**OWASP ZAP – JAVA - ALLURE**

Curso: Automatizacion de Pruebas

huRTADO PONCE, DILIAN ANABEL

MAMANI CAYO, MARIO

Rojas Salinas, ZARELA

Repositorio: https://github.com/zareliux/tallerSesion3

2020

Contenido

[**1.** **Crear Proyecto** 2](#_Toc57586720)

[**2.** **Configurar la versión del Gradle que se tiene en la computadora.** 4](#_Toc57586721)

[**3.** **Agregar dependencias** 5](#_Toc57586722)

[**4.** **Agregar Driver – ChromeDriver** 8](#_Toc57586723)

[**5.** **Buscar Localizadores página https://todoist.com/** 10](#_Toc57586724)

[**6.** **Implementación Package factoryBrowser** 12](#_Toc57586725)

[**7.** **Implementación Package session** 16](#_Toc57586726)

[**8.** **Implementación Package controls** 18](#_Toc57586727)

[**9.** **Implementación Package pages** 21](#_Toc57586728)

[**10.** **Implementación Package clientAPI** 23](#_Toc57586729)

[**11.** **Implementación Package helper** 29](#_Toc57586730)

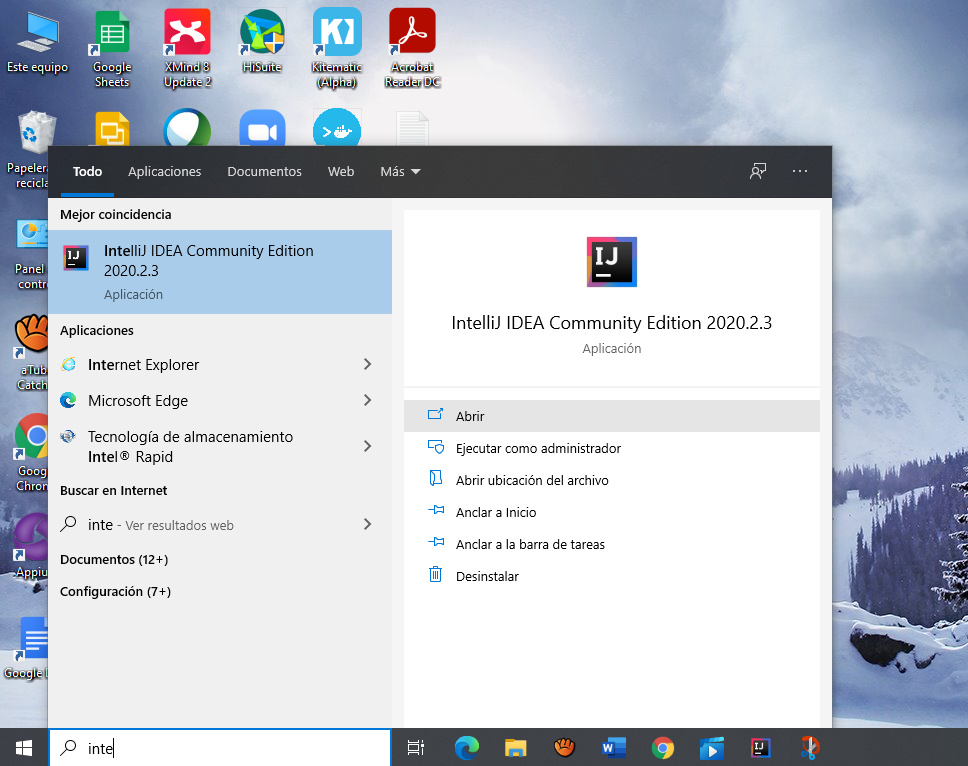
[**12.** **Implementación Package testing** 31](#_Toc57586731)

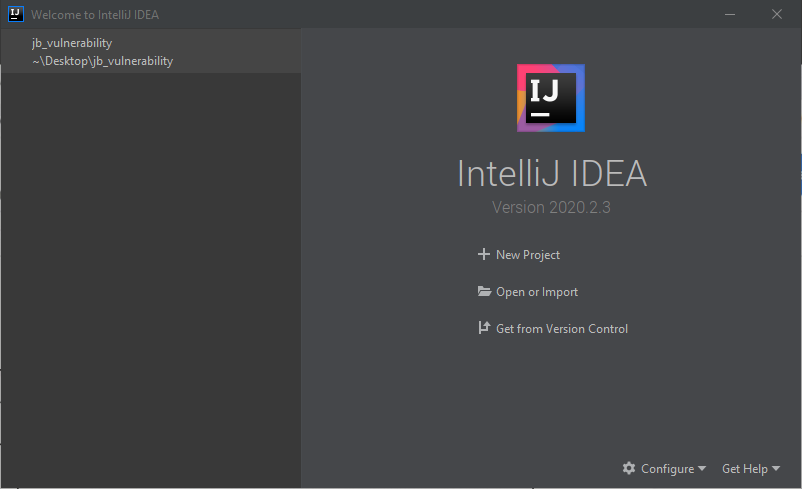
[**13.** **Configuración OWAS ZAP** 35](#_Toc57586732)

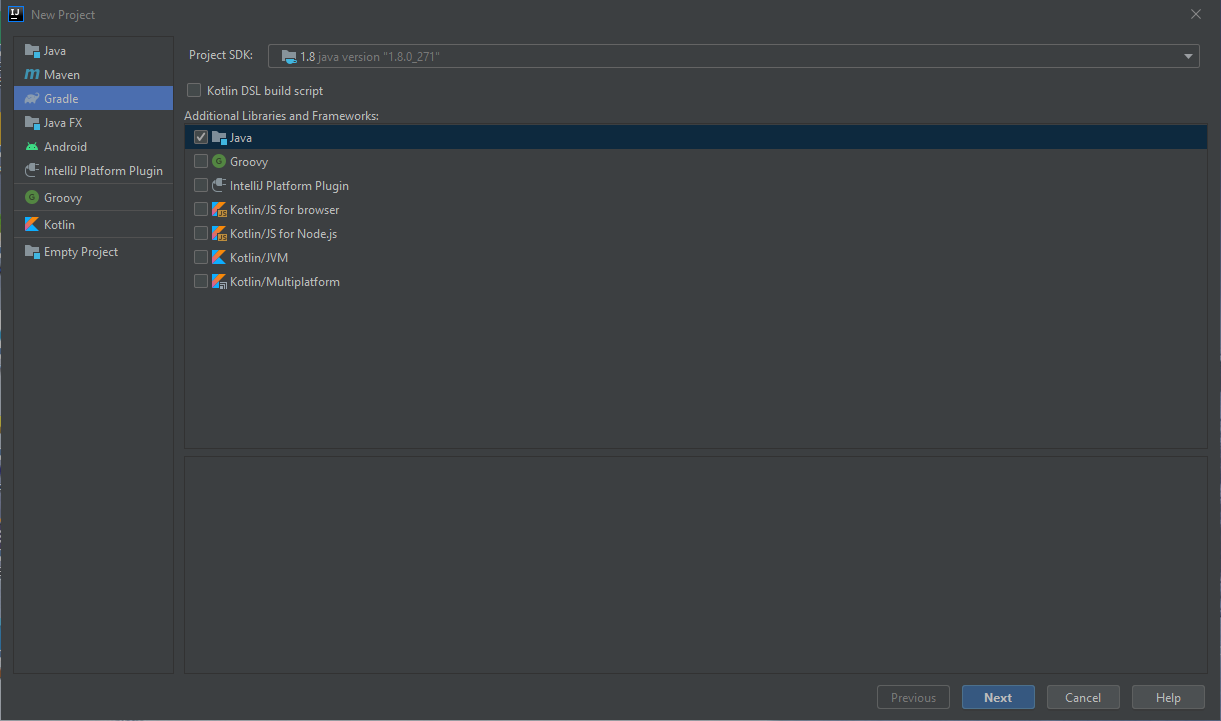
[**14.** **Ejecutar Script** 36](#_Toc57586733)

# **Crear Proyecto**

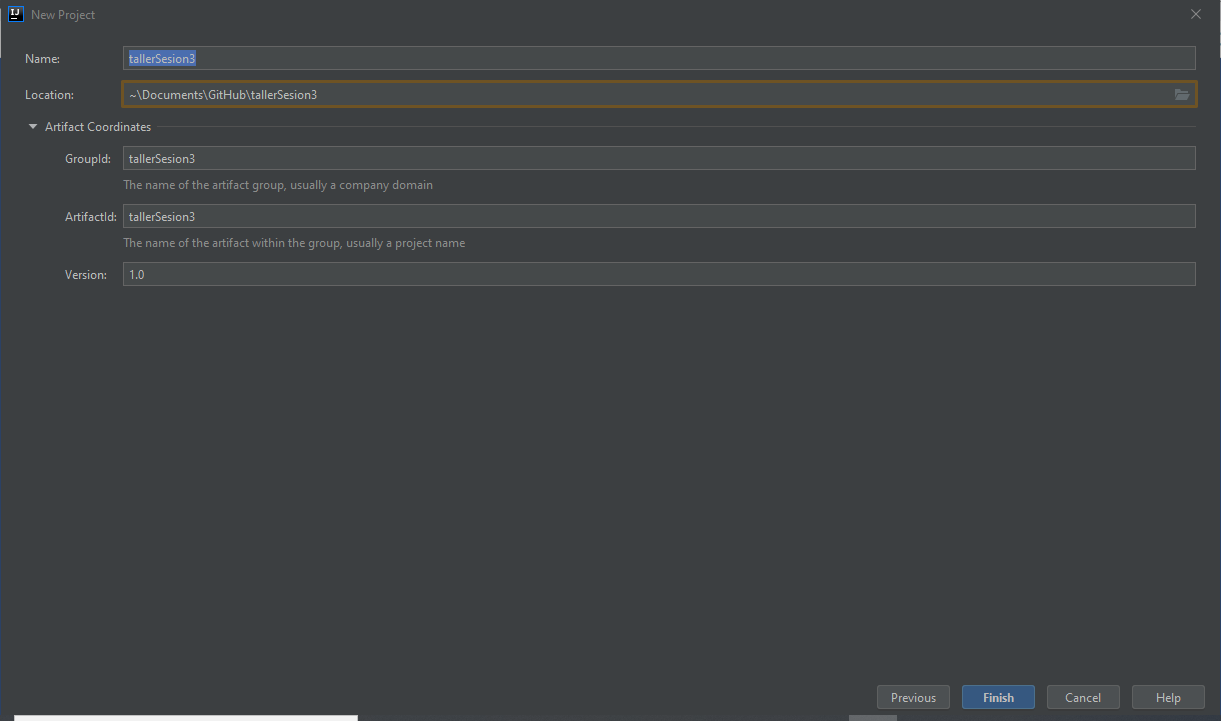
* 1. Abrir IntelliJ IDEA



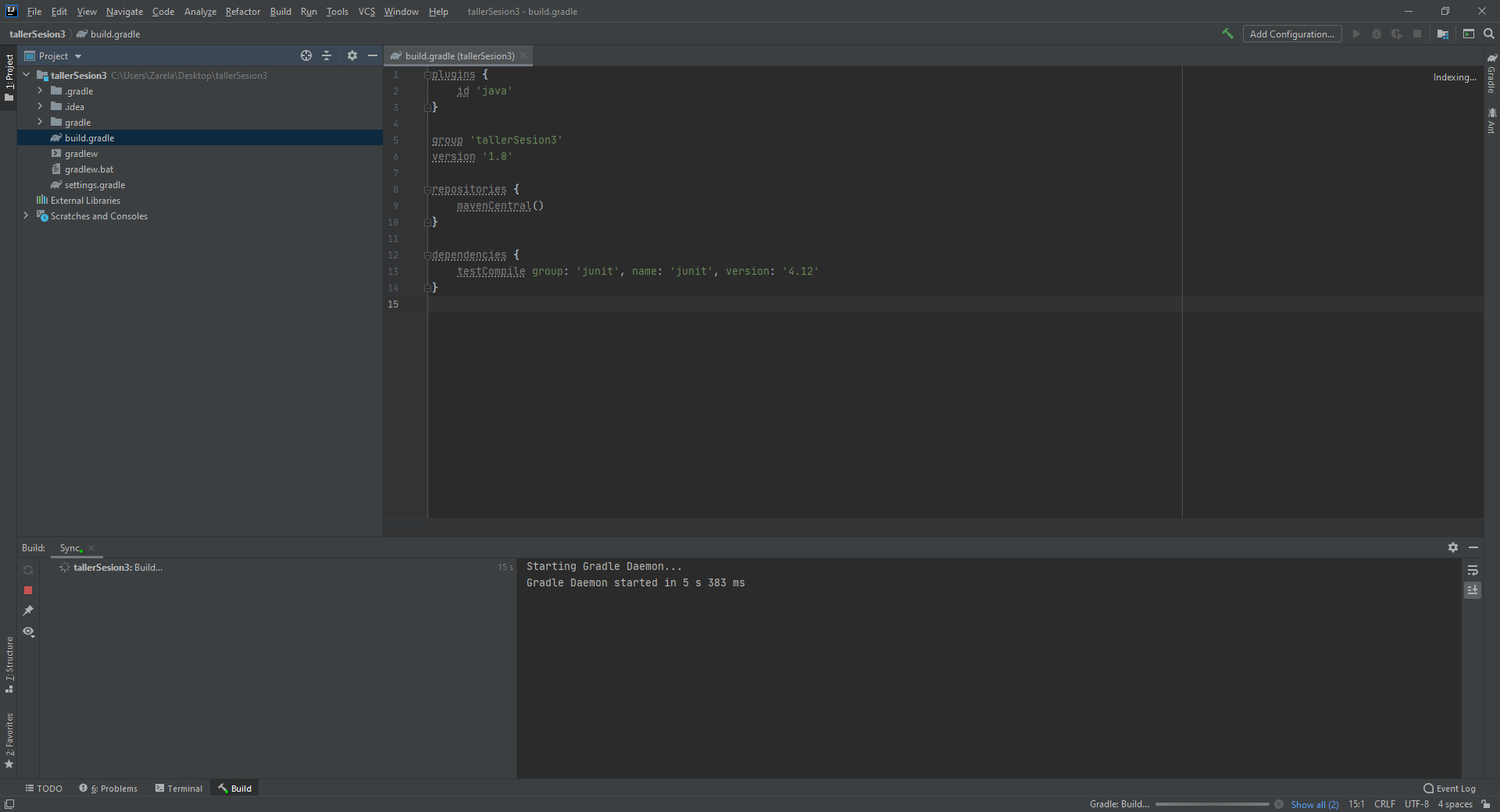
* 1. Seleccionar New Project
  2. Crear un proyecto Gradle con java. Luego selecciona Next.



* 1. Agrega un nombre al proyecto y selecciona la ubicación de la carpeta. Si deseas puedes cambiar los valores de Artifact Coordinates. Luego Selecciona Finish

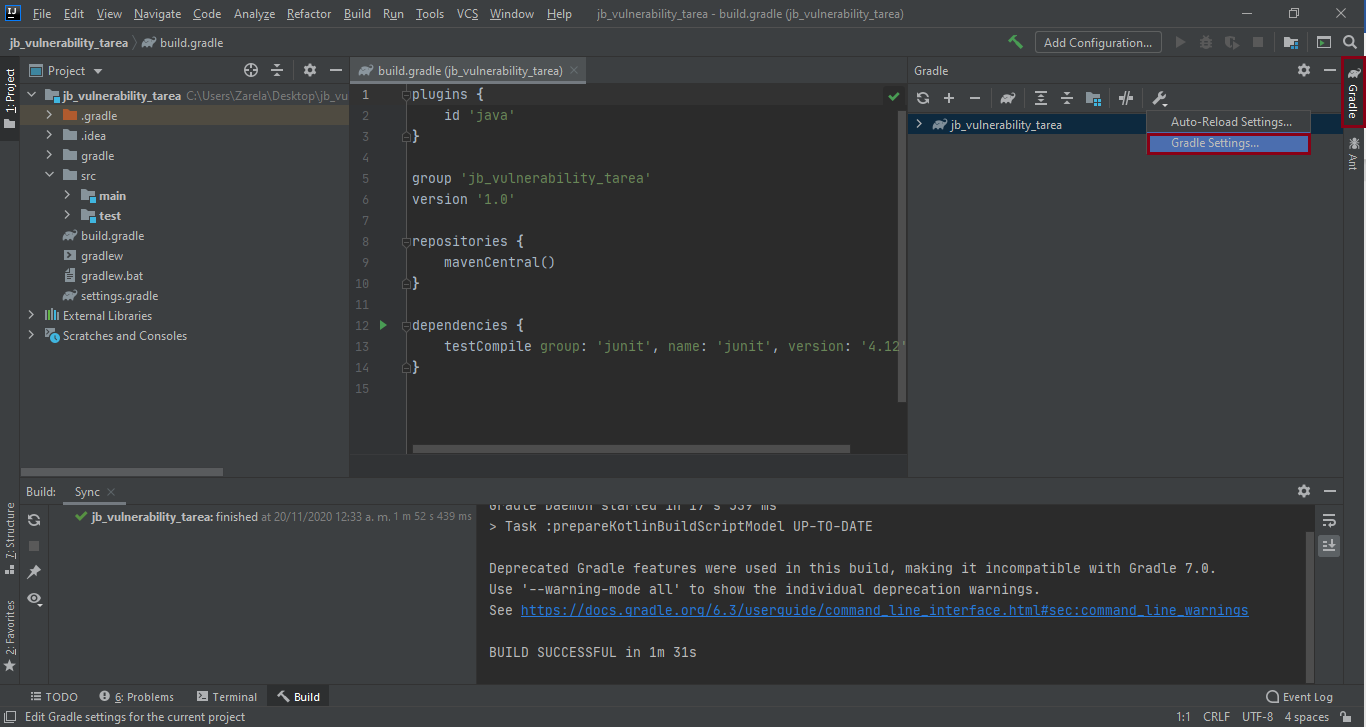


* 1. Se creará el proyecto.

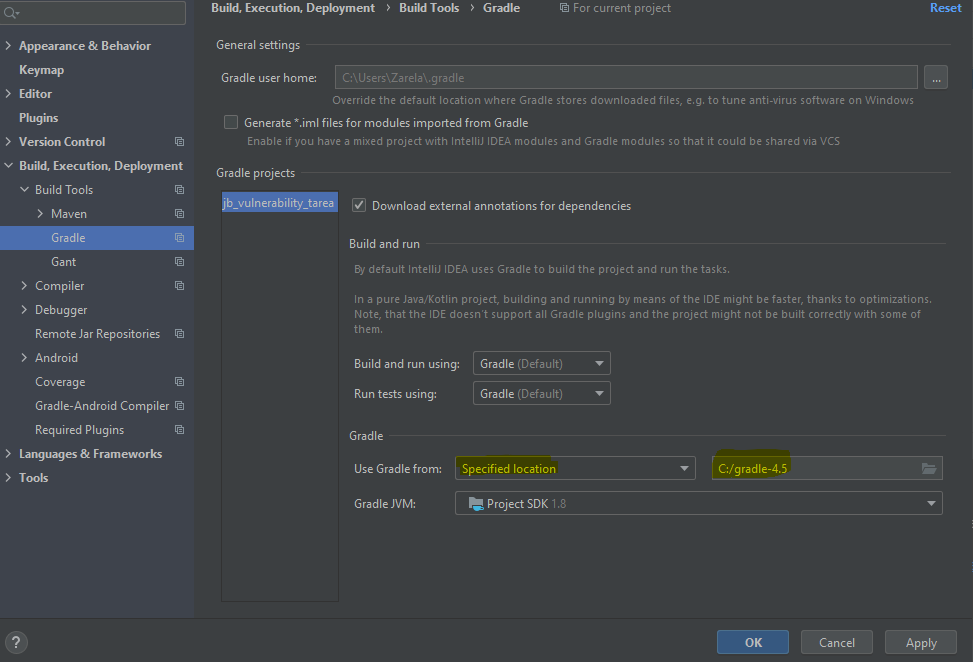


# **Configurar la versión del Gradle que se tiene en la computadora.**

* 1. Seleccionar Gradle y luego en la opción Build Tool Settings selección Gradle Setting.



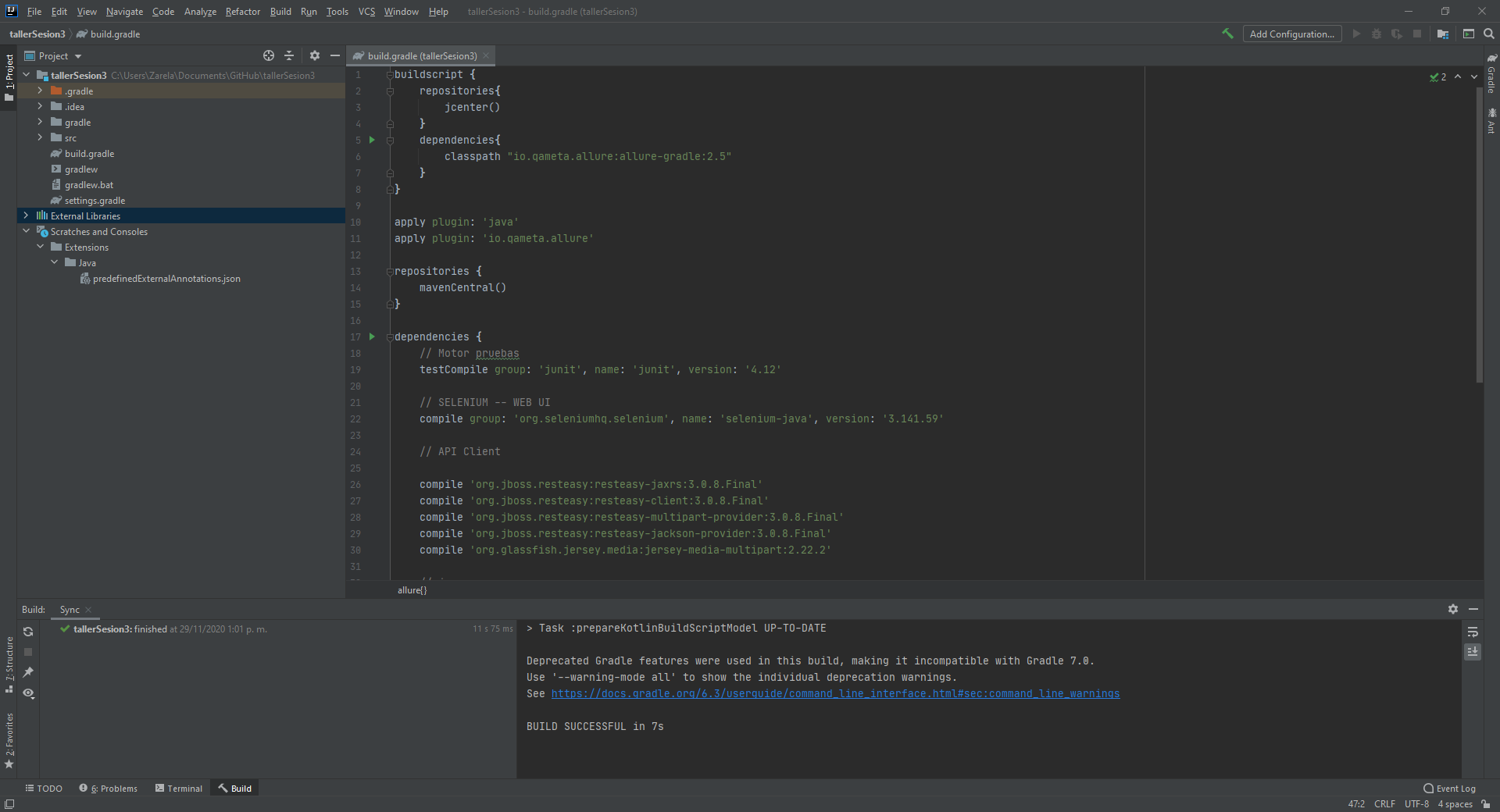
* 1. En la sección de Gradle. Colocar en Use Gradle from “Specified location” y automáticamente en el label del costado se mostrará la ubicación del Gradle en la máquina. Seleccionar OK.



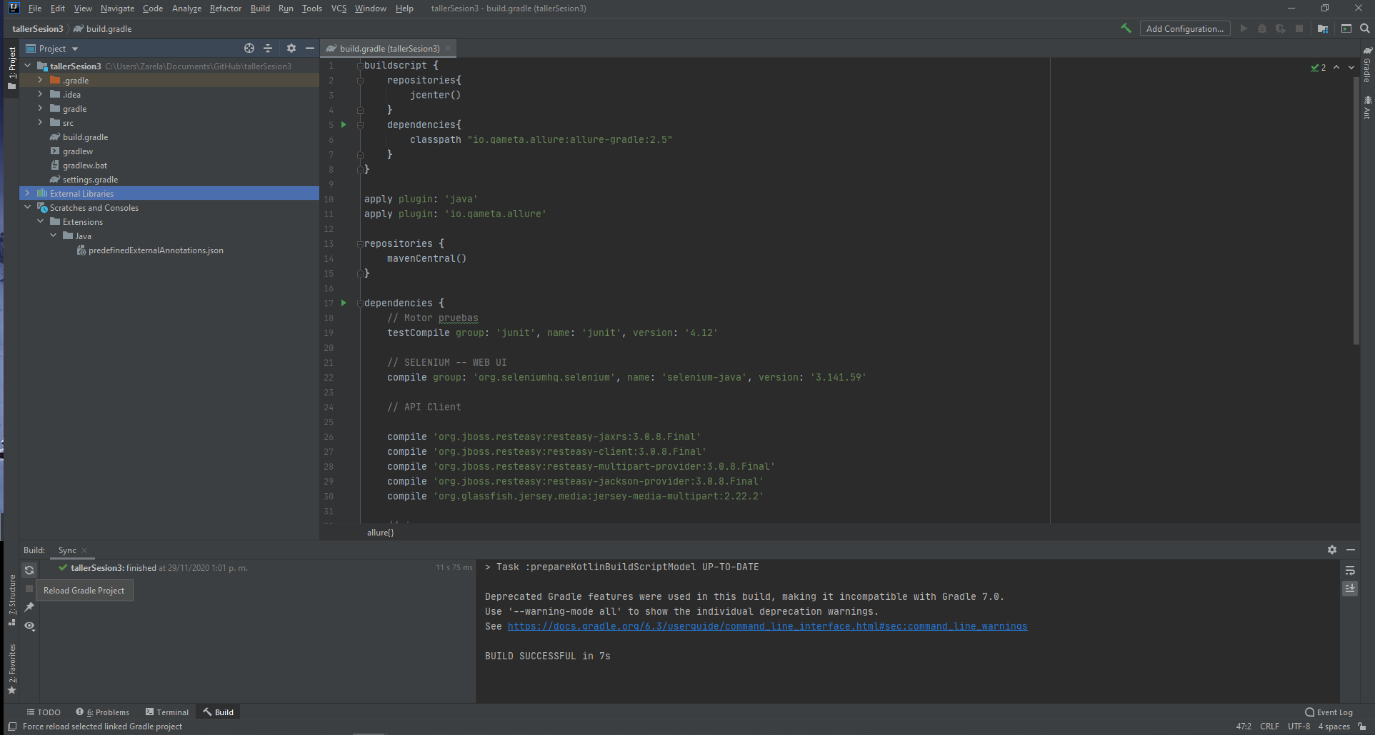
# **Agregar dependencias**

* 1. En el archivo build.gradle en la sección de dependencias agregar las siguientes dependencias:

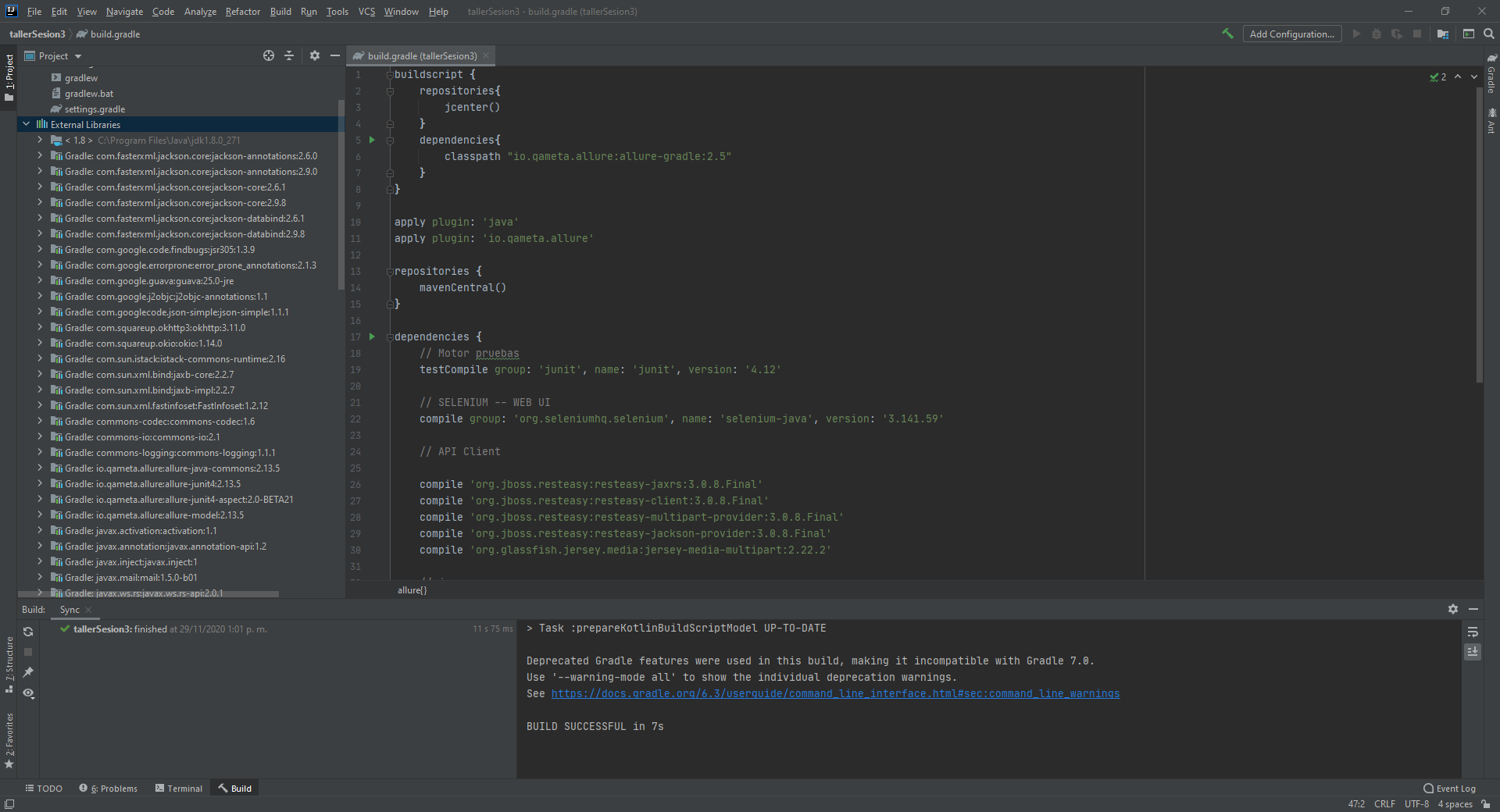
buildscript **{** repositories**{** jcenter()  
 **}** dependencies**{** classpath "io.qameta.allure:allure-gradle:2.5"  
 **}  
}**apply plugin: 'java'  
apply plugin: 'io.qameta.allure'  
  
repositories **{** mavenCentral()  
**}**dependencies **{** // Motor pruebas  
 testCompile group: 'junit', name: 'junit', version: '4.12'  
  
 // SELENIUM -- WEB UI  
 compile group: 'org.seleniumhq.selenium', name: 'selenium-java', version: '3.141.59'  
  
 // API Client  
  
 compile 'org.jboss.resteasy:resteasy-jaxrs:3.0.8.Final'  
 compile 'org.jboss.resteasy:resteasy-client:3.0.8.Final'  
 compile 'org.jboss.resteasy:resteasy-multipart-provider:3.0.8.Final'  
 compile 'org.jboss.resteasy:resteasy-jackson-provider:3.0.8.Final'  
 compile 'org.glassfish.jersey.media:jersey-media-multipart:2.22.2'  
  
 // json  
 compile 'com.googlecode.json-simple:json-simple:1.1.1'  
 compile 'org.json:json:20090211'  
 compile 'com.fasterxml.jackson.core:jackson-databind:2.6.1'  
  
 // Reportes  
 // https://mvnrepository.com/artifact/io.qameta.allure/allure-junit4  
 testCompile group: 'io.qameta.allure', name: 'allure-junit4', version: '2.13.5'  
  
**}**allure**{** autoconfigure = true  
 version = "2.6.0"  
 aspectjweaver = true  
**}**



* 1. Selecciona Reload Gradle Project para asegurarte que se cree las dependencias.

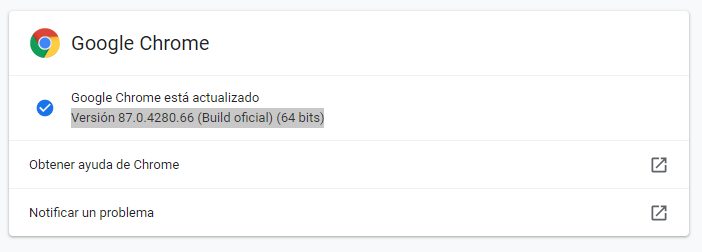


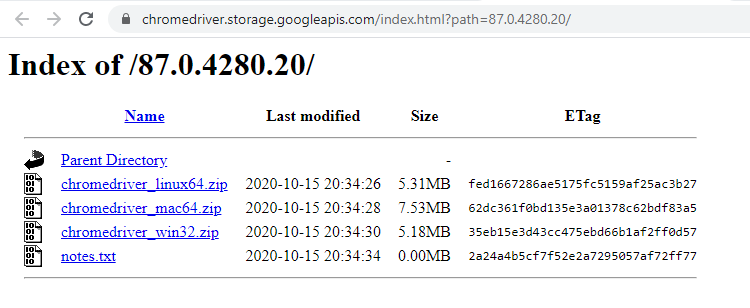
* 1. Para validar que se haya creado las dependencias. En la opción External Libraries se deben de mostrar las librerias.

**

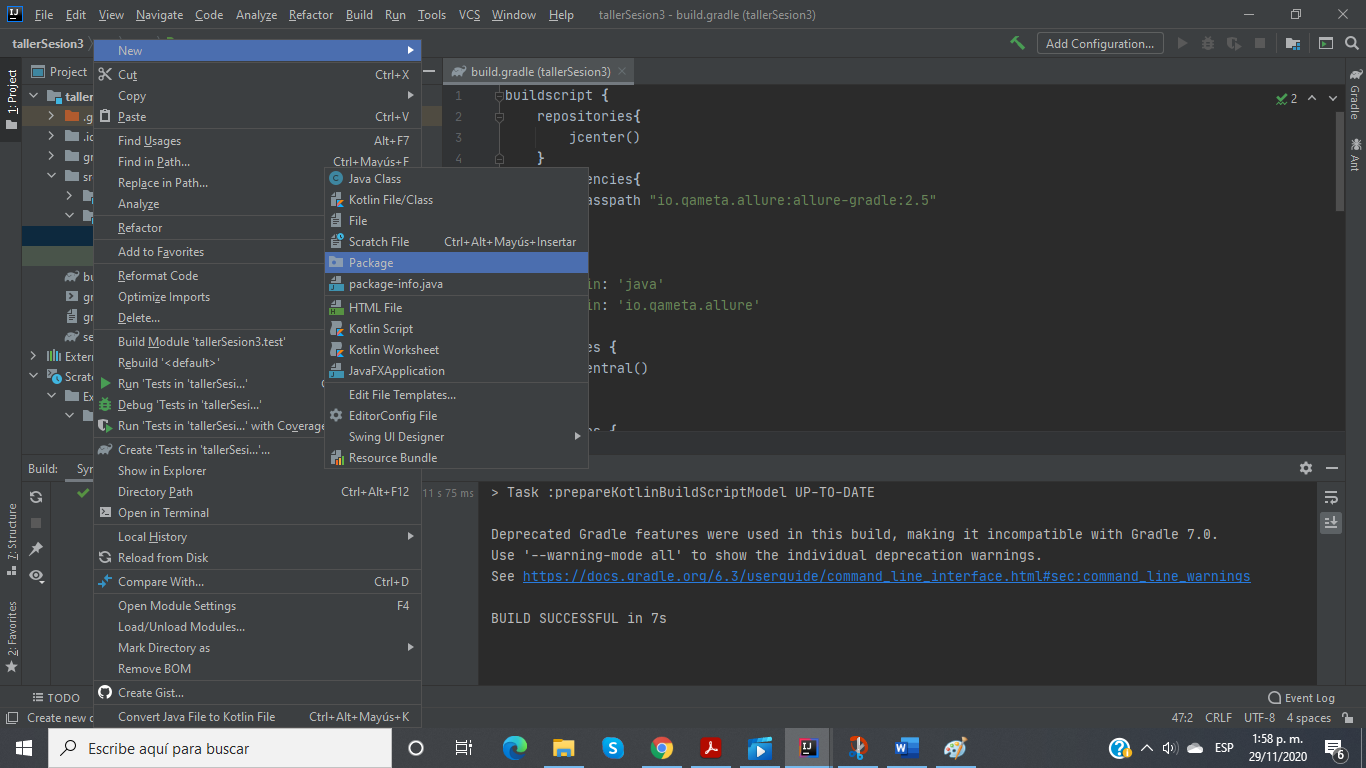
# **Agregar Driver – ChromeDriver**

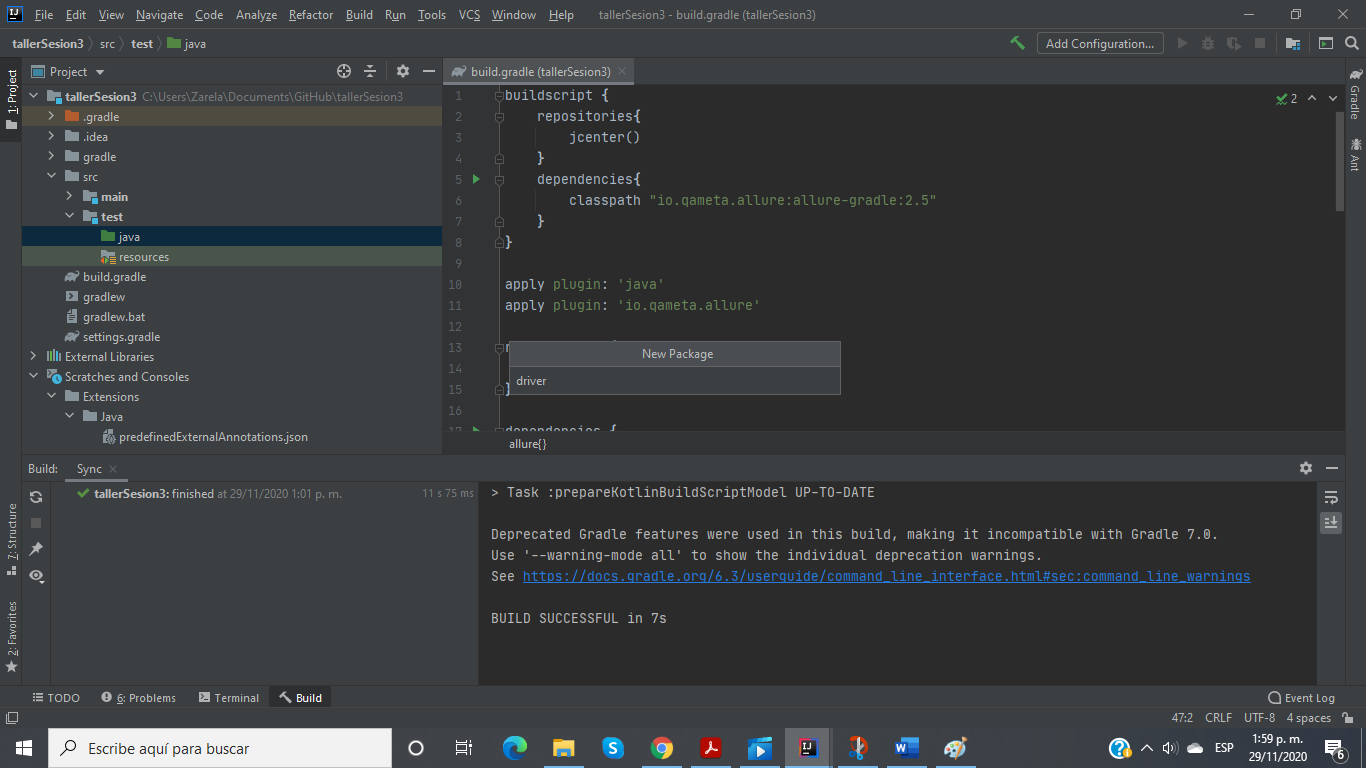
* 1. Buscar la versión del navegador Chrome que se encuentra en su máquina. Y descargar el driver de acuerdo al sistema operativo de la máquina.

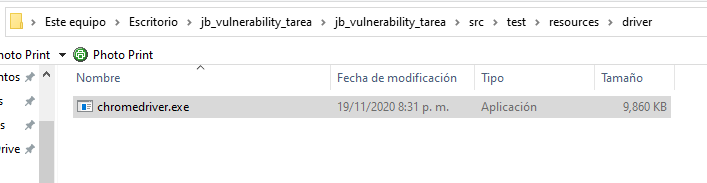


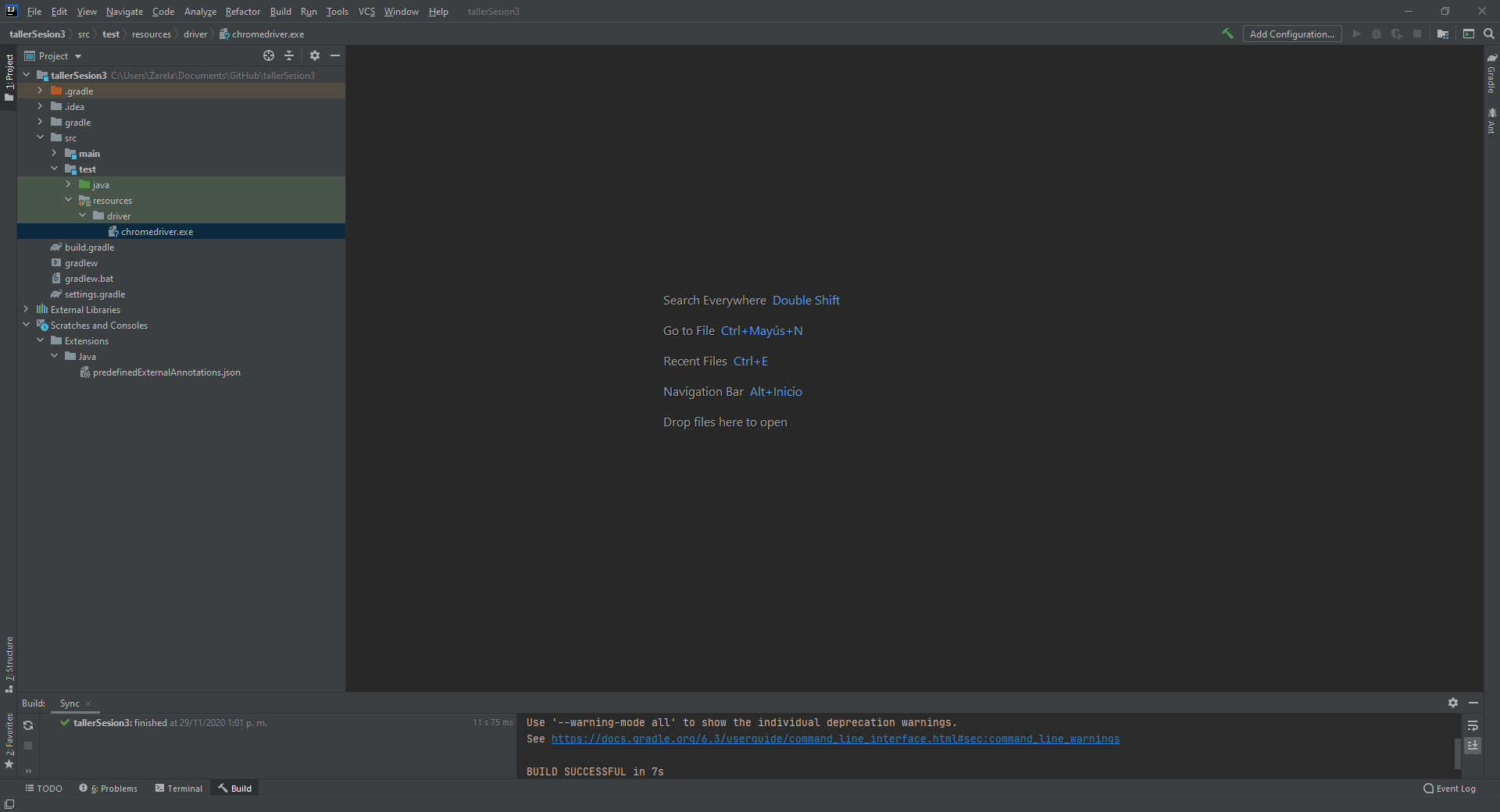


* 1. Crear una carpeta Driver en Test > Resources. En esta carpeta se agregará el driver descomprimido.



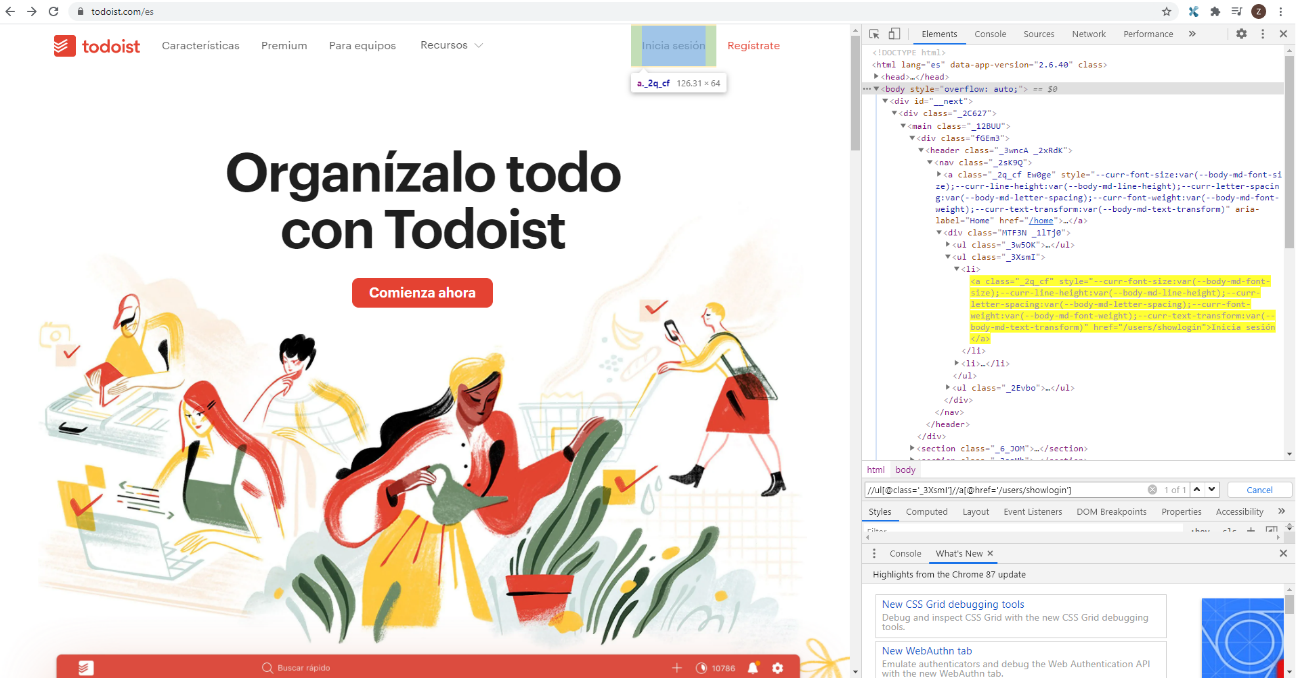




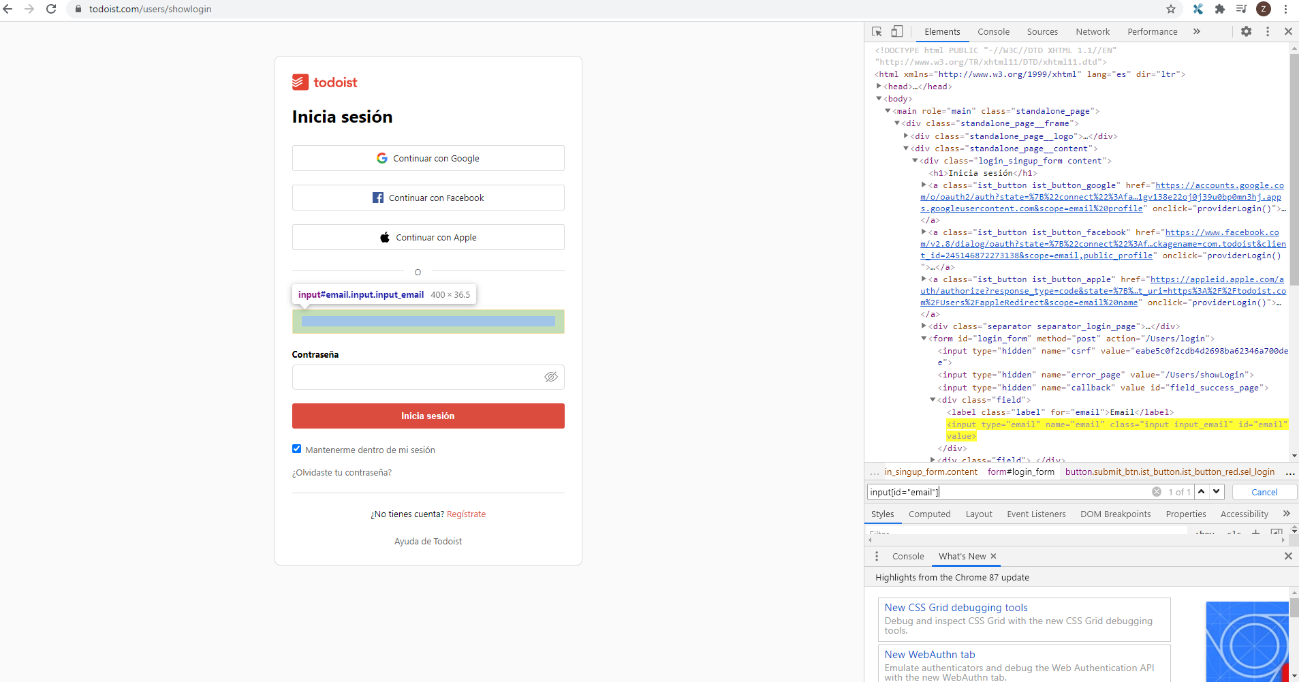


# **Buscar Localizadores página https://todoist.com/**

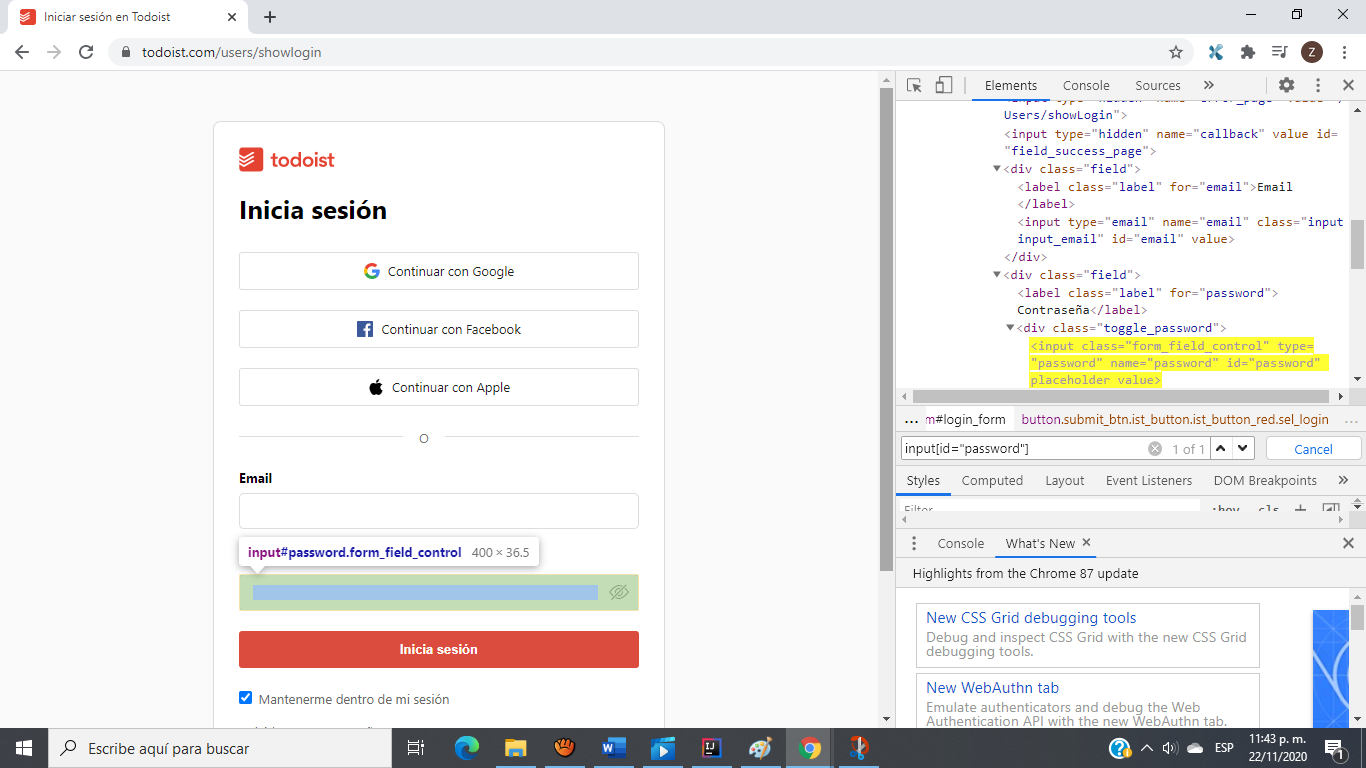
*loginFreeLabel = “xpath://ul[@class='\_3XsmI']//a[@href='/users/showlogin']”*



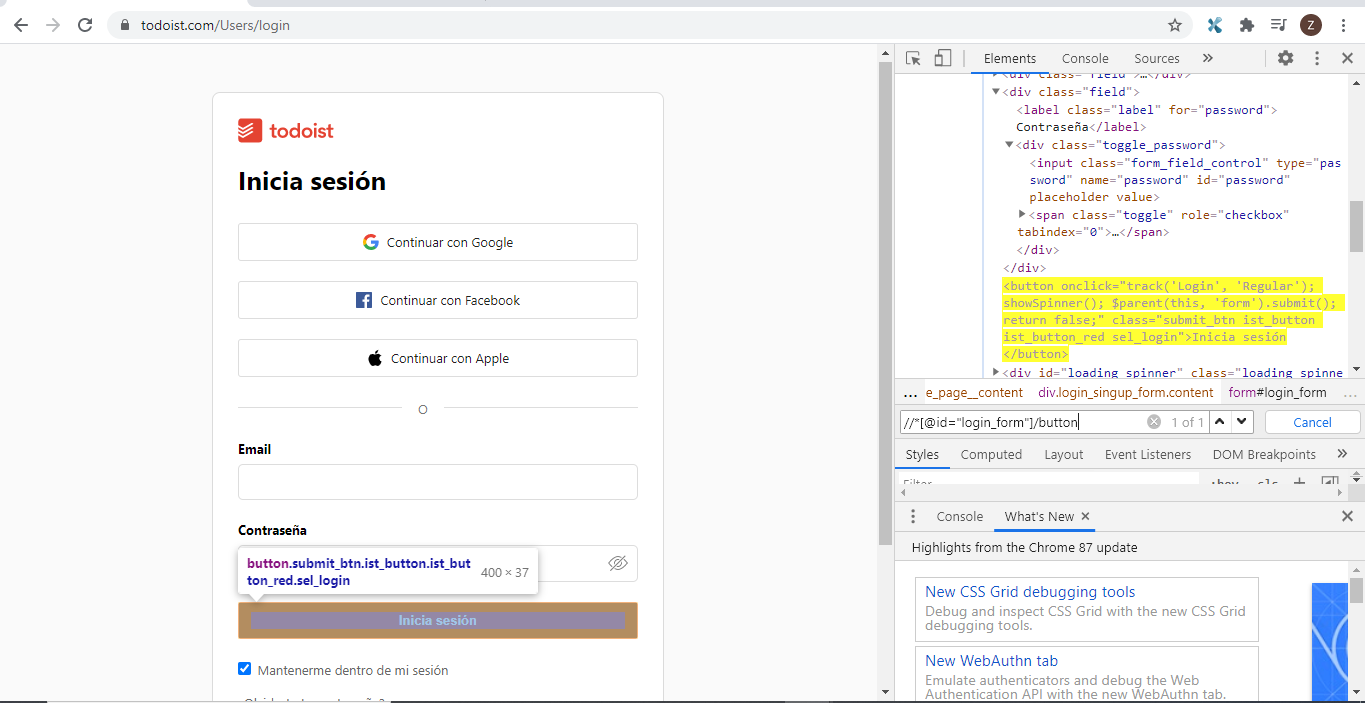
*Email = “id:email”*



*Contraseña = “id:password”*

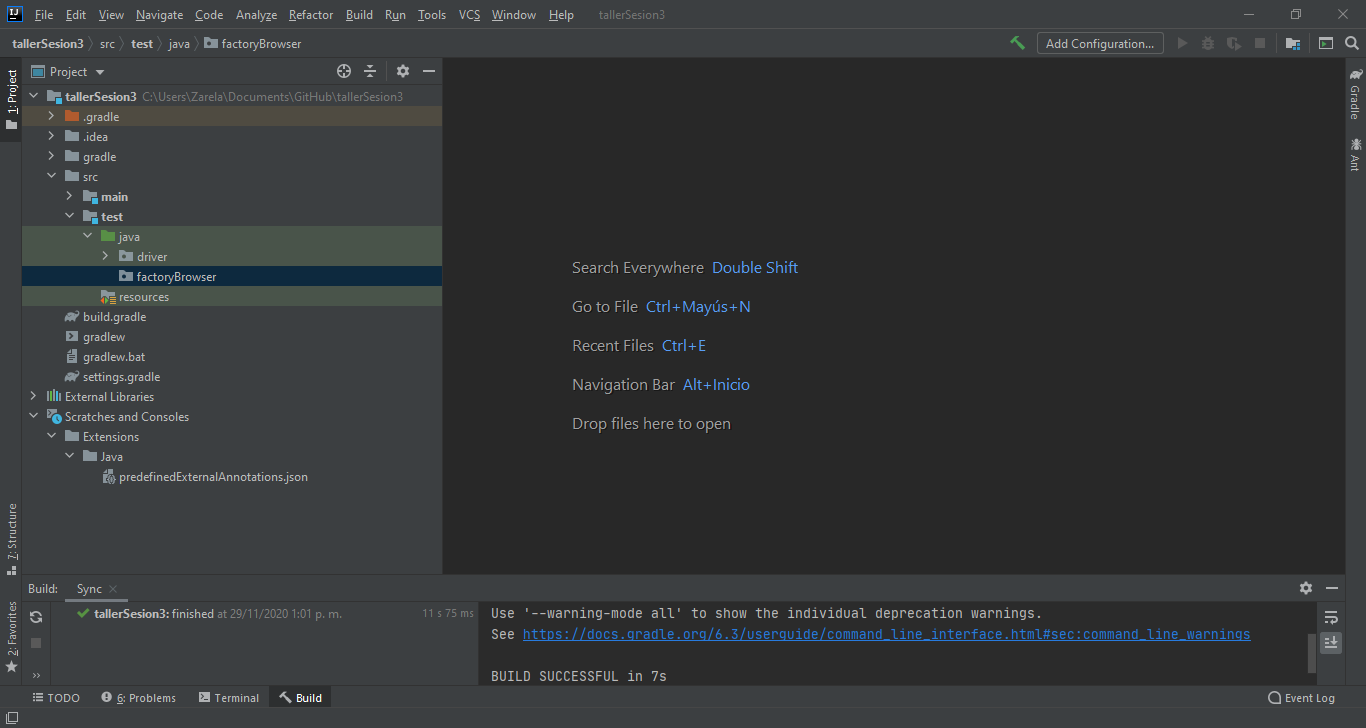


*IniciaSesion = “xpath: //\*[@id="login\_form"]/button”*



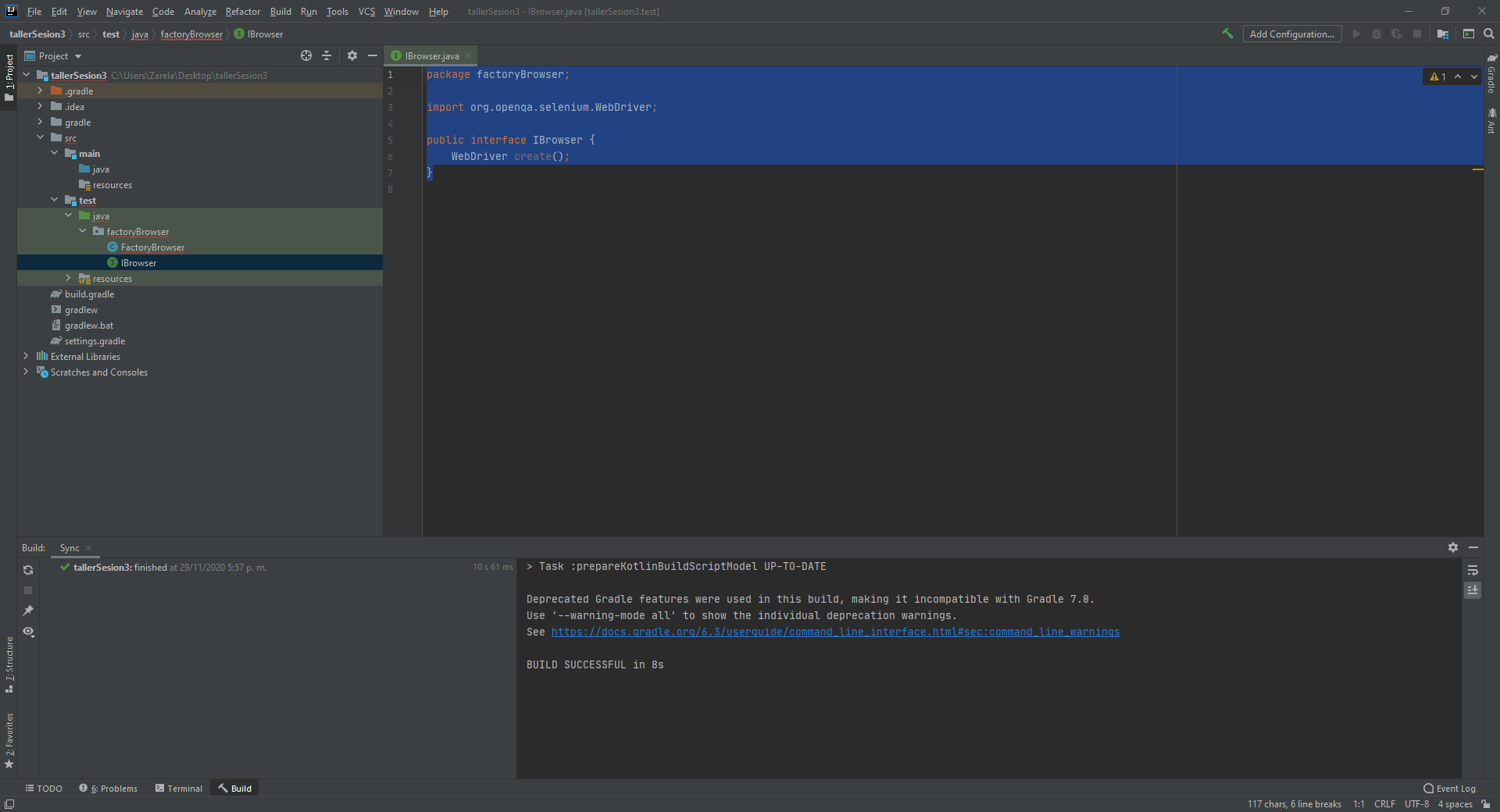
# **Implementación Package factoryBrowser**

* 1. Crear el Package factoryBrowser en la siguiente ruta: test>Java



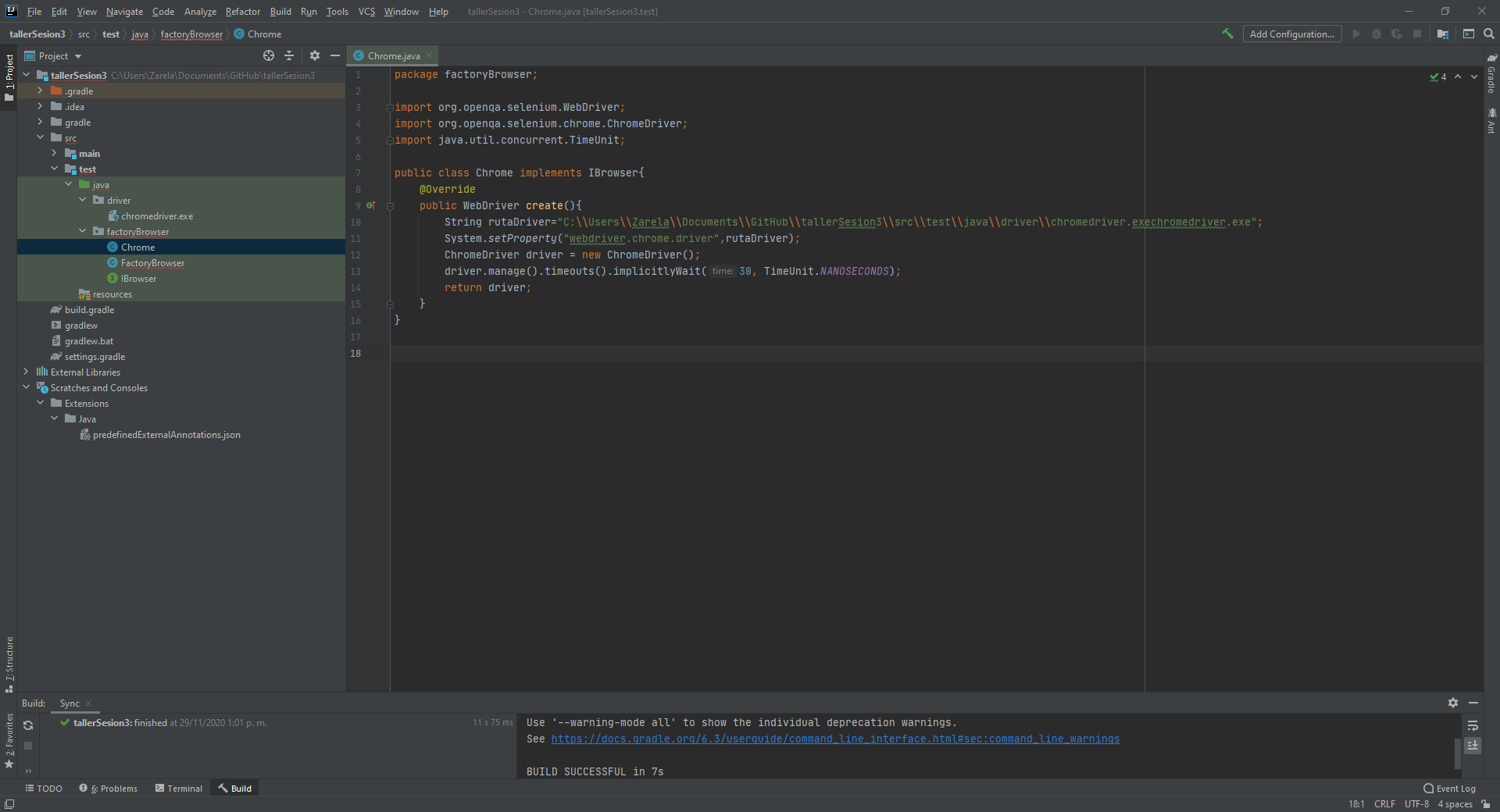
* 1. Crear la Interface IBrowser en la siguiente ruta: test>java>factoryBrowser

package factoryBrowser;  
  
import org.openqa.selenium.WebDriver;  
  
public interface IBrowser {  
 WebDriver create();  
}



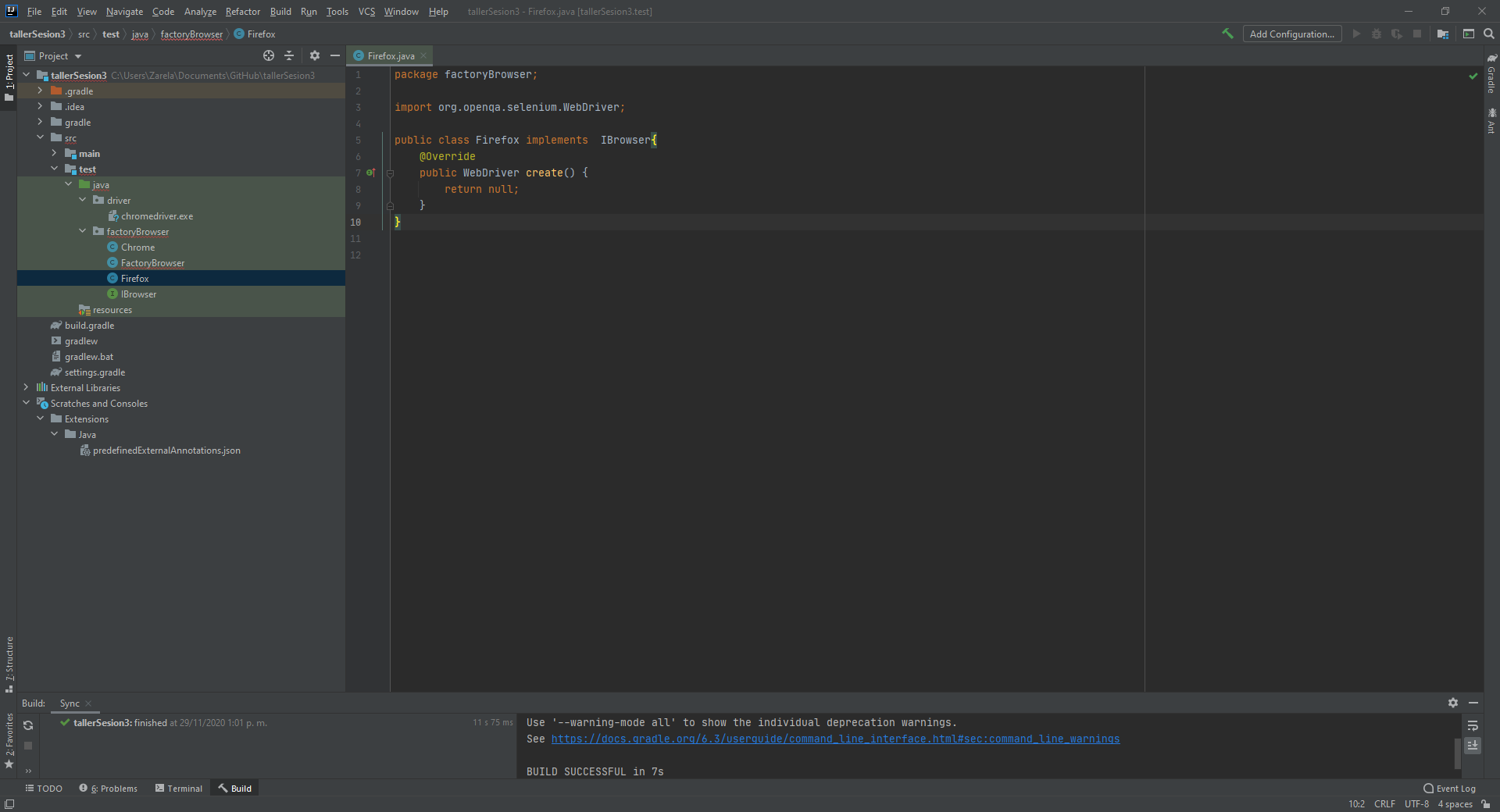
* 1. Crear la Clase Chrome en la siguiente ruta: test>java>factoryBrowser

package factoryBrowser;  
  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.chrome.ChromeDriver;  
import java.util.concurrent.TimeUnit;  
  
public class Chrome implements IBrowser{  
 @Override  
 public WebDriver create(){  
 String rutaDriver="C:\\Users\\Zarela\\Desktop\\tallerSesion3\\src\\test\\resources\\driver\\chromedriver.exe";  
 System.*setProperty*("webdriver.chrome.driver",rutaDriver);  
 ChromeDriver driver = new ChromeDriver();  
 driver.manage().timeouts().implicitlyWait(30, TimeUnit.*NANOSECONDS*);  
 return driver;  
 }  
}



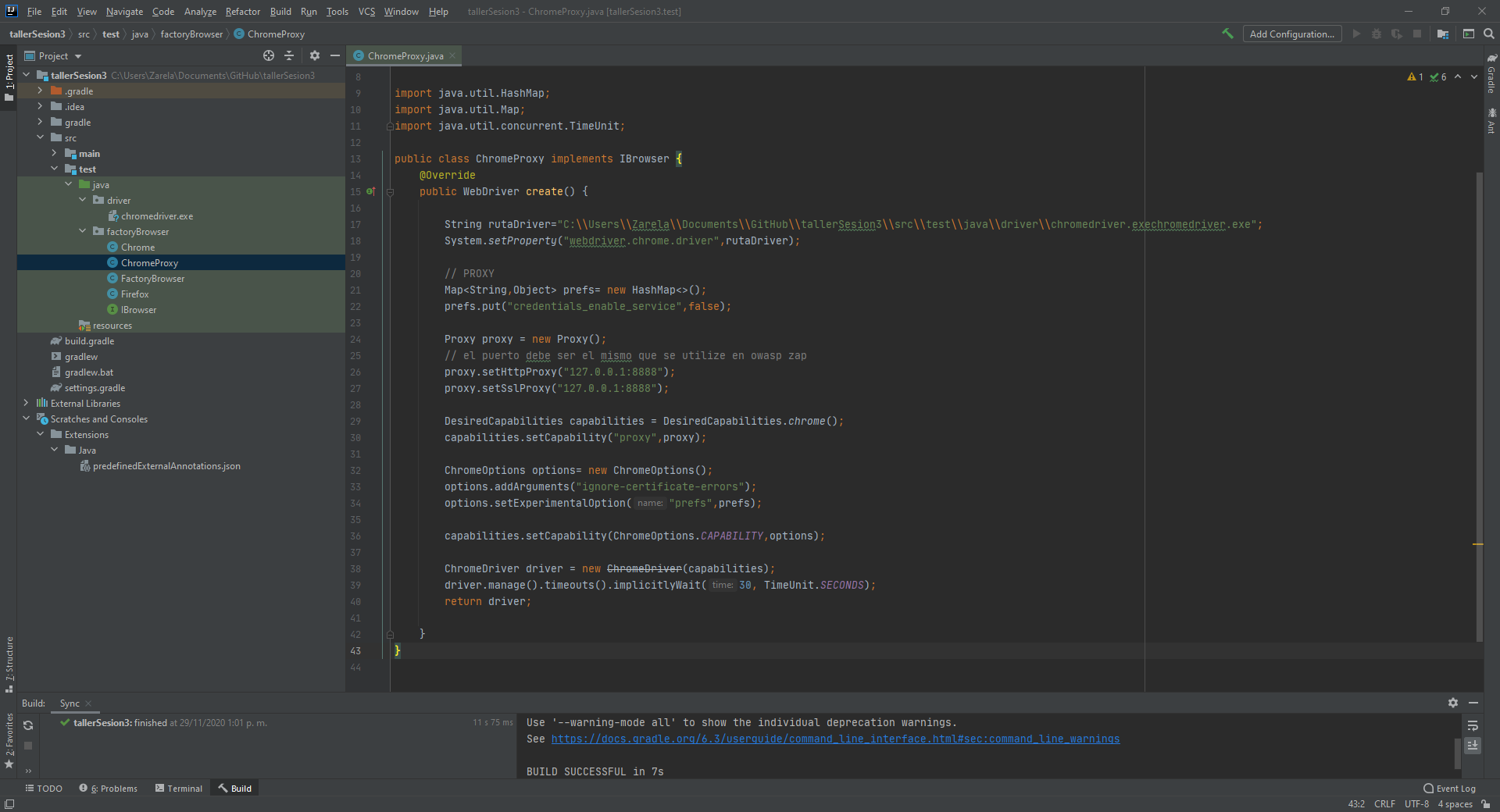
* 1. Crear la Clase Firefox en la siguiente ruta: test>java>factoryBrowser

package factoryBrowser;  
  
import org.openqa.selenium.WebDriver;  
  
public class Firefox implements IBrowser{  
 @Override  
 public WebDriver create() {  
 return null;  
 }  
}



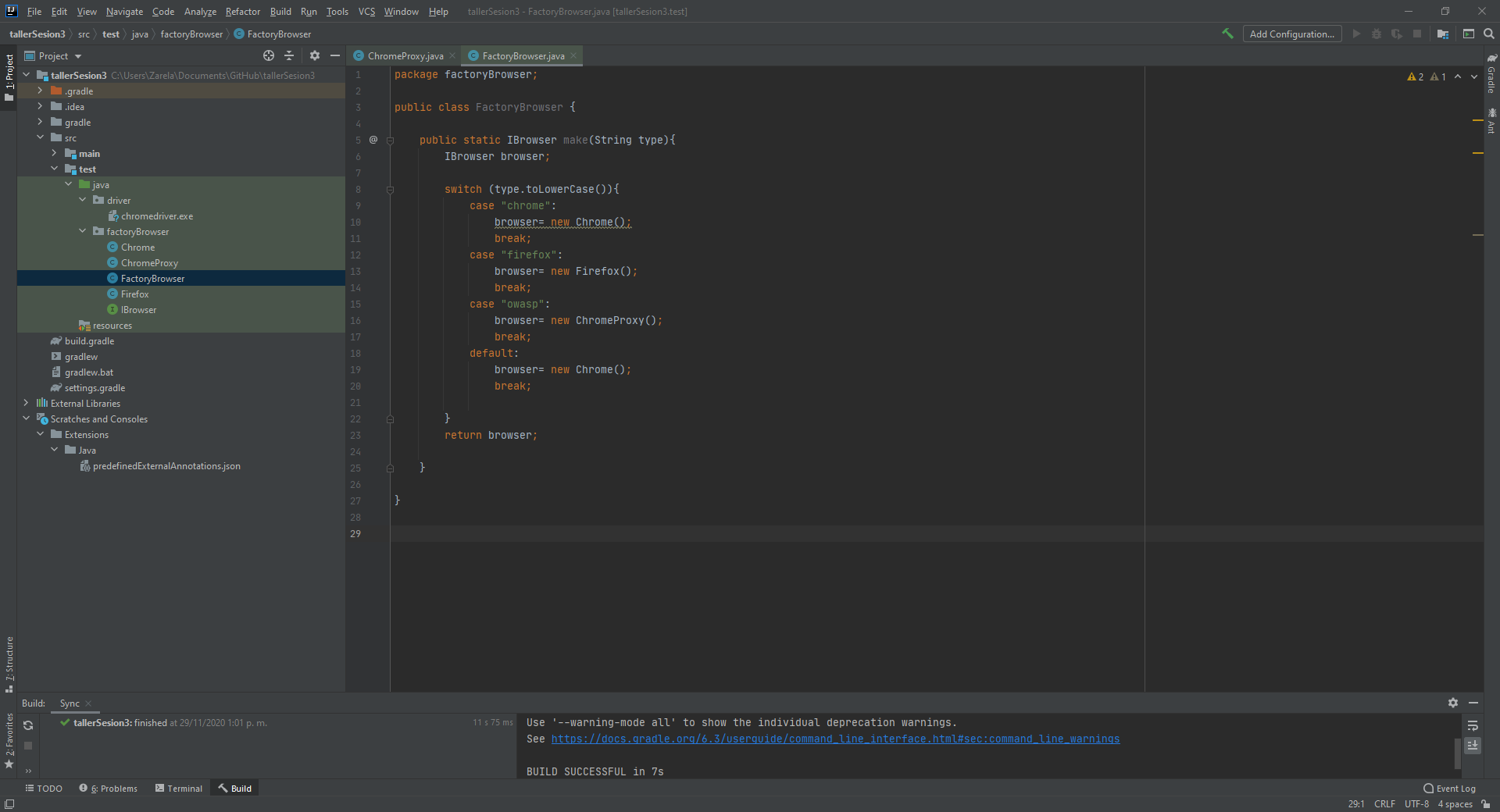
* 1. Crear la Clase ChromeProxy en la siguiente ruta: test>java>factoryBrowser

package factoryBrowser;  
  
import org.openqa.selenium.Proxy;  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.chrome.ChromeDriver;  
import org.openqa.selenium.chrome.ChromeOptions;  
import org.openqa.selenium.remote.DesiredCapabilities;  
  
import java.util.HashMap;  
import java.util.Map;  
import java.util.concurrent.TimeUnit;  
  
public class ChromeProxy implements IBrowser {  
 @Override  
 public WebDriver create() {  
  
 String rutaDriver="C:\\Users\\Zarela\\Documents\\GitHub\\tallerSesion3\\src\\test\\java\\driver\\chromedriver.exechromedriver.exe";  
 System.*setProperty*("webdriver.chrome.driver",rutaDriver);  
  
 // PROXY  
 Map<String,Object> prefs= new HashMap<>();  
 prefs.put("credentials\_enable\_service",false);  
  
 Proxy proxy = new Proxy();  
 // el puerto debe ser el mismo que se utilize en owasp zap  
 proxy.setHttpProxy("127.0.0.1:8888");  
 proxy.setSslProxy("127.0.0.1:8888");  
  
 DesiredCapabilities capabilities = DesiredCapabilities.*chrome*();  
 capabilities.setCapability("proxy",proxy);  
  
 ChromeOptions options= new ChromeOptions();  
 options.addArguments("ignore-certificate-errors");  
 options.setExperimentalOption("prefs",prefs);  
  
 capabilities.setCapability(ChromeOptions.*CAPABILITY*,options);  
  
 ChromeDriver driver = new ChromeDriver(capabilities);  
 driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);  
 return driver;  
  
 }  
}



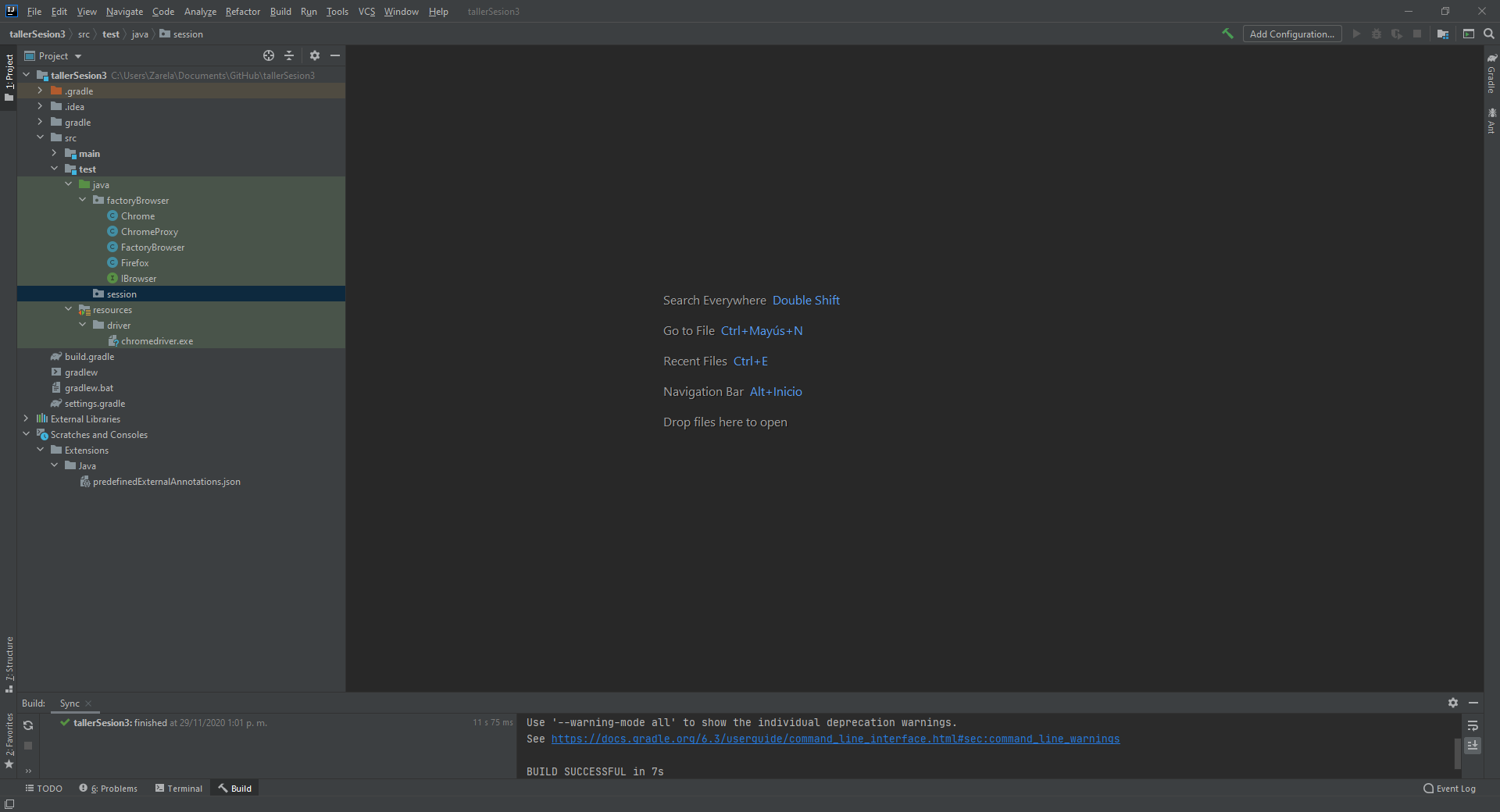
* 1. Crear la Clase FactoryBrowser en la siguiente ruta: test>java>factoryBrowser

package factoryBrowser;  
  
public class FactoryBrowser {  
  
 public static IBrowser make(String type){  
 IBrowser browser;  
  
 switch (type.toLowerCase()){  
 case "chrome":  
 browser= new Chrome();  
 break;  
 case "firefox":  
 browser= new Firefox();  
 break;  
 case "owasp":  
 browser= new ChromeProxy();  
 break;  
 default:  
 browser= new Chrome();  
 break;  
  
 }  
 return browser;  
  
 }  
  
}



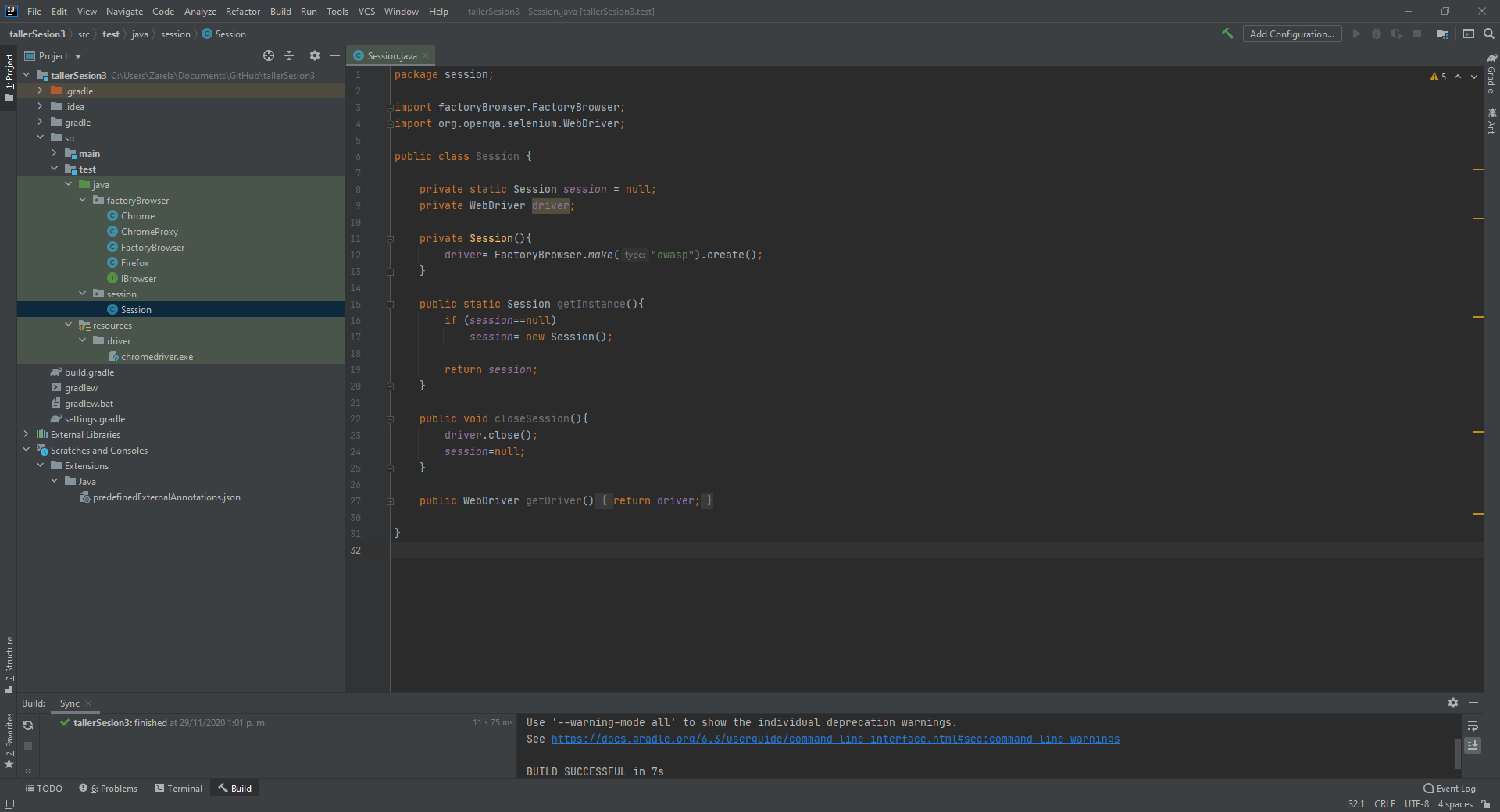
# **Implementación Package session**

* 1. Crear el Package Session en la siguiente ruta: test>Java



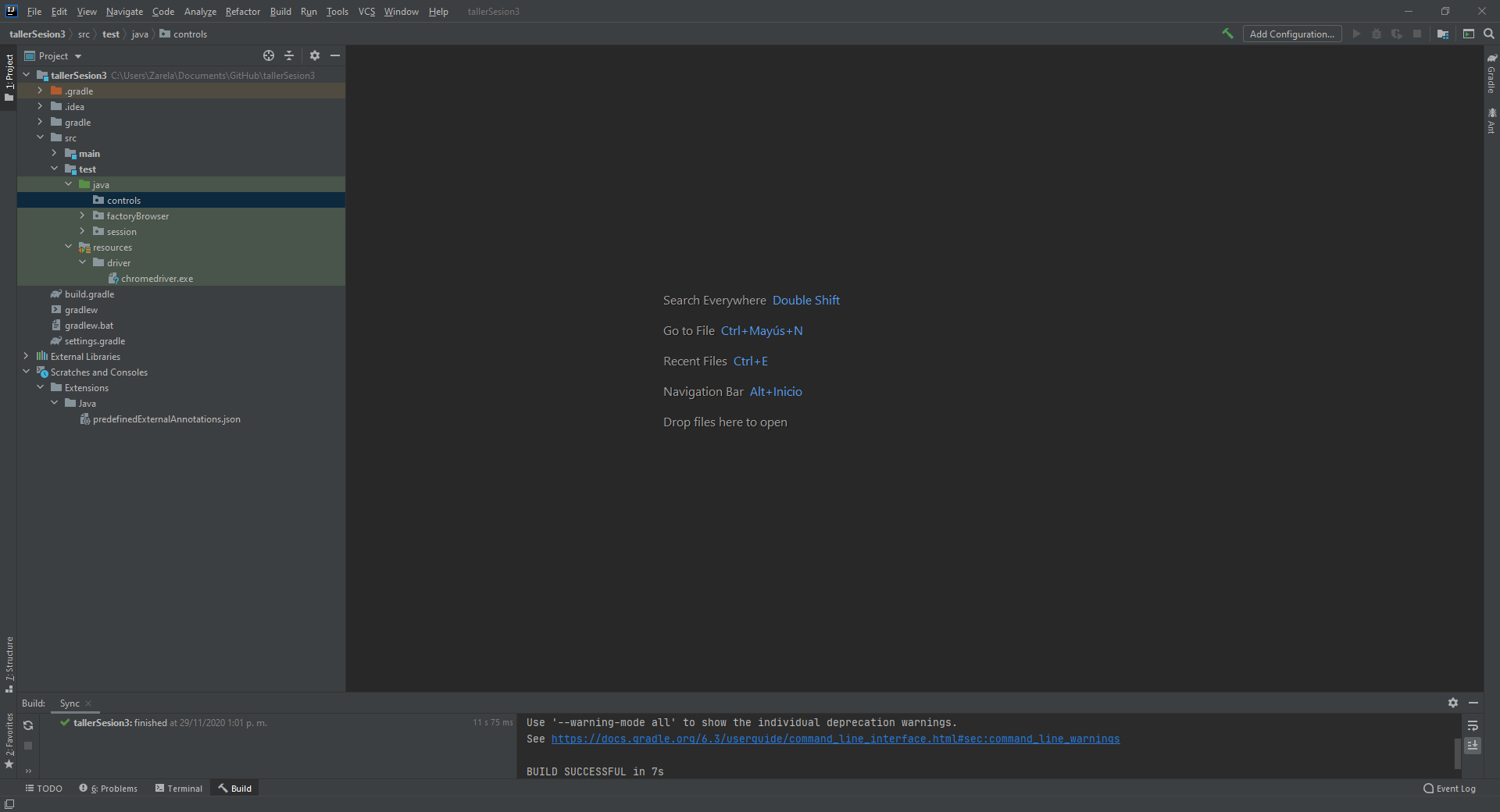
* 1. Crear la clase Session en la siguiente ruta: test>java>session

package session;  
  
import factoryBrowser.FactoryBrowser;  
import org.openqa.selenium.WebDriver;  
  
public class Session {  
  
 private static Session *session* = null;  
 private WebDriver driver;  
  
 private Session(){  
 driver= FactoryBrowser.*make*("owasp").create();  
 }  
  
 public static Session getInstance(){  
 if (*session*==null)  
 *session*= new Session();  
  
 return *session*;  
 }  
  
 public void closeSession(){  
 driver.close();  
 *session*=null;  
 }  
  
 public WebDriver getDriver() {  
 return driver;  
 }  
  
}



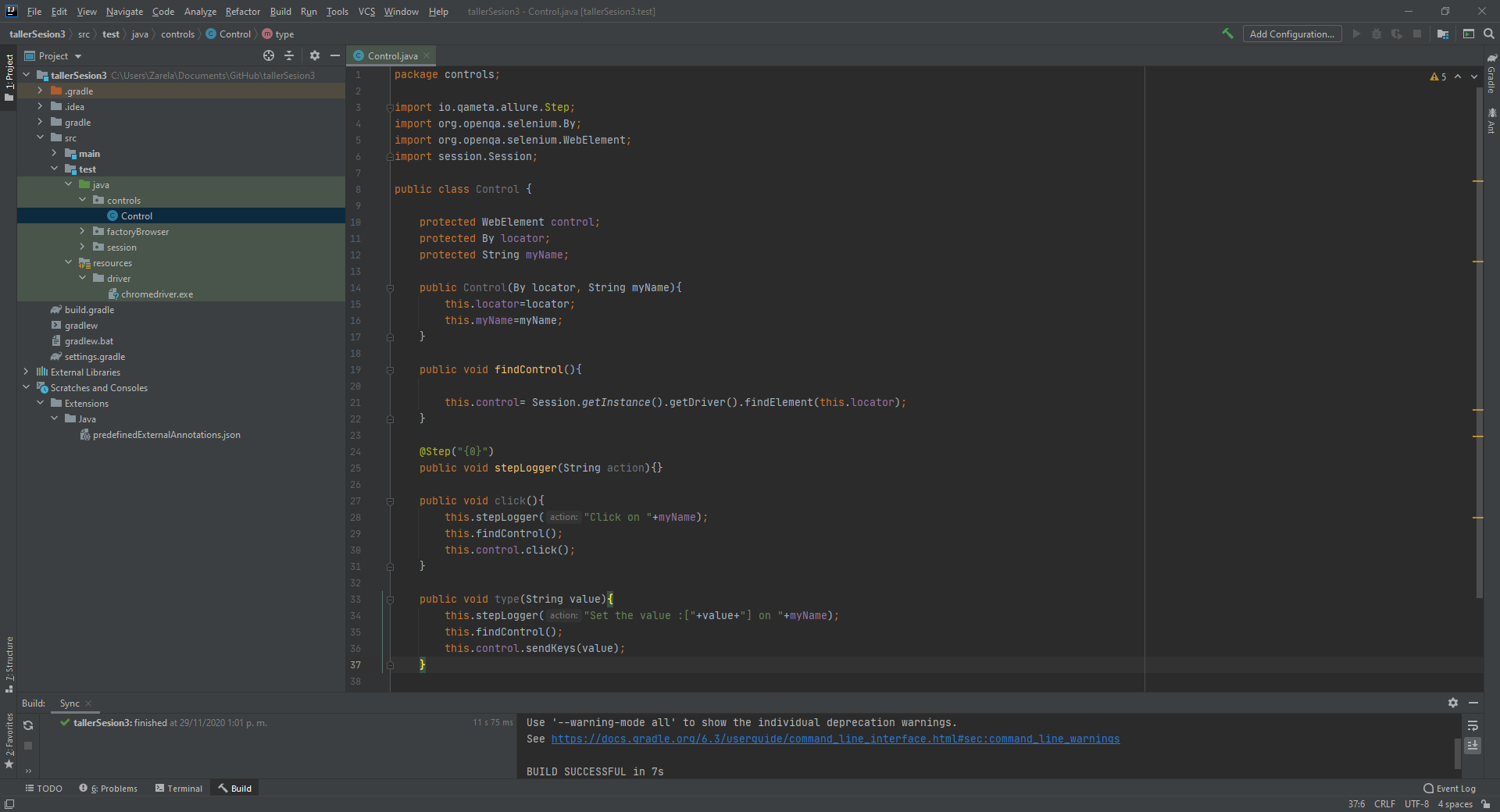
# **Implementación Package controls**

* 1. Crear el Package controls en la siguiente ruta: test>Java



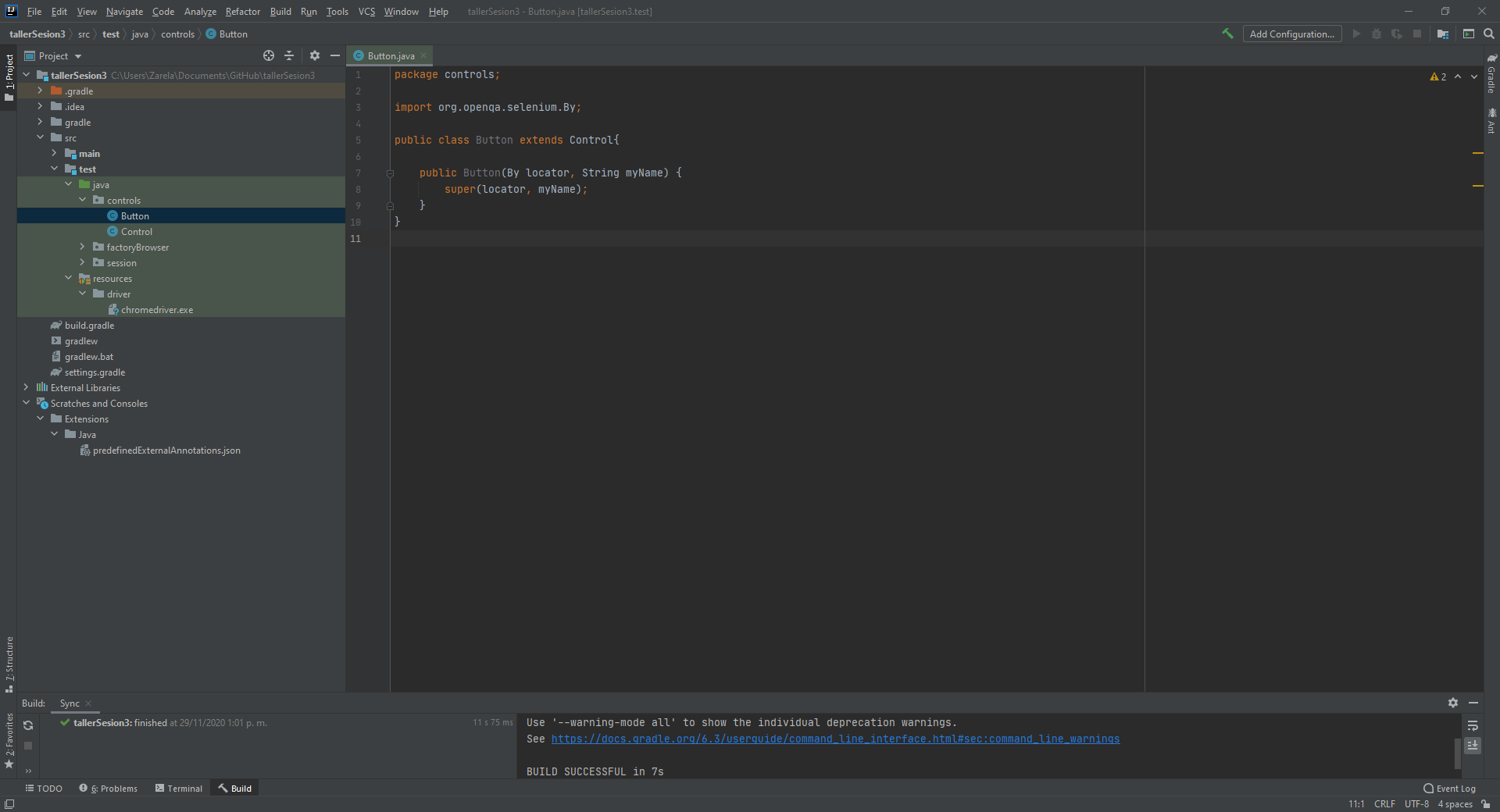
* 1. Crear la clase Control en la siguiente ruta: test>java>controls

package controls;  
  
import io.qameta.allure.Step;  
import org.openqa.selenium.By;  
import org.openqa.selenium.WebElement;  
import session.Session;  
  
public class Control {  
  
 protected WebElement control;  
 protected By locator;  
 protected String myName;  
  
 public Control(By locator, String myName){  
 this.locator=locator;  
 this.myName=myName;  
 }  
  
 public void findControl(){  
  
 this.control= Session.*getInstance*().getDriver().findElement(this.locator);  
 }  
  
 @Step("{0}")  
 public void stepLogger(String action){}  
  
 public void click(){  
 this.stepLogger("Click on "+myName);  
 this.findControl();  
 this.control.click();  
 }  
  
 public void type(String value){  
 this.stepLogger("Set the value :["+value+"] on "+myName);  
 this.findControl();  
 this.control.sendKeys(value);  
 }  
  
}



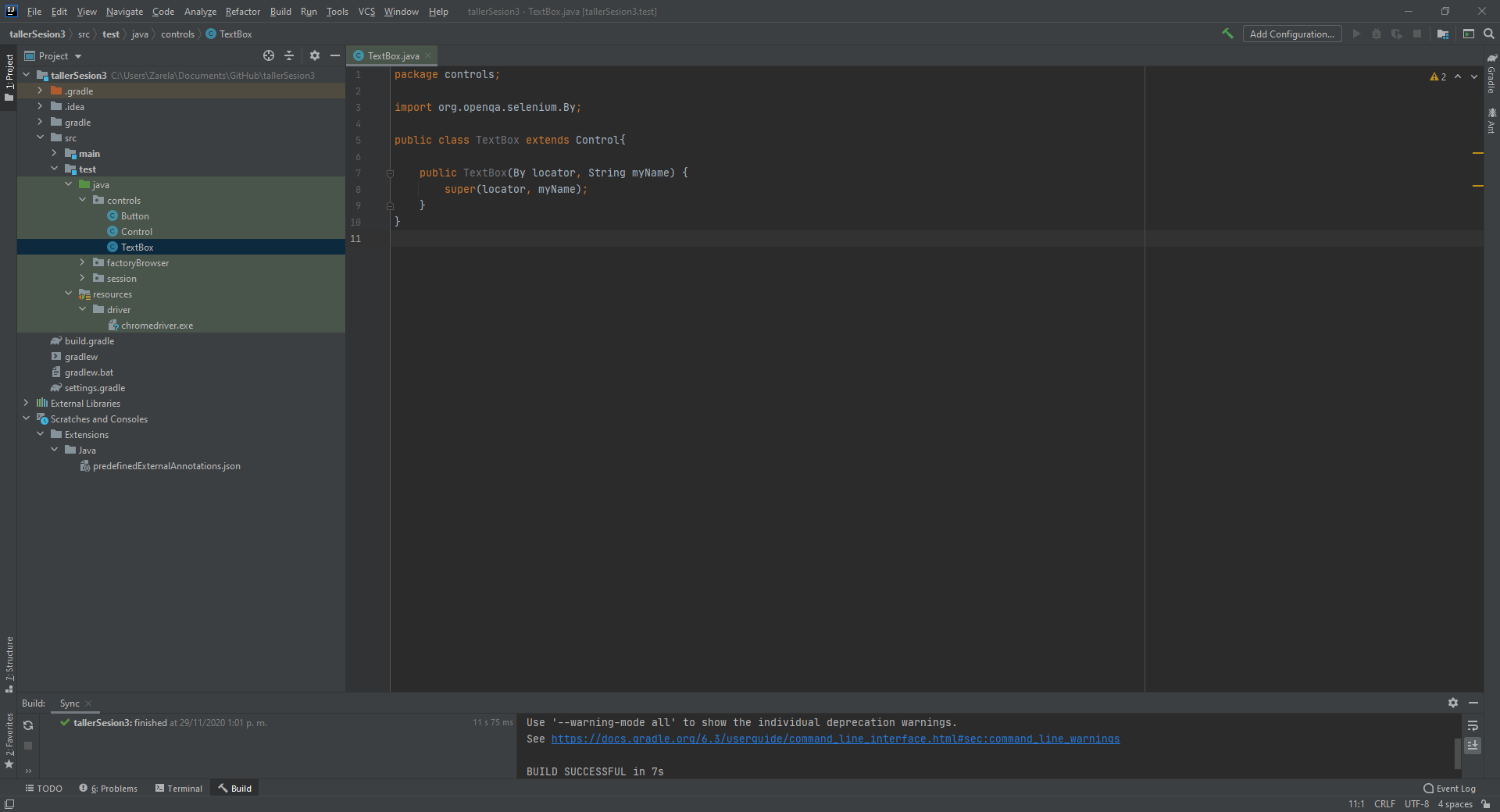
* 1. Crear la clase Button en la siguiente ruta: test>java> controls

package controls;  
  
import org.openqa.selenium.By;  
  
public class Button extends Control{  
  
 public Button(By locator, String myName) {  
 super(locator, myName);  
 }  
}



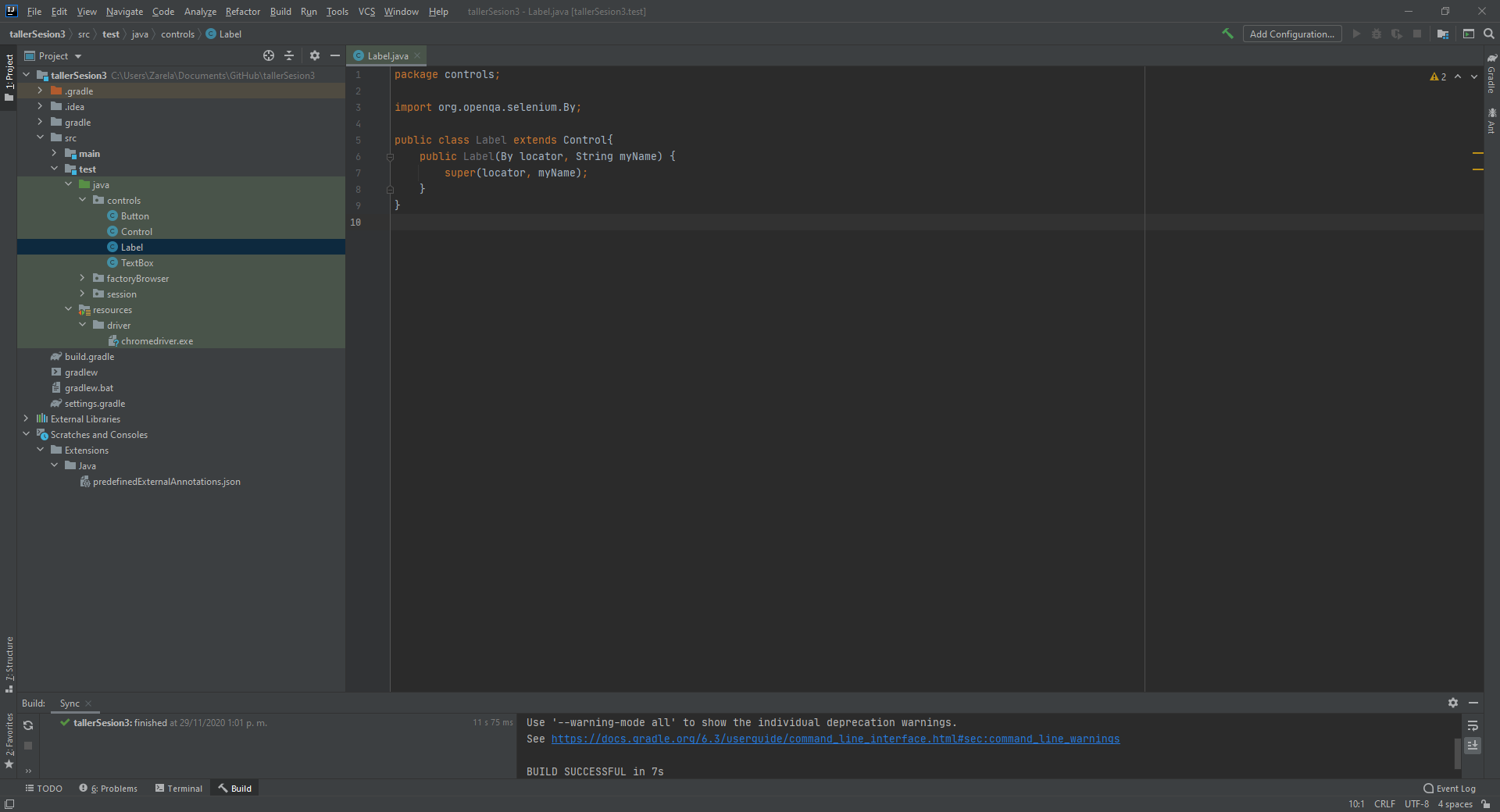
* 1. Crear la clase TextBox en la siguiente ruta: test>java> controls

package controls;  
  
import org.openqa.selenium.By;  
  
public class TextBox extends Control{  
  
 public TextBox(By locator, String myName) {  
 super(locator, myName);  
 }  
}



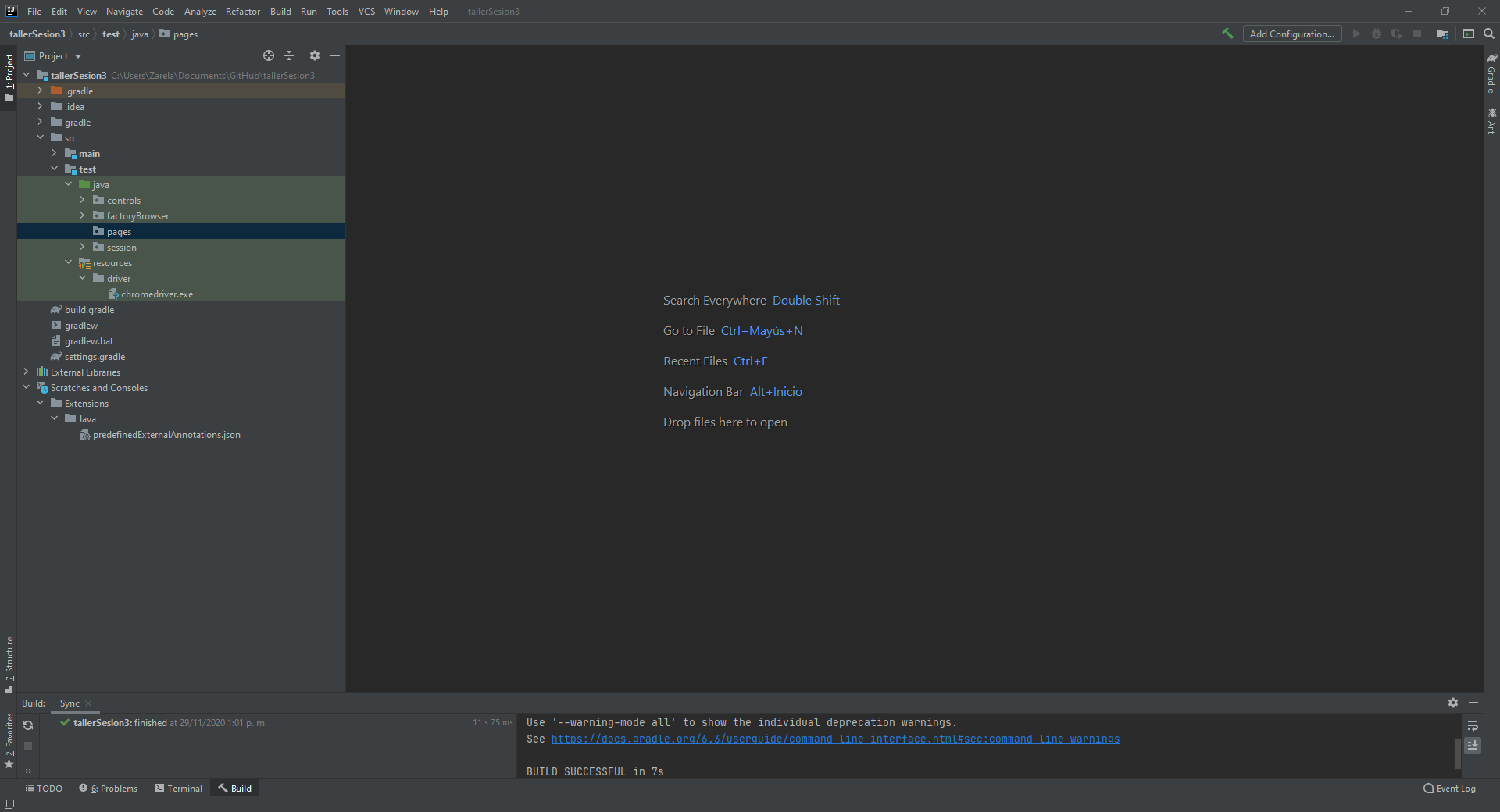
* 1. Crear la clase Label en la siguiente ruta: test>java> controls

package controls;  
  
import org.openqa.selenium.By;  
  
public class Label extends Control{  
 public Label(By locator, String myName) {  
 super(locator, myName);  
 }  
}



# **Implementación Package pages**

* 1. Crear el Package pages en la siguiente ruta: test>Java



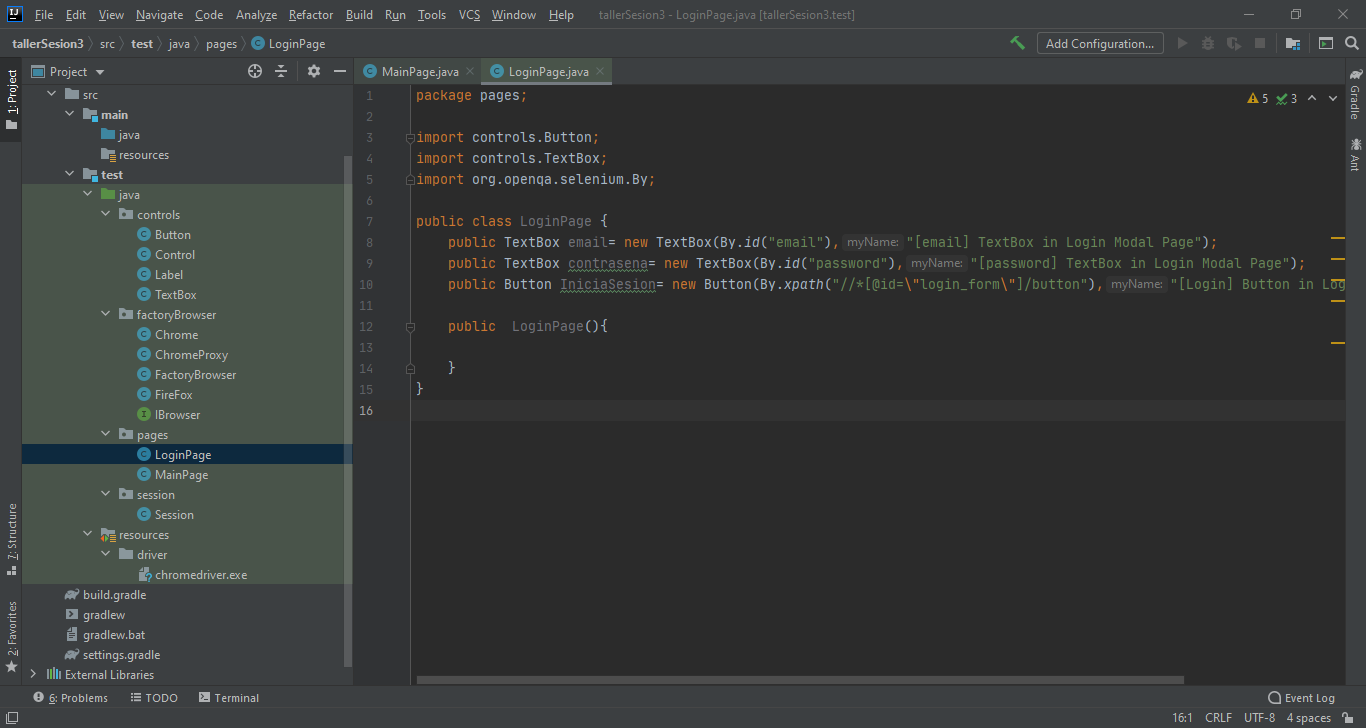
* 1. Crear la clase MainPage en la siguiente ruta: test>java> pages

package pages;  
  
import controls.Label;  
import org.openqa.selenium.By;  
  
public class MainPage {  
  
 public Label singUpFreeLabel = new Label(By.*xpath*("//ul[@class='\_3XsmI']//a[@href='/users/showlogin']")," [Login] Label in Main Page");  
  
 public MainPage(){}  
  
}



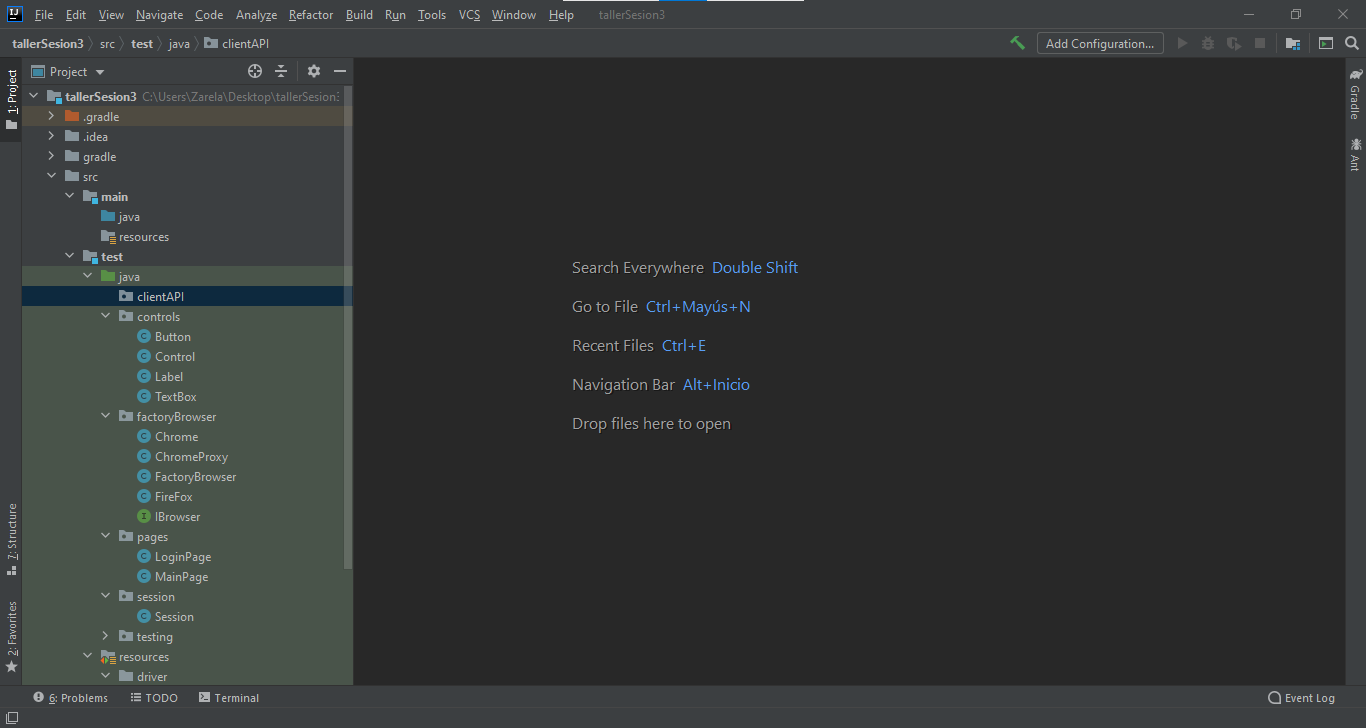
* 1. Crear la clase LoginPage en la siguiente ruta: test>java> pages

package pages;  
  
import controls.Button;  
import controls.TextBox;  
import org.openqa.selenium.By;  
  
public class LoginPage {  
 public TextBox email= new TextBox(By.*id*("email"),"[email] TextBox in Login Modal Page");  
 public TextBox contrasena= new TextBox(By.*id*("password"),"[password] TextBox in Login Modal Page");  
 public Button IniciaSesion= new Button(By.*xpath*("//\*[@id=\"login\_form\"]/button"),"[Login] Button in Login Modal Page");  
  
 public LoginPage(){  
  
 }  
}



# **Implementación Package clientAPI**

* 1. Crear el Package testing en la siguiente ruta: test>Java



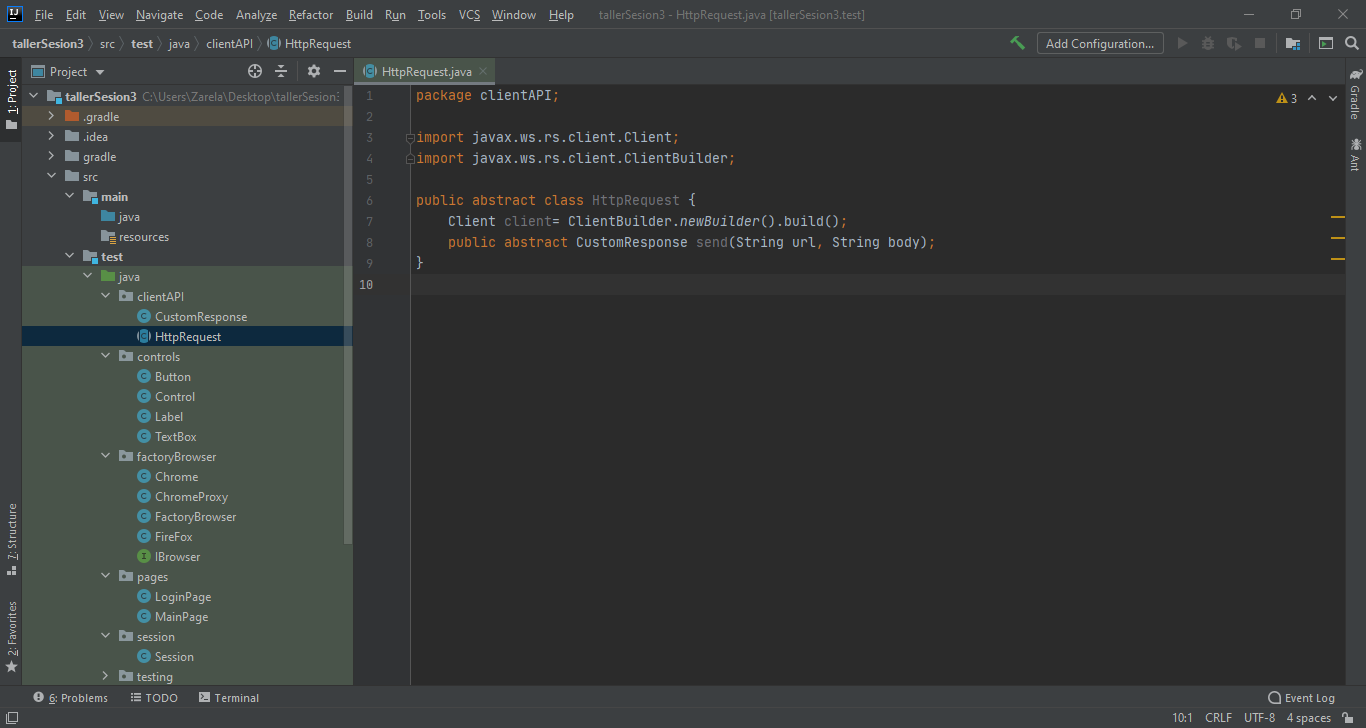
* 1. Crear la clase CustomResponse en la siguiente ruta: test>java> clientAPI

package clientAPI;  
  
public class CustomResponse {  
 private String responseCode;  
 private String responseBody;  
  
 public CustomResponse(){}  
  
 public String getResponseCode() {  
 return responseCode;  
 }  
  
 public void setResponseCode(String responseCode) {  
 this.responseCode = responseCode;  
 }  
  
 public String getResponseBody() {  
 return responseBody;  
 }  
  
 public void setResponseBody(String responseBody) {  
 this.responseBody = responseBody;  
 }  
}



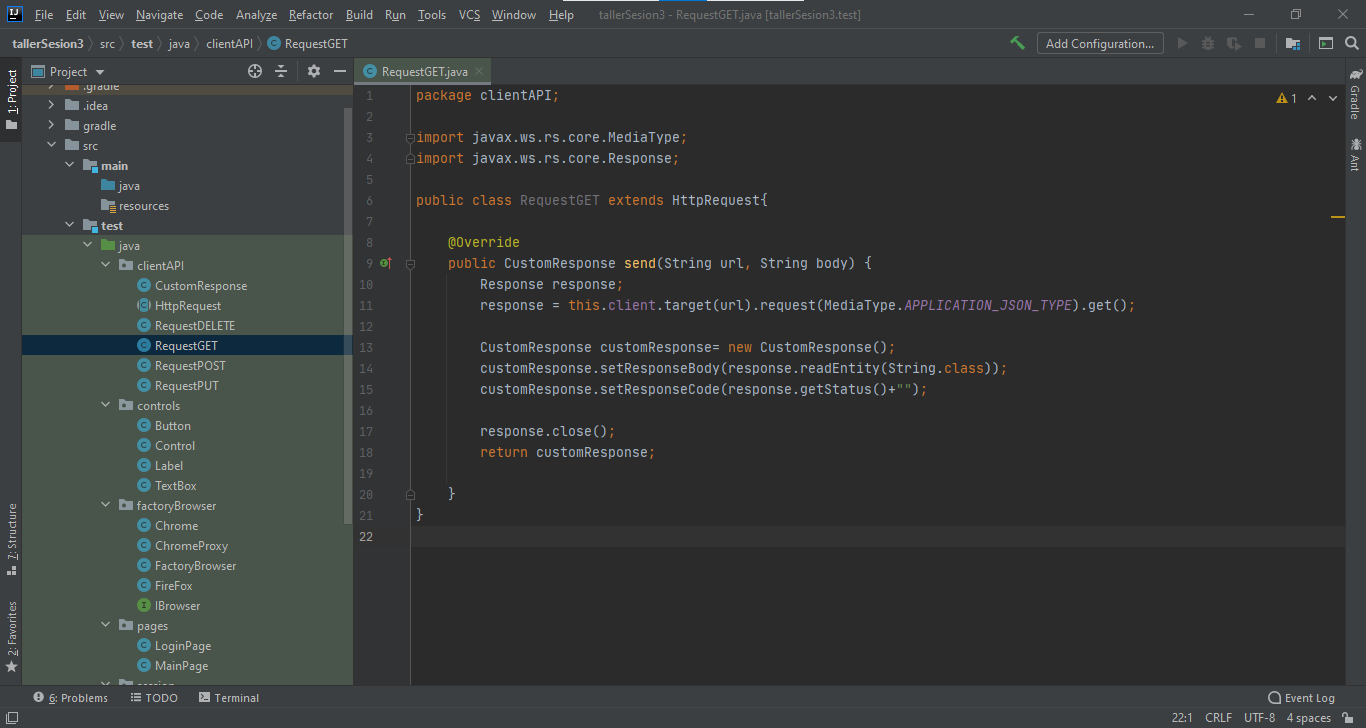
* 1. Crear la clase HttpRequest en la siguiente ruta: test>java> clientAPI

package clientAPI;  
  
import javax.ws.rs.client.Client;  
import javax.ws.rs.client.ClientBuilder;  
  
public abstract class HttpRequest {  
 Client client= ClientBuilder.*newBuilder*().build();  
 public abstract CustomResponse send(String url, String body);  
}



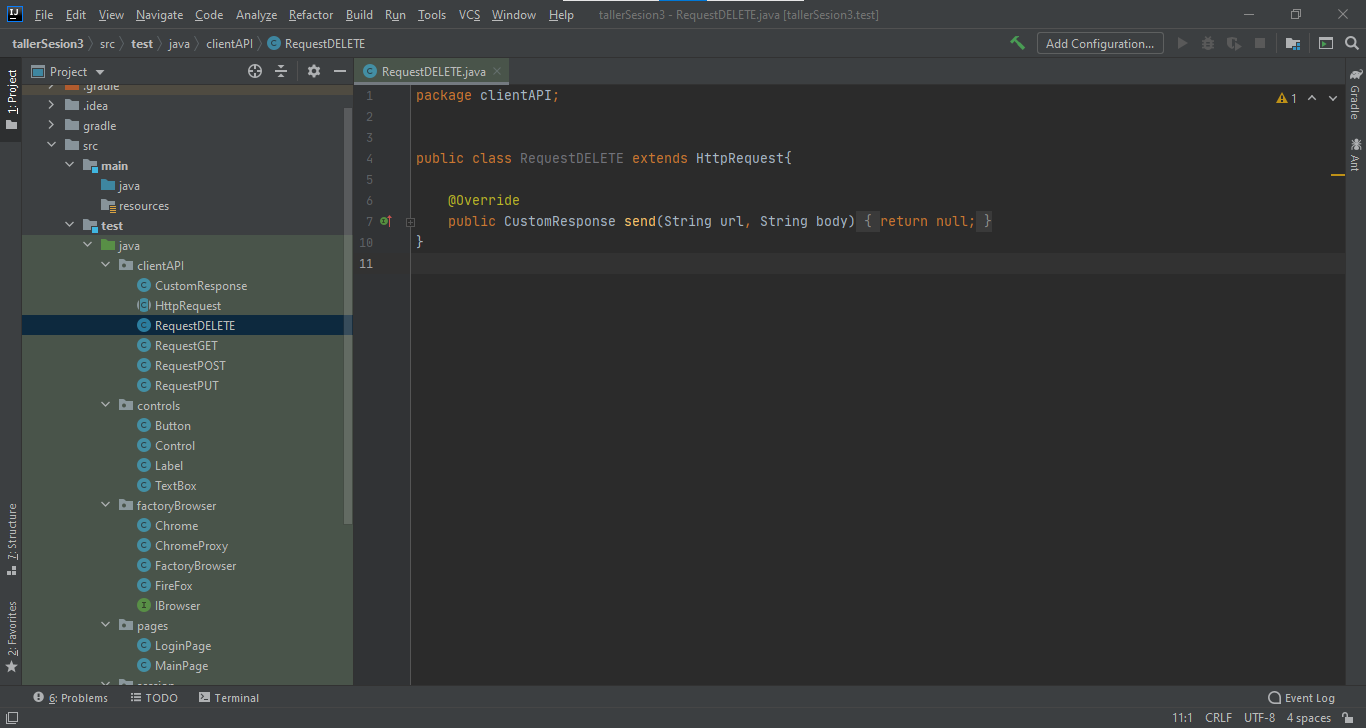
* 1. Crear la clase RequestGET en la siguiente ruta: test>java> clientAPI

package clientAPI;  
  
import javax.ws.rs.core.MediaType;  
import javax.ws.rs.core.Response;  
  
public class RequestGET extends HttpRequest{  
  
 @Override  
 public CustomResponse send(String url, String body) {  
 Response response;  
 response = this.client.target(url).request(MediaType.*APPLICATION\_JSON\_TYPE*).get();  
  
 CustomResponse customResponse= new CustomResponse();  
 customResponse.setResponseBody(response.readEntity(String.class));  
 customResponse.setResponseCode(response.getStatus()+"");  
  
 response.close();  
 return customResponse;  
  
 }  
}



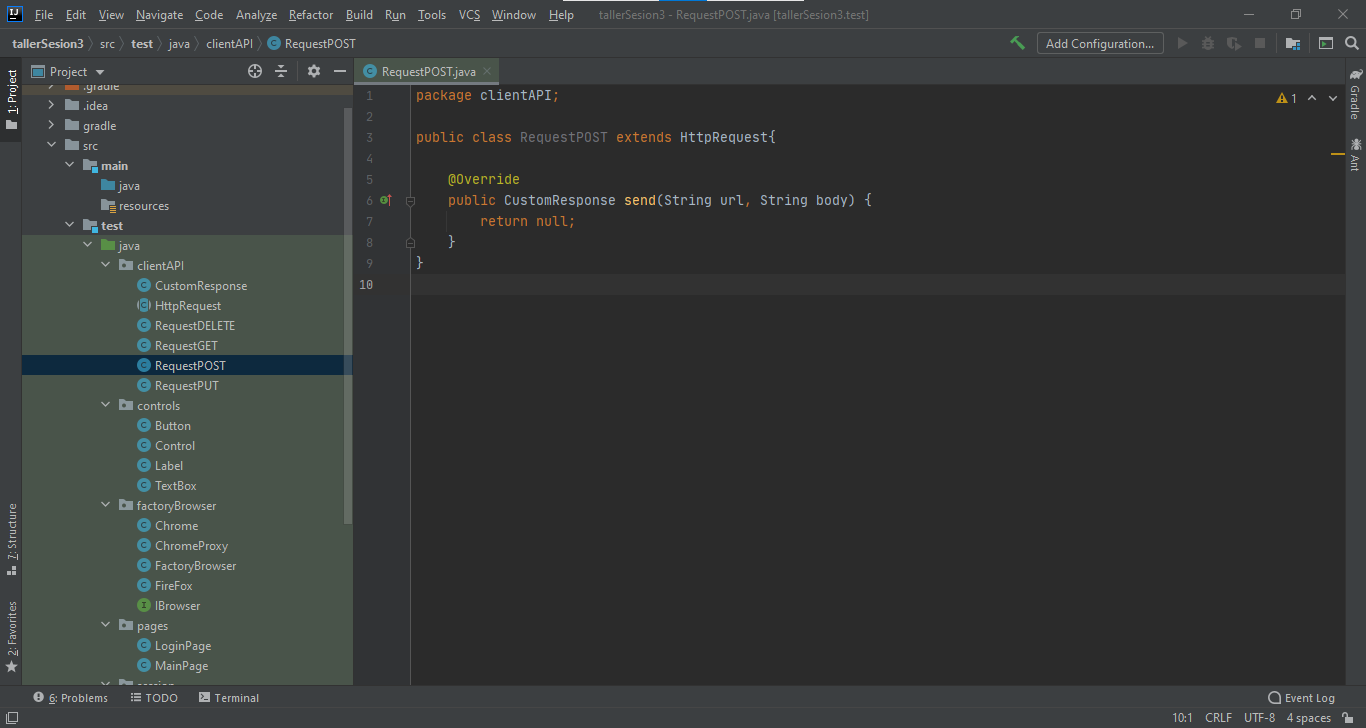
* 1. Crear la clase RequestDELETE en la siguiente ruta: test>java> clientAPI

package clientAPI;  
  
  
public class RequestDELETE extends HttpRequest{  
  
 @Override  
 public CustomResponse send(String url, String body) {  
 return null;  
 }  
}



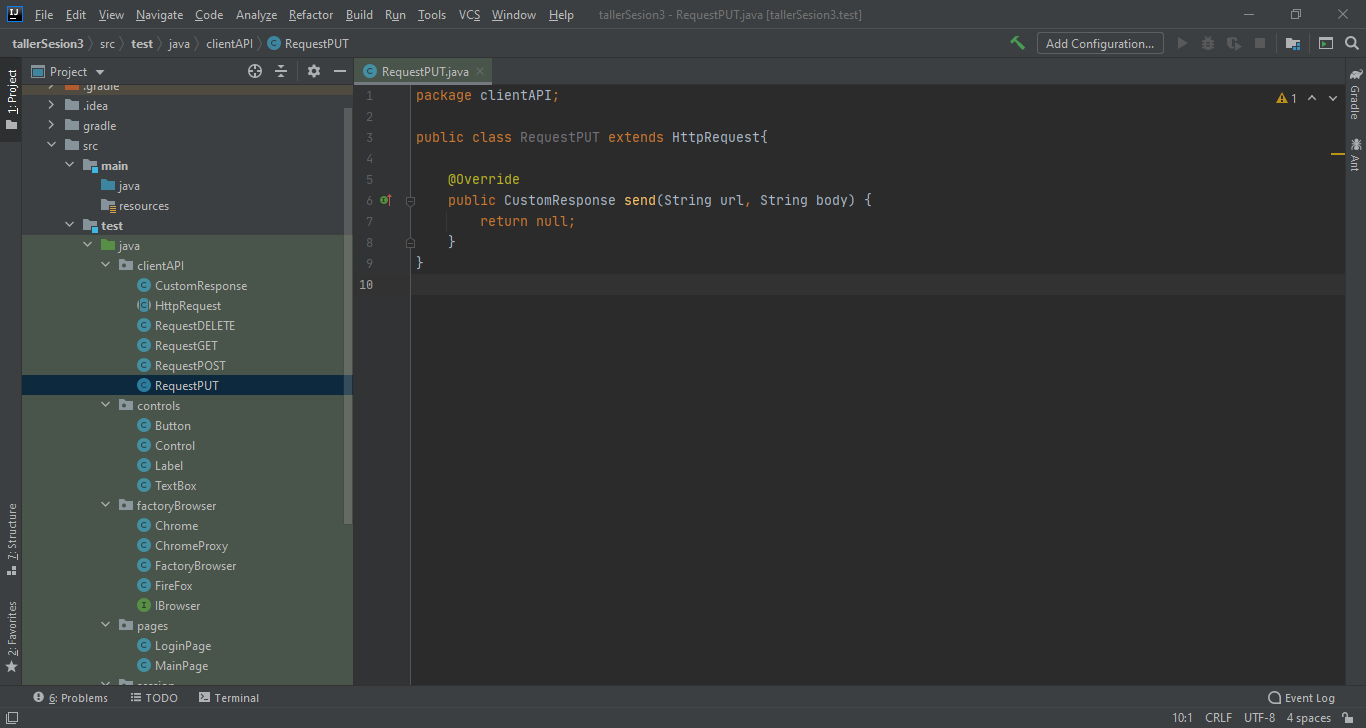
* 1. Crear la clase RequestPOST en la siguiente ruta: test>java> clientAPI

package clientAPI;  
  
public class RequestPOST extends HttpRequest{  
  
 @Override  
 public CustomResponse send(String url, String body) {  
 return null;  
 }  
}



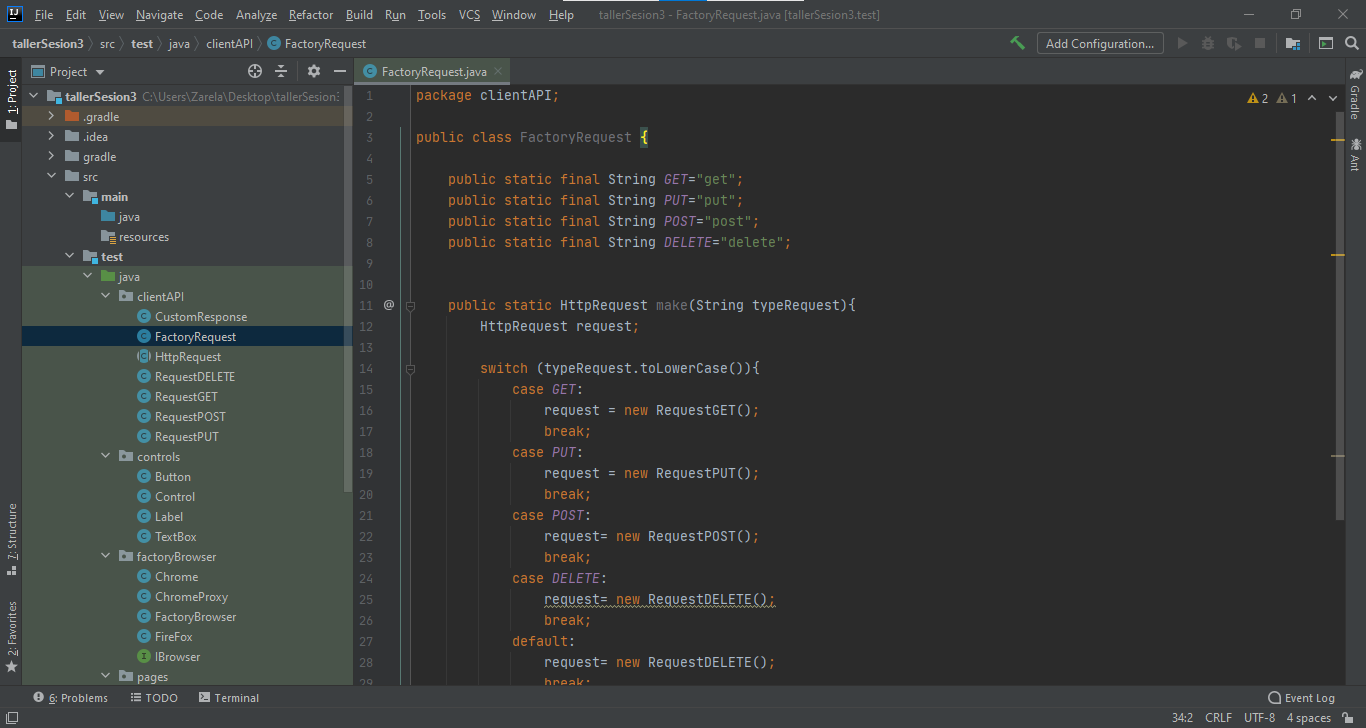
* 1. Crear la clase RequestPUT en la siguiente ruta: test>java> clientAPI

package clientAPI;  
  
public class RequestPUT extends HttpRequest{  
  
 @Override  
 public CustomResponse send(String url, String body) {  
 return null;  
 }  
}



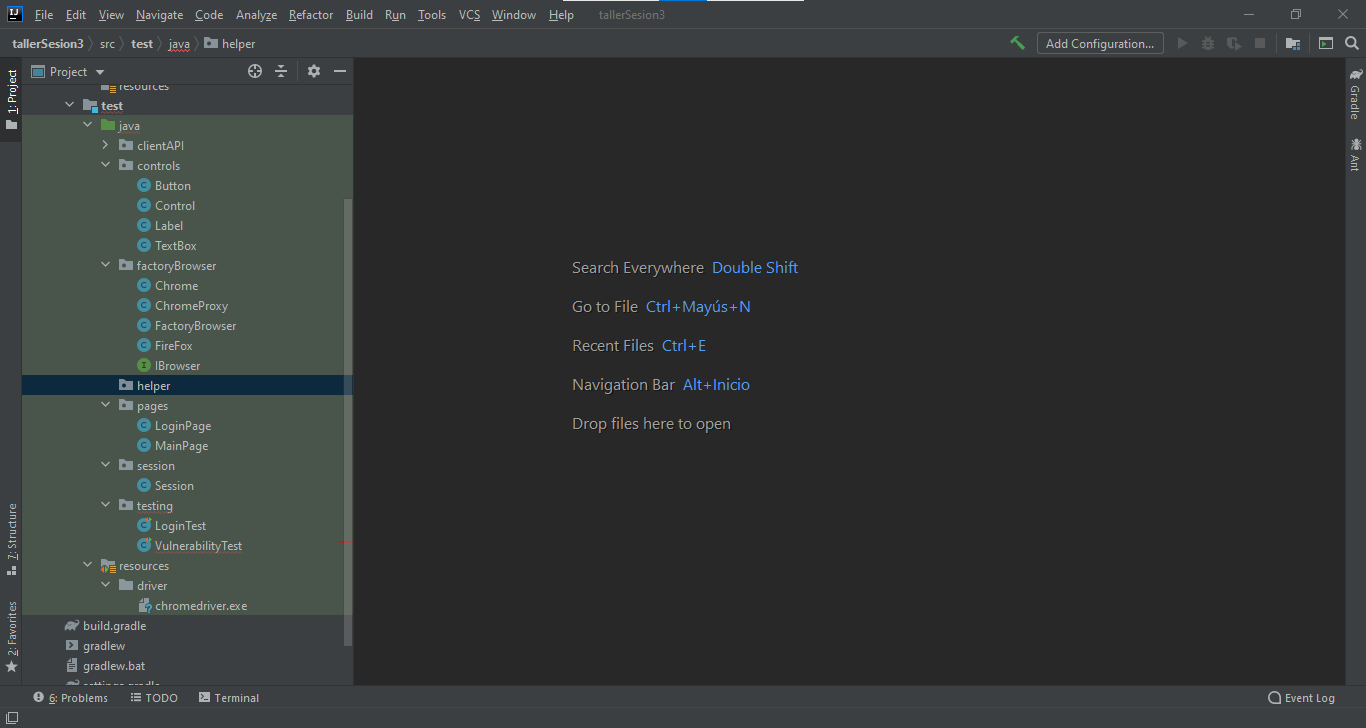
* 1. Crear la clase FactoryRequesten la siguiente ruta: test>java> clientAPI

package clientAPI;  
  
public class FactoryRequest {  
  
 public static final String *GET*="get";  
 public static final String *PUT*="put";  
 public static final String *POST*="post";  
 public static final String *DELETE*="delete";  
  
  
 public static HttpRequest make(String typeRequest){  
 HttpRequest request;  
  
 switch (typeRequest.toLowerCase()){  
 case *GET*:  
 request = new RequestGET();  
 break;  
 case *PUT*:  
 request = new RequestPUT();  
 break;  
 case *POST*:  
 request= new RequestPOST();  
 break;  
 case *DELETE*:  
 request= new RequestDELETE();  
 break;  
 default:  
 request= new RequestDELETE();  
 break;  
  
 }  
 return request;  
 }  
}



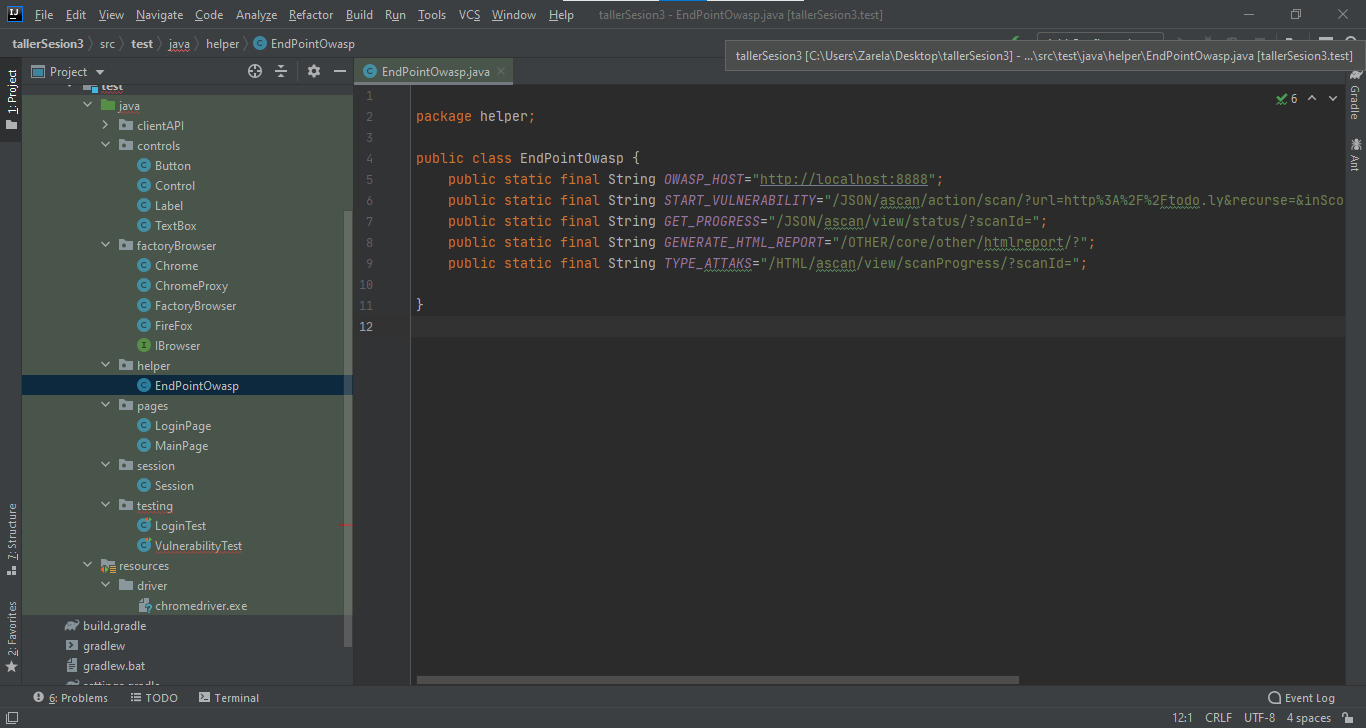
# **Implementación Package helper**

* 1. Crear el Package helper en la siguiente ruta: test>Java



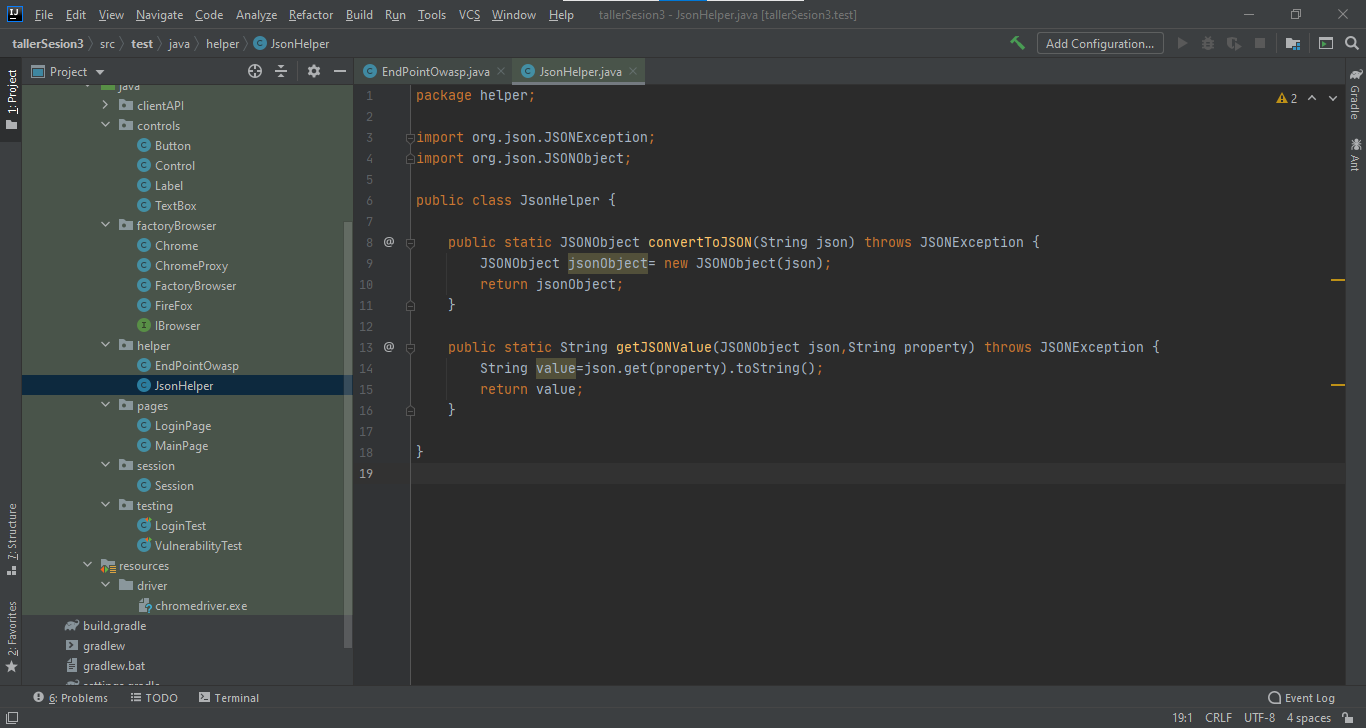
* 1. Crear la clase EndPointOwasp en la siguiente ruta: test>java> helper

package helper;  
  
public class EndPointOwasp {  
 public static final String *OWASP\_HOST*="http://localhost:8888";  
 public static final String *START\_VULNERABILITY*="/JSON/ascan/action/scan/?url=http%3A%2F%2Ftodo.ly&recurse=&inScopeOnly=&scanPolicyName=&method=&postData=&contextId=";  
 public static final String *GET\_PROGRESS*="/JSON/ascan/view/status/?scanId=";  
 public static final String *GENERATE\_HTML\_REPORT*="/OTHER/core/other/htmlreport/?";  
 public static final String *TYPE\_ATTAKS*="/HTML/ascan/view/scanProgress/?scanId=";  
  
}



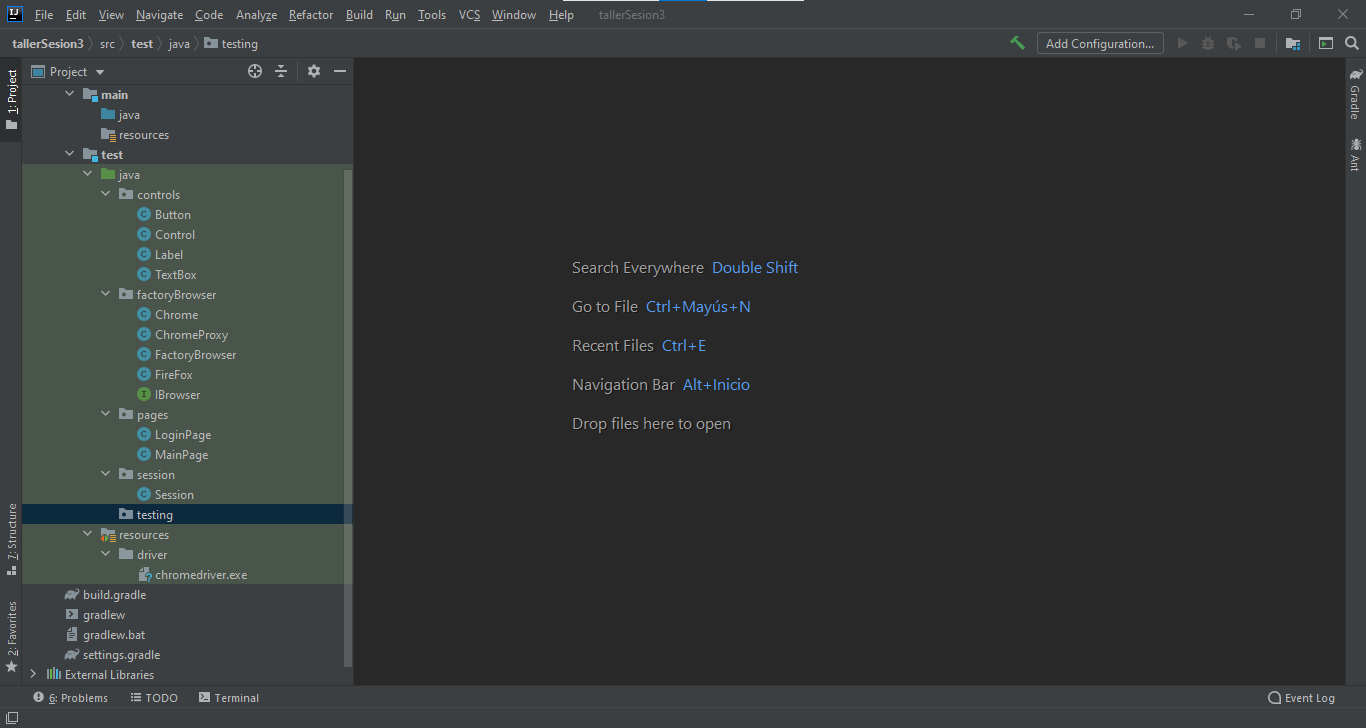
* 1. Crear la clase JsonHelper en la siguiente ruta: test>java> helper

package helper;  
  
import org.json.JSONException;  
import org.json.JSONObject;  
  
public class JsonHelper {  
  
 public static JSONObject convertToJSON(String json) throws JSONException {  
 JSONObject jsonObject= new JSONObject(json);  
 return jsonObject;  
 }  
  
 public static String getJSONValue(JSONObject json,String property) throws JSONException {  
 String value=json.get(property).toString();  
 return value;  
 }  
  
}



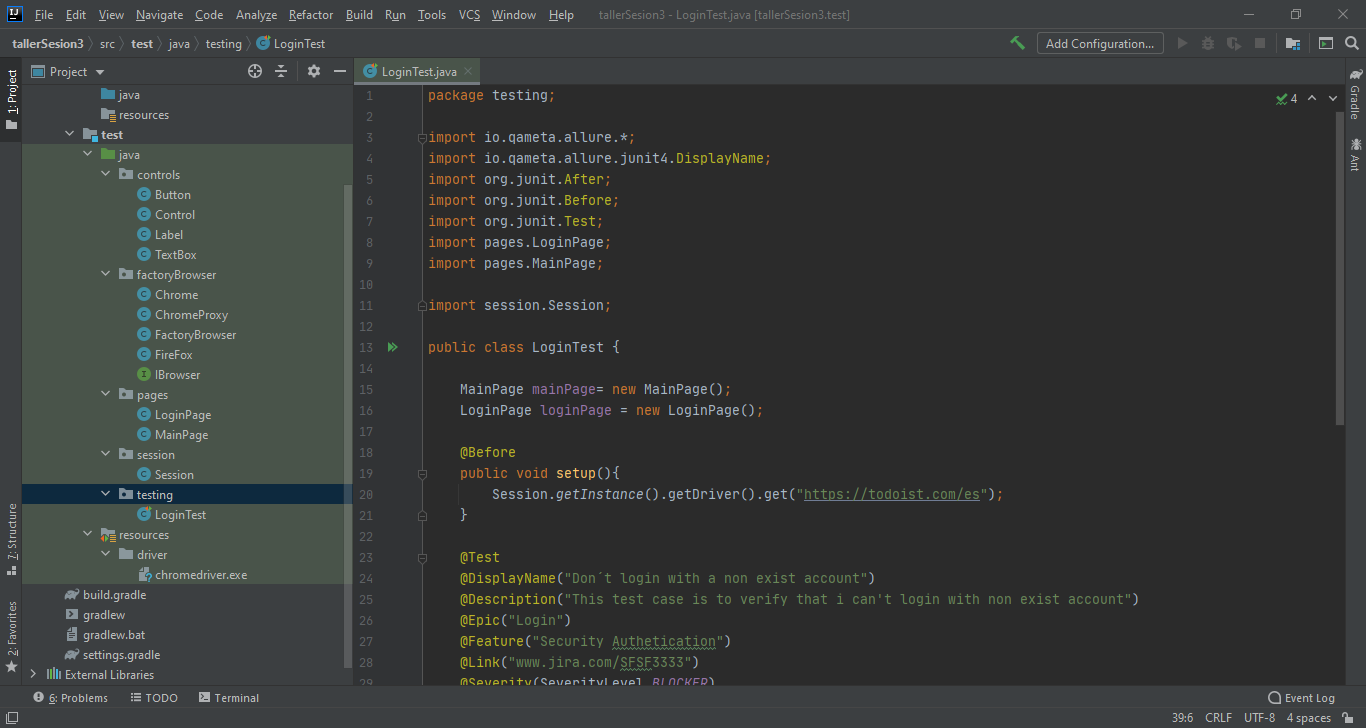
# **Implementación Package testing**

* 1. Crear el Package testing en la siguiente ruta: test>Java



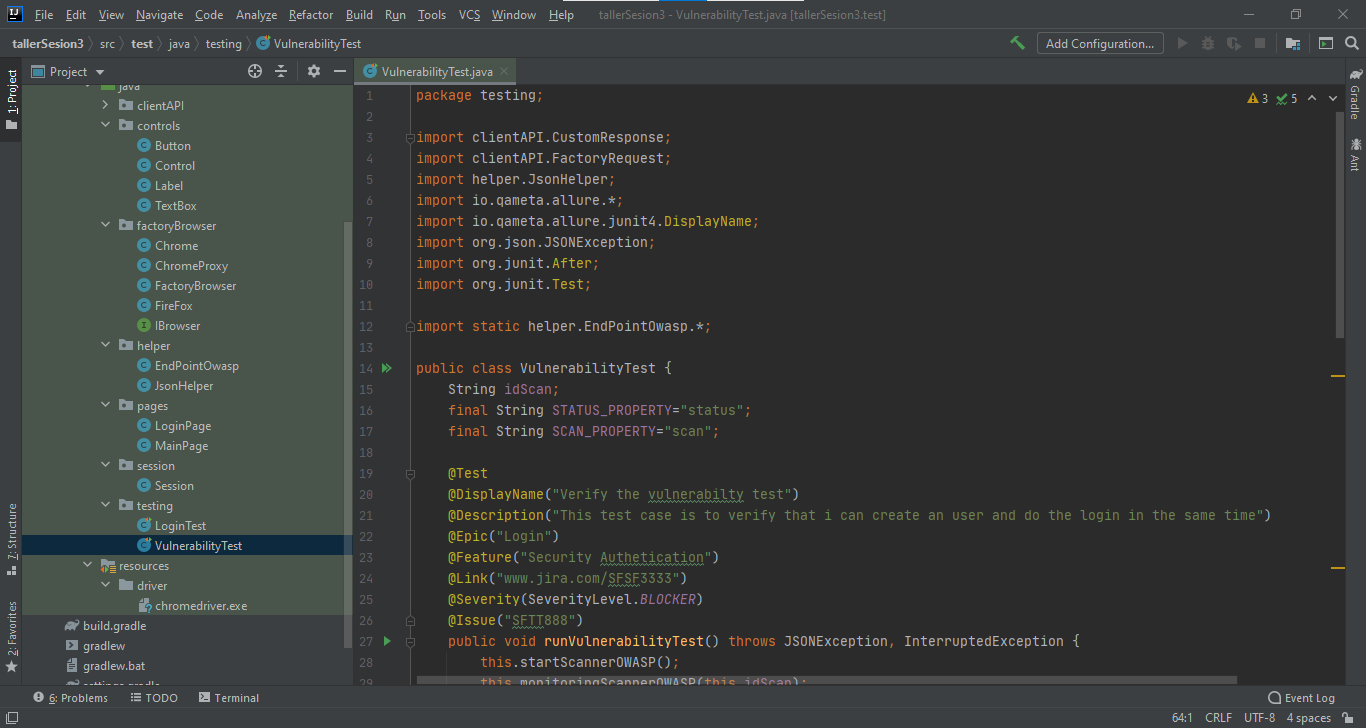
* 1. Crear la clase LoginTest en la siguiente ruta: test>java> testing

package testing;  
  
import io.qameta.allure.\*;  
import io.qameta.allure.junit4.DisplayName;  
import org.junit.After;  
import org.junit.Before;  
import org.junit.Test;  
import pages.LoginPage;  
import pages.MainPage;  
  
import session.Session;  
  
public class LoginTest {  
  
 MainPage mainPage= new MainPage();  
 LoginPage loginPage = new LoginPage();  
  
 @Before  
 public void setup(){  
 Session.*getInstance*().getDriver().get("https://todoist.com/es");  
 }  
  
 @Test  
 @DisplayName("Don´t login with a non exist account")  
 @Description("This test case is to verify that i can't login with non exist account")  
 @Epic("Login")  
 @Feature("Security Authetication")  
 @Link("www.jira.com/SFSF3333")  
 @Severity(SeverityLevel.*BLOCKER*)  
 @Issue("SFTT888")  
 public void createUserToSingup(){  
   
 mainPage.singUpFreeLabel.click();  
  
 loginPage.email.type("zarelarojas@gmail.com");  
 loginPage.contrasena.type("123456");  
 loginPage.IniciaSesion.click();  
  
 }  
  
 @After  
 public void cleanUp(){  
 Session.*getInstance*().closeSession();  
 }  
  
}



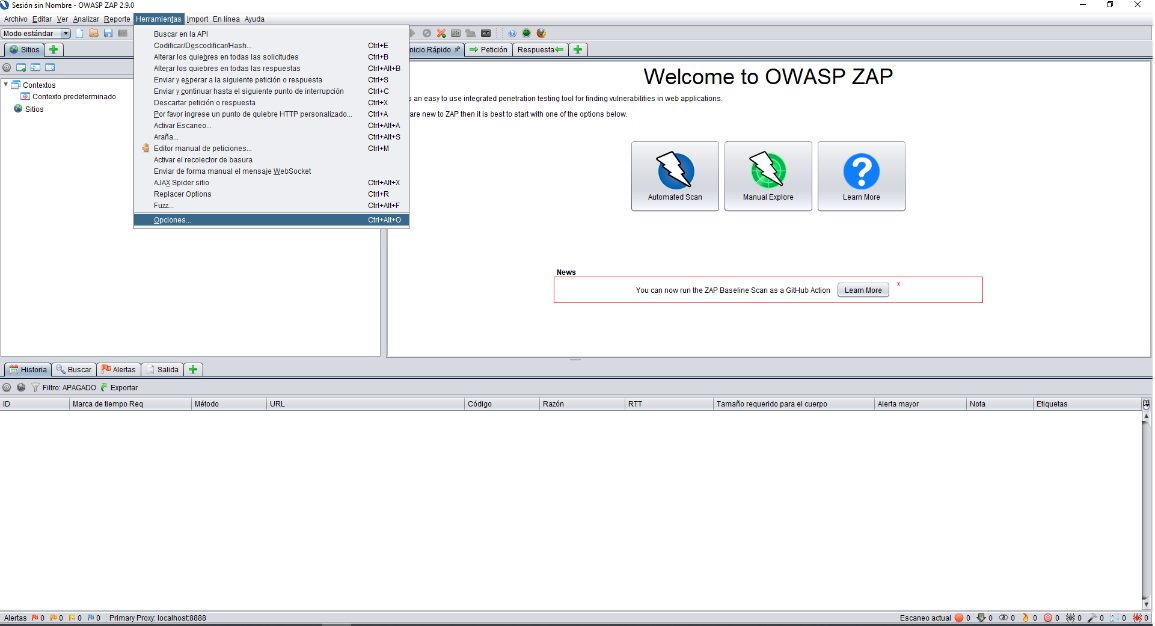
* 1. Crear la clase VulnerabilityTesten la siguiente ruta: test>java> testing

package testing;  
  
import clientAPI.CustomResponse;  
import clientAPI.FactoryRequest;  
import helper.JsonHelper;  
import io.qameta.allure.\*;  
import io.qameta.allure.junit4.DisplayName;  
import org.json.JSONException;  
import org.junit.After;  
import org.junit.Test;  
  
import static helper.EndPointOwasp.\*;  
  
public class VulnerabilityTest {  
 String idScan;  
 final String STATUS\_PROPERTY="status";  
 final String SCAN\_PROPERTY="scan";  
  
 @Test  
 @DisplayName("Verify the vulnerabilty test")  
 @Description("This test case is to verify that i can't login with non exist account")  
 @Epic("Login")  
 @Feature("Security Authetication")  
 @Link("www.jira.com/SFSF3333")  
 @Severity(SeverityLevel.*BLOCKER*)  
 @Issue("SFTT888")  
 public void runVulnerabilityTest() throws JSONException, InterruptedException {  
 this.startScannerOWASP();  
 this.monitoringScannerOWASP(this.idScan);  
 }  
  
  
 @Attachment(value="{0}",type="text/html")  
 public String attachHTML(String name,String htmlReport){  
 return htmlReport;  
 }  
  
 @After  
 public void generateReport(){  
 CustomResponse response =FactoryRequest.*make*(FactoryRequest.*GET*).send(*OWASP\_HOST*+*GENERATE\_HTML\_REPORT*,"");  
 this.attachHTML("OWASP Detail Report HTML",response.getResponseBody());  
 response =FactoryRequest.*make*(FactoryRequest.*GET*).send(*OWASP\_HOST*+*TYPE\_ATTAKS*+this.idScan,"");  
 this.attachHTML("OWASP Types Attack HTML",response.getResponseBody());  
 }  
  
 @Step("Start Scanner OWASP ZAP")  
 public void startScannerOWASP() throws JSONException {  
 CustomResponse response =FactoryRequest.*make*(FactoryRequest.*GET*).send(*OWASP\_HOST*+*START\_VULNERABILITY*,"");  
 this.idScan=JsonHelper.*getJSONValue*(JsonHelper.*convertToJSON*(response.getResponseBody()),SCAN\_PROPERTY);  
 System.*out*.println("ID SCAN : "+this.idScan);  
 }  
  
 @Step("Monitoring Current Scanner Process OWASP")  
 public void monitoringScannerOWASP(String id) throws InterruptedException, JSONException {  
 String isCompleted="";  
 while (!isCompleted.equals("100")){  
 //20 segundos  
 Thread.*sleep*(20\*1000);  
 CustomResponse response =FactoryRequest.*make*(FactoryRequest.*GET*).send(*OWASP\_HOST*+*GET\_PROGRESS*+id,"");  
 isCompleted=JsonHelper.*getJSONValue*(JsonHelper.*convertToJSON*(response.getResponseBody()),STATUS\_PROPERTY);  
 System.*out*.println("PROGRESS STATUS : "+isCompleted);  
 }  
 }  
  
}

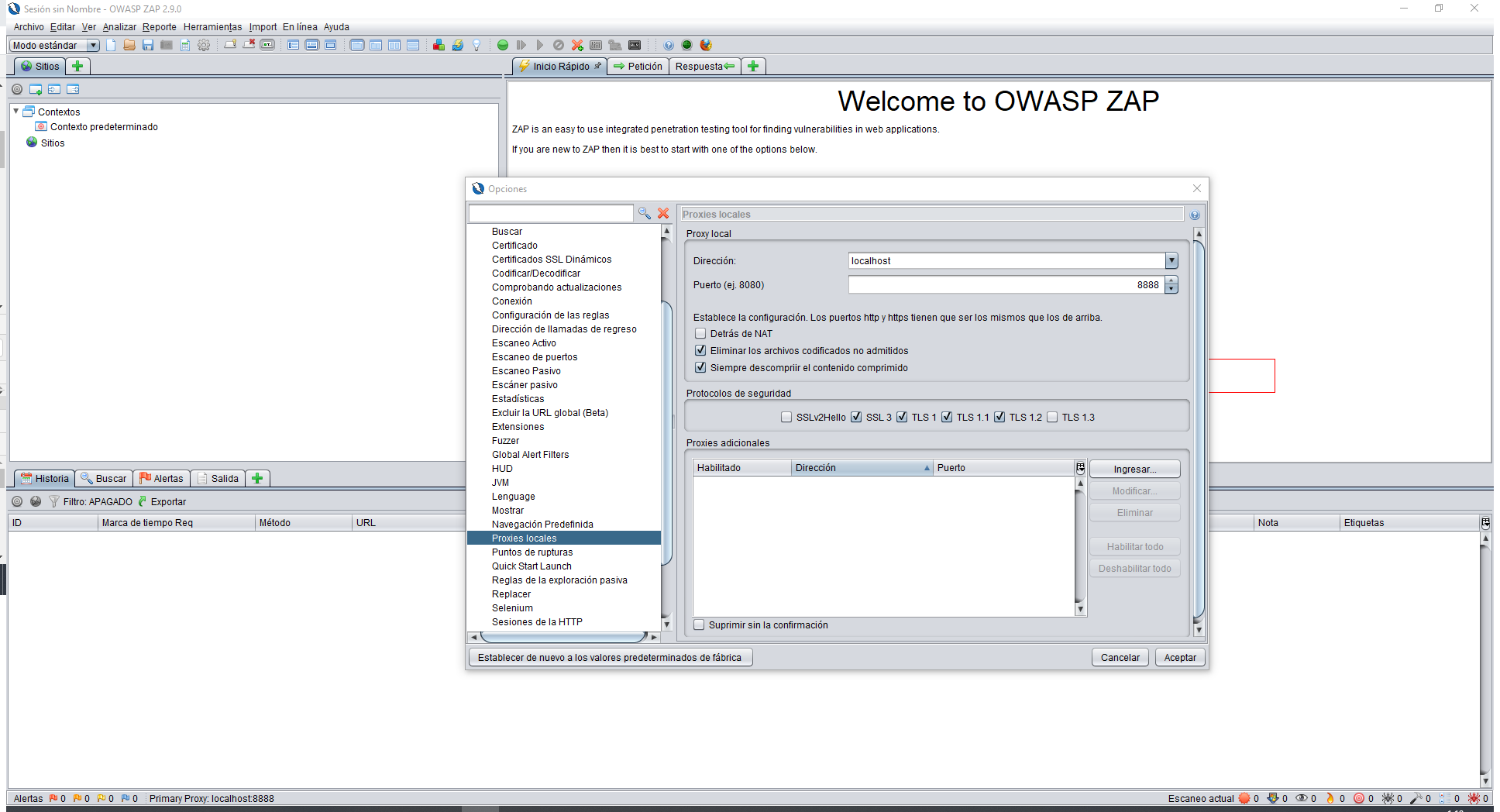


# **Configuración OWAS ZAP**

* 1. En el menú seleccionar la opción herramientas>opciones

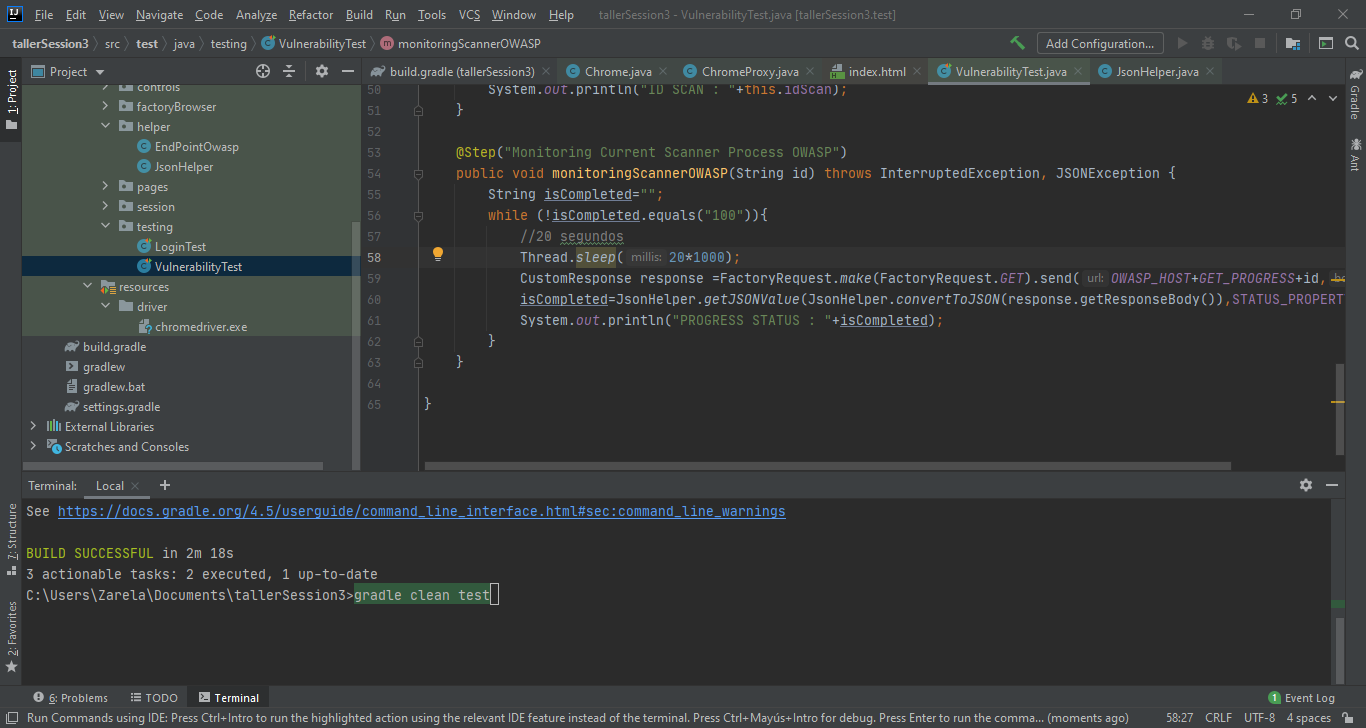


* 1. Buscar la opción “proxies locales” y en la opción puerto escribir “8888”. Seleccionar aceptar.

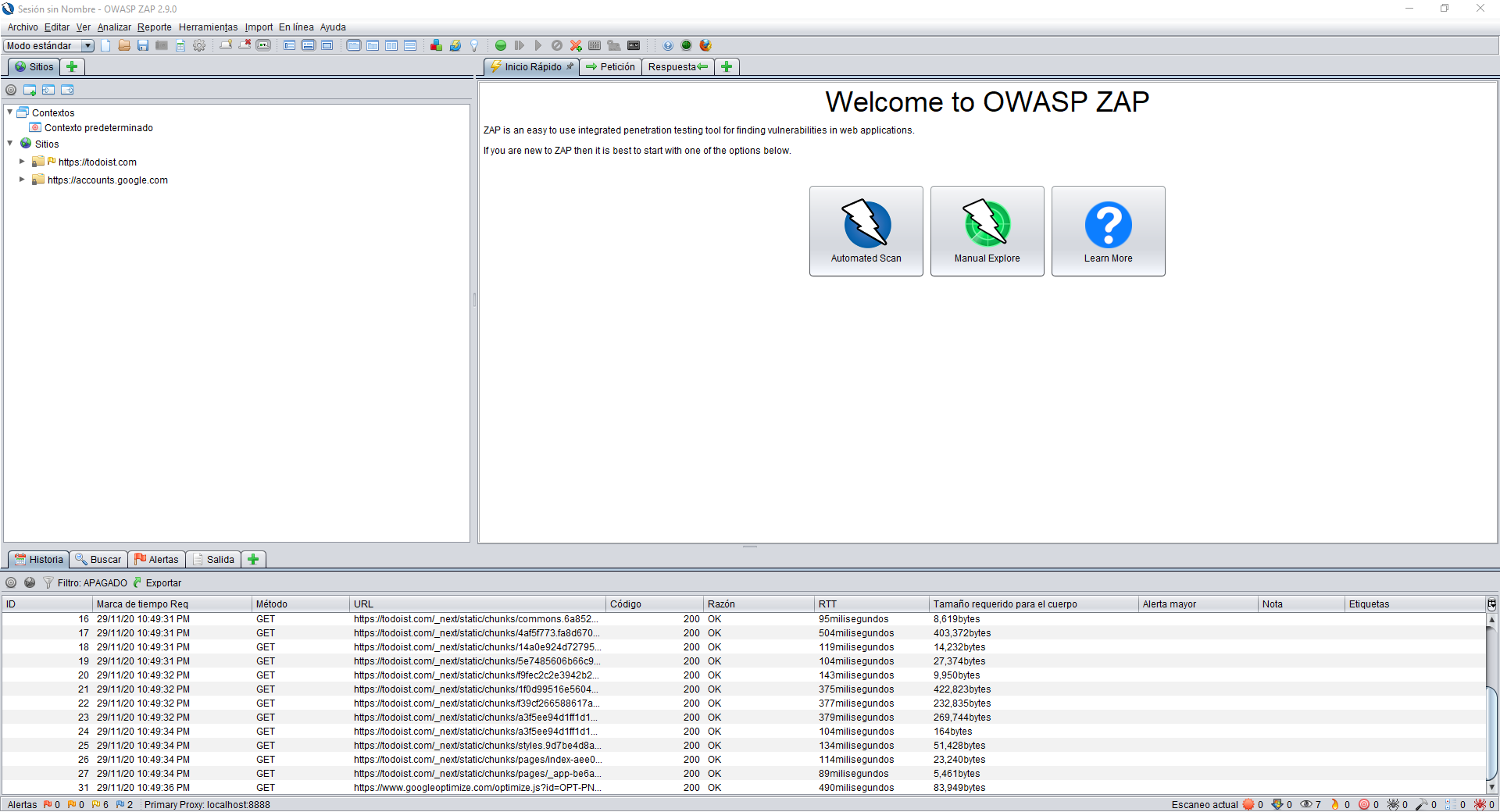


# **Ejecutar Script**

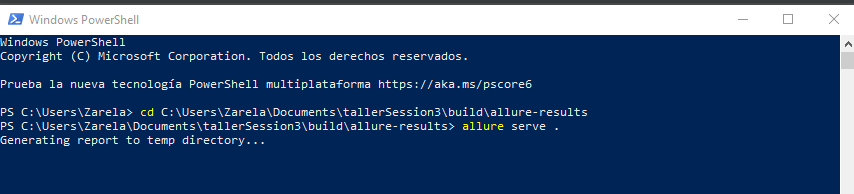
* 1. En el terminar ejecutar el siguiente comando
* Gradle test -> Solo si es la primera vez
* Gradle clean test -> Cuando se ejecuta por segunda vez.

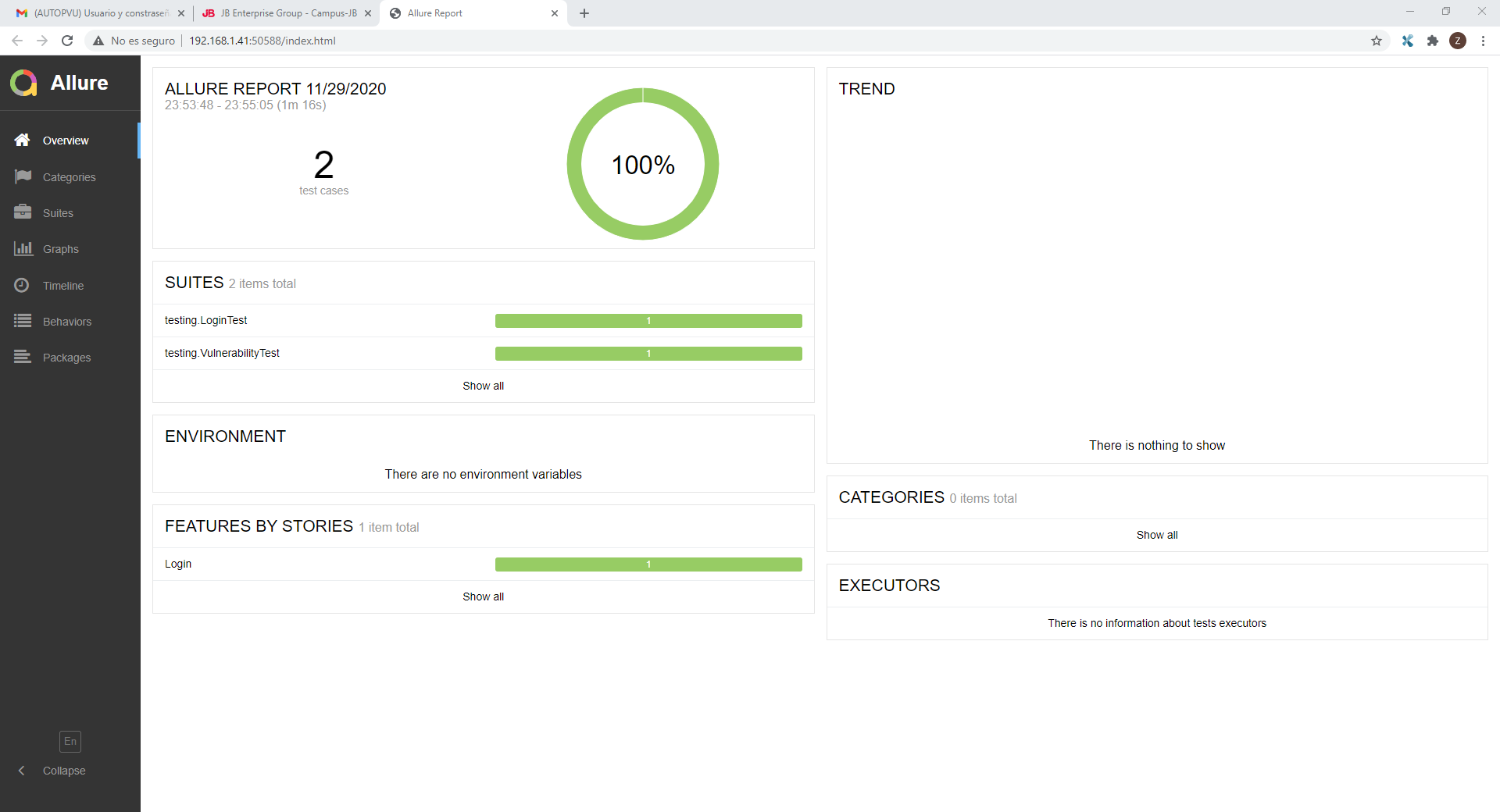


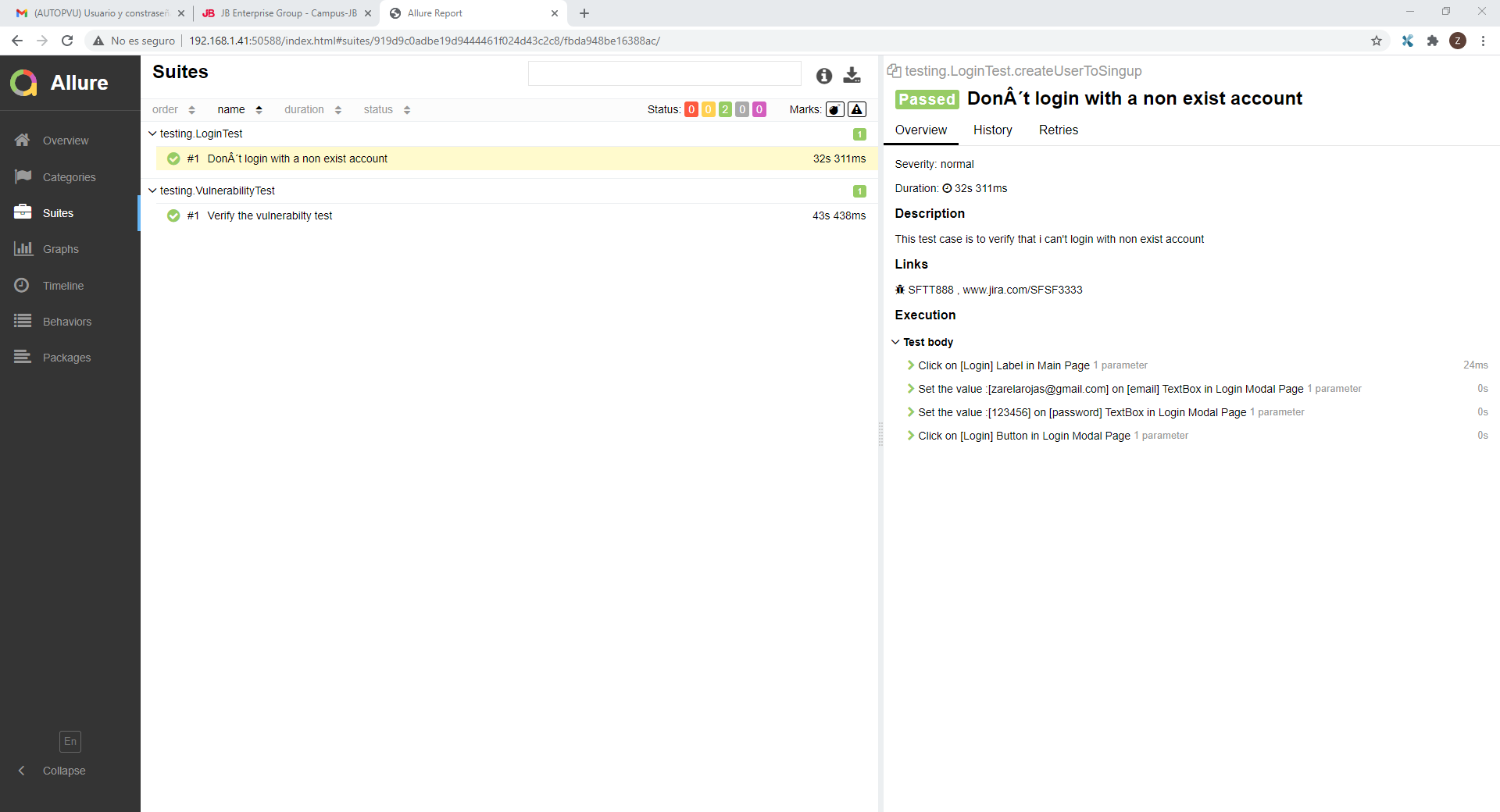
* 1. Al ingresar al OWASP ZAP se vera la grabación de la ejecución automática.

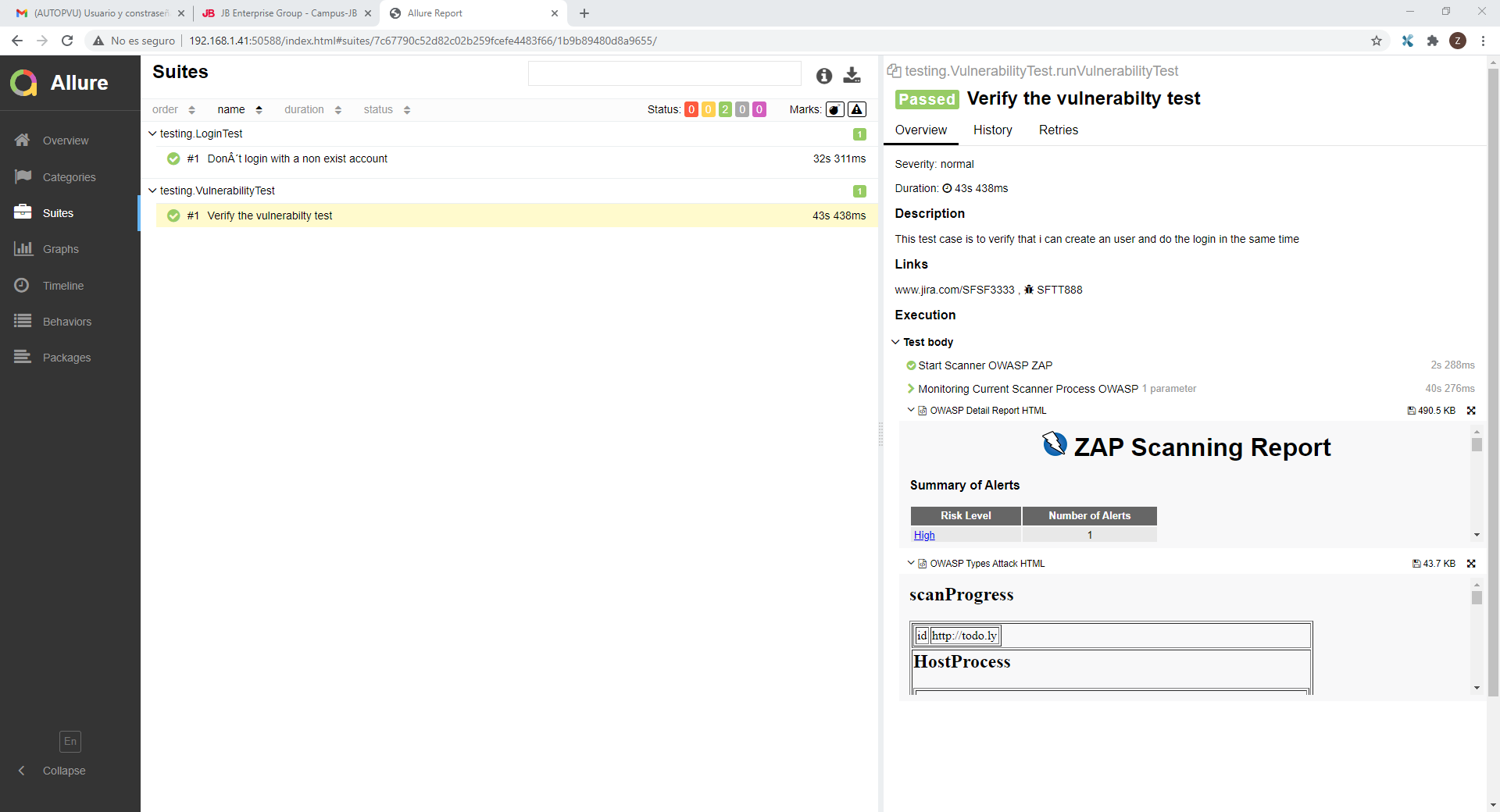
****

* 1. Para revisar los reportes ejecutar el siguiente comando en el Windows Power shell
* cd C:\Users\Zarela\Documents\tallerSession3\build\allure-results (Ruta carpeta allure results)
* Allure server .

****

****

****

****