El laboratorio incluye:

- Un proyecto Maven listo para ejecutar localmente.
- Un servidor HTTP ligero que sirve páginas de prueba (para no depender de recursos externos).
- Tests en **Java + Selenium 4 + TestNG** que cubren las ExpectedConditions más comunes y un **ejemplo de ExpectedCondition personalizada**.
- Archivos HTML de prueba mínimos (checkbox, formulario, drag&drop simulado, alert, iframe, tabla).
- testng.xml, instrucciones de instalación y comandos para ejecutar.

Resumen rápido

- Requisitos: Java 17+, Maven, Google Chrome (o Chromium).
- Comando para ejecutar: mvn test
- El test inicia un servidor local que sirve las páginas de prueba y arranca ChromeDriver local (usando WebDriverManager).

Estructura del proyecto (sugerida)

selenium-expectedconditions-lab/

├– pom.xml

├– src/

│ ├– main/

│ │ └– java/

│ │ └– lab.server/StaticFileServer.java

│ └– test/

│ ├– java/

│ │ └– lab.tests/

│ │ ├– BaseTest.java

│ │ ├– ExpectedConditionsTests.java

│ │ └– CustomExpectedCondition.java

│ └– resources/

│ └– pages/

│ ├– checkbox-demo.html

│ ├– simple-form-demo.html

│ ├– drag-drop-demo.html

```
│       ├─ dynamic-data-loading-demo.html
│       ├─ iframe-demo.html
│       ├─ alert-demo.html
│       └─ table-search-demo.html
└─ testng.xml
```

pom.xml (completo)

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                  http://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>

 <groupId>com.example</groupId>
 <artifactId>selenium-expectedconditions-lab</artifactId>
 <version>1.0-SNAPSHOT</version>

 <properties>
  <maven.compiler.release>17</maven.compiler.release>
  <selenium.version>4.23.0</selenium.version>
  <testng.version>7.8.0</testng.version>
  <webdrivermanager.version>5.4.0</webdrivermanager.version>
 </properties>

 <dependencies>
  <!-- Selenium -->
  <dependency>
   <groupId>org.seleniumhq.selenium</groupId>
   <artifactId>selenium-java</artifactId>
   <version>${selenium.version}</version>
  </dependency>
```

```
  <!-- TestNG -->
  <dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>${testng.version}</version>
    <scope>test</scope>
  </dependency>

  <!-- WebDriverManager para gestionar chromedriver automáticamente -->
  <dependency>
    <groupId>io.github.bonigarcia</groupId>
    <artifactId>webdrivermanager</artifactId>
    <version>${webdrivermanager.version}</version>
  </dependency>
 </dependencies>

 <build>
  <plugins>
   <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>3.0.0-M9</version>
    <configuration>
     <suiteXmlFiles>
      <suiteXmlFile>testng.xml</suiteXmlFile>
     </suiteXmlFiles>
    </configuration>
   </plugin>
  </plugins>
 </build>
</project>
```

Servidor estático ligero (sirve src/test/resources/pages)

src/main/java/lab/server/StaticFileServer.java

```
package lab.server;

import com.sun.net.httpserver.HttpServer;
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpHandler;

import java.io.*;
import java.net.InetSocketAddress;
import java.nio.file.Files;
import java.nio.file.Path;

public class StaticFileServer {
  private HttpServer server;
  private final int port;
  private final Path root;

  public StaticFileServer(int port, Path root) {
    this.port = port;
    this.root = root;
  }

  public void start() throws IOException {
    server = HttpServer.create(new InetSocketAddress(port), 0);
    server.createContext("/", new FileHandler(root));
    server.setExecutor(null);
    server.start();
    System.out.println("StaticFileServer started on http://localhost:" + port + " serving " +
root);
  }
```

```
public void stop() {
    if (server != null) {
        server.stop(0);
        System.out.println("StaticFileServer stopped");
    }
}

static class FileHandler implements HttpHandler {
    private final Path root;

    FileHandler(Path root) { this.root = root; }

    @Override
    public void handle(HttpExchange exchange) throws IOException {
        String path = exchange.getRequestURI().getPath();
        if (path.equals("/")) path = "/index.html";
        Path file = root.resolve(path.substring(1)).normalize();
        if (!file.startsWith(root) || !Files.exists(file) || Files.isDirectory(file)) {
            String resp = "404 (Not Found)\n";
            exchange.sendResponseHeaders(404, resp.length());
            try (OutputStream os = exchange.getResponseBody()) {
                os.write(resp.getBytes());
            }
            return;
        }
        byte[] bytes = Files.readAllBytes(file);
        String mime = Files.probeContentType(file);
        if (mime == null) mime = "application/octet-stream";
        exchange.getResponseHeaders().add("Content-Type", mime);
        exchange.sendResponseHeaders(200, bytes.length);
```

```
        try (OutputStream os = exchange.getResponseBody()) {
            os.write(bytes);
        }
    }
  }
}
```

Nota: com.sun.net.httpserver.HttpServer está disponible en la JDK; este servidor es suficiente para pruebas locales.

BaseTest.java

src/test/java/lab/tests/BaseTest.java

```
package lab.tests;

import lab.server.StaticFileServer;
import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.testng.annotations.*;

import java.nio.file.Path;
import java.time.Duration;

public class BaseTest {
  protected WebDriver driver;
  protected static StaticFileServer fileServer;
  protected static final int SERVER_PORT = 8000;

  @BeforeSuite(alwaysRun = true)
  public void startServer() throws Exception {
    Path pages = Path.of("src", "test", "resources", "pages");
    fileServer = new StaticFileServer(SERVER_PORT, pages);
```

```
        fileServer.start();
    }

    @AfterSuite(alwaysRun = true)
    public void stopServer() {
        if (fileServer != null) fileServer.stop();
    }

    @BeforeClass
    public void setup() {
        WebDriverManager.chromedriver().setup();
        ChromeOptions options = new ChromeOptions();
        // options.addArguments("--headless=new"); // descomenta si quieres headless
        driver = new ChromeDriver(options);
        driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(2)); // left small to show
explicit waits' benefit
        driver.manage().window().maximize();
    }

    @AfterClass
    public void tearDown() {
        if (driver != null) driver.quit();
    }

    protected String url(String page) {
        return "http://localhost:" + SERVER_PORT + "/" + page;
    }
}
```

Tests: ExpectedConditionsTests.java (todos los ejemplos)

src/test/java/lab/tests/ExpectedConditionsTests.java

```
package lab.tests;

import lab.tests.CustomExpectedCondition;
import org.openqa.selenium.*;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.ui.*;
import org.testng.annotations.Test;

import java.time.Duration;

import static org.testng.Assert.*;

public class ExpectedConditionsTests extends BaseTest {

  @Test
  public void testElementToBeSelected() {
    driver.get(url("checkbox-demo.html"));
    WebElement checkAllBtn = driver.findElement(By.id("checkAll"));
    checkAllBtn.click();

    WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
    WebElement cb1 = driver.findElement(By.id("cb1"));
    WebElement cb2 = driver.findElement(By.id("cb2"));
    assertTrue(wait.until(ExpectedConditions.elementToBeSelected(cb1)));
    assertTrue(wait.until(ExpectedConditions.elementToBeSelected(cb2)));
  }

  @Test
  public void testElementToBeClickable() {
    driver.get(url("simple-form-demo.html"));
    WebElement input = driver.findElement(By.id("user-message"));
```

```
        input.sendKeys("hola mundo");

        WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
        wait.until(ExpectedConditions.elementToBeClickable(By.id("showInput"))).click();

        String text = driver.findElement(By.id("message")).getText();
        assertEquals(text, "hola mundo");
    }

    @Test
    public void testVisibilityOfElementLocated() {
        driver.get(url("drag-drop-demo.html"));
        WebElement draggable = driver.findElement(By.id("draggable"));
        WebElement dropzone = driver.findElement(By.id("dropzone"));

        Actions actions = new Actions(driver);
        actions.dragAndDrop(draggable, dropzone).perform();

        WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
        WebElement droppedItem =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("dropped-item")));
        assertEquals(droppedItem.getText(), "Draggable 1");
    }

    @Test
    public void testTextToBePresentInElementLocated() {
        driver.get(url("dynamic-data-loading-demo.html"));
        driver.findElement(By.id("getRandom")).click();

        WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
```

```
assertTrue(wait.until(ExpectedConditions.textToBePresentInElementLocated(By.id("loadi
ng"), "First Name")));
  }

  @Test
  public void testFrameToBeAvailableAndSwitchToIt() {
    driver.get(url("iframe-demo.html"));
    WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
    WebElement iframe = driver.findElement(By.id("editorFrame"));
    wait.until(ExpectedConditions.frameToBeAvailableAndSwitchToIt(iframe));

    WebElement body = driver.findElement(By.id("editorBody"));
    body.clear();
    body.sendKeys("switched to iframe");
    assertEquals(body.getText(), "switched to iframe");

    // volver al top-level
    driver.switchTo().defaultContent();
  }

  @Test
  public void testAlertIsPresent() {
    driver.get(url("alert-demo.html"));
    driver.findElement(By.id("showAlert")).click();

    WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(5));
    Alert alert = wait.until(ExpectedConditions.alertIsPresent());
    assertEquals(alert.getText(), "Soy una alerta!");
    alert.accept();
  }
```

```
  @Test
  public void testPresenceOfElementLocatedAndTitleAndUrl() {
    driver.get(url("simple-form-demo.html"));
    WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(5));
    WebElement input =
wait.until(ExpectedConditions.presenceOfElementLocated(By.id("user-message")));
    assertTrue(input.isDisplayed());

    assertTrue(wait.until(ExpectedConditions.titleContains("Simple Form")));
    assertTrue(wait.until(ExpectedConditions.urlToBe(url("simple-form-demo.html"))));
  }

  @Test
  public void testCustomExpectedCondition() {
    driver.get(url("table-search-demo.html"));
    WebElement search = driver.findElement(By.id("searchBox"));
    search.sendKeys("Bennet");

    WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
    // Usamos nuestra condición personalizada:
    boolean found = wait.until(new CustomExpectedCondition(By.cssSelector("#table
tbody tr:nth-child(1) td.name"), "Bennet"));
    assertTrue(found, "No se encontró 'Bennet' en la tabla dentro del timeout");
  }
}
```

Condición personalizada: CustomExpectedCondition.java

src/test/java/lab/tests/CustomExpectedCondition.java

```
package lab.tests;

import org.openqa.selenium.*;
import org.openqa.selenium.support.ui.ExpectedCondition;
```

```
/**
 * Devuelve true cuando el texto esperado aparece en el elemento localizado.
 * Usamos esta clase para combinar lógica en un solo ExpectedCondition.
 */
public class CustomExpectedCondition implements ExpectedCondition<Boolean> {
  private final By locator;
  private final String expectedText;

  public CustomExpectedCondition(By locator, String expectedText) {
    this.locator = locator;
    this.expectedText = expectedText;
  }

  @Override
  public Boolean apply(WebDriver driver) {
    try {
      WebElement el = driver.findElement(locator);
      return el.isDisplayed() && el.getText().contains(expectedText);
    } catch (NoSuchElementException | StaleElementReferenceException e) {
      return false;
    }
  }

  @Override
  public String toString() {
    return String.format("texto '%s' presente en %s", expectedText, locator);
  }
}
```

Páginas HTML de ejemplo (mínimas)

Coloca estos archivos en src/test/resources/pages/

checkbox-demo.html

```
<!doctype html>
<html><head><meta charset="utf-8"><title>Checkbox Demo</title></head>
<body>
 <h1>Checkbox Demo</h1>
 <button id="checkAll" onclick="checkAll()">Check All</button>
 <br/><br/>
 <input type="checkbox" id="cb1"/> <label for="cb1">Checkbox 1</label><br/>
 <input type="checkbox" id="cb2"/> <label for="cb2">Checkbox 2</label><br/>
 <script>
  function checkAll(){
   setTimeout(()=>{ document.getElementById('cb1').checked = true;
document.getElementById('cb2').checked = true; }, 400);
  }
 </script>
</body></html>
```

simple-form-demo.html

```
<!doctype html>
<html><head><meta charset="utf-8"><title>Simple Form</title></head>
<body>
 <h1>Simple Form Demo</h1>
 <input id="user-message" placeholder="Message"/>
 <button id="showInput" onclick="show()">Get Checked Value</button>
 <p id="message"></p>

 <script>
  function show(){
   const v = document.getElementById('user-message').value;
   // Simular pequeña demora
   setTimeout(()=> document.getElementById('message').innerText = v, 300);
```

```
  }
 </script>
</body></html>
```

drag-drop-demo.html

```
<!doctype html>
<html><head><meta charset="utf-8"><title>Drag and Drop</title>
<style>#draggable{width:80px;height:30px;background:#cce;padding:5px;display:inline-block}#dropzone{width:200px;height:80px;border:1px solid #333;display:inline-block;margin-left:20px}</style>
</head>
<body>
 <div id="draggable" draggable="true">Draggable 1</div>
 <div id="dropzone">Drop here</div>
 <div id="droppedlist"></div>

<script>
 const drag = document.getElementById('draggable');
 const drop = document.getElementById('dropzone');
 drop.addEventListener('dragover', e => e.preventDefault());
 drop.addEventListener('drop', e => {
  e.preventDefault();
  // Simula que el elemento aparece luego de una operación
  setTimeout(()=> {
   const span = document.createElement('div');
   span.id = 'dropped-item';
   span.innerText = 'Draggable 1';
   document.getElementById('droppedlist').appendChild(span);
  }, 400);
 });
</script>
</body></html>
```

dynamic-data-loading-demo.html

```
<!doctype html>
<html><head><meta charset="utf-8"><title>Dynamic Data</title></head>
<body>
 <button id="getRandom" onclick="load()">Get Random User</button>
 <div id="loading"></div>

<script>
 function load(){
  document.getElementById('loading').innerText = 'Loading...';
  setTimeout(()=> {
   document.getElementById('loading').innerText = 'First Name: John';
  }, 700);
 }
</script>
</body></html>
```

iframe-demo.html

```
<!doctype html>
<html><head><meta charset="utf-8"><title>Iframe Demo</title></head>
<body>
 <h1>Iframe demo</h1>
 <iframe id="editorFrame" srcdoc='<div id="editorBody"
contenteditable="true">initial</div>'></iframe>
</body></html>
```

alert-demo.html

```
<!doctype html>
<html><head><meta charset="utf-8"><title>Alert Demo</title></head>
<body>
 <button id="showAlert" onclick="show()">Show Alert</button>
 <script>function show(){ setTimeout(()=> alert('Soy una alerta!'), 300); }</script>
</body></html>
```

table-search-demo.html

```
<!doctype html>
<html><head><meta charset="utf-8"><title>Table Search</title></head>
<body>
 <input id="searchBox" placeholder="search"/>
 <table id="table" border="1">
  <thead><tr><th>Name</th><th>Age</th></tr></thead>
  <tbody>
   <tr><td class="name">Bennet</td><td>30</td></tr>
   <tr><td class="name">Smith</td><td>25</td></tr>
  </tbody>
 </table>
 <script>
  document.getElementById('searchBox').addEventListener('input', function(){
     const q = this.value.toLowerCase();
     const rows = document.querySelectorAll('#table tbody tr');
     rows.forEach(r => r.style.display = r.innerText.toLowerCase().includes(q) ? '' : 'none');
  });
 </script>
</body></html>
```

testng.xml

```
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >
<suite name="ExpectedConditions Suite">
 <test name="ExpectedConditionsTests">
  <classes>
   <class name="lab.tests.ExpectedConditionsTests"/>
  </classes>
 </test>
</suite>
```

Cómo ejecutar (pasos)

1. Clona/crea el proyecto con la estructura mostrada.
2. Asegúrate de tener Java 17+ y Maven instalados.
3. Desde la raíz del proyecto ejecuta:
4. mvn test

Esto:

- Iniciará el servidor local en http://localhost:8000.
- Lanzará ChromeDriver localmente (WebDriverManager descarga el chromedriver si hace falta).
- Ejecutará los tests de TestNG.

Si prefieres ejecutar en modo headless, descomenta la línea options.addArguments("--headless=new"); en BaseTest.setup().

FAQ (preguntas frecuentes — paso a paso)

P: ¿Por qué usar ExpectedConditions en lugar de Thread.sleep() o solo implicitlyWait?
R: Thread.sleep() es fijo y ralentiza; implicitlyWait se aplica en todas las búsquedas y no espera condiciones como visibilidad o clickabilidad. ExpectedConditions con WebDriverWait permite esperar *condiciones concretas* (visible, clickable, texto presente…) y mejora la fiabilidad y velocidad de los tests.

P: ¿Cuándo usar elementToBeClickable vs visibilityOfElementLocated?
R: visibilityOfElementLocated espera que el elemento esté presente y visible (tamaño > 0). elementToBeClickable además requiere que esté habilitado (enabled) para hacer click. Para clickar, prefiere elementToBeClickable.

P: ¿Qué hace frameToBeAvailableAndSwitchToIt?
R: Espera a que el iframe/frame esté disponible y automáticamente hace driver.switchTo().frame(…). Después, recuerda volver con driver.switchTo().defaultContent().

P: ¿Cómo creo una ExpectedCondition personalizada?
R: Implementa ExpectedCondition<T> (por ejemplo ExpectedCondition<Boolean>) y sobreescribe apply(WebDriver). He incluido CustomExpectedCondition como ejemplo.

P: ¿Qué timeout pongo?
R: Depende de tu app. En tests locales 5–15s suele ser suficiente. Usa WebDriverWait(driver, Duration.ofSeconds(n)). No uses timeouts excesivos por defecto.

P: ¿Puedo reutilizar estas pruebas en CI?
R: Sí. Asegúrate de ejecutar Chrome/Chromedriver en el runner o usar una solución de grid/servicio con capacidades equivalentes. Nuestro ejemplo usa Chrome local, por tanto en CI necesitarás un runner que permita navegadores (o usar contenedores con xvfb).

P: ¿Y si quiero usar Firefox?
R: Cambia WebDriverManager.firefoxdriver().setup() y crea new FirefoxDriver() con las opciones correspondientes.