NOMBRE: **Nodier Jose Pineda Villa** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Caso de negocio**:

La empresa de producción de prendas de vestir “**New experience**” está desarrollando una solución para la ampliación de sus canales de ventas, los clientes podrán *comprar los productos* mediante una plataforma virtual (Tienda virtual), en el que *seleccionarán las diferentes prendas de temporada*, agregarlas a su compra, navegar las diferentes secciones, *registrarse*, detallar sus datos para el envió de sus productos a domicilio (es posible que sea diferente para cada compra), tener una *vista previa de los productos* seleccionados y la posibilidad de *pagar en línea*. Es necesario registrarse para probar algunas funcionalidades, como por ejemplo completar los flujos de la compra.

Acuden a usted para realizar las pruebas ya que el responsable de este no está disponible. Asimismo, le entregan los diferentes criterios que se tienen que cumplir :

Ruta de la aplicación: <http://automationpractice.com/index.php>

1. **Criterios tácticos:**
2. Determine que debe probar (Para esto puede explorar la sección como guste y parta de la condición de que existen errores) para asegurar que se cumple con el mínimo de calidad.
3. Considere la creación de casos de prueba. (Considerar pasos si es necesario)
4. De estos casos, clasifique que casos son automatizables y no automatizables.
5. Indique si existen y cuáles son las precondiciones para ejecutar los casos de prueba.
6. Presente estadísticas de prueba que permitan describir la conclusión de un indicador que defina.
7. Sugiera mejoras al módulo probado.
8. **Criterios técnicos:**
9. Realice la automatización (programación) de los casos de prueba que seleccionó como automatizables. Se requiere implementar el uso de las siguientes Herramientas:
   * + Lenguaje: Java
     + Tipo de proyecto: Deseable Gradle
     + Framework de ejecución: Junit
     + Framework de pruebas: Serenity/Selenium
     + Driver o navegador: Chrome
     + Arquetipo: Page Object Model o Screenplay
     + Escribir HU en: Cucumber
     + Ejecución: Junit o Jenkins Local
10. Realice la ejecución de lo automatizado, con sus respectivas evidencias.
11. Colocar el código automatizado en **github** y enviarlo. (Repositorio público)
12. **Agilismo**
13. De las sugerencias de mejora propuestas, defina historias de usuario que plantearía para un nuevo desarrollo, con sus respectivos criterios de aceptación.
14. Teniendo en cuenta que estas historias de usuario se seleccionaron en un sprint, liste que ***tareas de prueba y de desarrollo*** son necesarias realizar estas historias.
15. **Conceptos Programación / CI / CD**

Describa con sus propias palabras cada uno de los siguientes conceptos:

1. Principios SOLID
2. Patrón Singleton
3. Patrón FIRST
4. Patrón AAA
5. Pull Request
6. Release Train
7. Quality Gates
8. Diferencias servicios SOAP / REST

¡Este es un reto! Por ende, puede consultar toda la información que sea necesaria u otra opción es preguntar las dudas que tenga a personas conocidas.

# DESARROLLO

# 

Identificación de los stakeholders, publico objetivo y nicho de mercado (lógica del negocio) Extraído de Taller práctico de testing #1 [Taller práctico de testing #1/Choucair Testing - Digital School Colombia](https://www.youtube.com/watch?v=t5AJAgG9DG4&t=4815s)

**1) Quien o quienes lo van a usar:**

Tipos de usuario:

⦁ cualquier persona (aun cuando el catálogo de compra va enfocado exclusivamente para mujeres) que tenga la necesidad de realizar compras de manera virtual. [Publico objetivo](https://drive.google.com/file/d/1ZzKq75k0YjlzzzOmPUL_wNomNkDQ4wqb/view?usp=sharing)

⦁ no existe restriccion por estrato socio-economico (no existe opcion premium) es necesario registrarse para probar algunas funcionalidades, como por ejemplo completar los flujos de la compra (precondicion).

⦁ no existe restriccion de edad

**2) Para que lo van a usar:**

los usuarios accederan a una tienda virtual de ropa en la cual tendran las opciones de:

⦁ Realizar compra y pago en linea de ropa

⦁ Ver promociones especiales de ropa

⦁ Comparar diferentes opciones de ropa segun criterios de estilo, composicion y propiedades

⦁ Explorar las diferentes opciones en un catalogo de ropa actual (catalogo de moda en stock)

⦁ Previsualización de las prendas por comprar

⦁ Compartir por redes opciones de compra [Compartir los momentos](https://drive.google.com/file/d/1z8alaQlgDqkvX7OgVM02LdOGxtXhoyOb/view?usp=sharing)

**3) Como lo van a usar:**

Ya que la plataforma virtual desarrollada es de tipo "Web Responsive" es posible utilizarla desde diferentes dispositivos como:

⦁ Dispositivos moviles (celulares y tablets).

⦁ Dispositivos de mesa (computadoras y televisores inteligentes).

los usuarios pueden acceder a la APP ingresando a la url en donde se encuentra alojada http://automationpractice.com/index.php.

**4) Que necesita para usarlo:**

⦁ La APP es compatible con diferentes sistemas operativos (probado en chrome y mozilla) y puesto que la APP es web responsive lo fundamental es tener un dispositivo con conexion a internet y un navegador web.

**5) Que es lo que mas valora:**

⦁ lo mas valioso de la aplicacion es que funciona desde diferentes dispositivos, desde diferentes sistemas operativos, es adaptable, no se requiere de una cuenta premium para acceder a todas las opciones, es posible compartir preferencias de compra, es posible realizar pagos en linea, la APP funciona de manera fluida, sin interrupciones 24/7 y de manera fluida.

**Criterios tácticos:**

⦁ Determine que debe probar (Para esto puede explorar la sección como guste y parta de la condición de que existen errores) para asegurar que se cumple con el mínimo de calidad.

los criterios que se deben probar en la APP son: (teniendo en cuenta el artículo [https://www.choucairtesting.com/Ecommerce](https://www.choucairtesting.com/aspectos-importantes-para-la-construccion-de-un-e-commerce/))

* Funcionalidad **(selección)**
* Usabilidad (no se probara pues es preferible retroalimentación del usuario final de la aplicación para mayor fiabilidad)
* Rendimiento
* Seguridad
* Compatibilidad

**Prueba de funcionalidad:**

[**Enlace a casos de prueba**](https://docs.google.com/spreadsheets/d/1Pc94yT4e-pzqibNjwRME8VcnzKbIFE7v/edit?usp=sharing&ouid=101923620441160405706&rtpof=true&sd=true) **(Excel con casos de prueba)**

Escenarios de prueba a verificar:

* **[nombre de la funcionalidad / vista a probar] + [ruta de acceso a la vista]**
* Funcionalidad del **registro de usuario** (sign in/register)
* Funcionalidad del **inicio de sesión** (sign in)
* Funcionalidad **selección articulo de catalogo** (women/[producto])
* Funcionalidad del **carro de compras** inventario y pago (cart)

***Escenarios a evaluar Alternativos:***

* Funcionalidad del campo de búsqueda (search)
* Funcionalidad contacto empresarial (contact us)

Checklist **prueba de funcionalidad** (aplica a las funcionalidades anteriormente citadas )

* Verificación: los campos no pueden estar vacíos a menos que sean opcionales
* Verificación: los campos obligatorios deben ser identificados mediante un asterisco
* Verificación: no se pueden presentar mensajes de error para campos opcionales vacios
* Verificación: en los campos numéricos no se deben aceptar caracteres alfabéticos
* Verificación: se muestra el mensaje de confirmación para operaciones de actualización y eliminación
* Verificación: las políticas de privacidad y preguntas frecuentes está bien definida

⦁ Considere la creación de casos de prueba. (Considerar pasos si es necesario)

[Enlace a Casos de prueba](https://docs.google.com/spreadsheets/d/1Pc94yT4e-pzqibNjwRME8VcnzKbIFE7v/edit?usp=sharing&ouid=101923620441160405706&rtpof=true&sd=true)

⦁ De estos casos, clasifique que casos son automatizables y no automatizables.

Casos de uso automatizables:

⦁ Indique si existen y cuáles son las precondiciones para ejecutar los casos de prueba.

[Enlace a Casos de prueba](https://docs.google.com/spreadsheets/d/1Pc94yT4e-pzqibNjwRME8VcnzKbIFE7v/edit?usp=sharing&ouid=101923620441160405706&rtpof=true&sd=true)

⦁ Presente estadísticas de prueba que permitan describir la conclusión de un indicador que defina.

⦁ Sugiera mejoras al módulo probado.

* Funcionalidad del **registro de usuario** (sign in/register)
* No es posible **eliminar una cuenta** de usuario una vez a sido dado de alta en el sistema (en caso de existir la opcion el problema seria entonces que la aplicacion en cuanto a usabilidad es poco intuitiva y dicha opción no fue encontrada).
  + crear la opción “eliminar mi cuenta” y no solo la opcion de modificar datos de la cuenta
* La interfaz de registro de usuario deberia aparecer como una opcion visible en la pagina HOME puesto que es indispensable iniciar sesion para poder realizar compras en la aplicacion (ademas está información deberia ser difundida al momento de ingresar al sitio por primera vez )

**Criterios técnicos:**

Realice la automatización (programación) de los casos de prueba que seleccionó como automatizables. Se requiere implementar el uso de las siguientes Herramientas:

⦁ Lenguaje: Java

⦁ Tipo de proyecto: Deseable Gradle

⦁ Framework de ejecución: Junit

⦁ Framework de pruebas: Serenity/Selenium

⦁ Driver o navegador: Chrome

⦁ Arquetipo: Page Object Model o Screenplay

⦁ Escribir HU en: Cucumber

⦁ Ejecución: Junit o Jenkins Local

⦁ Realice la ejecución de lo automatizado, con sus respectivas evidencias.

⦁ Colocar el código automatizado en github y enviarlo. (Repositorio público)

enlace:

**Agilismo**

⦁ De las sugerencias de mejora propuestas, defina historias de usuario que plantearía para un nuevo desarrollo, con sus respectivos criterios de aceptación.

**Sugerencias de mejora detectadas:**

* No es posible **eliminar una cuenta** de usuario una vez a sido dado de alta en el sistema (en caso de existir la opcion el problema seria entonces que la aplicacion en cuanto a usabilidad es poco intuitiva y dicha opción no fue encontrada).
* No existe la opción de **multilenguaje** por lo cual solo un usuario con conocimiento del idioma predeterminado podria interactuar con la aplicacion de manera flluida.
* No existen **iconos y elementos visuales representativos** que mejoren la experiencia in situ teniendo en cuenta que los usuarios del sitio se caracterizarian por buscar un espacio que les resulte llamativo (tienda virtual de **“*moda*”**) por lo cual resulta muy cuadriculada la navegabilidad.
* No existe la opcion de **preguntas frecuentes y politicas de privacidad**
* La funcionalidad más importante **“compra y pago en linea”** de la aplicacion requiere de minimo 8 pasos para efectuarla.
* En la interfaz **“formulario de registro”** en el areade **información residencial solicita el nombre y apellido de usuario;** información que habia sido recopilada en la parte de información personal en la misma interfaz. [Informacion repetida y en un area equivocada](https://drive.google.com/file/d/1hZTaCiFqnyjDcnTQx2HY9zk-wzM7Ei1U/view?usp=sharing)
* Ausencia de **sugerencia de contraseña** segura al momento de registrarse en el sistema (solo tiene en cuenta el tamaño de la cadena)
* Ausencia de **placeholder** para facilitar la información requerida en los campos de texto
* La **validación de campos** de texto en la **“interfaz de registro”** se realiza solo al momento de hacer submit por lo que por ejemplo el usuario puede ingresar datos erroneos como su fecha de nacimiento (**31 de febrero**) y solo se percataria cuando ya envio la información.
* La pagina no cuenta con **certificado SSL** siendo este un hueco de seguridad y un problema para el tratamiento de la información de los usuarios (lo cual es un punto clave en un sitio de e-commerce)

**Mejora encontrada y posible historia de usuario:**

* No existe la opción de **multilenguaje** por lo cual solo un usuario con conocimiento del idioma predeterminado podria interactuar con la aplicacion de manera fluida.

[Enlace a la historia de usuario](https://docs.google.com/spreadsheets/d/1OF63NK8ayaPCX3pjCArUmjncNm2uZyOy/edit?usp=sharing&ouid=101923620441160405706&rtpof=true&sd=true)

⦁ Teniendo en cuenta que estas historias de usuario se seleccionaron en un sprint, liste que tareas de prueba y de desarrollo son necesarias realizar estas historias.

**Conceptos Programación / CI / CD**

Describa con sus propias palabras cada uno de los siguientes conceptos:

⦁ Principios SOLID

Es la recopilación de una serie de principios que generan buenas practicas de programacion buscando se desarrolle codigo en el cual exista un nivel *bajo de acoplamiento* y de *alta cohesion* generando así un codigo más limpio y mantenible.

se dividen en: ***S*** (single responsability) cada modulo debe ser responsable de una unica funcion, ***O*** (open/close principle) un modulo de software debe estar abierto a extender su funcionalidad per cerrado a modificar su codigo (a menos que se intente corregir un funcionamiento errado), ***L*** (Liskov substitution principle) una clase que hereda de otra deberia poder reemplazar a su clase padre como clase base sin que ocurran problemas de funcionalidad del codigo ya que cuenta con los metodos y atributos de su sustituto, ***I*** (Interface segregation principle) se deben utilizar interfaces con responsabilidades especificas y concretas (evitar interfaces muy grandes) para evitar errores y dependencias que puedan volver el codigo poco mantenible y ***D*** (Dependency inversion principle) se delega la creacion de objetos realizando una dependencia de ellos a las clases y la instanciacion la hace un tercero mediante el uso de interfaces

⦁ Patrón Singleton

patron de tipo creacional cuyo objetivo es asegurar que una clase solo pueda ser instanciada una vez proporcionando un punto global de acceso a ella.

se usa generalmente para controlar el acceso a un recurso compartido como por ejemplo una base de datos o un archivo restringido (privacidad de la información); ya que ante su llamado se llama al objeto pero no se volvería a instanciar como clase

⦁ Patrón FIRST (¿Principio FIRST?)

son propiedades que deben cumplir las pruebas unitarias

***F*** (fast), ***I*** (independent), ***R*** (repeteable), ***S*** (Self-validating) y  ***T*** (Timely).

las pruebas unitarias deben poder realizarse de manera agil y rapida evitando que se lleguen a considerar como una tarea tediosa, dichas pruebas deben funcionar de manera independiente unas de otras sin que se produzcan efectos adversos entre ellas, ademas deben poder ejecutarse sin problemas en diferentes entornos sin que dependan de la configuracion de usuario o herramientas siempre y cuando se ejecuten en las mismas condiciones entre los diferentes ambientes, las pruebas deben ser suficientemente claras en sus resultados para determinar si han pasado con exito o ha sucedido alguna falla sin que el testeador tenga que realizar alguna otra operacion para determinar la congruencia de la prueba y estas pruebas deben ser escritas just in time (en el tiempo justo) evitando que sea demasiado tarde par realizar correcciones o muy pronto que quizas se olviden o no utilicen.

⦁ Patrón AAA [Imagen patronAAA](https://drive.google.com/file/d/1KlovCFgeCX6wfJsCBTGBVe-2R8gKFPUj/view?usp=sharing)

***A*** (Arrange), ***A*** (Act) y ***A*** (Assert)

patron de diseño que busca se generen las pruebas unitarias de una manera organizada separando las etapas de desarrollo en 3 partes bien diferenciadas **preparación** etapa en la cual se organiza y establece el ambiente de la prueba (escenario del test) objetos, variables, dependencias, entradas y mocks necesarios para hacer el llamado a la prueba; **actuacion** en está parte nos encargamos de invocar a la funcionalidad que se quiere probar con los parametros que se generaron en el “arrange”y por ultimo **comprobacion** en donde lo que se busca es verificar que el resultado obtenido en el “act” coincide con el comportamiento que se tenia previsto.

⦁ Pull Request (peticion de validacion)

Es una solicitud que se hace a un sistema de control de versiones como por ejemplo GIT para notificar que se ha realizado cambio a un codigo en común y se debe realizar una validacion de este antes de realizar un merge que permita unificar el codigo al main (principal). es basicamente una solicitud de incorporacion de cambios en un ambiente colaborativo basado en control de versiones.

⦁ Release Train

es una metodología de entrega (liberación - release) de resultados que se aplica a los equipos agiles autorganizados y autogestionados cuya finalidad es planificar entregas en tiempos de calendario publicos preestablecidos en los cuales cada miembro aporta a la entrega (generando un compromiso individual) y solo se integra a la solucion el software “just in time” (ya que es una metodología *rigida* que se adecua o ciñe al calendario).

⦁ Quality Gates

es un conjunto de condiciones / limites minimos que se establecen permitiendo asegurar la calidad del codigo por ejemplo en ambitos como mantenimiento, seguridad, duplicidad de lineas, confiabilidad, cobertura, etc; tambien son aplicables a proyecto enteros con herramientos como por ejemplo SonarCube .

⦁ Diferencias servicios SOAP / REST

Ambos son aplicables en la transmisión de datos en linea. [SOAP y REST](https://drive.google.com/file/d/1AqSH8B57quzxtDUnv-3uIaU_6WkYlJqW/view?usp=sharing)

**SOAP (protocolo simple de acceso a objetos)**

SOAP es un protocolo estándar que se creo con el fin de comunicar aplicaciones en diferentes lenguajes y plataformas dado que es un protocolo es menos flexible e impone sus reglas aún así ofrece normas integradas que son ideales para el entorno empresarial incluyendo seguridad, atomicidad, uniformidad aislamiento y durabilidad (ACID) y

**REST (transferencia de estado representacional)**

No es un protocolo sino un estilo arquitectonico, depende de la implementacion particular que le den los desarrolladores a las recomendaciones planteadas por lo cual REST es más flexible configurandose con mayor facilidad.

¡Este es un reto! Por ende, puede consultar toda la información que sea necesaria u otra opción es preguntar las dudas que tenga a personas conocidas.

guion corto