**Catalina Mejía Merino**

**Solución**:

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.

R= Se probarán funcionalidades relacionadas con el loguin, registro de usuario. Proceso de selección de productos y adicionarlos al carro de compras

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

R= Se adjunta archivo Excel separado por Hojas, donde cada una corresponde a casos de prueba para web. IOS, Android. Se relaciona en la columna “outcome” si pasa o no la prueba

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

R= En el Excel adjunto, se especifica cuáles serán automatizados

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

R= En el Excel adjunto, se especifica en la columna Precondition este ítem para cada caso de prueba

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

R= para las pruebas manuales, se adjunta un informe en Word “Informe de Avance\_Pruebas\_New Experience”. Para las automatizadas, se adjunta reporte resultante de la automatización aplicada

1. Sugiera mejoras al módulo probado.

R=

1. Implementar la app en Idioma Español

2. Las notificaciones vía e-mail deben ser en español

3. Permitir al usuario personalizar la aplicación con sus colores corporativos

De la marca

1. **Criterios técnicos:**
2. 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
3. Realice la ejecución de lo automatizado, con sus respectivas evidencias.
4. Colocar el código automatizado en **github** y enviarlo. (Repositorio público)

**// falta**

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

R= HU de las mejoras propuestas

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Historia de Usuario | Descripción | Criterios Aceptación | Tamaño |
| HU1 | COMO Usuario de la aplicación  QUIERO elegir el idioma Español PARA poder visualizar la información en este idioma además del inglés | *Se requiere Habilitar la aplicación para el idioma Español* | CA1- Todo lo que se visualice en la aplicación debe estar en Idioma español con  ortografía y gramática correctos   CA2- Si el usuario quiere volver a mirar el contenido en inglés, debe poder hacerlo | 5 |
| HU2 | COMO Usuario de la aplicación  QUIERO que las notificaciones que se generan me lleguen al correo en español PARA entender de mejor manera lo que me quieren decir por este medio | *Se requiere Habilitar notificaciones enviadas al correo en Idioma español* | CA1 - Los emails deben redactarse en español y tener la estructura  definida como se muestra en la plantilla ubicada en la documentación de la HU | 5 |
| HU3 | COMO Usuario admin de la aplicación  QUIERO poder elegir los colores  PARA que estén de acuerdo con la marca y así esté más personalizada la aplicación | *Se requiere permitir al usuario personalizar el diseño de la aplicación* | CA1- EL usuario podrá personalizar la aplicación con los colores de su marca y el módulo de configuración inicial deberá permitirle esto y sus cambios se deben ver reflejados de manera correcta | 2 |

1. 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.

R =

|  |  |  |
| --- | --- | --- |
| **Actividad** | **Etapa** | **Estado** |
| Actualización Documentación Diaria | Seguimiento y Cierre | Pendiente |
| Análisis de Eventos y estrategia | Diseño | Pendiente |
| Comités de Cambio | Ejecución | Pendiente |
| Creación del set de datos | Ejecución | Pendiente |
| Diseño de Querys para Set de datos y Smoke test | Diseño | Pendiente |
| Documentación y entrega | Seguimiento y Cierre | Pendiente |
| Elaboración plan de pruebas y cronograma | Planeación | Pendiente |
| Estimación de Tiempos | Estimación | Pendiente |
| Gestión de la prueba | Ejecución | Pendiente |
| Recepción del aplicativo y Lectura documentación | Planeación | Terminada |
| Regresión | Regresión | Pendiente |
| Reunión de aprobación | Planeación | Pendiente |
| Revisión de casos y aprobación | Diseño | Pendiente |
| Smoke Test | Ejecución | Terminada |
| Diseño de Casos de prueba | Diseño | Pendiente |
| Ejecución de casos de prueba | Ejecución | Pendiente |
| Diseño de Script para automatización | Diseño | Pendiente |

1. **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

R=

1. **Principios SOLID**: son necesarios para que se puedan crear desarrollos de software de una manera más fácil de entender por otros de manera estructurada y clara y más fáciles de darles soporte o mantenimiento ante los cambios que pueda presentar así como poder reusar el código y usar la encapsulación Se usa en la orientación a objetos cuando se agrupan funcionalidades según sus caracteristicasSe deben aplicar estos principios que son unas características que se deben cumplir, son 5: responsabilidad única, (cada clase se responsabiliza de una sola cosa)abierto o cerrado, (las entidades deben estar abiertas a extensión y cerradas para modificación, es decir, se puede heredar o extender de ellas pero no modificarlas en si porque es posible que la usen otras mas) sustitución de liskov, (es importante en la jerarquía de clases, ya que una subclase se puede reemplazar por su superclase si no se cumple, no es de calidad el software) segregación de la interfaz,(el usuario no debe depender de interfaces que no usa) inversión de dependencia (en las abstracciones deben estar ubicadas las dependencias. Los módulos de alto nivel no deberían depender de módulos de bajo nivel. Ambos deberían depender de abstracciones. Las abstracciones no deben cambiar si cambian los detalles sino que los detalles deberían depender de ellas por que en la abstracción esta la lógica
2. **Patrón Singleton:** es un patrón de diseño que indica que solo se puede tener una instancia de una clase que aplica a toda la aplicación. se usa "new", además se impone un constructor privado y un método estático. se usa con el fin de que solo haya una única instancia de una clase con una referencia única que la distinga en toda la aplicación y así evita duplicidad
3. **Patrón FIRST:** un patrón de diseño es una forma estandarizada de resolver los problemas que más se dan o se presentan cuando se desarrolla un software, se usa el patrón first para usar 5 características que se deben cumplir cuando se hacen pruebas unitarias con calidad, estas son (rapido,independiente, repetible, auto evaluable y oportuno)si se aplican de manera correcta evita errores en producción
4. **Patrón AAA:** es una buena práctica para las pruebas unitarias que se consiste en repartir una prueba unitaria en 3 partes, Arrange, Act, Assert. (AAA)
   * Arrange organiza o inicializa los objetos y hace que se den valores a los datos.
   * Act, llama al método a probar teniendo ya listos los parámetros
   * Assert, es la confirmación, de que lo que se probó está bien
5. **Pull Request:** Es una acción que realiza un desarrollador de software en el momento que acaba de creary hacer pruebas a una funcionalidad o cuando da solución a un error entonces debe pedir que lo que hizo, se sume al repositorio principal que se tenga en el proyecto Entonces los demás desarrolladores pueden revisar el cambio en el software para decir cómo les parece y si les parece bien para aceptarlo o dar sus comentarios para rechazarlo porque el software se hace en equipo
6. **Release Train:** es un equipo multifuncional con muchas capacidades para dar soluciones que trabaja en grupo apuntando a metas iguales, que sean reales y que se basan en ciertos valores o características. También se conoce como art agile release train, que es efectivo al momento de entregar resultados y operar en equipo. se unen bajo una misma misión que se despliega en conjunto entregando beneficios a los usuarios
7. **Quality Gates:** es una app que le da a las empresas la opción de controlar y revisar la

Integridad de la calidad de datos para que de una forma continua y eficaz efectúe la la calidad de la información cuando por ejemplo trabaja con procesos ETL en conjunto con datos

1. **Diferencias servicios SOAP / REST: SOAP** (Simple Object Access Protocol); REST (Representational State Transfer). ambos envían y reciben datos o mensajes en línea con API pero la diferencia es que Soap es un protocolo, rest no lo es, rest es para transferencia de estado y expone las api públicas para mejorar el crud de los datos,

Accede a recursos con una interfaz y soap es solo de acceso. REst es más flexible, no define tantos estándares como soap que es más especifico