INDEX

[1. Modelo de Arquitectura general de pruebas automatizadas 3](#_Toc150249431)

[2. Proceso de automatización de pruebas 4](#_Toc150249432)

[2.1 Procesos generales 4](#_Toc150249433)

[2.2 Tipo de Automatización 5](#_Toc150249434)

[2.3 Herramientas 5](#_Toc150249435)

[2.4 Mantenibilidad 5](#_Toc150249436)

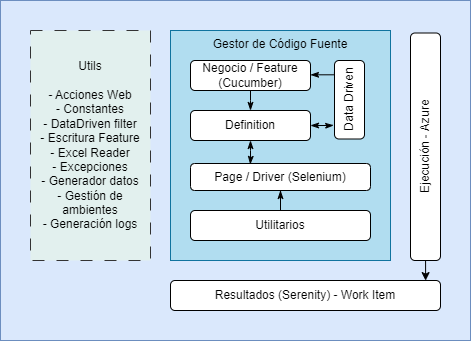
[3. Obtención de data 6](#_Toc150249437)

[4. Manejo de control de versiones 6](#_Toc150249438)

[5. Buenas prácticas automatización de pruebas 7](#_Toc150249439)

[6. Buenas prácticas para creación de casos automatizados 8](#_Toc150249440)

# Modelo de Arquitectura general de pruebas automatizadas



**Capa de Negocio:** Esta capa incluye todo lo que está relacionado con las reglas de negocio, la descripción de casos y/o escenarios de prueba.

Los componentes de esta capa se utilizan para editar y navegar por las estructuras de la suite de pruebas, relacionar casos de prueba con objetivos de pruebas (criterios de aceptación) o requisitos del SUT (Sistema Bajo Prueba), documentar en diseño de la prueba.

En esta capa se tiene en cuenta aspectos como:

* Escenarios de prueba en formato Cucumber (como, cuando, entonces).
* Desarrolla, capturar o derivar los datos para la prueba.
* Documentar el diseño de la prueba, es decir, cada caso de prueba o escenario.

**Capa de Definición:** Esta capa comprende el desarrollo de las definiciones para los escenarios o casos de prueba descritos en la capa de negocio, los cuales se pueden describir en definiciones y pasos.

Algunos aspectos para tener en cuenta:

* Especificar métodos de prueba para un caso de prueba o conjunto de casos de prueba.
* Definición de scripts de prueba para la ejecución de los casos de prueba.
* Especificar secuencias de prueba o comportamientos de prueba completos.
* Es conveniente documentar los datos de prueba, casos de prueba y/o los métodos de la prueba.

**Capa Page/Driver**: Esta capa define la iteración entre el SUT (Sistema Bajo Prueba) con las acciones a ejecutar en cada uno de los elementos objeto que son parte de las pruebas. En resumen, es el que maneja los recursos propios de la herramienta de automatización que posibilitan esta interacción.

**Capa de Datos (Data Driven):** Esta capa define la interacción con diferentes fuentes de datos que se tengan, con el fin de que los datos sean canalizados a las capas que corresponden y que a su vez se mantengan un mismo patrón resultado. Independiente de la fuente esta deberá ser consumido de una misma manera en la automatización. En este caso se hace consumo de los datos de una fuente externa al código fuente, pero presente en el proyecto.

**Ejecución:** Esta capa consta de herramientas de soporte que están en capacidad de ejecutar casos de prueba automáticamente (en el caso de que se hayan construido, asegurarse de la disponibilidad de ambientes, notificar que la ejecución fue exitosa o tuvo algún error).

En esta capa lo importante para tener en cuenta es:

• Se ejecutan los casos de prueba automáticamente vía consola de comandos.

• Se generan los reportes de los resultados de la ejecución de los casos de prueba

**Resultados:** Los resultados corresponden al informe de pruebas y las evidencias que soportan las mismas.

Lo resultados se pueden ver de diferentes formas:

* Informes tipo HTML: Este tipo de informe de Serenity se publica cómo un link en la ruta interna del proyecto correspondiente y presenta los Screenshots, steps, tiempos, resultados específicos e informe de resultados resumen.
* Logs de resultados: Generados de manera personalizada pro el robot de pruebas. Están en formato de texto, TXT o PROPERTIES.

# Proceso de automatización de pruebas

## Procesos generales

El proceso base de Automatización de pruebas comprende desde la implementación hasta la operación de esta.

|  |  |  |
| --- | --- | --- |
| **Grupo / Capa / Criterio** | **Criterio de segundo nivel** | **Lineamiento** |
| Proceso general | Necesidad de Automatizar | Se debe iniciar el proceso con una necesidad de Automatización identificada. |
| Proceso general | Validación de Aplicabilidad | Se debe realizar una validación de aplicabilidad utilizando los lineamientos de Aplicabilidad descritos. |
| Proceso general | Validación flujo de negocio | Se debe determinar si el flujo funcional de negocio es automatizable. |
| Proceso general | Validación Técnica | Se debe determinar si la(s) tecnología(s) a automatizar dentro del flujo de negocio tiene soporte con herramienta de Automatización. |
| Proceso general | Costo/Beneficio | Se debe realizar el análisis de Costo Beneficio de la Automatización. |
| Proceso general | Gestión de Pruebas (Test Management) | Se debe crear y agrupar el modelado de negocio en archivos extensión feature |
| Proceso general | Modelado de Negocio (Business) | Se debe crear el modelado de negocio de acuerdo con los lineamientos establecidos en la Arquitectura de pruebas automatizadas |
| Proceso general | Modelado de Negocio (Data Driven) | Se debe desarrollar el modelado de datos de acuerdo con los lineamientos establecidos en la Arquitectura de pruebas automatizadas |
| Proceso general | Modelado Técnico (Definición) | Se debe desarrollar el modelado técnico en definición de acuerdo con los lineamientos establecidos en la Arquitectura de pruebas automatizadas |
| Proceso general | Modelado Técnico (Driver) | Se debe desarrollar el modelado técnico en controlador de acuerdo con los lineamientos establecidos en la Arquitectura de pruebas automatizadas |
| Proceso general | Ejecución | La ejecución se debe establecer de acuerdo con la necesidad de implementación haciendo uso de las opciones establecidas en el lineamiento de la Arquitectura de pruebas automatizadas. |
| Proceso general | Resultados | Los resultados de ejecución de pruebas deben contener el resumen y las evidencias de soporte definida en los pasos anteriores del proceso. |

## Tipo de Automatización

Como parte de la cobertura y planteamiento base de la estrategia de calidad y la aplicación de los estándares, se crea una plantilla base para una automatización de aplicaciones Web de tipo FrontEnd, automatizando pruebas empleando la UI del aplicativo.

El tipo de Automatización FrontEnd es a nivel de interfaz gráfica la que simula el comportamiento de un usuario haciendo uso del sistema, esta se correlaciona al nivel de Automatización de UI.

## Herramientas

|  |  |  |
| --- | --- | --- |
| **Area** | **Herramienta** | **Administración / Configuración** |
| Repositorios | Azure | Cliente Ecopetrol |
| Lenguaje de Programación | Java | NTT DATA |
| Constructores | Maven | NTT DATA |
| Entorno integrado de Desarrollo (IDE) | Eclipse | NTT DATA |
| Lenguajes | Gherkin | NTT DATA |
| Framework Automatización | Cucumber | NTT DATA |
| Framework Automatización | Junit | NTT DATA |
| Framework Automatización | Selenium | NTT DATA |
| Reporteria | Serenity | NTT DATA |

## Mantenibilidad

El código dispuesto es una base pensada para ser incremental y complementada de acuerdo con la necesidad en cada proyecto en el que sea empleado. Se encuentran métodos disponibles que facilitan el uso y configuración de cada proyecto, aparte de permitir iniciar de una forma rápida con las herramientas preconfiguradas.

Se debe tener en cuenta la configuración base del proyecto, gestión de ambientes, apuntamientos, mapeo de localizadores de la UI, además crear los escenarios con los tags de Gherkin y su asociación a la capa de definition.

# Obtención de data

Los datos serán obtenidos de un archivo DataDriven en formato .xlsx (Excel), donde cada hoja de este archivo corresponderá a un caso automatizado construido, estas hojas deben llevar el nombre correspondiente al escenario/tag asociado al feature correspondiente. Esta información será consultada y cargada en un mapa de datos (HashMap), como primer paso definido en los features correspondientes y que al hacer la trazabilidad hasta la capa PageObject, se puede encontrar el método cargarDatos() donde se realiza la interacción con el DataDriven.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

# Manejo de control de versiones

Sistema de control de revisiones distribuido, Cada directorio de trabajo git es un repositorio completamente funcional y con total capacidad independientemente.

Su principal propósito es que se pueda regresar a un estado anterior del proyecto o conocer, incluso, toda su evolución en el tiempo, desde su inicio hasta donde la actualidad.

Gráfico, Diagrama, Gráfico de burbujas

Descripción generada automáticamente

# Buenas prácticas automatización de pruebas

Los principios SOLID fueron introducidos por Robert C Martin, estas guías buscan crear código limpio que sea mantenible, reutilizable, legible entre otros aspectos en una solución de software e indican que:

* S: Principio de responsabilidad única (Single responsibility principle)
* O: Principio de abierto/cerrado (Open/closed principle)
* L: Principio de sustitución de Liskov (Liskov substitution principle)
* I: Principio de segregación de la interfaz (Interface segregation principle)
* D: Principio de inversión de la dependencia (Dependency inversion principle)

Las **CLASES** deben ser sustantivos y en singular con la primera letra en mayúscula y sus siguientes palabras todas con la primera letra en mayúscula también:

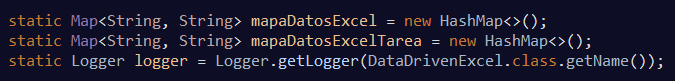
* **Uso Correcto**: MiClase, Vehiculo, FormularioRetiro, ConexionBaseDatos
* **Uso Incorrecto:** miClase, vehiculo, formularioregistro, Conexionbasedatos

Texto

Descripción generada automáticamente

Las **VARIABLES** también deben ser escritas en notación lowerCamelCase, esto quiere decir, que la primera letra siempre debe iniciar en minúscula, mientras que las demás palabras siguientes la primera letra en mayúscula:

* **Uso Correcto:** monto, esAfiliado, personaNaturalPensionado
* **Uso Incorrecto:** Monto, Esafiliado, PersonanaturalPensionado



Las CONSTANTES se deben escribir con letras mayúsculas, separando las palabras con raya al piso:

* **Uso Correcto:** NUMERO\_HIJOS, NAVEGADOR, PERSONAS\_POR\_DEPARTAMENTO
* **Uso Incorrecto:** numero\_hijos, Navegador, Personas\_por\_Departamento

Texto

Descripción generada automáticamente

# Buenas prácticas para creación de casos automatizados

**Behavior-Driven Development** o **BDD** fue el origen de la creación de Dan North en un conjunto de prácticas o procesos de ingeniería de software diseñados para crear impacto de negocio. Se basa en Ágil y en prácticas LEAN; sin embargo, el logro más importante generado por BDD es proveer un lenguaje (Gherkin) que facilita la comunicación entre los miembros del proyecto y stakeholders.

Para escribir los escenarios en dicho lenguaje y a su vez hacer uso de ellos cómo insumo en las pruebas automatizadas, se podría usar una herramienta cómo Cucumber; Adicionalmente, se van a necesitar estos archivos:

**Feature File**:Los archivos de características son aquellos donde los analistas de negocio (BA) almacenan los requisitos y pueden crear el puente entre los requisitos y las pruebas automatizadas. Deben estar escritos en lenguaje Gherkin, el cual es un lenguaje simple que se utiliza para explicar el comportamiento esperado de un sistema. Admite escritura en cualquier idioma y se define en ellos los datos de prueba.

Es importante que exista un Feature por cada historia de usuario generada, esto facilita la comprensión del robot y la mantenibilidad en el futuro.

**Creación de una Feature**: La creación de una Feature se deberá realizar en el robot de pruebas; en la carpeta pertinente del folder “resources” para uso posterior de los scripts automatizados.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Escenarios:** Los escenarios de prueba describen un ejemplo de comportamiento del sistema, con condiciones de aceptación de una historia de usuario. Es importante aclarar que la buena práctica es especificar qué debe hacer el sistema y no cómo debe hacerlo.

El **escenario** contiene una lista de pasos que comienza con algunas de estas palabras claves:

* + **Given** (|Dado|Dada|Dados|Dadas)
  + **When** (Cuando)
  + **Then** (Entonces)
  + **And** (Y)

**Given**: Es usado para presentarnos los actores (y su entorno) de un escenario; Siempre va en el comienzo de un escenario,

**When**: Representa la acción que ejecuta el usuario; Aunque se pueden concatenar varios When no es recomendable, para ello existe la palabra clave And**.**

**Then:** Es usado para verificar el resultado esperado después de ejecutarse la acción (When); Siempre va en el final de un escenario y para concatenarlos se usa And.

