Cucumber – lenguaje Getkins

**Para MAC**

Paso 1 Abrir la consola / terminal

Paso 2 En la terminal escribir: touch ~/.bash-profile (dar enter)

Paso 3 En la terminal escribir: sudo nano ~/.bash-profile(dar enter)

Paso4: Posiblemente te pida la contraseña de tu máquina (capturas contraseña y enter... te cambiará de pantalla y escripes lo siguiente...)

Paso 5 En la terminal escribir: export JAVA\_HOME=$(/usr/libexec/java\_home) dar enter

Paso 6 En la terminal escribir: export PATH=${JAVA\_HOME}/bin:$PATH (guardas los cambios con ctlr + o, le das enter y para salir ctrl + x)

Paso 7 En la terminal escribir: source ~/.bash-profile

**Cuenta de git:**

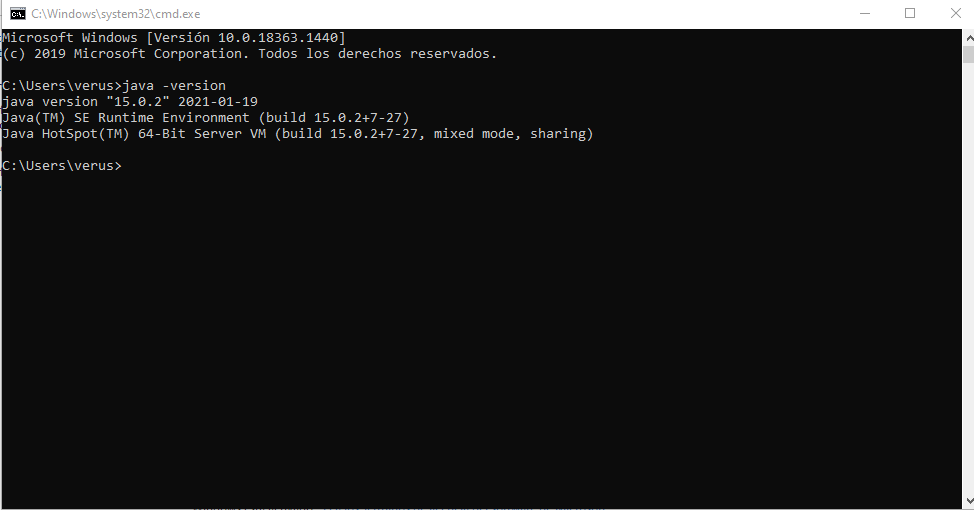
[team.automation2021@gmail.com](mailto:team.automation2021@gmail.com) / pass: Team2021

**Windows/variables de entorno**

Cuando no se tiene acceso a las variables del sistema, se utilizan las variables de usuario

Para acceder más rápido a la configuración:

Windows + r --- abre una consola (cmd), posteriormente se teclea java – version y se debe mostrar la versión de java instalada



Posteriormente se descarga IntelliJ (IDE) – Entorno de desarrollo, puede ser IntelliJ, Eclipse o Visual

Google\download IntelliJ en la página descargar la opción de Community

Instalación

1. Descargar JAVA SE, última versión de nuestro sistema operativo
2. Instalar Java
3. Configuración de variables de entorno
   1. Ir a ventana de Windows (explorador)
   2. Este equipo, clic derecho, propiedades y aparece ventana
   3. Configuración avanzada del sistema
   4. Variables de sistema (cuando no se tiene el pws, se seleccionan variables de usuario)
      1. Crear Java\_Home, seleccionar donde esta Java y hasta donde está el JDK
      2. En el path, en la misma carpeta del JDK, pero hasta el BIN

En una ventana de DOS se escribe Java -version para que nos muestre donde está instalado el JDK

Para la automatización se requiere conocer los cuatro pilares de la programación orientada a objetos (POO):

1. Herencia
2. Encapsulamiento
3. Abstracción
4. Poliformismo

Bucles utilizados: For, While and Do while

Condicionantes: For

**Reglas para codificación:**

Snake case: se utilizan \_ para generar nombres - Python

Cammel case: se inicia con minúsculas y las conjunciones son con mayúsculas – Java

En algunos casos se pone .COM al final del nombre del proyecto, por convención de programación

Indexar el código

Ctrl + Alt + L



== sirve para igualar o relacionar algo

!= diferente de

Para instalación en MAC

JAVA\_HOME=$(/usr/libexec/java\_home)

PATH=${JAVA\_HOME}/bin:$PATH

~/.bash-profile (dar enter)

sudo nano ~/.bash-profile

source ~/.bash-profile

public static void main (String args []) { System.out.println("Text"); }

public para que se pueda acceder desde otra clase

static porque es estático y solo se ejecuta una vez

void porque no espera que se le regrese alguna información

main es el motor que hace que se ejecute el programa

Cuando se crea una clase solo lleva el cuerpo de la clase y es entre llaves, no lleva punto y coma

Dentro de las clases van métodos, el método recibe argumentos que van entre paréntesis, no lleva punto y coma

La declaración de variables llevan punto y coma al final de la línea

public static void main(String[] args)  
// Método PSVM  
// public: Método de accesibilidad para instanciar la clase u objeto desde cualquier parte de nuestro proyecto  
// static: Solo se ejecuta una vez  
// void: No regresa o no espera algo (No usamos la palabra reservada return  
// main: Método de arranque o principal



Diferencia entre automatizar y grabar

Herramienta de Selenium

1. Selenium IDE \*, es un grabador que inicialmente solo tenía integración con Mozila, actualmente también tiene integración con FireFox. Herramientas similares son Catalon,
2. Selenium WebDriver \*, sirve para Programar ejecuciones automatizadas. Se creo en 1997.
3. Selenium Grid, sirve para hacer automatizaciones en múltiples equipos (pruebas automatizadas en paralelo). Lanza los scprits en distintos equipos.
4. Selenium RemoteControl (descontinuado)

**COMANDOS DE CONSOLA Y GIT**

vero-git2021 / R3c0n0c1d4$ - verus.mex@gmail.com

[vero.automatiza@gmail.com](mailto:vero.automatiza@gmail.com) – vero-automatiza / r3c0n0c1d4

Para verificar la lista de configuraciones que tiene nuestro repositorio

git config –list

git config – global user.name “Veronica Montoya”

git config --global user.email [team.automation2021@gmail.com](mailto:team.automation2021@gmail.com)

Victor Saucedo

12:25

git clone

Victor Saucedo

12:35

git config --list

Victor Saucedo

12:38

git config --global user.name "Victor Saucedo"

Victor Saucedo

12:40

git config --global user.email team.automation2021@gmail.com

git status – para identificar los cambios realizados

git add – para agregar archivos

git commit - m "Mi primer commit" – para agregar los archivos que no tiene mi proyecto

1. Siempre iniciamos con git status para saber qué tenemos
2. Segundo git add + nombre del archivo para agregar un archivo
   1. En caso de que sean varios archivos usar git add -A // A significa All
3. Después se hace commit -m
4. Se hace git push para pasar los archivos de modo local al repositorio

**NOTA: En caso de darle git commit sin la -m, se hace lo siguiente:**

1. Se agrega un mensaje en la parte superior de la pantalla
2. Se da ESC
3. Y después en el promt se pone “:wq” y con eso se recuera la sesión

Para ver cuántos commits se han hecho se utiliza git log

Con este comando se trae todos los cambios que están en el repositorio **git pull**

//Utilizando xpath con tag (tag: span, img, button, input, select)y un atributo

//Utilizando xpath por contains usando "\*", el asterisco se puede reemplazar por el tag o tipo de elemento

Victor Saucedo

<http://automationpractice.com/index.php>

USUARIO = "test.0@test.com"; CONTRASENA = "test1234";

Sesión sábado 12 de junio 2021

//button[@type='submit' and @name='submitNewsletter'

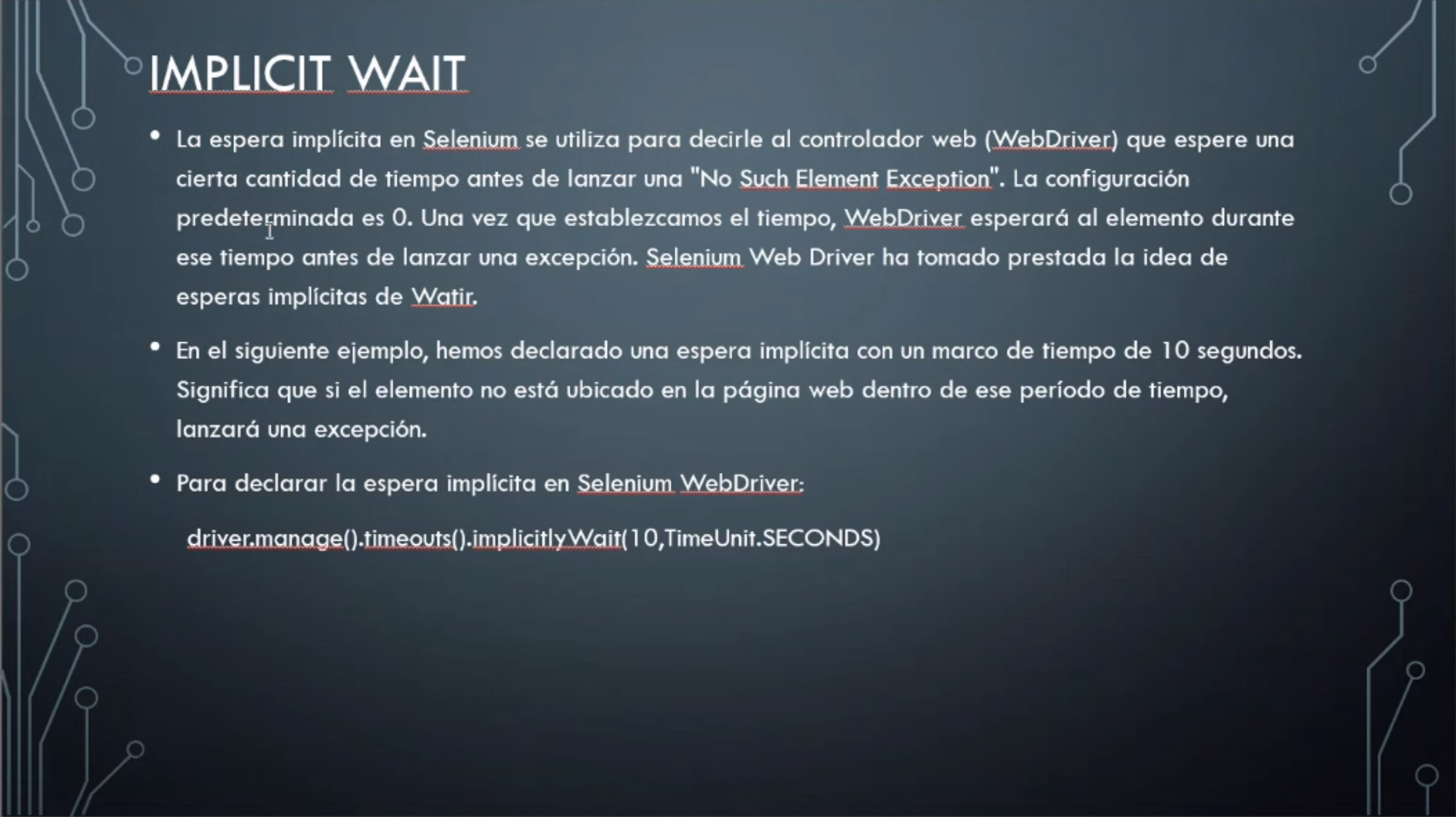
//\*[contains(text(),'Authentication failed')]

**Código para archivo CONFIG donde se van a poner los**

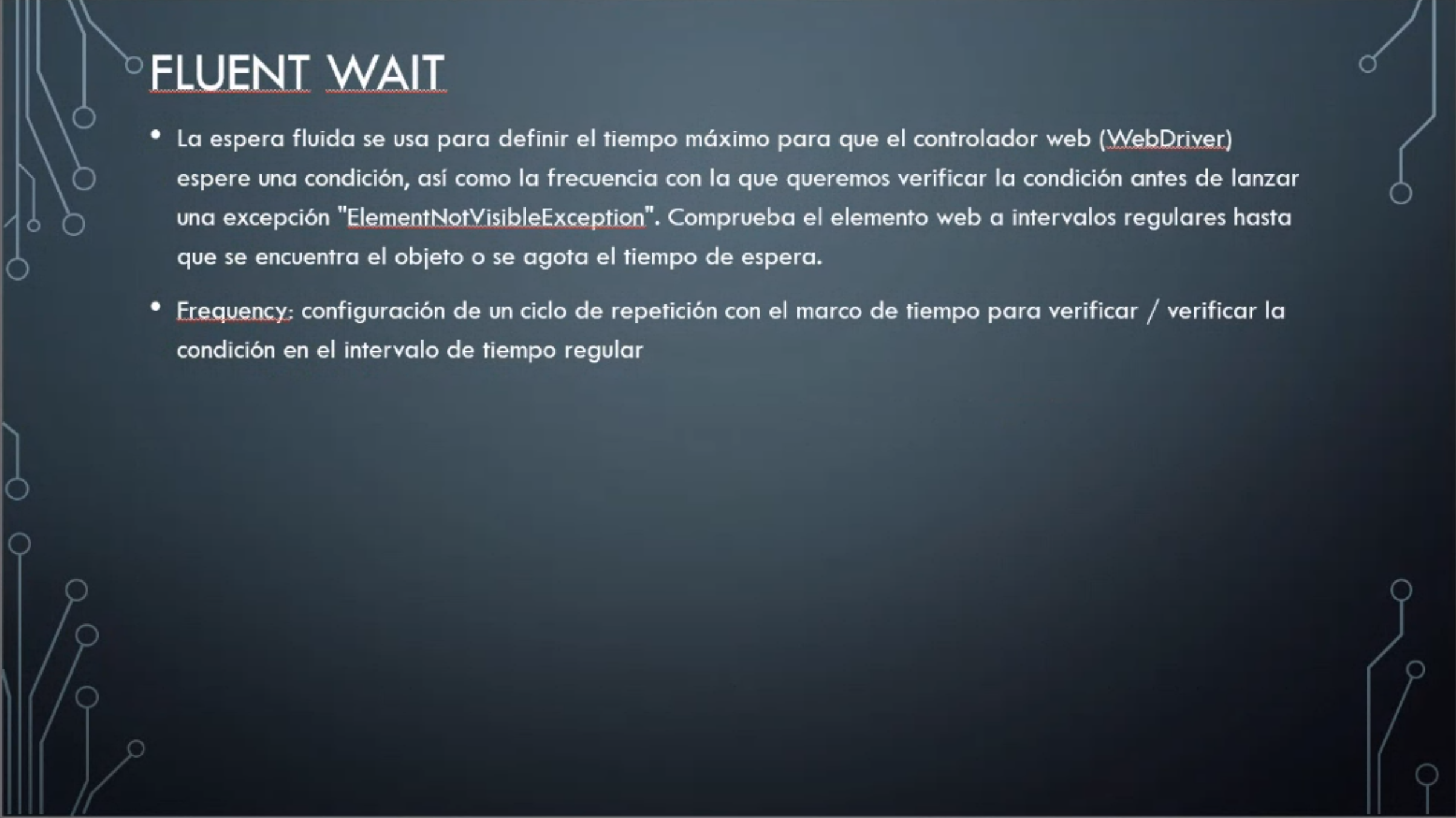
public static String URL\_QA = "<http://automationpractice.com/index.php>"; public static String URL\_DEV = "<http://automationpractice.com/index.php>"; public static String user = "test.0@test.com"; public static String password = "test1234";

**Un archivo JSON se controlan los datos que se van a enviar a los parámetros o bien mediante un Excel**

<https://medium.com/@wc.testing.qa/la-famosa-pir%C3%A1mide-de-cohn-y-la-dura-realidad-e1250dfbe5f3>







**Repaso**

ElementsLogin elementsLogin = new ElementsLogin();

**Las instancias nos sirven para refactorizar, es decir, para definir código que se utiliza en diversas partes del sistema, o bien, que se utiliza en todos los programas, como el caso de “abrir carpeta”, “maximizar pantalla”, etc…**

**Estamos creando funciones para poderlas usar en un proceso más complejo, pero utilizando un código más compacto.**

**ElementsLogin, hace referencia a la clase ElementsLogin que se encuentra dentro de Login. El objeto elementsLogin es un objeto de la clase y es de tipo referenciado.**

**En este tipo de instancias, la sentencia “= new” sirve para indicar que regrese la información que está en ElementsLogin().**

**Cuando usamos public, private, etc… son tipo de accesibilidad de las variables**

**Para poder hacer nuestros programas de automatización, en Base test debemos tener la carga y apertura del navegador (@Before) y el cierre del mismo (@After).**

**En cada clase Elements que se crea se almacenan diferentes elementos que se van a utilizar, para cada tipo, por ejemplo: Login, Registro, ConsultaSaldo, MétodosPago, etc… (sería el equivalente a las diferentes tablas que van a conformar mi sistema).**

**Estos elementos no refieren ninguna función, solo son la definición de mis elementos necesarios para construir un script.**

**Features (métodos)**

**Permite crear utilidades (Utils), que permitan a hacer test más eficientes, pero solo implica acciones que se le pueden hacer a un elemento, o acciones para nuestro sistema (ej. LimpiarYLlenar, loading, LimpiarInput, desplazamiento, etc.**

**Para Login:**

1. **Me traigo mis elementos**
2. **Me traigo mis utilerías**
3. **Creo mis funcionalidades**
   1. **Hacer login**
   2. **Hacer cerrarSesion (puede crearse dentro de la clase ElementsLogin o se puede crear una por separado).**

**Nota: para moverse entre pantallas de InteliJi, CTRL + Tab, para regresar Shift + Tab**

**Las acciones se mandan a los métodos WebElement, no a los By**

**En Login, la sentencia:**

// SE DA CLICK EN EL BOTON SING IN Y SE LLENA EL USUARIO  
utils.ClickButoon(elementsLogin.InputUser(driver));

**Está mandando como parámetro “driver”, porque en Elements tengo:**

// ELEMENTO BOTON SIGNIN  
public By byButtonSigning = By.*xpath*("//a[@class='login']");  
public WebElement ButtonSignin (WebDriver driver)

**Mi elemento WebElement que tiene el ButtonSignin le paso el parámetro driver de tipo WebDriver.**

**Nota: En login se ponen las acciones secuenciales que forman parte de mi test (esto es FEATURES).**

**Ojo**

Victor Saucedo

12:15

//SE DA CLIC Y SE LLENA EL INPUT PASSWORD WebElement inputPassword = driver.findElement(By.id("passwd")); inputPassword.click(); inputPassword.clear(); inputPassword.sendKeys("test1234"); Thread.sleep(2000); //SE DA CLIC EN EL BOTON SIGIN WebElement buttonSignIn = driver.findElement(By.id("SubmitLogin")); buttonSignIn.click(); Thread.sleep(2000);

**Cómo me doy cuenta si tengo errores en mis clases??**

1. **Viendo la alerta que está en la esquina superior derecha**
2. **En el package y /o en el element**