# Pruebas funcionales automatizadas para aplicaciones Web: usando Selenium para aplicar pruebas de regresión automatizadas

**Autor**: Adriano Tobías Vega Llobell
**Tutores**: Patricio Orlando Letelier Torres
María Carmen Penadés Gramage
Curso: 2017 - 2018

UNIVERSITAT POLITÈCNICA DE VALÈNCIA
Escola Tècnica Superior d'Enginyeria Informàtica
etsinf

# Índice

# 1. Introducción

# 1. Introducción

- **Pruebas de regresión**: Pruebas para comprobar si un cambio ha afectado negativamente al comportamiento existente.
    - Nos ayudan a ganar confianza antes del lanzamiento de una versión.
    - Las pruebas manuales de regresión en programas con cambios frecuentes pueden resultar ineficientes.
    - Su automatización puede resultar interesante.

# 1. Introducción – Motivación

- Desarrollado en el contexto de unas prácticas de empresa.
- Surge la necesidad de establecer un proceso de pruebas automatizadas de regresión para las aplicaciones web de la empresa.

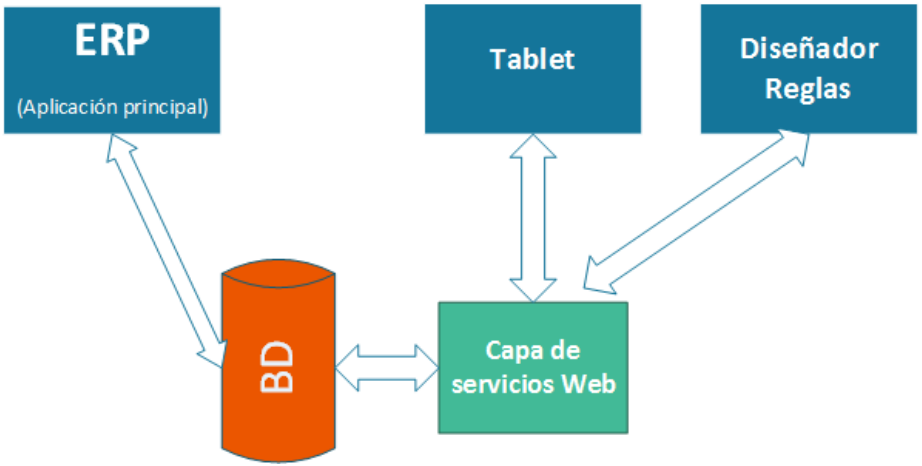
# 1. Introducción - Objetivos

- Establecer un proceso de pruebas automatizadas para aplicaciones web que cubra el ciclo completo:
  - Diseño y automatización de la prueba.
  - Ejecución de la batería de pruebas.
  - Revisión de los resultados.
- Seleccionar o desarrollar herramientas que den soporte a este proceso.
- Desarrollar una batería de pruebas funcionales para una aplicación de la empresa.

# 2. Estrategia de pruebas

# 2. Estrategia de pruebas - Contexto

- Se definió una estrategia de pruebas para una de las aplicaciones web.
    - Objetivo: **identificar necesidades del proceso y la herramienta**.
- La aplicación web a probar es una aplicación complementaria a un ERP.



# 2. Estrategia de pruebas

- Se detectaron las siguientes necesidades:
  - **Automatización de pruebas funcionales:**
    - Contar con una batería de pruebas de regresión.
    - Detectar fallos provocados por cambios en la aplicación web.
    - Controlar que cambios en el ERP no afecten negativamente a su contraparte web.
  - **Integración con la infraestructura y proceso de pruebas del ERP.**

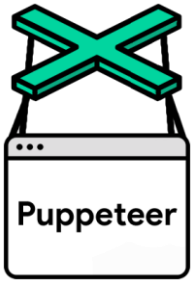
# 2. Estrategia de pruebas

- Se detectaron los siguientes requisitos:
  - **Ejecución de pruebas crossbrowser**:
    - Google Chrome y Microsoft Edge.
  - **Ejecución de pruebas *responsive***:
    - Resolución Desktop: 1920x1080.
    - Resolución Tablet: 768x1024.

# 3. Estudio de las herramientas

# 3. Estudio de las herramientas

- Se compararon 3 herramientas distintas mediante la automatización de una prueba de ejemplo:




Puppeteer

Katalon Studio


Selenium

# 3. Estudio de las herramientas – Resumen

| | Puppeteer | Katalon Studio | Selenium |
|---|---|---|---|
| **Soporte para todas las acciones básicas de la prueba de ejemplo.** | Sí | Sí | Sí |
| **Ejecución de las pruebas en múltiples resoluciones para pruebas responsive.** | Sí | Sí | Sí |
| **Ejecución de pruebas en distintos navegadores.** | **No. Solo soporta Chrome.** | Sí | Sí |
| **Posibilidad de integrar con la infraestructura existente de la empresa.** | Sí | **No** | Sí |

# 4. Propuesta basada en Selenium

# 4. Propuesta basada en Selenium – Tecnología usada

- Solución basada en .NET Framework, usando el lenguaje C#.
    - Elegida debido a que la infraestructura existente para pruebas utiliza esta plataforma.
- Bibliotecas desarrolladas internamente: restauración de *scripts* SQL.
- Bibliotecas de Selenium WebDriver.

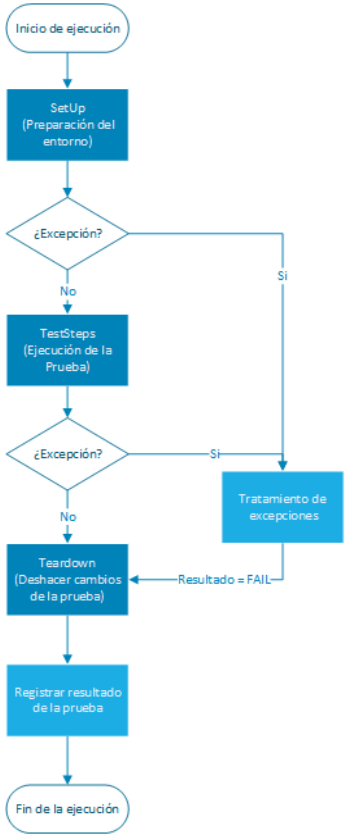
# 4. Propuesta basada en Selenium

- Objetivo prioritario: Establecer cómo se organizará el código de las pruebas.
- Reutilizar el máximo código posible: será esencial cuando contemos con un gran número de pruebas automatizadas.
- Esto derivó en el diseño de dos clases fundamentales reutilizables: *Test* y *Page Object*.
    - Estas clases condicionaron la arquitectura de la herramienta.

# 4. Propuesta basada en Selenium

**Clase Test**: clase genérica y base para la implementación de todas las pruebas.

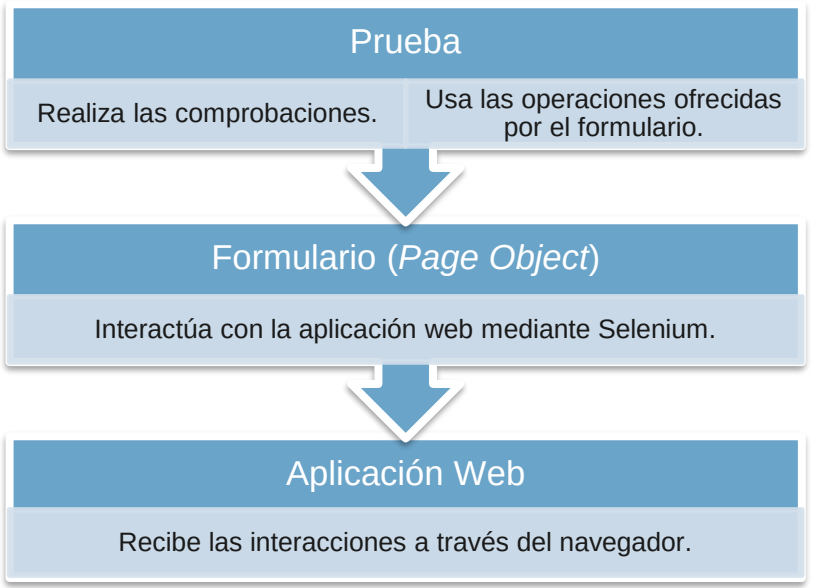
- Una prueba está compuesta por tres fases distintas:
  - ***SetUp***: preparación de datos y entorno.
  - ***TestSteps***: ejecución de la prueba.
  - ***Teardown***: restauración del entorno.



# 4. Propuesta basada en Selenium

**Clase *Page Object***: Clase para la implementación de un formulario de la aplicación web.

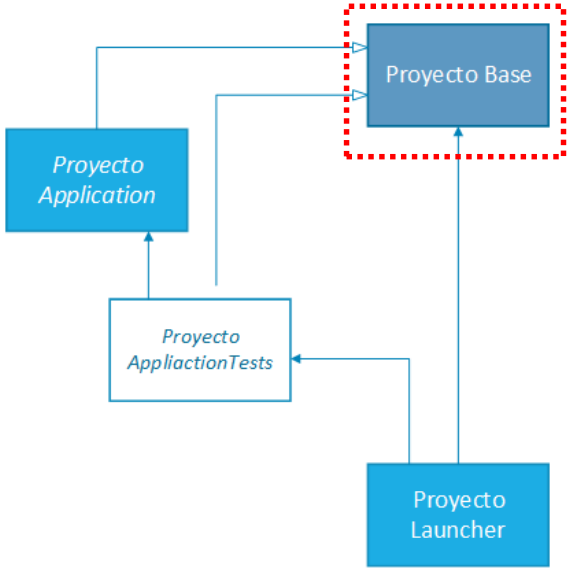
- Ofrece a las pruebas las operaciones para interactuar con un formulario.
  - Reutilización de código.
  - Abstrae a las pruebas de las interacciones con la aplicación.
  - Centraliza la lógica de interacción facilitando los cambios.



# 4. Propuesta basada en Selenium – Arquitectura

- El uso de las clases *Page Object* y *Test* derivó en la siguiente arquitectura:
    - **Proyecto base**: contiene utilidades básicas para todos los proyectos. Incluye las clases *Test* y *Page Object*.

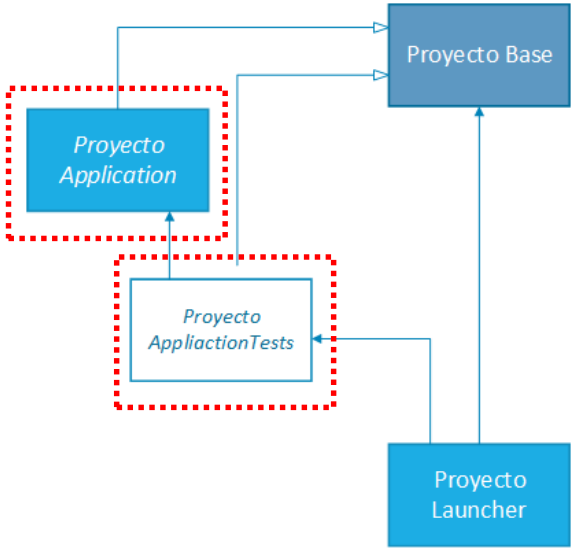
      Todos los proyectos dependen de él.



# 4. Propuesta basada en Selenium – Arquitectura

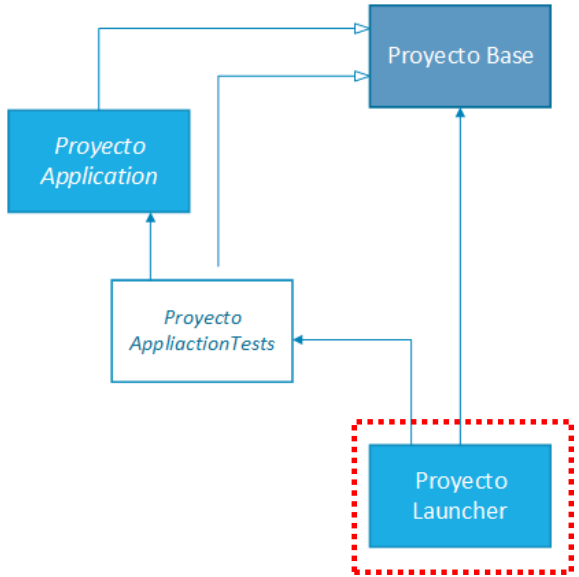
- El uso de las clases *Page Object* y *Test* derivó en la siguiente arquitectura:
  - **Proyecto *Application***: contiene todos los Page Object de los formularios de la aplicación. Este proyecto es el que hará de interfaz entre la prueba y la aplicación en sí.
  - **Proyecto *ApplicationTests***: contiene las pruebas de una aplicación determinada. Va siempre asociado a un proyecto *Application*.

Se creará uno de cada, para cada aplicación distinta que se quiera probar.



# 4. Propuesta basada en Selenium – Arquitectura

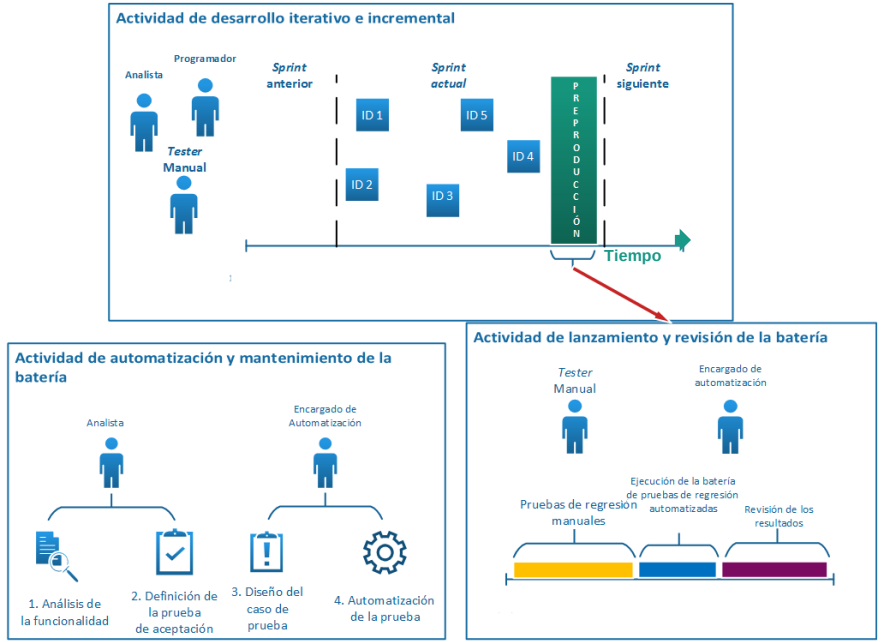
- El uso de las clases *Page Object* y *Test* derivó en la siguiente arquitectura:
  - **Proyecto *Launcher***: encargado de gestionar la ejecución de las pruebas.
  - Recibe los parámetros de la ejecución para las baterías de pruebas, como:
    - La lista de pruebas a ejecutar.
    - Los navegadores.
    - Las resoluciones.



# 5. Proceso de pruebas automatizadas para aplicaciones web

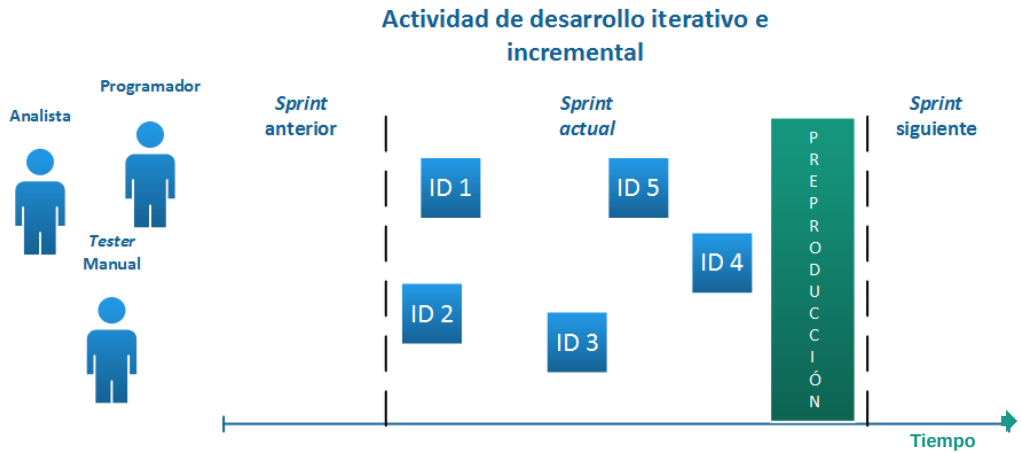
# 5. Proceso de pruebas automatizadas para aplicaciones web

- Una vez establecida la herramienta basada en Selenium, se procedió a definir el proceso de pruebas.
- Compuesto por tres elementos fundamentales:
  - Desarrollo iterativo e incremental.
  - Automatización y mantenimiento de la batería de pruebas.
  - Lanzamiento y revisión de la batería.



# 5. Proceso de pruebas automatizadas para aplicaciones web

- **Desarrollo iterativo e incremental**: Es la actividad principal del ciclo de desarrollo.
  - Un *sprint* se corresponde con una versión del producto.
  - Las unidades de trabajo son incidencias.
  - Cada incidencia tiene sus propias actividades de análisis, diseño, implementación y pruebas manuales.
  - Al final del sprint hay un **periodo de preproducción**, donde tendrán lugar las pruebas manuales y automatizadas.



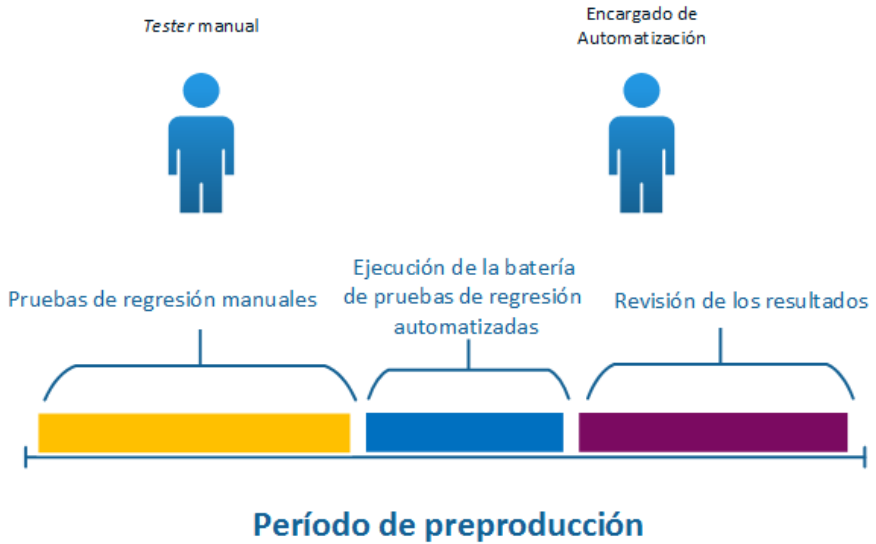
# 5. Proceso de pruebas automatizadas para aplicaciones web

- **Automatización y mantenimiento de la batería:** Tiene lugar en paralelo con la actividad de desarrollo iterativo e incremental. Los encargados de automatización se dedicarán a:
  - Diseñar e implementar más pruebas.
  - A mantener la batería, reparando pruebas que estén desactualizadas.


Proceso de automatización de pruebas

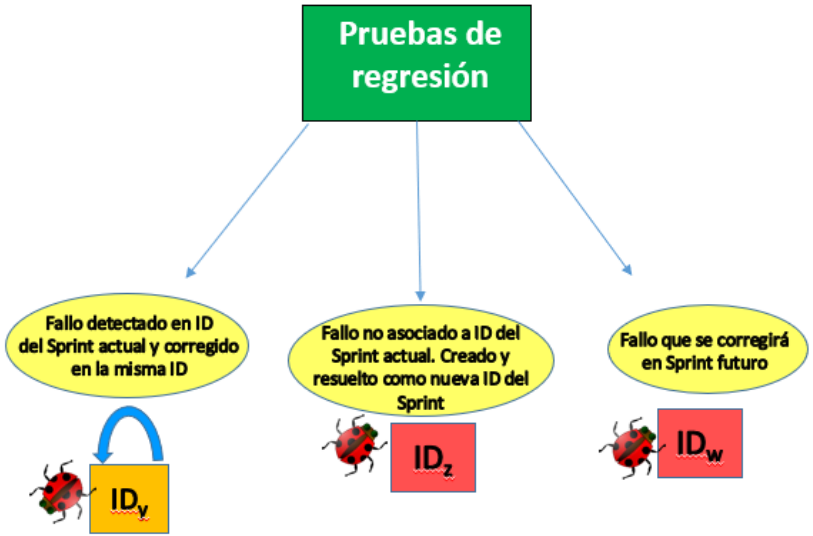
# 5. Proceso de pruebas automatizadas para aplicaciones web

- **Lanzamiento y revisión de la batería**: durante el periodo de preproducción de la actividad de desarrollo, se lanza la batería de pruebas.
  - Los *testers* manuales deben terminar antes sus pruebas.
  - Se procede a revisar los resultados de las pruebas.
    - Fallos de entorno, fallos de automatización o fallos de aplicación.

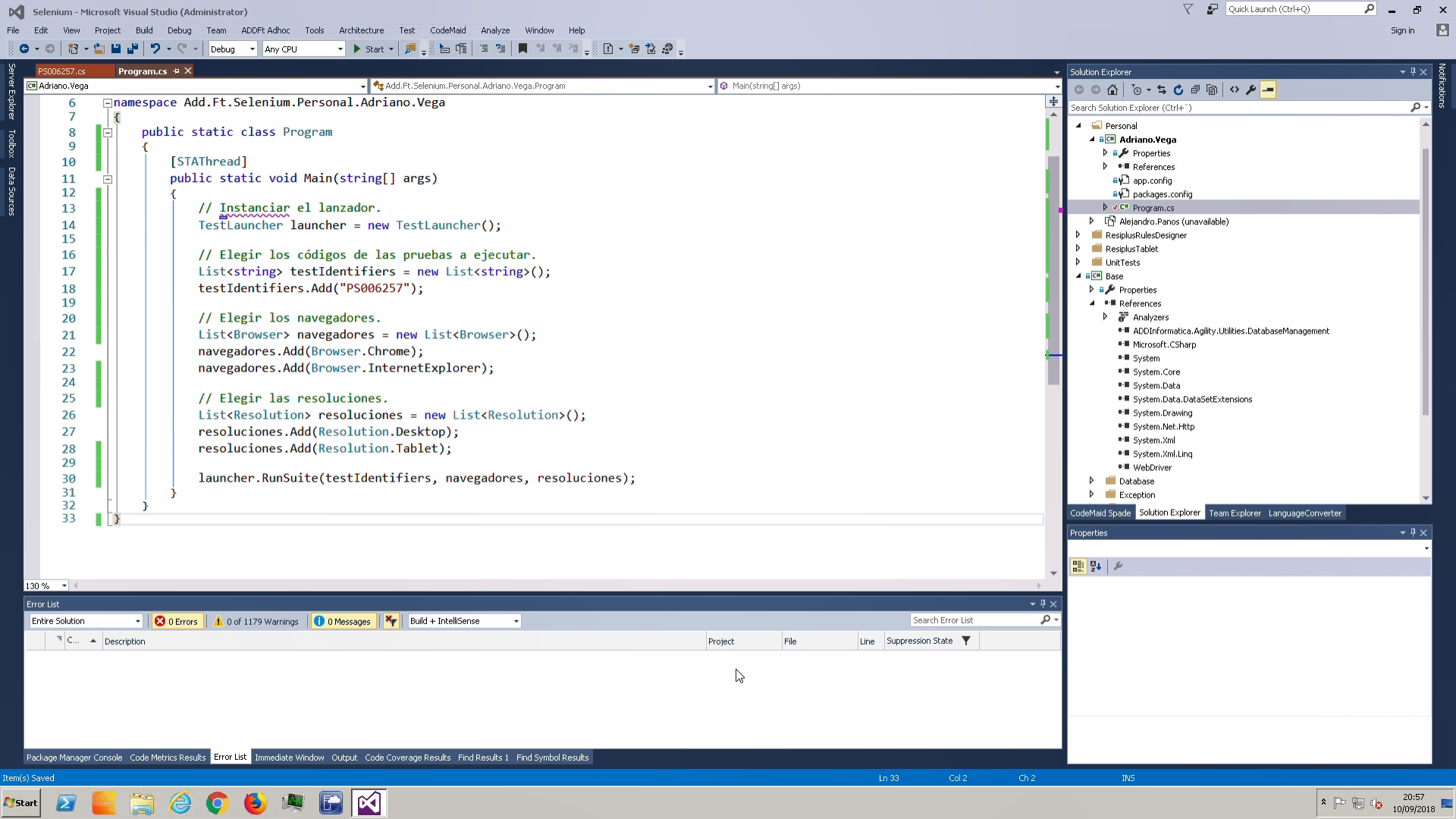


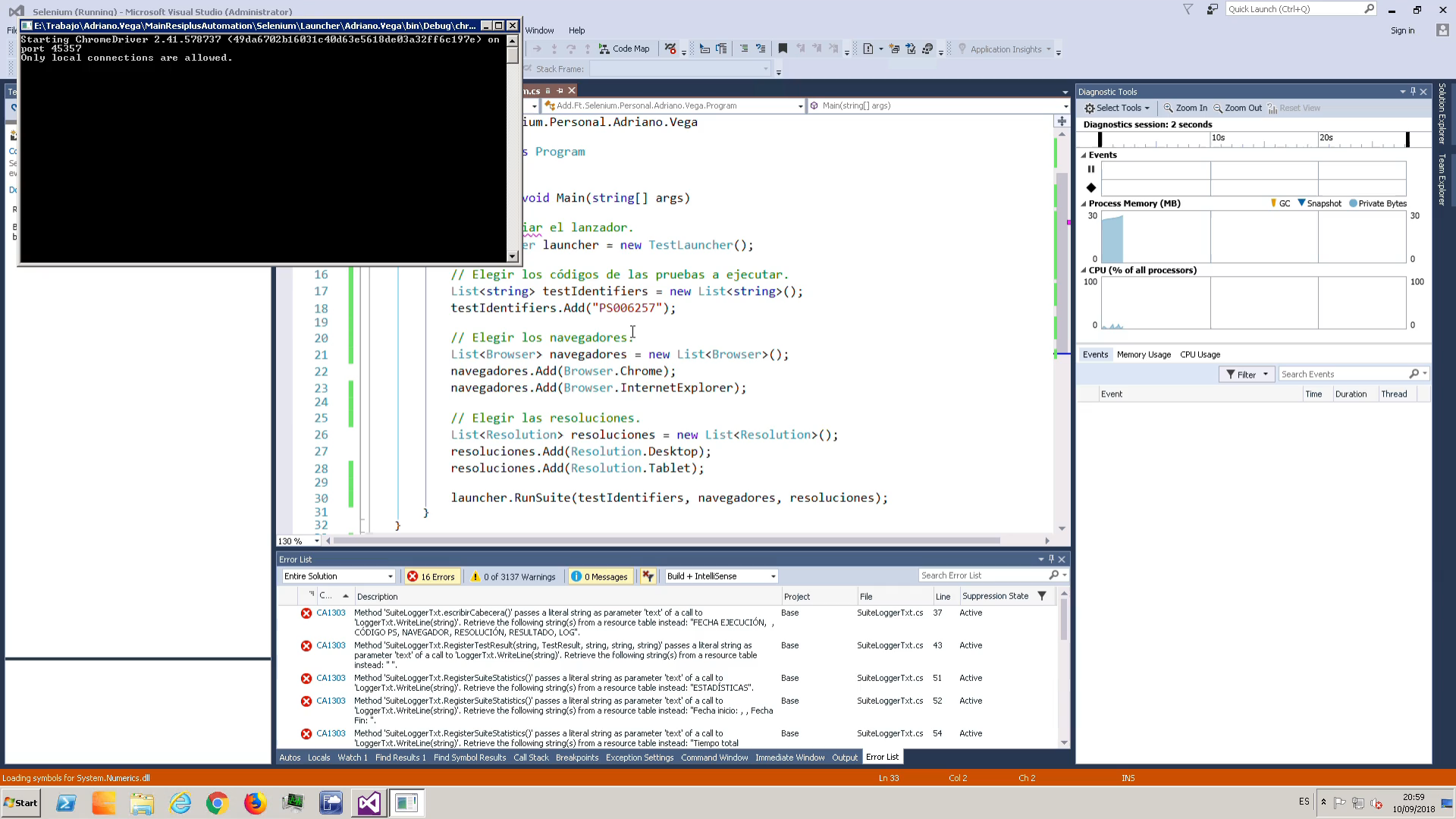
# 5. Proceso de pruebas automatizadas para aplicaciones web

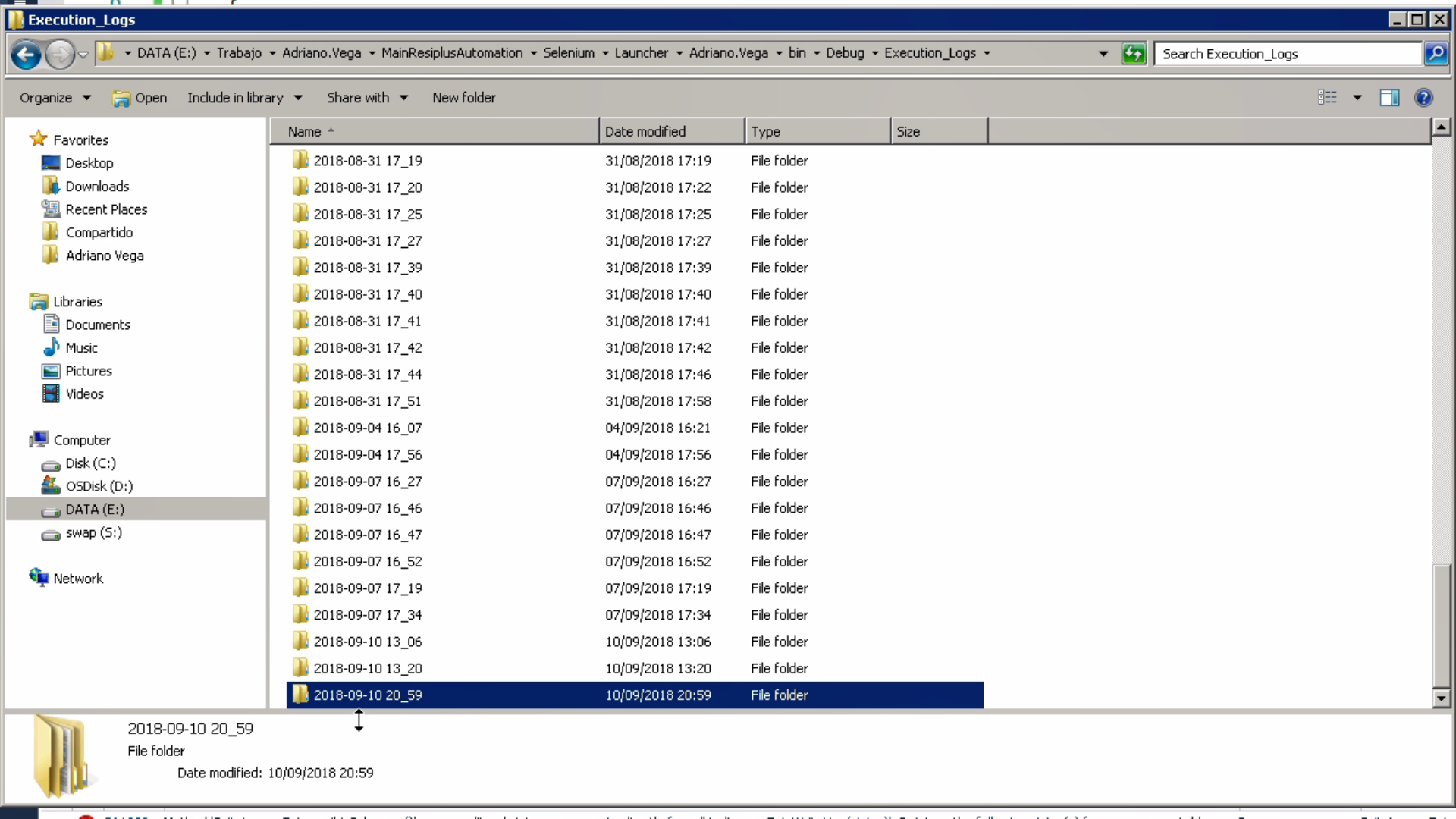
- **Gestión de los fallos:** se aplica al detectar un fallo en pruebas de regresión.
    - La forma de actuar dependerá de:
        - La severidad del fallo
        - El momento de detección
        - La dificultad de su solución.

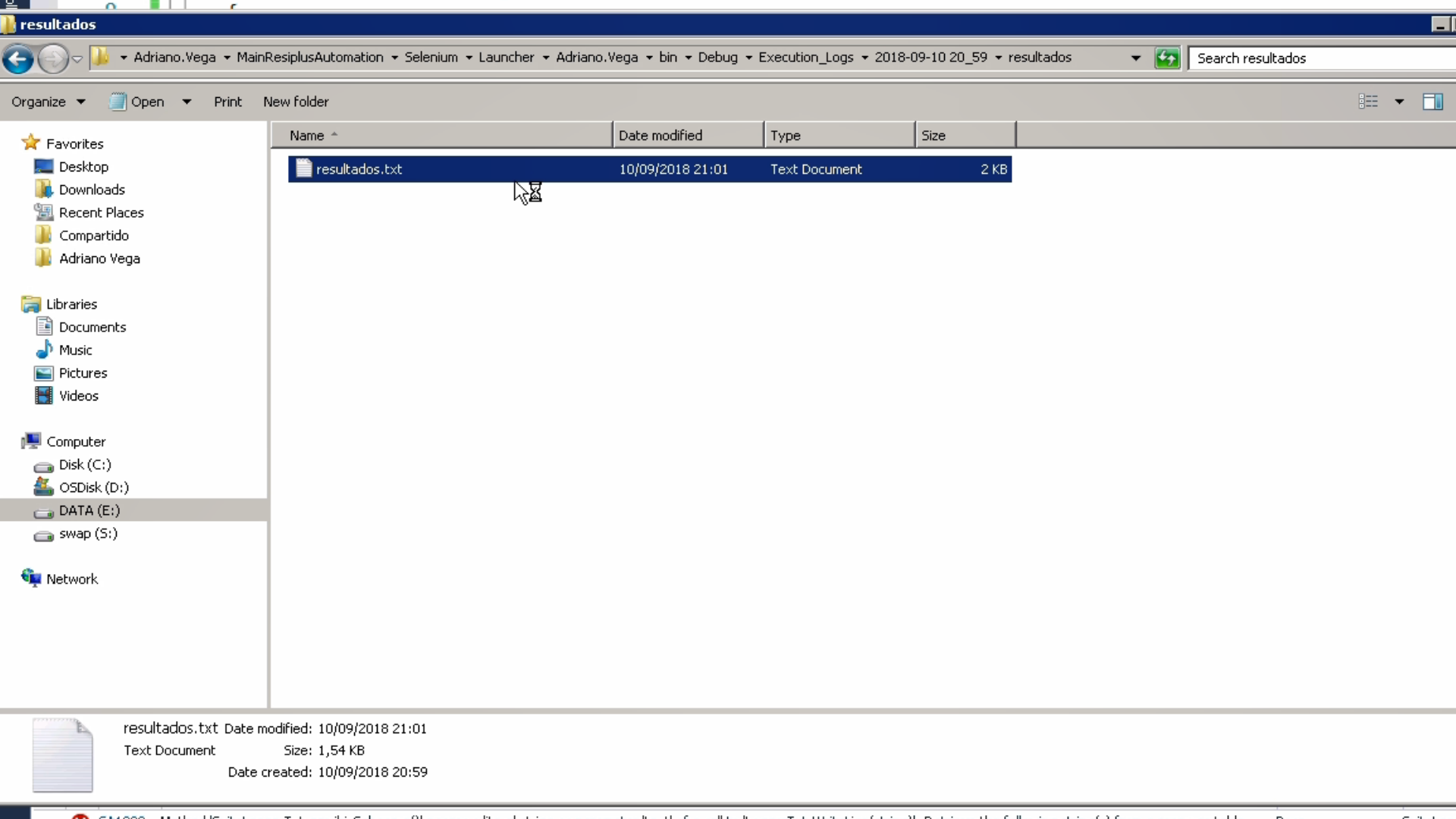


# 6. Demostración


Selenium - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Team ADDFt Adhoc Tools Architecture Test CodeMaid Analyze Window Help
Debug
Any CPU
Start
PS006257.cs
Program.cs
Adriano.Vega
Add.Ft.Selenium.Personal.Adriano.Vega.Program
Main(string[] args)
namespace Add.Ft.Selenium.Personal.Adriano.Vega
{
public static class Program
{
[STAThread]
public static void Main(string[] args)
{
// Instanciar el lanzador.
TestLauncher launcher = new TestLauncher();
// Elegir los códigos de las pruebas a ejecutar.
List<string> testIdentifiers = new List<string>();
testIdentifiers.Add("PS006257");
// Elegir los navegadores.
List<Browser> navegadores = new List<Browser>();
navegadores.Add(Browser.Chrome);
navegadores.Add(Browser.InternetExplorer);
// Elegir las resoluciones.
List<Resolution> resoluciones = new List<Resolution>();
resoluciones.Add(Resolution.Desktop);
resoluciones.Add(Resolution.Tablet);
launcher.RunSuite(testIdentifiers, navegadores, resoluciones);
}
}
}
130 %
Solution Explorer
Search Solution Explorer (Ctrl+;)
Personal
Adriano.Vega
Properties
References
app.config
packages.config
Program.cs
Alejandro.Panos (unavailable)
ResiplusRulesDesigner
ResiplusTablet
UnitTests
Base
Properties
References
Analyzers
ADDInformatica.Agility.Utilities.DatabaseManagement
Microsoft.CSharp
System
System.Core
System.Data
System.Data.DataSetExtensions
System.Drawing
System.Net.Http
System.Xml
System.Xml.Linq
WebDriver
Database
Exception
CodeMaid Spade
Solution Explorer
Team Explorer
LanguageConverter
Properties
Error List
Entire Solution
0 Errors
0 of 1179 Warnings
0 Messages
Build + IntelliSense
Search Error List
Description
Project
File
Line
Suppression State
Package Manager Console
Code Metrics Results
Error List
Immediate Window
Output
Code Coverage Results
Find Results 1
Find Symbol Results
Item(s) Saved
Ln 33
Col 2
Ch 2
INS
20:57
10/09/2018


Selenium (Running) - Microsoft Visual Studio (Administrator)
E:\Trabajo\Adriano.Vega\MainResiplusAutomation\Selenium\Launcher\Adriano.Vega\bin\Debug\chr...
Starting ChromeDriver 2.41.578737 (49da6702b16031c40d63e5618de03a32ff6c197e) on port 45357
Only local connections are allowed.
Add.Ft.Selenium.Personal.Adriano.Vega.Program
Main(string[] args)
// Elegir los códigos de las pruebas a ejecutar.
List<string> testIdentifiers = new List<string>();
testIdentifiers.Add("PS006257");
// Elegir los navegadores.
List<Browser> navegadores = new List<Browser>();
navegadores.Add(Browser.Chrome);
navegadores.Add(Browser.InternetExplorer);
// Elegir las resoluciones.
List<Resolution> resoluciones = new List<Resolution>();
resoluciones.Add(Resolution.Desktop);
resoluciones.Add(Resolution.Tablet);
launcher.RunSuite(testIdentifiers, navegadores, resoluciones);
Diagnostic Tools
Diagnostics session: 2 seconds
Events
Process Memory (MB)
CPU (% of all processors)
Error List
Entire Solution
16 Errors
0 of 3137 Warnings
0 Messages
Build + IntelliSense
CA1303 Method 'SuiteLoggerTxt.escribirCabecera()' passes a literal string as parameter 'text' of a call to 'LoggerTxt.WriteLine(string)'. Retrieve the following string(s) from a resource table instead: "FECHA EJECUCIÓN, , CÓDIGO PS, NAVEGADOR, RESOLUCIÓN, RESULTADO, LOG".
CA1303 Method 'SuiteLoggerTxt.RegisterTestResult(string, TestResult, string, string, string)' passes a literal string as parameter 'text' of a call to 'LoggerTxt.WriteLine(string)'. Retrieve the following string(s) from a resource table instead: " ".
CA1303 Method 'SuiteLoggerTxt.RegisterSuiteStatistics()' passes a literal string as parameter 'text' of a call to 'LoggerTxt.WriteLine(string)'. Retrieve the following string(s) from a resource table instead: "ESTADÍSTICAS".
CA1303 Method 'SuiteLoggerTxt.RegisterSuiteStatistics()' passes a literal string as parameter 'text' of a call to 'LoggerTxt.WriteLine(string)'. Retrieve the following string(s) from a resource table instead: "Fecha inicio: , , Fecha Fin: ".
CA1303 Method 'SuiteLoggerTxt.RegisterSuiteStatistics()' passes a literal string as parameter 'text' of a call to 'LoggerTxt.WriteLine(string)'. Retrieve the following string(s) from a resource table instead: "Tiempo total
Loading symbols for System.Numerics.dll


Execution_Logs
DATA (E:) ▸ Trabajo ▸ Adriano.Vega ▸ MainResiplusAutomation ▸ Selenium ▸ Launcher ▸ Adriano.Vega ▸ bin ▸ Debug ▸ Execution_Logs
Search Execution_Logs
Organize
Open
Include in library
Share with
New folder
Favorites
Desktop
Downloads
Recent Places
Compartido
Adriano Vega
Libraries
Documents
Music
Pictures
Videos
Computer
Disk (C:)
OSDisk (D:)
DATA (E:)
swap (S:)
Network
Name
Date modified
Type
Size
2018-08-31 17_19 31/08/2018 17:19 File folder
2018-08-31 17_20 31/08/2018 17:22 File folder
2018-08-31 17_25 31/08/2018 17:25 File folder
2018-08-31 17_27 31/08/2018 17:27 File folder
2018-08-31 17_39 31/08/2018 17:39 File folder
2018-08-31 17_40 31/08/2018 17:40 File folder
2018-08-31 17_41 31/08/2018 17:41 File folder
2018-08-31 17_42 31/08/2018 17:42 File folder
2018-08-31 17_44 31/08/2018 17:46 File folder
2018-08-31 17_51 31/08/2018 17:58 File folder
2018-09-04 16_07 04/09/2018 16:21 File folder
2018-09-04 17_56 04/09/2018 17:56 File folder
2018-09-07 16_27 07/09/2018 16:27 File folder
2018-09-07 16_46 07/09/2018 16:46 File folder
2018-09-07 16_47 07/09/2018 16:47 File folder
2018-09-07 16_52 07/09/2018 16:52 File folder
2018-09-07 17_19 07/09/2018 17:19 File folder
2018-09-07 17_34 07/09/2018 17:34 File folder
2018-09-10 13_06 10/09/2018 13:06 File folder
2018-09-10 13_20 10/09/2018 13:20 File folder
2018-09-10 20_59 10/09/2018 20:59 File folder
2018-09-10 20_59
File folder
Date modified: 10/09/2018 20:59


resultados
Adriano.Vega ▸ MainResiplusAutomation ▸ Selenium ▸ Launcher ▸ Adriano.Vega ▸ bin ▸ Debug ▸ Execution_Logs ▸ 2018-09-10 20_59 ▸ resultados
Search resultados
Organize
Open
Print
New folder
Favorites
Desktop
Downloads
Recent Places
Compartido
Adriano Vega
Libraries
Documents
Music
Pictures
Videos
Computer
Disk (C:)
OSDisk (D:)
DATA (E:)
swap (S:)
Network
Name
Date modified
Type
Size
resultados.txt
10/09/2018 21:01
Text Document
2 KB
resultados.txt Date modified: 10/09/2018 21:01
Text Document Size: 1,54 KB
Date created: 10/09/2018 20:59

resultados.txt - Notepad

File Edit Format View Help

| FECHA EJECUCIÓN | CÓDIGO PS | NAVEGADOR | RESOLUCIÓN | RESULTADO | LOG |
|---|---|---|---|---|---|
| 10/09/2018 20:59:41 | PS006257 | Chrome | Desktop | FAIL | E:\Trabajo\Adriano.Ve |
| 10/09/2018 21:00:13 | PS006257 | Chrome | Tablet | PASS | E:\Trabajo\Adriano.Ve |
| 10/09/2018 21:01:00 | PS006257 | InternetExplorer | Desktop | PASS | E:\Trabajo\Adriano.Ve |
| 10/09/2018 21:01:42 | PS006257 | InternetExplorer | Tablet | PASS | E:\Trabajo\Adriano.Ve |

ESTADÍSTICAS

Fecha inicio: 10/09/2018 20:59:26 , Fecha Fin: 10/09/2018 21:01:49

Tiempo total ejecucción: 0 horas 2 minutos 23 segundos

Total pruebas ejecutadas: 4

Total pruebas PASS: 3

Total pruebas WARN: 0

Total pruebas FAIL: 1

Total pruebas desconocido: 0

Ln 1, Col 1

Start

ES 21:02 10/09/2018

PS006257_Chrome_Desktop.txt - Notepad

File Edit Format View Help

```
[10/09/2018 20:59:30] - INFO: Iniciando ejecución de la PS
[10/09/2018 20:59:35] - INFO: Iniciando sesión como TestUser02
[10/09/2018 20:59:40] - FAIL: Se ha producido la siguiente excepción:  System.NullReferenceException
   at Add.Ft.Selenium.ResiplusTablet.Componentes.SelectResidentComponent.Select(String nombreResiden
   at Add.Ft.Selenium.ResiplusTabletTests.ID29426.PA023452.PS006257.TestSteps() in E:\Trabajo\Adrian
   at Add.Ft.Selenium.Base.Pruebas.PruebaSistema.Execute(RemoteWebDriver driver, ITestLogger testLog
   at Add.Ft.Selenium.Launcher.TestLauncher.Launcher.RunTest(PruebaSistema test, RemoteWebDriver bro
Screenshot capturada en: E:\Trabajo\Adriano.Vega\MainResiplusAutomation\Selenium\Launcher\Adriano.Ve
```

Ln 1, Col 1

Start

ES 21:03 10/09/2018

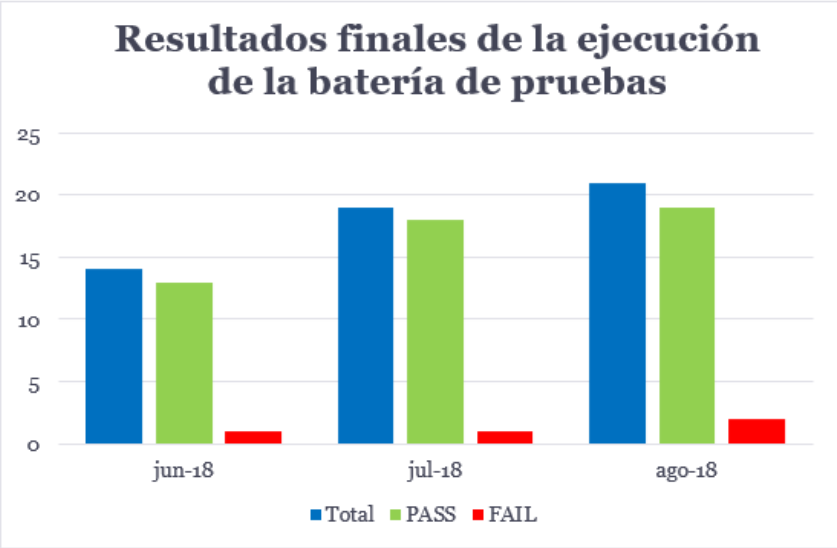
# 7. Validación del proceso y la herramienta

# 7. Validación del proceso y la herramienta

- **Instrucción de un encargado de automatización:**
  - A fecha de julio de 2018 se comenzó la instrucción de un miembro del equipo de automatización en la aplicación del proceso.
  - Aplicó el proceso completo en dos *sprints* distintos.
  - Automatizó también pruebas.
  - Se espera el resto del equipo de automatización se incorpore progresivamente.

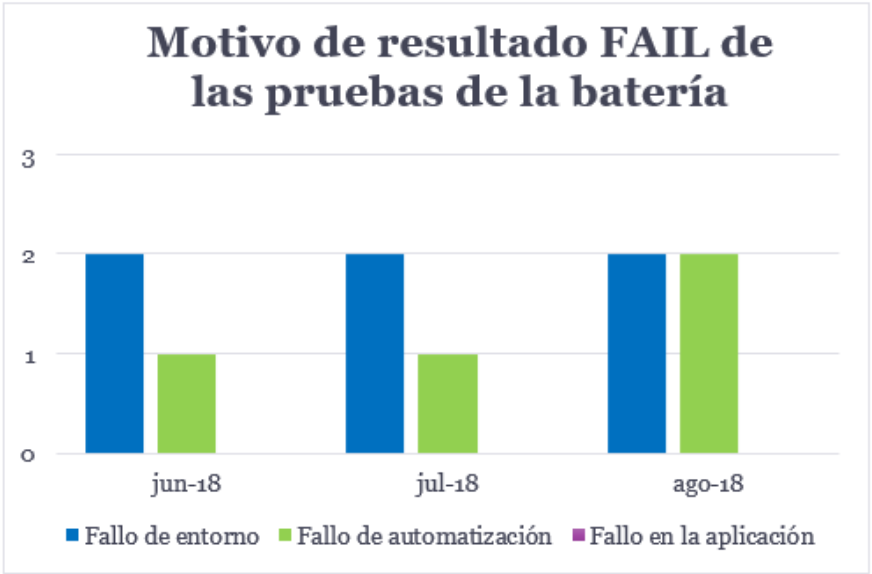
# 7. Validación del proceso y la herramienta

- Se desarrolló una batería de pruebas que cuenta con **23 pruebas** a fecha de agosto de 2018.
- El proceso fue aplicado completamente en tres ocasiones: *sprints* de junio, julio y agosto de 2018.

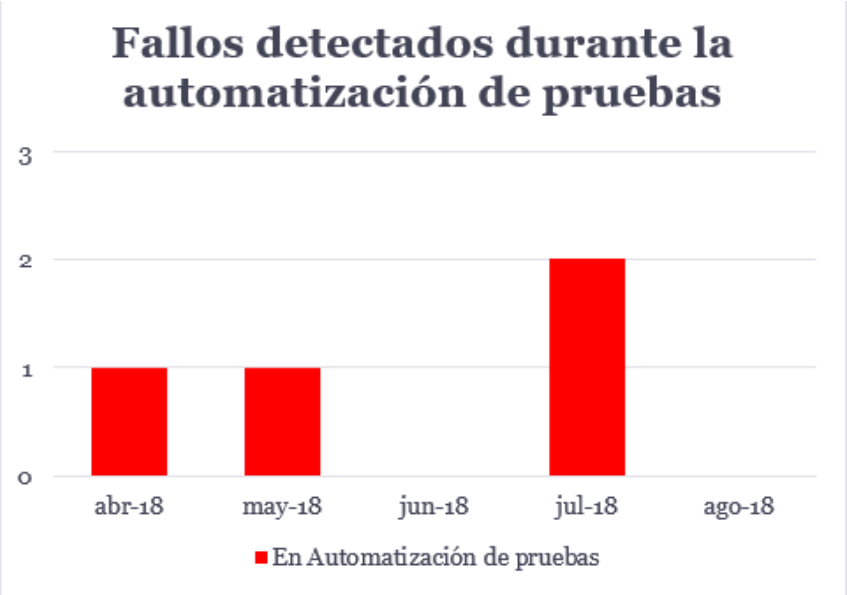


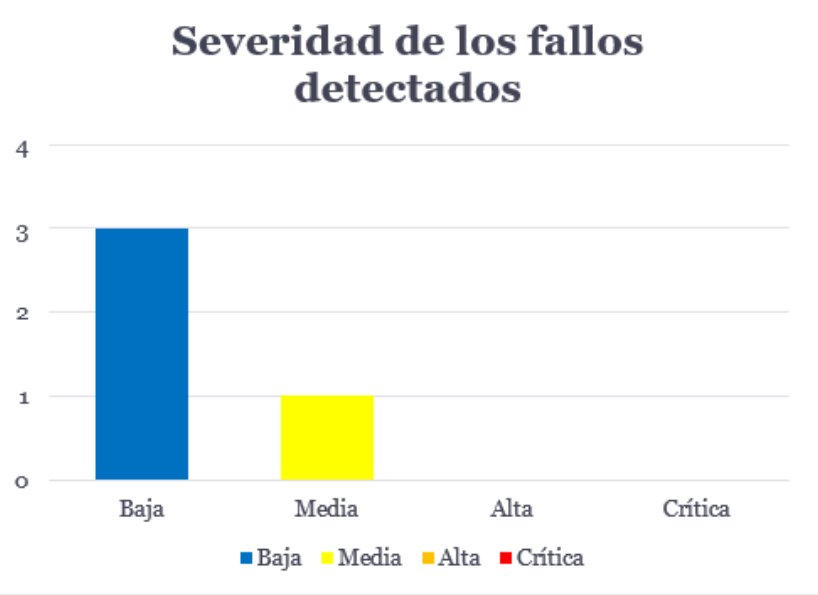
# 7. Validación del proceso y la herramienta

- **Desglose de los fallos de ejecución de las pruebas durante los lanzamientos de la batería:**
  - La mayoría fueron debido a errores de entorno.
  - También hubo fallos de automatización.
  - No se detectó ningún fallo de aplicación.



# 7. Pruebas aplicadas sobre el proceso y la herramienta



# 8. Conclusiones y trabajos futuros

# 8. Conclusiones y trabajos futuros

- En cuanto el proceso de pruebas automatizadas:
  - Ha sido implantado satisfactoriamente y se ha probado en condiciones reales.
  - Se aplicó en tres lanzamientos de versión distintos.
    - Aplicado en dos ocasiones por personas ajenas al proyecto.
  - Se continuará aplicando en los futuros *sprints* del desarrollo.

# 8. Conclusiones y trabajos futuros

- En cuanto a la herramienta basada en Selenium:
  - Se alcanzaron los objetivos de automatización de pruebas.
    - Ejecución *crossbrowser* y *responsive* de las pruebas.
  - La batería desarrollada cuenta con 23 pruebas.
    - Ha servido para detectar fallos dentro de la aplicación bajo pruebas.
  - Se ha constatado que durante la automatización de las pruebas se detectan más errores que durante la ejecución de las pruebas.

# 8. Conclusiones y trabajos futuros

**Posibles trabajos futuros – Herramienta:**

- Integración con la infraestructura existente de pruebas.
  - Usada para la ejecución de pruebas del ERP.
- Creación de un entorno de ejecución de las pruebas en emuladores:
  - Usando la biblioteca Appium, simular la ejecución en dispositivos Android y iOS mediante un entorno con emuladores.

# FIN

# Pruebas funcionales automatizadas para aplicaciones Web: usando Selenium para aplicar pruebas de regresión automatizadas

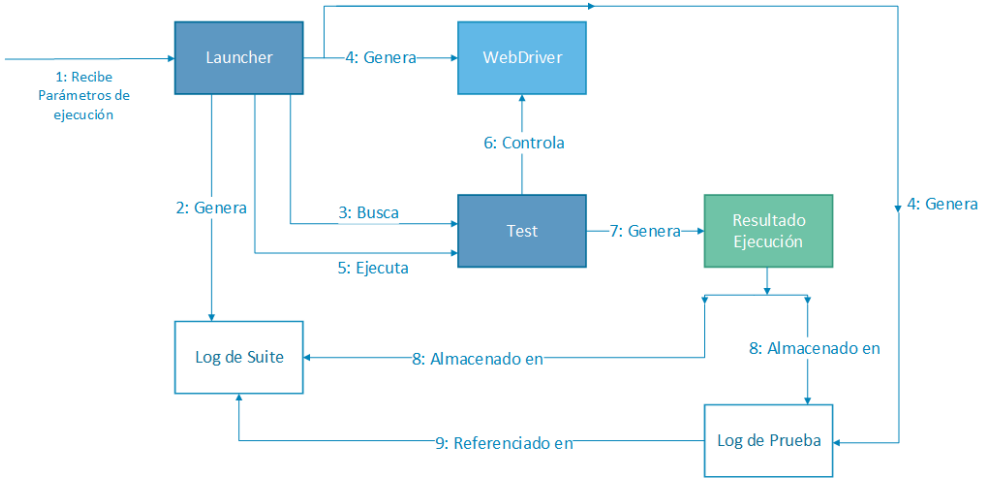
**Autor**: Adriano Tobías Vega Llobell
**Tutores**: Patricio Orlando Letelier Torres
María Carmen Penadés Gramage
Curso: 2017 - 2018

UNIVERSITAT POLITÈCNICA DE VALÈNCIA
Escola Tècnica Superior d'Enginyeria Informàtica
etsinf

# 4. Propuesta basada en Selenium

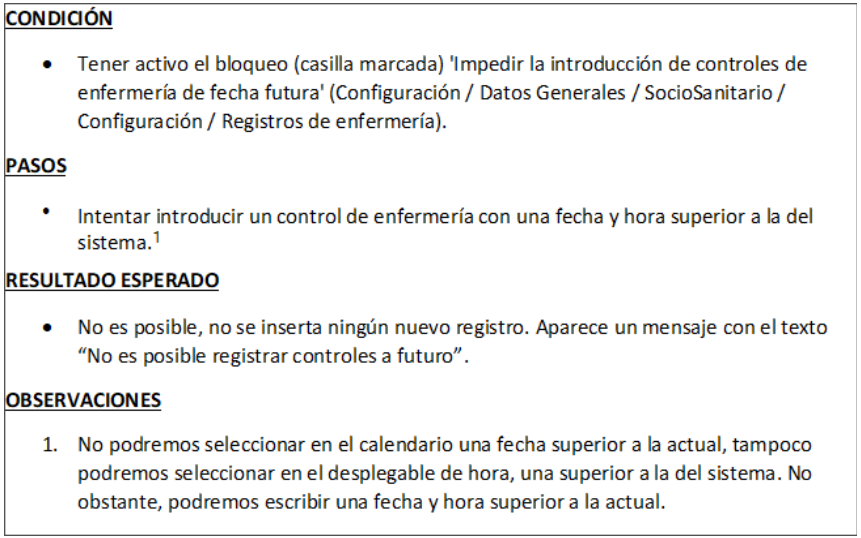
**Flujo de ejecución de una batería**:
Representa un formulario o componente de la aplicación web.

- Implementado siguiendo el patrón *Page Object*.
- Abstrae a las pruebas de las interacciones con la aplicación.
- Centraliza la lógica de interacción, facilitando los cambios.



# 5. Proceso de pruebas automatizadas para aplicaciones web

- **Automatización y mantenimiento de la batería. Definición de la prueba de aceptación.**
  - Durante la fase de análisis, un analista define la prueba de aceptación (PA).
  - Los *testers* elegirán que PA son más prioritarias para automatizar.
  - De la prueba de aceptación, se derivan los distintos casos de prueba.



**CONDICIÓN**

- Tener activo el bloqueo (casilla marcada) 'Impedir la introducción de controles de enfermería de fecha futura' (Configuración / Datos Generales / SocioSanitario / Configuración / Registros de enfermería).

**PASOS**

- Intentar introducir un control de enfermería con una fecha y hora superior a la del sistema.[1]

**RESULTADO ESPERADO**

- No es posible, no se inserta ningún nuevo registro. Aparece un mensaje con el texto "No es posible registrar controles a futuro".

**OBSERVACIONES**

1. No podremos seleccionar en el calendario una fecha superior a la actual, tampoco podremos seleccionar en el desplegable de hora, una superior a la del sistema. No obstante, podremos escribir una fecha y hora superior a la actual.

# 5. Proceso de pruebas automatizadas para aplicaciones web

- **Automatización y mantenimiento de la batería. Diseño del caso de prueba.** Los casos de prueba se pueden dividir en tres apartados:
    - **Acciones previas**: Se definen todas las acciones de preparación del entorno.
        - Opciones de configuración necesarias.
        - Datos necesarios.
        - Etc.

**Acciones previas**

*Restaurar los scripts de base de datos:*

- ID29426_DatosResidencia_01.sql
- ID29426_DatosUsuarios.sql

*Estado previo del sistema:*

- Utilizar el usuario TestUser02, que tiene permisos para ver y crear controles de diuresis.
- Tener activo el bloqueo (casilla marcada) *Impedir la introducción de controles de enfermería de fecha futura*.
- Existe un residente llamado 'Nuevo Residente 00001'.

# 5. Proceso de pruebas automatizadas para aplicaciones web

- **Automatización y mantenimiento de la batería. Diseño del caso de prueba.** Los casos de prueba se pueden dividir en tres apartados:
    - **Pasos de la prueba**: Se define paso a paso la ejecución de la prueba.
        - Las interacciones con la aplicación.
        - Los datos a introducir.
        - Las comprobaciones que se deben realizar.
        - Etc.

**Pasos de la prueba**

1. Iniciar sesión con TestUser02.
2. Seleccionar el residente 'Nuevo Residente 00001'.
3. Acceder a Controles → Diuresis.
4. Crear un nuevo registro con fecha un año mayor a la actual.
5. Comprobar que ha aparecido un mensaje de alerta con el texto "No es posible registrar controles a futuro"
6. Comprobar que no se ha insertado ningún registro nuevo en la tabla.
7. Cerrar la sesión.

# 5. Proceso de pruebas automatizadas para aplicaciones web

- **Automatización y mantenimiento de la batería. Diseño del caso de prueba.** Los casos de prueba se pueden dividir en tres apartados:
    - **Resultado esperado**: En este apartado se describen los resultados de las comprobaciones que se describieron en los pasos de la prueba.
        - Determinaran si la prueba ha pasado o no.

**Resultado esperado**

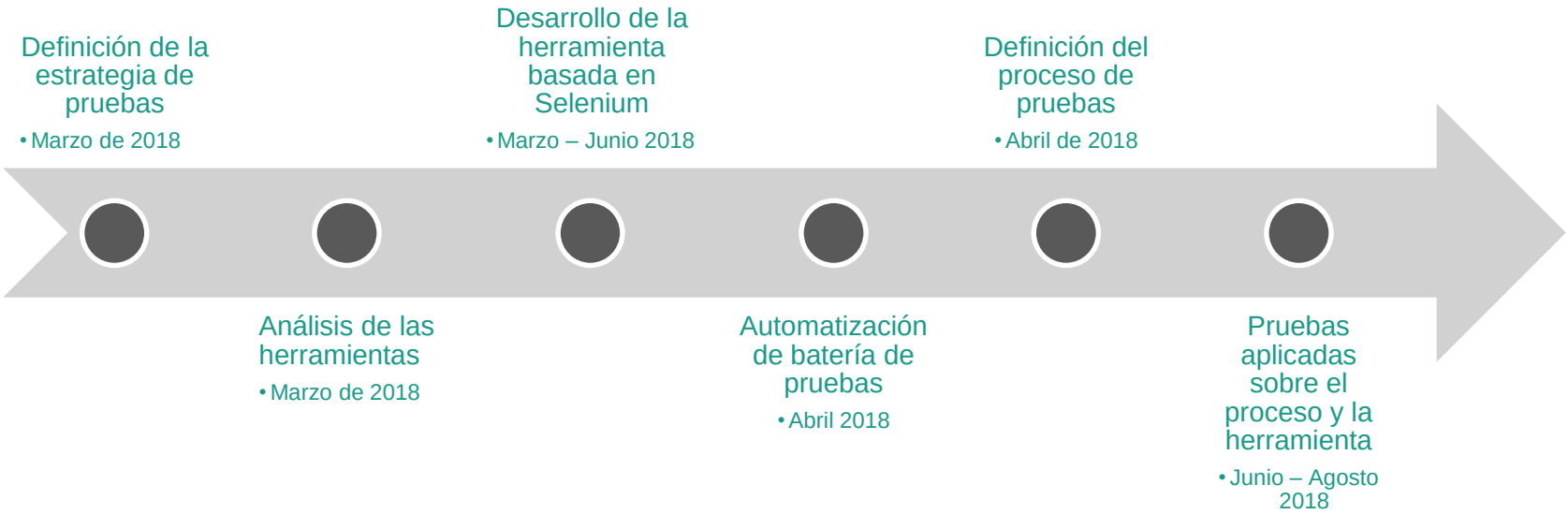
1. La aparición de un mensaje de información con el texto 'No es posible registrar controles a futuro'. (Paso 5)
2. No se ha introducido ningún nuevo registro con los datos de ejemplo. (Paso 6)

# 5. Proceso de pruebas automatizadas para aplicaciones web

- **Automatización y mantenimiento de la batería. Automatización de la prueba:** Siguiendo el diseño descrito anteriormente, se implementa la prueba.
    - Una vez se compruebe que funciona correctamente, se añade a la batería de pruebas.

```
public class PS006257 : Test
{
    // …
    protected override void SetUp()
    {
        // …

        restoreScript($@"{ Workspace.WORKSPACE_DIRECTORY }
   \ID29426\ID29426_DatosResidencia_01.sql", "DatosResidencia_001_01");
        restoreScript($@"{ Workspace.WORKSPACE_DIRECTORY }
   \ID29426\ID29426_DatosUsuarios.sql", "DatosUsuarios");
    }

    protected override void TestSteps()
    {
        // 1. Login con 'TestUser02'.
        login.NavigateToPage();
        info("Iniciando sesión como TestUser02");
        login.LogIn("TestUser02", "Selenium");

        // 2. Seleccionar el residente "Nuevo Residente 00001"
        seleccionarResidente.Select("Nuevo Residente 00001");

        // 3. Acceder a Controles -> Diuresis del 'Nuevo Residente 00001'.
        home.SelectOptionHome(HomePage.OpcionMenuPrincipal.Controles);
        controlesEnfermeria.SelectControl(
            ControlesPage.OpcionControles.Diuresis
        );

          // …

        // 7. Cerrar la sesión.
        info("Cerrando sesión");
        login.LogOut();
    }

    protected override void Teardown()
    {
    }
}
```

# 8. Conclusiones y trabajos futuros - Cronograma



# 8. Conclusiones y trabajos futuros

- Relación con asignaturas de la carrera:
    - **Proceso de *software*:** mejora de un proceso de desarrollo existente, mediante la implantación de un proceso de pruebas automatizadas para productos de una empresa
    - **Diseño de *software***: diseño modular de la herramienta. Aplicación de distintos patrones de programación, como *Page Object*.
    - **Mantenimiento y Evolución del *software***: se estableció una estrategia de pruebas para una aplicación. Esto formó parte del mantenimiento y evolución de esta aplicación.