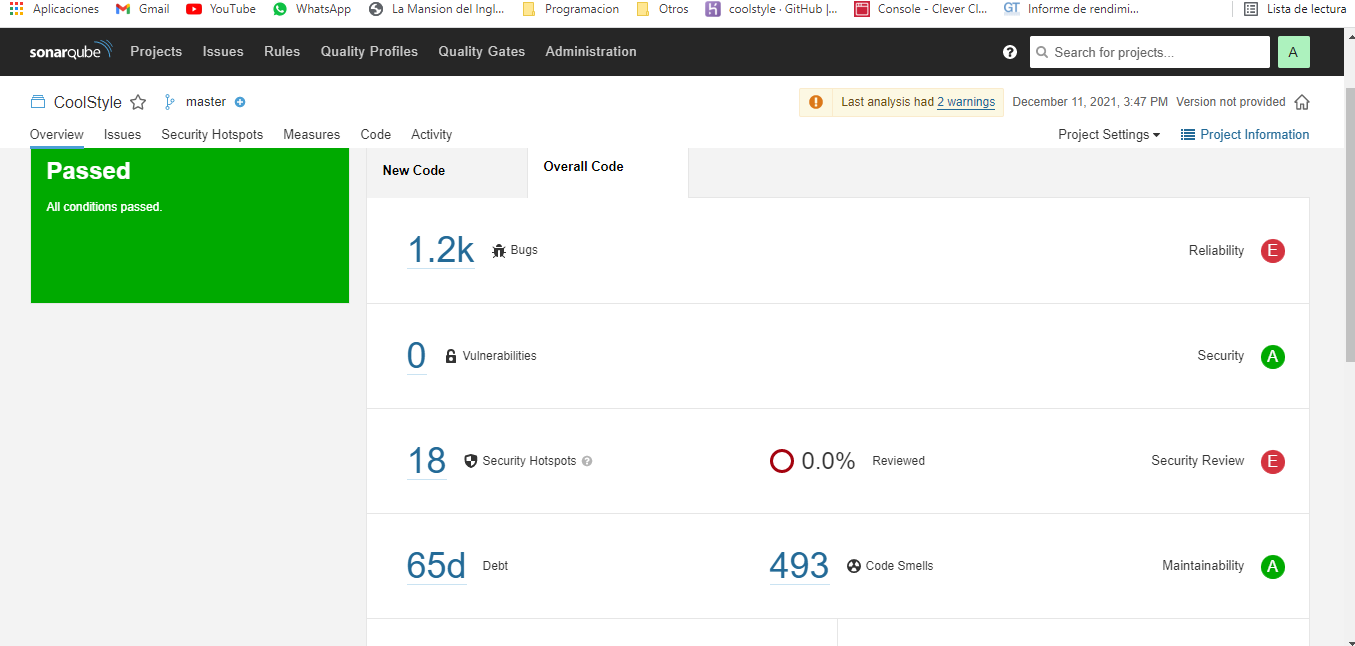
OWASP es un proyecto de código abierto dedicado a determinar y combatir las causas que hacen que el software sea inseguro. La Fundación OWASP es un organismo sin ánimo de lucro que apoya y gestiona los proyectos e infraestructura de OWASP.

## PRUEBA SONARQUBE

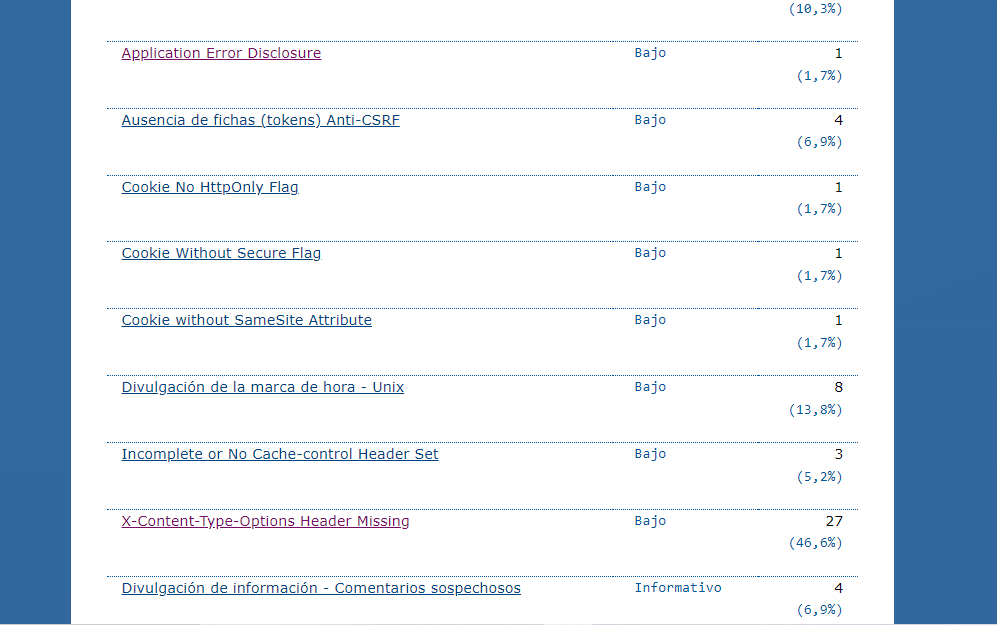
SonarQube es una plataforma para evaluar código fuente. Es software libre y usa diversas herramientas de análisis estático de código fuente como Checkstyle, PMD o FindBugs para obtener métricas que pueden ayudar a mejorar la calidad del código de un programa. ​

## METRICAS

### SONARQUBE

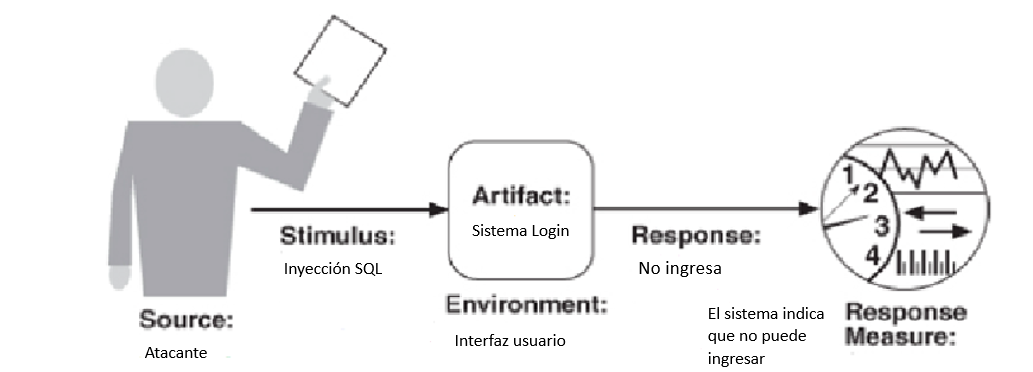


### OWASP



# ESCENARIO 3 SEGURIDAD

El atacante hace un ataque de inyección SQL al Login para tratar de tener acceso



Prueba de inyección SQL clásica

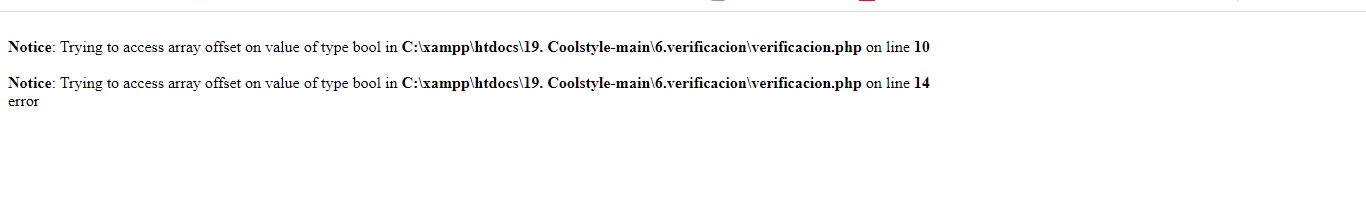
1. OR 1=1

2. $username = 1' or '1' = '1

## METRICAS

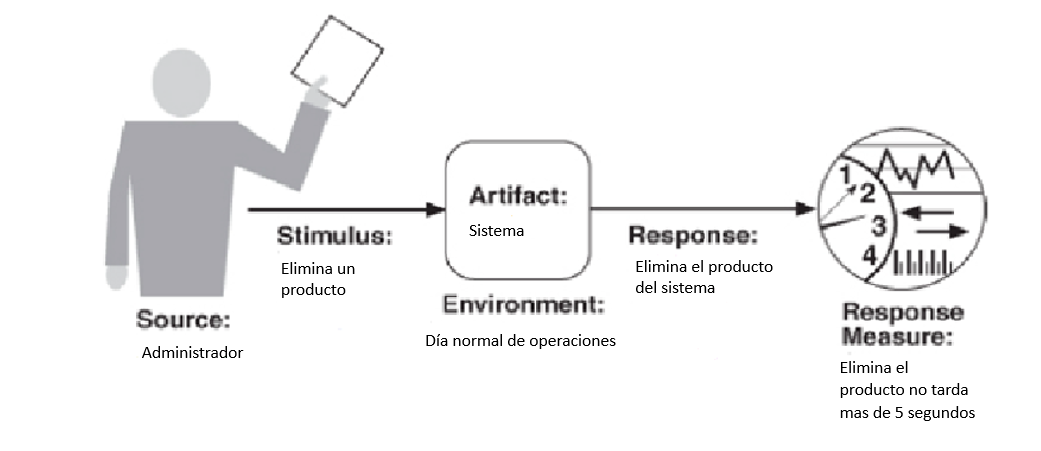


Response



# ESCENARIO RENDIMIENTO

Administrador realiza una consulta en el catálogo de productos en un día normal de operación del sistema, el sistema muestra el resultado de dicha consulta en un tiempo no máximo a 5 segundos



Prueba de cronometro de cuanto se tarda en ir a consultar a la base de datos y ejecutar una query

## METRICAS

### 

# ESCENARIO MODIFICABILIDAD

Un desarrollador agrega un nuevo atributo a una tabla de productos de la base de datos en tiempo de desarrollo. El atributo se agrega a la tabla de productos y no requiere la modificación de más de una tabla

Diagram

Description automatically generated

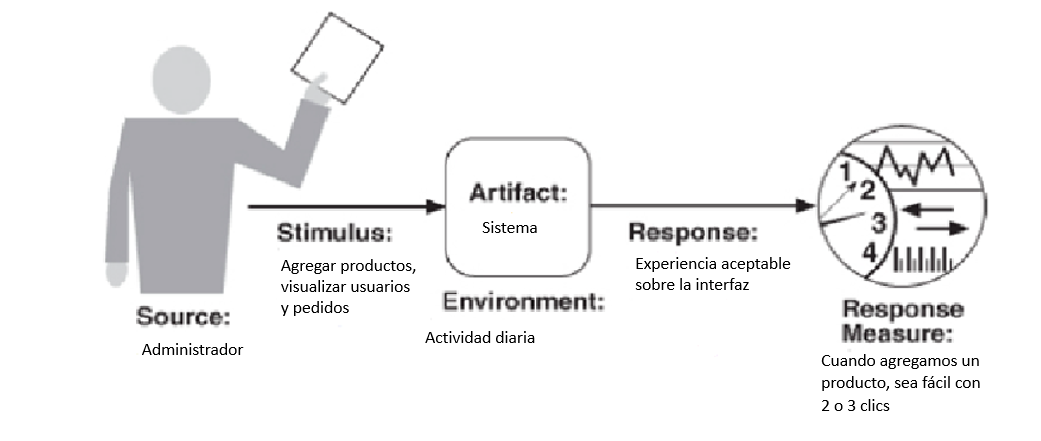
Esta prueba consta es que se puede modificar fácilmente el crear consultas nuevas a la base de datos, si quisiéramos crear una nueva funcionalidad en la que me trajera datos concretos lo haría sencillamente creando solo una función y llamarla desde el Front.

## METRICAS

## 

# ESCENARIO DE USABILIDAD

El Administrador busca diferentes pedidos en un día normal de operación del sistema, el sistema muestra en la etiqueta pedidos la información actualizada y también necesita agregar productos en el apartado de productos.



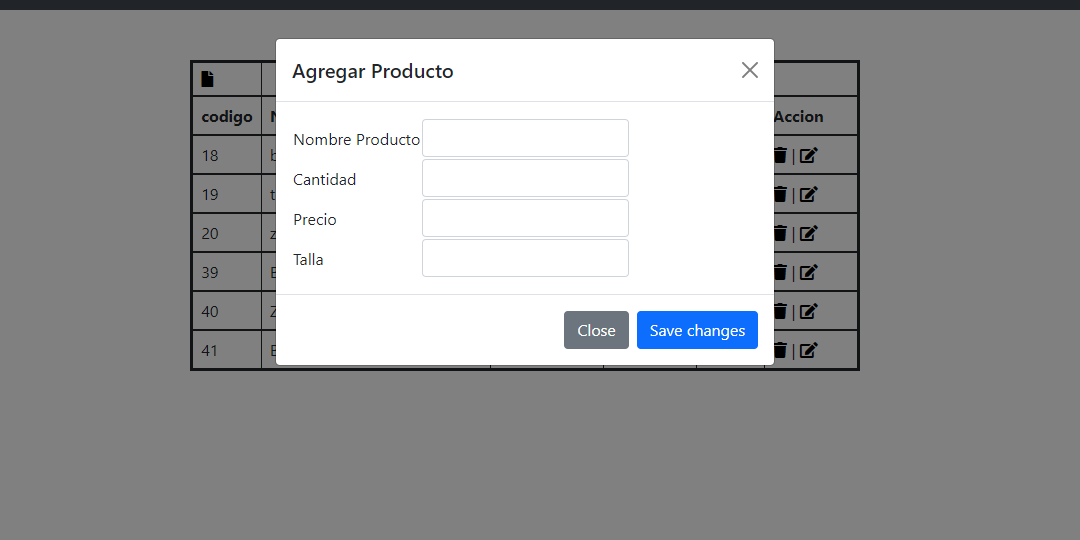
## METRICAS

Este escenario plantea la situación del trabajo normal del administrador para ver los pedidos y agregar productos donde cada una de estas acciones no representa un esfuerzo grande para ser realizadas solo con dos clics ya estábamos en el menú que se necesitaba

## 

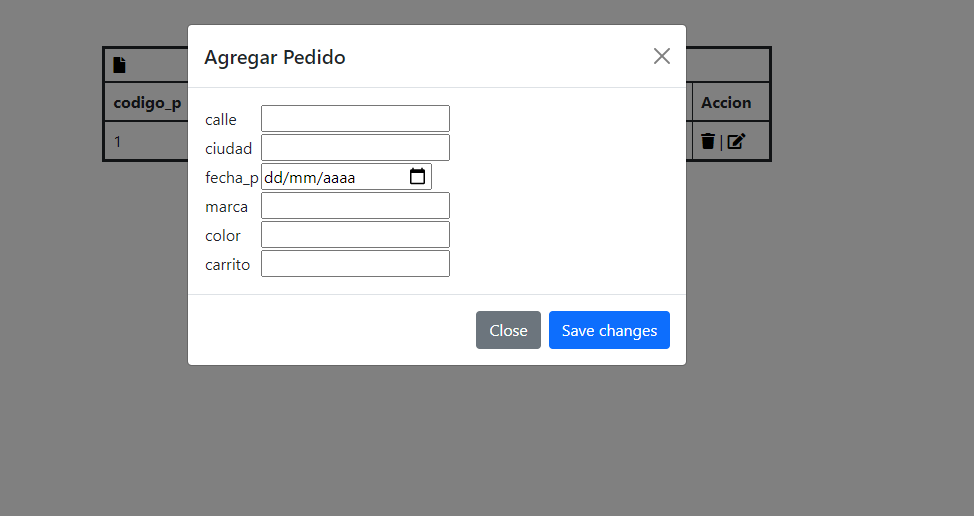
Clic1 > Agregar producto

Clic2 > Save Changes



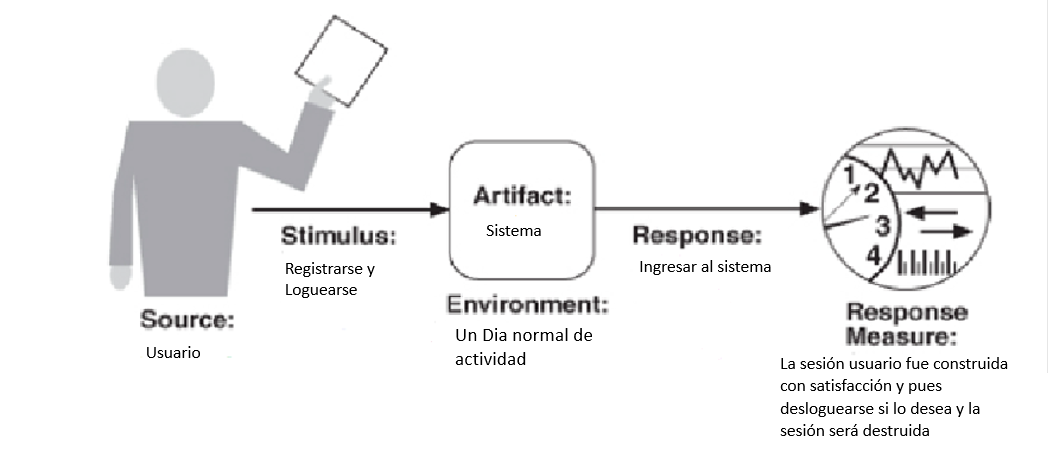
Clic1 > Agregar Pedido

Clic2 > Save Changes



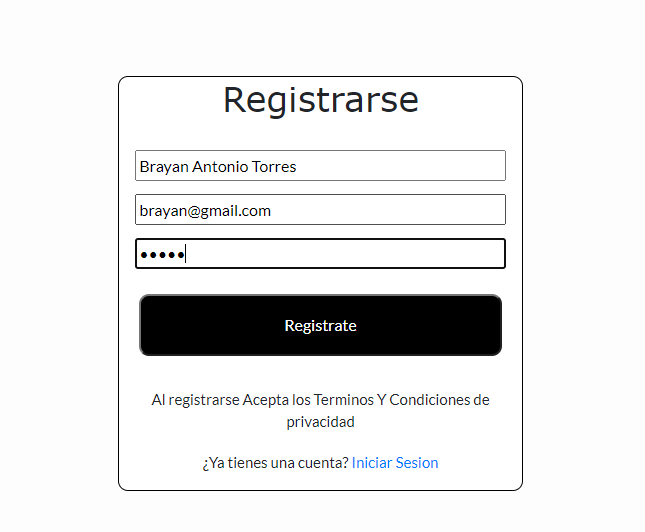
# ESCENARIO INTEROPERABILIDAD

un usuario hace una consulta al catálogo de productos en un día normal de operación del sistema, el sistema se comunica con la base de datos para verificar si el producto esta existente y se envía una nueva respuesta si está o no está el producto

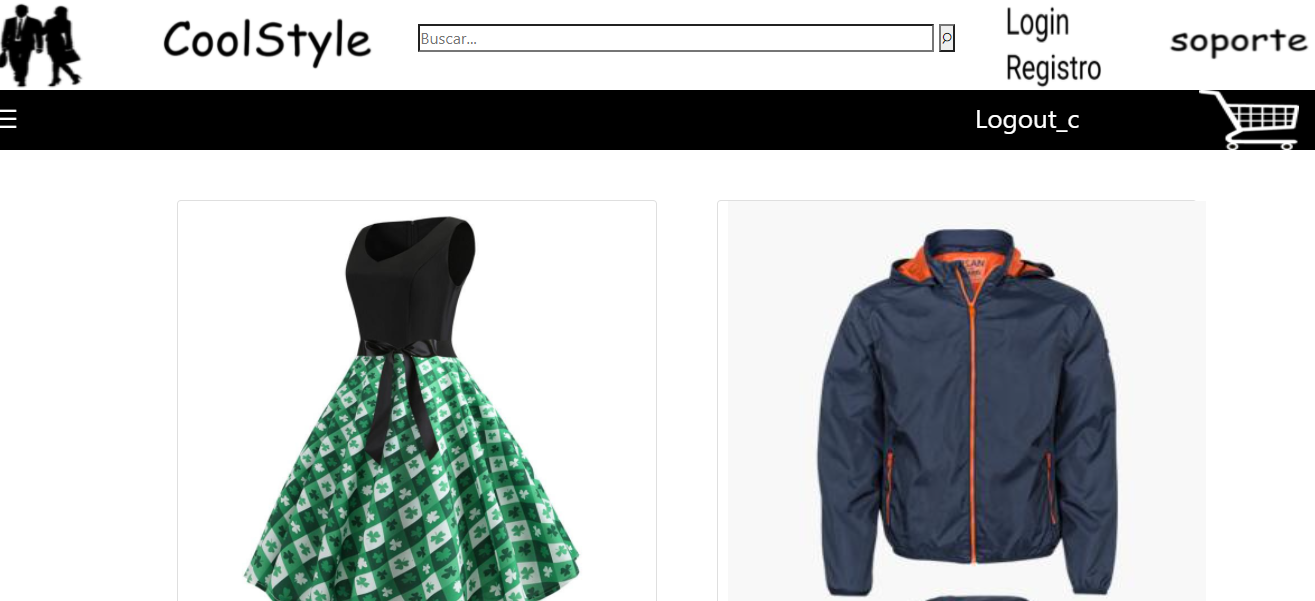


En este escenario vamos a usar la táctica de localizar servicio, entonces planteamos el escenario donde el usuario se va a registrar y después se va loguear y esto va consumir el servicio de verificar que este ya registrado y la conexión de la base de datos funcione

## METRICAS



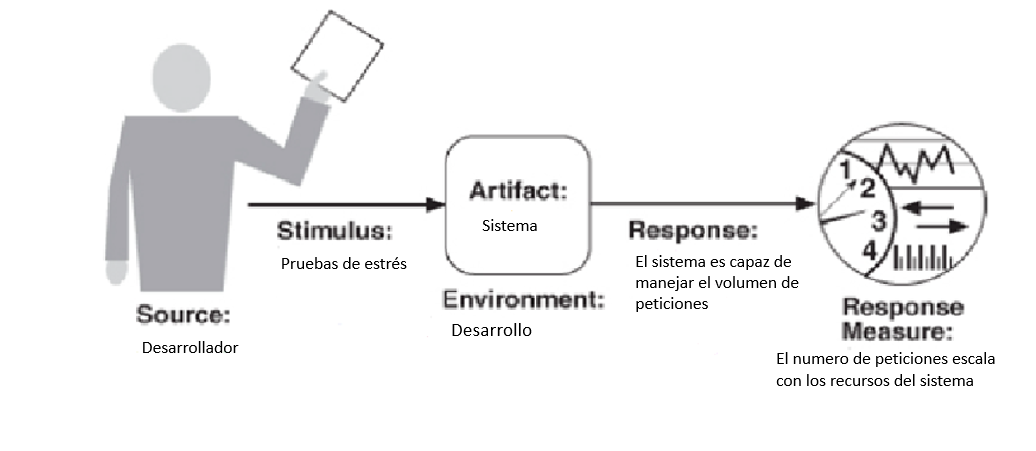




Como podemos ver el registro y el logueo se realizó con éxito además de que se creó el botón logout\_c que es para cerrar sesión y destruir la sesión

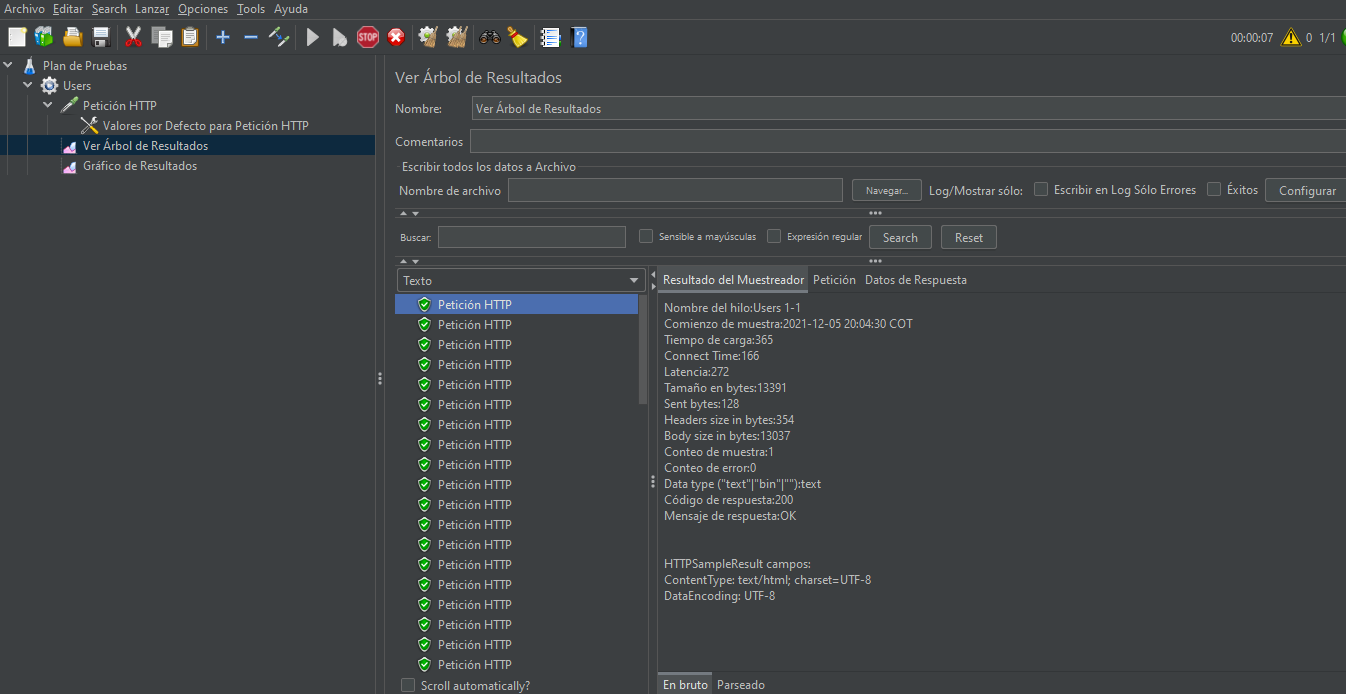
# ESCENARIO ESCALABILIDAD

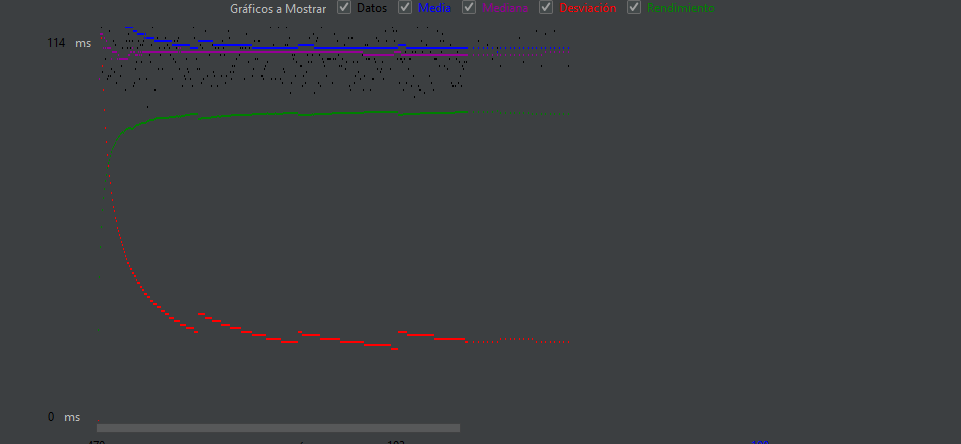
El escenario planteado es de ejecutar pruebas de estrés donde ejecutamos x cantidad de peticiones concurrentes a la página para comparar como crecen proporcionalmente los usuarios por los recursos requeridos

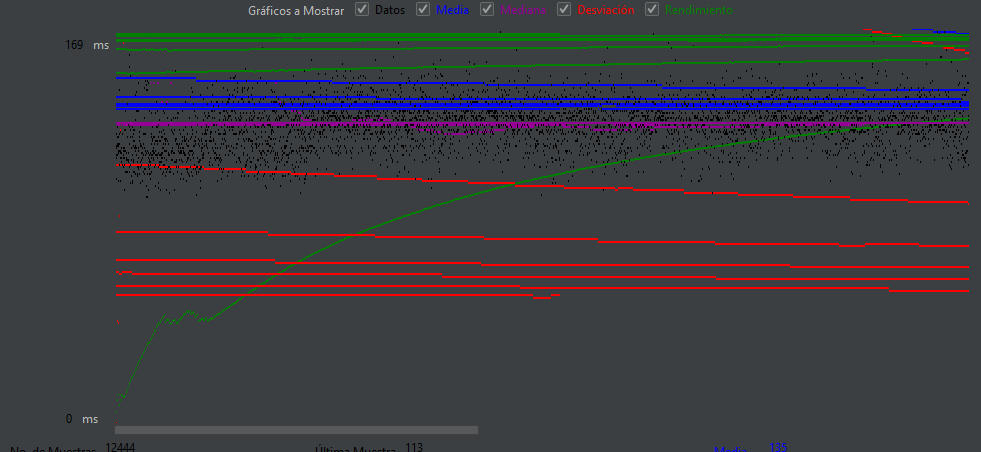


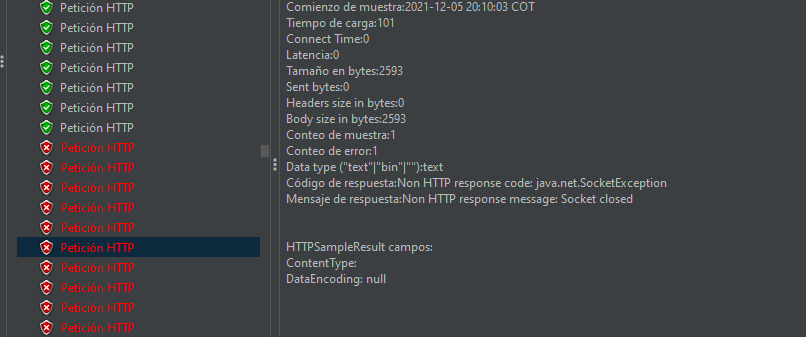
Vamos a usar Jmeter para ejecutar las pruebas de stress para ir mirando como decae el rendimiento mientras más crecen las solicitudes, así se pueden sacar estimaciones de cuantos recursos hay que dedicar para escalar la aplicación

## METRICAS

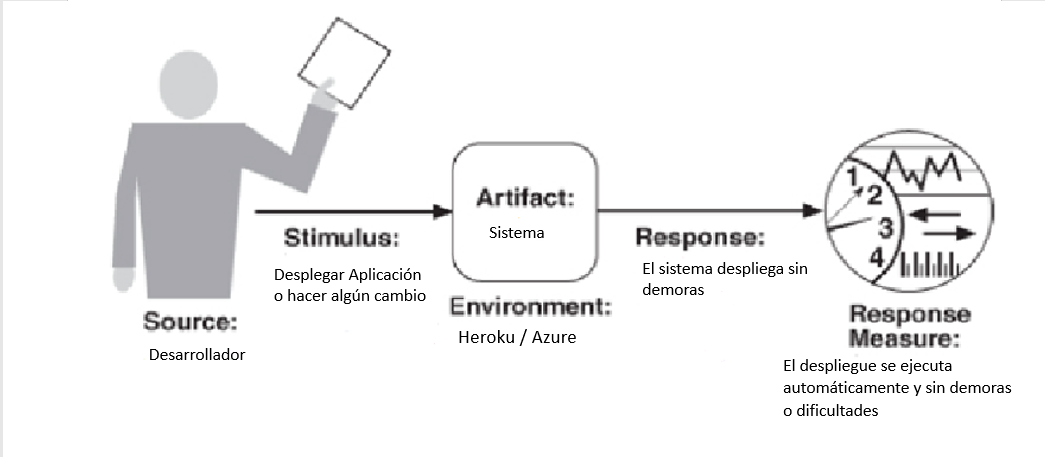








# ESCENARIO DESPLEGABILIDAD

Un desarrollador realiza algún cambio en el código y lo sube la rama máster del repositorio donde esté alojado, el despliegue se realiza automáticamente tras haber subido el cambio a la rama máster sin complicaciones 

Prueba de despliegue en heroku y despliegue automatico cuando se realiza algun cambio

## METRICAS

## 