

UNIVERSIDAD DEL ISTMO

FACULTAD DE INGENIERÍA

PROYECTO FINAL: DOCUMENTACION

Arquitectura de Sistemas, Comportamiento y Optimizacion

JORGE EDUARDO GUERRERO GARCIA

Carne: 2019-1096

Guatemala, 21 de octubre de 2022

**ÍNDICE**

[**1.** **INTRODUCCIÓN** 1](#_Toc117252862)

[**2.** **DESARROLLO** 2](#_Toc117252863)

[**2.1 Diagrama General del sistema** 2](#_Toc117252864)

[**2.2 ADD del Sistema** 2](#_Toc117252865)

[**3.** **CONCLUSIONES** 17](#_Toc117252866)

[**4.** **RECOMENDACIONES** 18](#_Toc117252867)

# **INTRODUCCIÓN**

El presente informe contiene toda la documentación relacionada con el proyecto realizado para el curso

1. **DESARROLLO**

## **2.1 Diagrama General del sistema**

Gráfico, Diagrama, Esquemático

Descripción generada automáticamente

## **2.2 ADD del Sistema**

**Escenarios de calidad: Proyecto Final**

Escenario de calidad: Comunicación del sistema

|  |  |
| --- | --- |
| Element | Statement |
| Stimulus | Un llamado a la aplicación React |
| Stimulus Source | Se ingresa al URL del portal o se realiza una acción del JS dentro de la aplicación |
| Environment | Se tiene un modelo MVVM conformado por un FrontEnd, un BackEnd y una base de datos, donde el BE administra todas las peticiones y se encargará de la comunicación del sistema |
| Artifact | Servidor Java Spring Boot |
| Response | Respuestas en archivos JSON |
| Response Measure | En base al recurso accedido y a los parámetros otorgados, se devolverá un resultado válido o erróneo de la base de datos al FrontEnd |

Escenario de calidad: WebServices

|  |  |
| --- | --- |
| Element | Statement |
| Stimulus | Llamado del servidor BE a otro servidor por medio de Rest API |
| Stimulus Source | Se designa una IP otorgada por el FE junto con datos adicionales y se realiza el llamado a la IP destino |
| Environment | Se posee un servidor con una configuración de CORS, por lo que deben ser entregados los headers correspondientes e IP válidas para la autorización de la comunicación  El servidor es capaz de manejar 100 solicitudes al mismo tiempo |
| Artifact | Web Service Method |
| Response | En caso de ser exitosa la operación se retorna el objeto JSON obtenido de la solicitud, en caso de fallo se le otorga una alerta al FE por medio de un objeto JSON sobre el fallo en la operación |
| Response Measure | El tiempo de respuesta debe ser menos de 10 segundos, en caso de fallo en la comunicación con el destino debe registrarse en un log y se cancela la operación y se le indica al usuario que lo intente de nuevo |

Escenario de calidad: Creación y administración de clientes individuales, grandes clientes y clientes mayoristas

|  |  |
| --- | --- |
| Element | Statement |
| Stimulus | Creación de un usuario y compra de elemento con un cliente |
| Stimulus Source | Se introducen los datos del cliente y se indica el tipo de suscripción que se desea adquirir para quedar registrada en la base de datos, por lo que ya se puede ingresar al portal para realizar una compra |
| Environment | Dentro de la base de datos existirá una diferenciación entre los distintos clientes registrados en el portal, los clientes individuales no tendrán ningún beneficio, los grandes clientes tendrán un descuento y podrán realizar pedidos por adelantado y los grandes distribuidores tendrán un mayor descuento, pedidos por adelantado y compras al crédito |
| Artifact | Modulo de clientes |
| Response | Compras en el portal de ventas |
| Response Measure | Al momento de introducir los datos del cliente de las diferentes opciones de pago según la suscripción y aplica los respectivos descuentos a la compra |

Escenario de calidad: Facturas de compras

|  |  |
| --- | --- |
| Element | Statement |
| Stimulus | Compra de un producto dentro del portal |
| Stimulus Source | Se ingresan los datos del cliente y la cantidad de elementos a comprar |
| Environment | Dentro del sistema se tendrá un catálogo con variedad de productos, donde se tendrá un inventario y los respectivos precios e impuestos, por lo que al realizar las compras se eliminarán las unidades del inventario con su número de serie y se procederá a aplicar la orden de compra. En el caso de realizar una compra a crédito se queda registrado y se notifica el total a pagar a fin de mes a los clientes y los pedidos anticipados se cobran en el momento y se entregan cuando estén listos los productos. |
| Artifact | Catálogo, inventario y módulo de compras |
| Response | Se indica la confirmación de la compra y se entrega una factura digital |
| Response Measure | Se genera una factura que incluye todo el detalle de la compra, datos del cliente y el detalle de las unidades adquiridas con su número de serie |

Escenario de calidad: Búsqueda de productos en el portal

|  |  |
| --- | --- |
| Element | Statement |
| Stimulus | Texto plano en una barra de búsqueda |
| Stimulus Source | Se ingresa en el portal en la parte del catálogo es posible encontrar todos los elementos y filtrarlos en base a texto ya sea por nombre, modelo, marca, color, etc. y se puede ordenar en base a precios y al ingresar en el producto es posible ver todo el detalle |
| Environment | El portal tendrá un catálogo que mostrar productos provenientes de fábricas distribuidoras donde se podrá buscar elementos deseados entre los distintos fabricantes, pero toda la información externa queda registrada dentro del sistema de ventas |
| Artifact | Catálogo |
| Response | Elementos coincidentes |
| Response Measure | Luego de la búsqueda, se mostrarán los resultados que coincidan con el texto ingresado, ordenados de precio de mayor a menor o de menor a mayor, permitiendo ingresar en los detalles para realizar la compra |

Escenario de calidad: El sistema debe permitir el manejo de cualquier cantidad de usuarios

|  |  |
| --- | --- |
| Element | Statement |
| Stimulus | Solicitud al FE del sistema |
| Stimulus Source | Se realiza un llamado al FE que a su vez realiza un llamado al servidor BE el cual va a procesar una solicitud |
| Environment | Se tiene un servidor BE Restful el cual realizara el rápido procesamiento de solicitudes para poder soportar cualquier carga de |
| Artifact | Portal de ventas |
| Response | Respuesta del servidor BE |
| Response Measure | El servidor BE debe responder al FE dentro de los próximos 5 segundo |

Escenario de calidad: Ambiente de desarrollo en GitHub para el servidor BE

|  |  |
| --- | --- |
| Element | Statement |
| Stimulus | Desarrollo de nuevo código en el repositorio |
| Stimulus Source | Los desarrolladores podrán integrar código nuevo en base a las reglas del repositorio |
| Environment | El código del servidor BE se encuentra en un controlador de versiones en la nube el cual debe contar con 3 branches que se conectaran con plataforma CI/CD y el código será actualizado de manera constante realizando un proceso de CI/CD verificando los unit test, compilación del código y analisis del codigo |
| Artifact | Repositorio privado de GitHub |
| Response | Llamado a la plataforma de CI/CD para el inicio de los test y build del código |
| Response Measure | El repositorio hace un llamado por medio de un webhook donde envía información del pull request e información de las branchs afectadas |

Escenario de calidad: Registro de pedidos

|  |  |
| --- | --- |
| Element | Statement |
| Stimulus | Confirmación de un pedido recibido o cancelación de un pedido no recibido |
| Stimulus Source | Se selecciona una orden de pedido y se le da click a la cancelación o confirmación y el sistema recoge todos los datos del pedido y los envía a la fábrica para que quede registrado |
| Environment | Los sistemas de fabricas y ventas tienen sus propias bases de datos, por lo que los sistemas de ventas tienen su propio manejo de pedidos y enviar los datos de vuelta a la fábrica para que ellos registren el estado del pedido |
| Artifact | Registro de pedidos |
| Response | Actualización de la base de datos y ejecución del web service |
| Response Measure | En la base de datos se modifica el estado del pedido y ya no aparece en el portal de elementos pendientes de confirmar o cancelar |

Escenario de calidad: Registro de fábricas

|  |  |
| --- | --- |
| Element | Statement |
| Stimulus | Registro de una nueva fábrica |
| Stimulus Source | Se ingresa el nombre de la fábrica, la dirección ip y puerto del servidor de la fábrica y las credenciales en el registro de fábricas del sistema |
| Environment | Para poder realizar comunicación con el sistema de fábricas es necesaria la autenticación previa para recibir las credenciales y conectarnos al sistema para acceder al catálogo y realizar pedidos |
| Artifact | Marcas registradas |
| Response | Confirmación exitosa del registro |
| Response Measure | Se hace registro de la marca y es posible acceder al catálogo de la fábrica y hacer pedidos a la misma |

Escenario de calidad: Manejo de garantías

|  |  |
| --- | --- |
| Element | Statement |
| Stimulus | Ingreso de los datos de una unidad |
| Stimulus Source | Dentro del portal en la sección de garantías se ingresa el numero de serie del dispositivo que se desea devolver por garantía |
| Environment | Las fábricas establecen la garantía de sus productos y se manejan en base a su número de serie y se valida la fecha en que fue vendido el dispositivo y la fecha actual para comprobar la validez de la garantía |
| Artifact | Validación de garantías |
| Response | Confirmación o denegación de la vigencia de la garantía |
| Response Measure | Se obtiene la confirmación de la garantía del producto y en caso positivo se obtienen los datos del dispositivo para confirmar la devolución del producto defectuoso y realizar un nuevo pedido con la unidad nueva |

**Attribute-Driven Design - Iteración 1 (Sistema de Ventas)**

Paso 1: Se tiene un sistema de venta de electrónicos el cual se conforma de un FrontEnd, un BackEnd y una base de datos, a su vez este sistema puede comunicarse con sistemas externos siendo estos fábricas donde se pueden adquirir el catalogo de productos que se va a ofrecer en el portal de ventas, por lo que este sistema permitirá:

* Ofrecimiento de un catálogo en base a fabricas registradas en el sistema
* Creación de clientes dentro de la plataforma con suscripciones que otorgaran privilegios segun la categoria a la hora de comprar productos
* Se realizará un registro de las compras y emisiones de facturas al momento en que los clientes realicen las compras
* Manejo, confirmación y cancelación de pedidos a las fábricas
* Todos los dispositivos se manejan en base a un número de serie
* Se podrá hacer uso de la garantía si se encuentra dentro del plazo
* Todas las peticiones al BE deben ser procesadas dentro de los próximos 5 segundos

Paso 2 Refinar el módulo:

2.a Elegir los controles arquitectónicos

* Un cliente podrá realizar compras dentro de un portal web
* Comunicación a otros sistemas por medio de web services y únicamente se realizará mediante el BE
* Se poseerá un servidor BE Restful y se hará manejo de archivos JSON
* Se contará con una base de datos relacional
* Se podrá hacer la búsqueda de productos mediante texto plano
* El sistema debe permitir cualquier cantidad de usuarios
* Se debe hacer manejo de un número de serie por dispositivo

2.b Elegir un patrón arquitectónico

* El sistema debe ser accedido por medio de un portal web
* El sistema debe contar con un Frontend, un Backend y una base de datos
* Solamente el BE puede comunicarse con otros servidores

Se determino que el patron mas adecuado para la solución es el Modelo Vista Controlador, ya que en este caso contamos con un sistema que cuenta con una interfaz de usuario en la web siendo la vista, tenemos un servidor BE, que sera el controlador, el cual va a recibir peticiones de la interfaz y se comunicará con una base de datos, que será el modelo o con otros servidores para procesar y dar respuesta al usuario.

2.c Instanciar módulos y asignar funcionalidades usando múltiples vistas:

* Para iniciar el servidor B, se debe establecer primero la conexión con la base de datos, de lo contrario no iniciara el sistema
* El servidor BE tendrá una configuración de seguridad donde se establecerán los headers y dominios permitidos a la hora de procesar las peticiones
* Se contará con un único servidor de bases de datos y de BE y se contará con una cantidad indefinida de usuarios que se comunican desde la interfaz web
* El BE podra establecer comunicación con más de un servidor de los sistemas de fábrica
* Los usuarios empleados y administradores podrán administrar el portal, registro de clientes, los pedidos y el catálogo del sistema
* Los clientes podrán hacer la búsqueda de productos, realizar compras, adquirir, renovar y cancelar suscripciones, y devolver productos

2.d Definir las interfaces de módulos secundarios

* Integración de nuevas funcionalidades del servidor BE
  + Funcionales: Debe ser posible la integración del nuevo código para el servidor BE mejorando el código actual y agregar nuevas funcionalidades mediante la compilación del proyecto guardado en un controlador de versiones
  + De calidad: El desarrollo debe cumplir con un proceso de CI/CD, donde no se permitirá el aumento de la deuda técnica cuando se hacen nuevas integraciones de código, todo este proceso debe ser validado por el Lead Developer del proyecto
  + De negocio: Reducir la deuda técnica y ofrecer un producto de calidad al consumidor
* Responsiveness del portal en plataformas móviles
  + Funcionales: El portal debe adaptar los elementos al tamaño de la pantalla de manera que cualquier tipo de pantalla sea compatible
  + De calidad: Los elementos deben adaptarse de manera correcta con los tamaños de pantalla más estandarizados
  + De negocio: Una interfaz de usuario que no limite la interacción del cliente
* Envio de informacion
  + Funcionales: El servidor BE debe garantizar que la información enviada a servidores externos ha sido recibida correctamente
  + De calidad: Debe quedar un log de operaciones donde se tengan registro de los datos enviados y aquellos que ya se han confirmado de recibidos, para volver a enviar aquellos que aún no han confirmado
  + De negocio: Consistencia de datos entre negocios

2.e Verificar y refinar casos de uso y escenarios de calidad como restricciones para los módulos secundarios

* Ante un fallo del servidor BE, se puede hacer uso del log para completar aquellas transacciones que han quedado incompletas
* Ante la pérdida de comunicación con la base de datos, el servidor BE de manera constante deberá intentar la conexion cada segundo, llegado a los 15 segundos se manda una alerta de que se ha perdido conexión con la base de datos
* En caso de que falle el proceso de CI/CD no debe ser posible cambiar el servidor actual hasta que sea validado por el Lead Developer

**Attribute-Driven Design - Iteración  (CI/CD Jenkins)**

Paso 1: Se tiene un sistema de venta de electrónicos el cual se conforma de un FrontEnd, un BackEnd y una base de datos, en el cual el BackEnd tendrá un proceso de CI/CD para garantizar la calidad y buen funcionamiento del código antes de ser subido al servidor para ello se tendrá subido el código en el controlador de versiones GitHub y se integrará con la herramienta Jenkins para manejar el CI/CD a través de pipelines analizando el código con la herramienta Sonarqube. Para ello se deberá cumplir:

* El repositorio debe contener las branches Dev, UAT y Master, de la misma manera Jenkins contendrá pipelines individuales para cada una de estas branches
* Se deberá notificar a Jenkins iniciar el proceso del pipeline correspondiente cuando se haga un pull request en cualquiera de las branches
* Solo el Lead Developer puede aprobar el Merge
* Jenkins obtendrá el código entrante del pull request y correrá los unit test, compilara el proyecto y enviara el codigo a sonarqube para realizar el análisis
* El código debe cumplir con el quality gate de sonarqube para validar el proceso de CI/CD
* En caso de algún fallo se debe notificar al product owner y lead developer mediante correo

Paso 2 Refinar el módulo:

2.a Elegir los controles arquitectónicos

* Las comunicaciones se realizaran mediante peticiones HTTP y WebHooks de Github
* Se hará uso de tokens para la autenticación en las comunicaciones
* En github se podrá acumular código para luego realizar
* Se podrán crear diferentes branches basadas en Dev, UAT, Master para luego hacer merge a las mismas

2.b Elegir un patrón arquitectónico

* El proceso de CI/CD tendrá una ejecución basada en pasos donde cada paso es un procesamiento de datos donde se puede verificar el punto en que fallo
* Los pipelines serán activados según el branch que haya sido modificado
* Es un proceso recursivo que se va a realizar la línea de ejecución múltiples veces

Se determinó que el patrón más adecuado para la solución es el de Tuberías y Filtros ya que el proceso de CI/CD se basa en el proceso y validación de los pasos especificados y existe la posibilidad que existan fallos dentro de esto por lo que se toman diversas acciones cuando estos ocurren

2.c Instanciar módulos y asignar funcionalidades usando múltiples vistas:

* Se contará con un controlador de versiones en la nube, por lo que múltiples usuarios pueden acceder al mismo repositorio y pueden implementar cambios directamente en el sitio web o subir los cambios que mantienen los desarrolladores de manera local en sus dispositivos
* Solo podrá haber un Lead Developer en el proyecto
* Solo se realizará una ejecución de un pull request a la vez dentro de un mismo branch
* Existirá un único servidor de Sonarqube
* Existirá un único servidor de Jenkins el cual contará con 3 líneas de ejecución simultáneas

2.d Definir las interfaces de módulos secundarios

* Exposición de métricas
  + Funcionales: El servidor de Jenkins debe tener instalado un plugin que permita la adquisición de métricas útiles para ver datos sobre el servidor y las ejecuciones dentro del mismo
  + De calidad: Las métricas debe ser expuestas en un formato compatible para su recopilación con una herramienta de monitoreo
  + De negocio: Conocer el estado del servidor y de las ejecuciones
* Cancelación de una línea de ejecución
  + Funcionales: Debe ser posible detener la ejecución de cualquier pipeline cuando sea requerido
  + De calidad: El servidor debe notificar al Github que la validación ha sido denegada, para impedir la integración del código
  + De negocio: Detener procesos debido a errores humanos
* Fallos en el pipeline
  + Funcionales: Cuando ocurre un error en el pipeline debe enviarse una notificación mediante un correo electrónico a las direcciones registradas
  + De calidad: El cuerpo del correo debe contener qué pipeline fallo, numero de build que fallo, identificador del pull request, branch de la que proviene el cambio entrante y responsable del código y url de los logs del pipeline
  + De negocio: Administración de los cambios al código

2.e Verificar y refinar casos de uso y escenarios de calidad como restricciones para los módulos secundarios

* No debera ser posible la ejecucion manual de los pipelines, unicamente seran activados cuando Github realice la comunicación con todos los datos del pull request
* Al reabrir un pull request luego de la corrección en el código debe volver a ser invocado el pipeline para un nuevo procesamiento
* El Lead developer puede aceptar cambios sin la validación del pipeline si se requiere en caso que exista un error en alguno de los servidores

**Attribute-Driven Design - Iteración  (CI/CD Drone/Harness)**

Paso 1: Se tiene un sistema de venta de electrónicos el cual se conforma de un FrontEnd, un BackEnd y una base de datos, en el cual el BackEnd tendrá un proceso de CI/CD para garantizar la calidad y buen funcionamiento del código antes de ser subido al servidor para ello se tendrá subido el código en el controlador de versiones GitHub y se integrará con la herramienta Drone para manejar el CI/CD a través de pipelines analizando el código con la herramienta Sonarqube. Para ello se deberá cumplir:

* El repositorio debe contener las branches Dev, UAT y Master, de la misma manera Drone contendrá pipelines individuales para cada una de estas branches
* Se deberá notificar a Drone iniciar el proceso del pipeline correspondiente cuando se haga un pull request en cualquiera de las branches
* Solo el Lead Developer puede aprobar el Merge
* Drone obtendrá el código entrante del pull request y correrá los unit test, compilara el proyecto y enviara el codigo a sonarqube para realizar el análisis
* El código debe cumplir con el quality gate de sonarqube para validar el proceso de CI/CD
* En caso de algún fallo se debe notificar al product owner y lead developer mediante correo

Paso 2 Refinar el módulo:

2.a Elegir los controles arquitectónicos

* Las comunicaciones se realizaran mediante peticiones HTTP y WebHooks de Github
* Se hará uso de tokens para la autenticación en las comunicaciones
* En github se podrá acumular código para luego realizar
* Se podrán crear diferentes branches basadas en Dev, UAT, Master para luego hacer merge a las mismas

2.b Elegir un patrón arquitectónico

* El proceso de CI/CD tendrá una ejecución basada en pasos donde cada paso es un procesamiento de datos donde se puede verificar el punto en que fallo
* Los pipelines serán activados según el branch que haya sido modificado
* Es un proceso recursivo que se va a realizar la línea de ejecución múltiples veces

Se determinó que el patrón más adecuado para la solución es el de Tuberías y Filtros ya que el proceso de CI/CD se basa en el proceso y validación de los pasos especificados y existe la posibilidad que existan fallos dentro de esto por lo que se toman diversas acciones cuando estos ocurren

2.c Instanciar módulos y asignar funcionalidades usando múltiples vistas:

* Se contará con un controlador de versiones en la nube, por lo que múltiples usuarios pueden acceder al mismo repositorio y pueden implementar cambios directamente en el sitio web o subir los cambios que mantienen los desarrolladores de manera local en sus dispositivos
* Solo podrá haber un Lead Developer en el proyecto
* Solo se realizará una ejecución de un pull request a la vez dentro de un mismo branch
* Existirá un único servidor de Sonarqube
* Existirá un único servidor de Drone el cual contará con 3 líneas de ejecución simultáneas

2.d Definir las interfaces de módulos secundarios

* Cancelación de una línea de ejecución
  + Funcionales: Debe ser posible detener la ejecución de cualquier pipeline cuando sea requerido
  + De calidad: El servidor debe notificar al Github que la validación ha sido denegada, para impedir la integración del código
  + De negocio: Detener procesos debido a errores humanos
* Fallos en el pipeline
  + Funcionales: Cuando ocurre un error en el pipeline debe enviarse una notificación mediante un correo electrónico a las direcciones registradas
  + De calidad: El cuerpo del correo debe contener qué pipeline fallo, numero de build que fallo, identificador del pull request, branch de la que proviene el cambio entrante y responsable del código y url de los logs del pipeline
  + De negocio: Administración de los cambios al código

2.e Verificar y refinar casos de uso y escenarios de calidad como restricciones para los módulos secundarios

* No debera ser posible la ejecucion manual de los pipelines, unicamente seran activados cuando Github realice la comunicación con todos los datos del pull request
* Al reabrir un pull request luego de la corrección en el código debe volver a ser invocado el pipeline para un nuevo procesamiento
* El Lead developer puede aceptar cambios sin la validación del pipeline si se requiere en caso que exista un error en alguno de los servidores

**Attribute-Driven Design - Iteración  (Actualización del Servidor BE)**

Paso 1: Se tiene un sistema de venta de electrónicos el cual se conforma de un FrontEnd, un BackEnd y una base de datos, en el cual el BackEnd tendrá un proceso de CI/CD para garantizar la calidad y buen funcionamiento del código antes de ser subido al servidor y ser levantado nuevamente el servidor con los cambios aplicados. Para ello se deberá cumplir:

* Se deben haber ejecutado correctamente los procesos de CI/CD del repositorio
* El Lead Developer debe aprobar el merge del código a la branch raíz
* Se debe crear una imagen de Docker para la creación de un contenedor
* La ejecución debe iniciarse inmediatamente luego de realizar merge

Paso 2 Refinar el módulo:

2.a Elegir los controles arquitectónicos

* GitHub enviara la compilacion a todos los servidores especificados para cierta branch
* Se generará la imagen la cual se enviará al servidor respectivo actualizando la imagen y se reemplazará el contenedor y se ejecutará inmediatamente
* Se podrán añadir y remover servidores para la actualización

2.b Elegir un patrón arquitectónico

* Todo el proceso inicia al realizar merge
* Se pueden especificar múltiples servidores en donde se va a correr el build
* Se actualizarán diferentes servidores según la branch que se haya concretado el merge

Se determinó que el patrón más adecuado para la solución es el de Eventos Publish-Suscribe ya que en este caso tenemos que se van a actualizar diferentes contenedores del ecosistema dependiendo de la branch que haga merge, por lo que todos los servidores suscritos recibirán estos cambios

2.c Instanciar módulos y asignar funcionalidades usando múltiples vistas:

* Luego de la actualización de los contenedores no puede ser revertida la acción
* Github tendrá 3 líneas de ejecuciones de actualización simultaneas
* Si existe un fallo en la actualización se enviará inmediatamente un correo al lead developer

2.d Definir las interfaces de módulos secundarios

* Rollback
  + Funcionales: Debe ser posible revertir los cambios en un branch de manera manual y volver a hacer merge para revertir la actualización de los contenedores
  + De calidad: Todos los commits deben contener una descripción que sean de utilidad para revertir cambios
  + De negocio: Revertir cambios
* Logs
  + Funcionales: Se debe tener un registro de las compilaciones y actualizaciones para su verificación en caso de un fallo
  + De calidad: En caso de fallo estos logs son enviados al Lead Developer
  + De negocio: Actualización correcta de los servidores

2.e Verificar y refinar casos de uso y escenarios de calidad como restricciones para los módulos secundarios

* El proceso de actualización podrá activarlo el lead developer de manera manual para que inicie la ejecución en caso de cualquier error o rollback

## **2.2 Servidores del Sistema**

1. **CONCLUSIONES**

* E

1. **RECOMENDACIONES**

* E