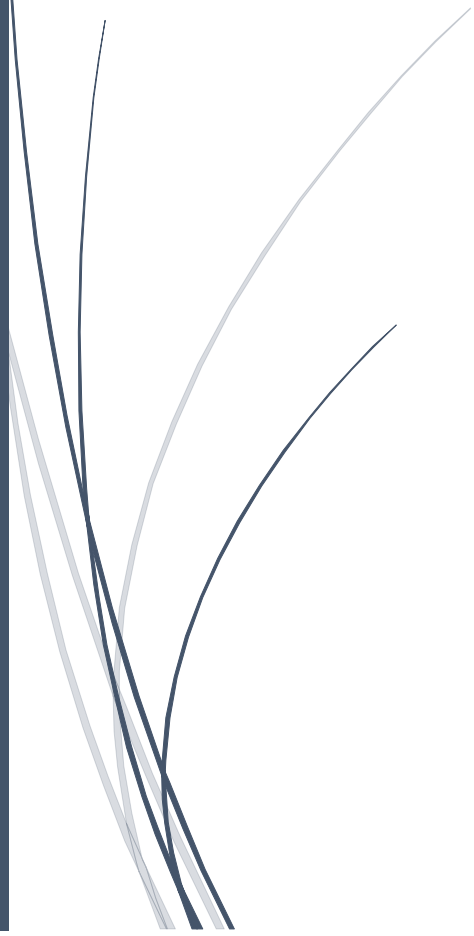
23-2-2024

Documentación Práctica Final

Integración Continua en el Desarrollo Ágil

Javier González Soldado



ÍNDICE

INTRODUCCIÓN

En el presente documento se realiza una explicación sobre los procedimientos y desarrollo seguidos por el alumno para la implementación de la nueva versión de la aplicación.

Quiero recalcar que en este documento no se va a explicar la primera versión de la aplicación ya que se corresponde con la práctica guiada realizada conjuntamente en la última clase presencial de la asignatura.

La nueva versión de la aplicación viene dada por unos requisitos solicitados por el profesor de la asignatura. En el siguiente apartado de la documentación, se procede a explicar los pasos llevados a cabo para lograr cumplimentar esos requisitos de forma satisfactoria.

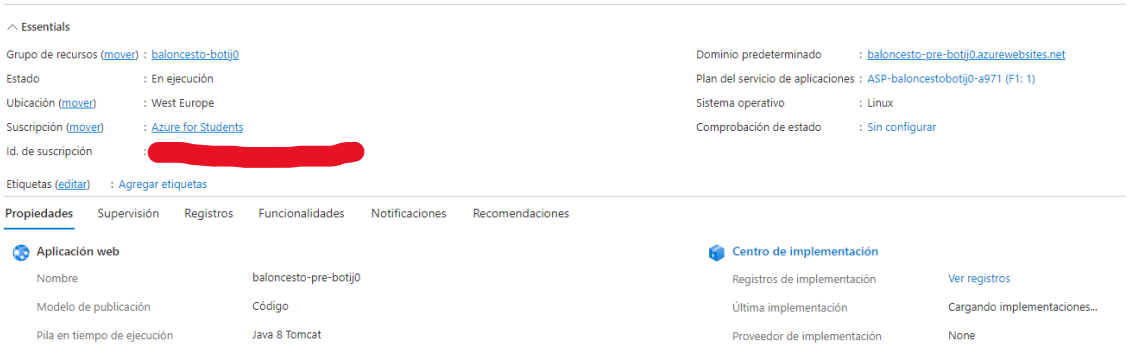
DESARROLLO

A continuación, se procede a explicar los pasos llevados a cabo para cada uno de los requisitos de la asignatura. Para ello se ha seguido el mismo orden sugerido en el enunciado del trabajo final.

Trabajo Stage.

El proceso para llevar a cabo este requisito ha sido similar al proceso seguido por el trabajo "deploy" implementado en la práctica guiada realizada en clase.

Para ello me he dirigido a Microsoft Azure, entrando con mi cuenta educativa donde tengo desplegada la máquina virtual que simula el entorno de producción. Una vez allí lo que he realizado es crear otra máquina virtual con las mismas características que la anterior, pero cambiando el nombre para diferenciarlas.



Una vez creada la máquina, he configurado el archivo "main.yaml" para añadir el nuevo trabajo entre el trabajo "qa" y el de "deploy". Para ello se han añadido las siguientes líneas al archivo.

```
  stage:
      needs: qa
      runs-on: ubuntu-latest
      steps:
          - name: descargar repostorio
            uses: actions/checkout@v3
          - name: compilar la aplicación sin repetir pruebas
```

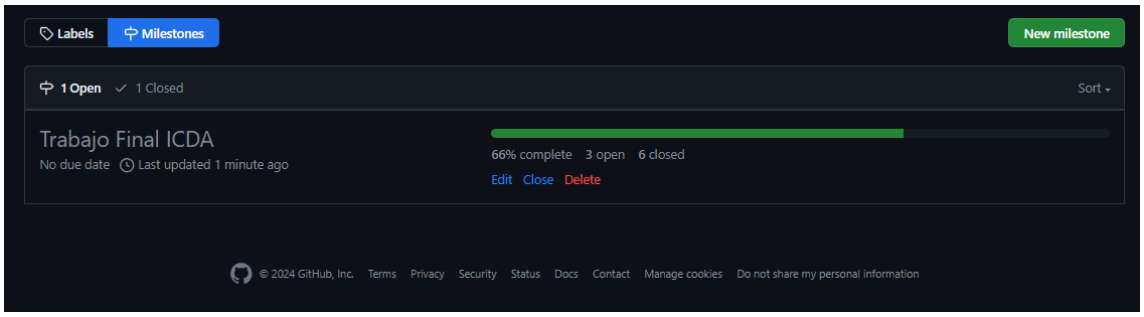
```
            run: mvn package -DskipTests=true
          - name: desplegar en Azure
            uses: Azure/webapps-deploy@v2
            with:
                app-name: baloncesto-pre-botij0
                publish-profile: ${{
secrets.AZURE_WEBAPP_PRE_PUBLISH_PROFILE }}
                package: target/*.war
```

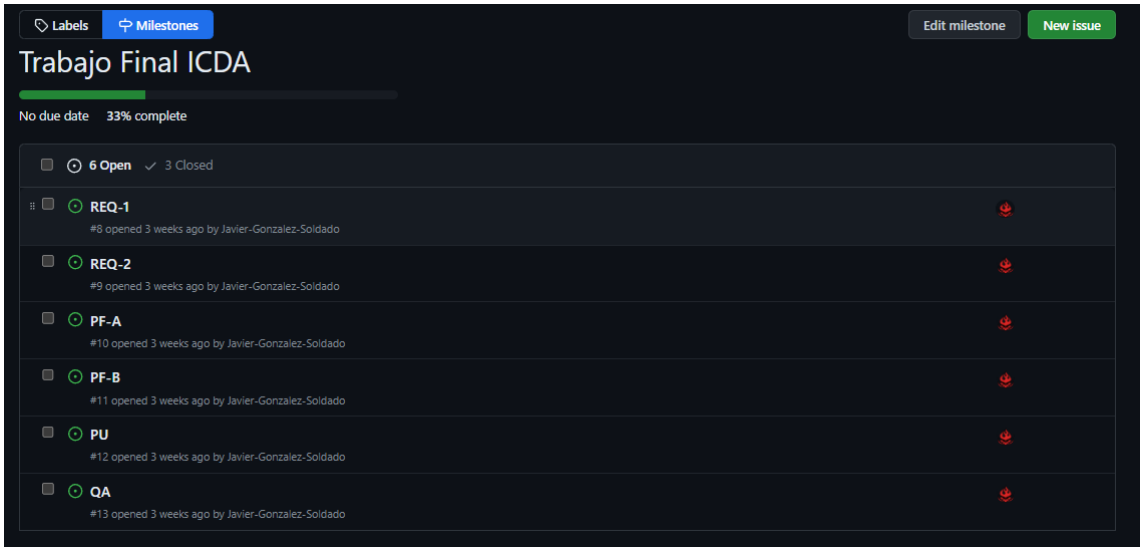
Es importante tener en cuenta que se ha configurado un nuevo secreto con el perfil de publicación de la nueva máquina que alojará el estado de preproducción. Así pues cada vez que realicemos un push y no se produzca ningún fallo en los trabajos previos, se desplegará la aplicación en esta máquina.

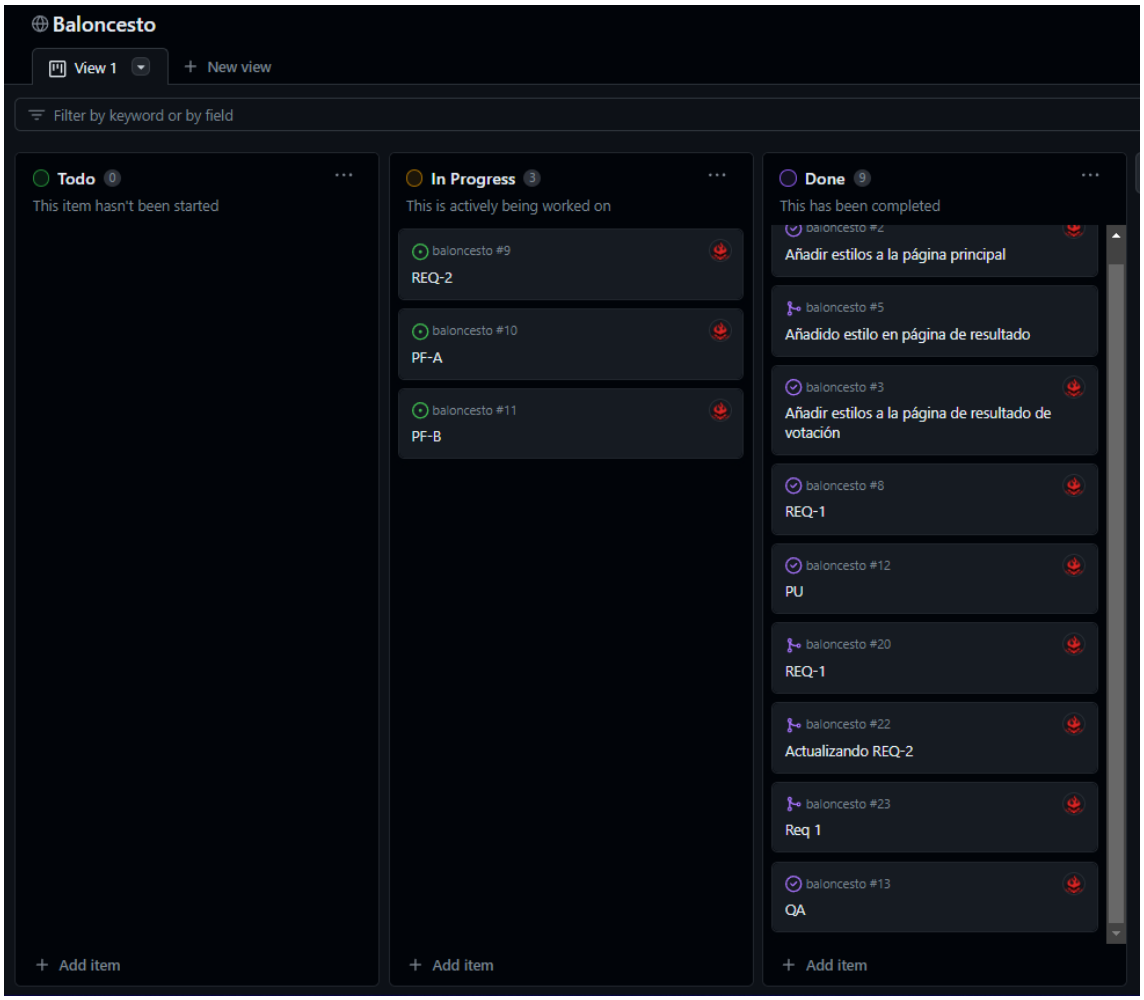
Creación del Sprint (Milestone).

Para llevar a cabo los demás requisitos se realizó un nuevo Sprint, con los requisitos solicitados en el enunciado como issues. Para ello se siguieron los mismos pasos que en la práctica guiada y se utilizó un tablero Kanban para visualizar el progreso de la realización de las issues.

A continuación, se muestran una serie de imágenes del Sprint/Milestone y de las issues:





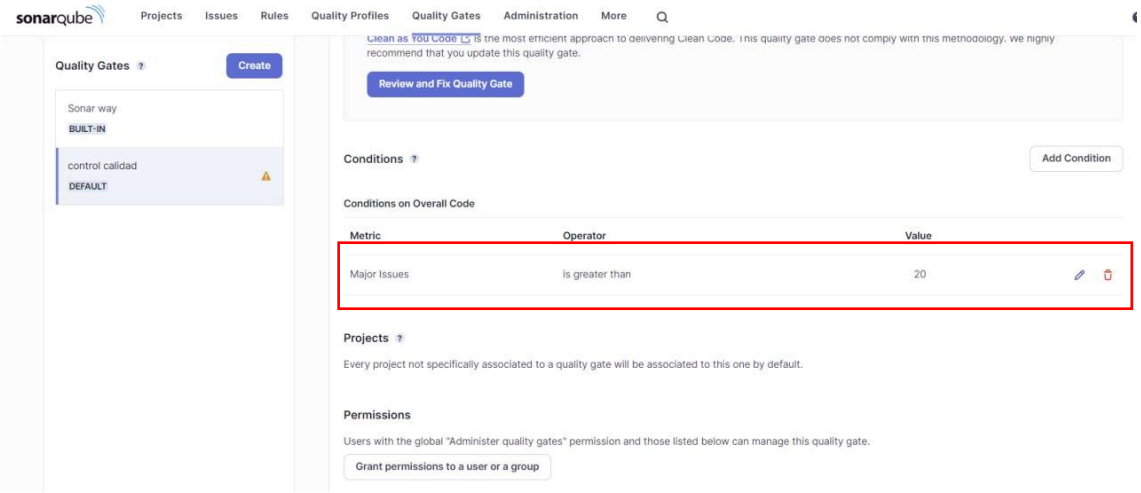


QA: 20 major issues.

Para garantizar el mínimo de calidad de que el código total del proyecto no tuviera más de 20 major issues, se configuró SonarQube.

Para ello accedimos a la interfaz web que nos ofrece, accediendo desde el navegador con los credenciales configuradas previamente. Una vez dentro, nos dirigimos al apartado “Quality Gates”.

Una vez ahí, en el apartado “Conditions”, específicamente en el apartado “Conditions on Overall Code” simplemente asignamos/modificamos una regla para que quede de la siguiente forma:



Me gustaría mencionar que, al comienzo del desarrollo de la práctica, había varias condiciones en SonarQube para el código nuevo, siendo estas condiciones muy estrictas lo que hizo que fallara el despliegue. Opté por eliminar todas y simplemente dejar como única condición el requisito solicitado en el enunciado.

Hay que añadir que se mejoró la calidad del código proporcionado, quitando elementos del html que estaban obsoletos, o añadiendo loggers a los métodos de la clase ModeloDatos.java.

Finalmente, para evitar que se despliegue el código si el criterio de calidad no se cumple, se modificó el archivo “main.yaml” poniendo el campo “continue-on-error” en falso:

```
qa:
        needs: test
        runs-on: self-hosted
        continue-on-error: false
        steps:
            - name: calidad-codigo
              run: |
                  mvn sonar:sonar -
Dsonar.host.url=http://sonarqube:9000 -Dsonar.qualitygate.wait=true -
Dsonar.login=admin -Dsonar.password=javier
```

REQ-1: Poner votos a cero.

Para el cumplimiento de este requisito se creó primero una rama (REQ-1) desde main. Una vez creada la rama, se procedió con el desarrollo de código. Primero se creó un botón, dentro de un formulario con la misma acción y método que el anterior, en la página principal “index.html”:

```
        <form action="Acb" method="POST">
            <button type="submit" name="votosACero" id="votosACero">
                Poner votos a cero
            </button>
        </form>
```

Después en el método service del archivo “Acb.java” se añadieron las siguientes líneas que comprobaba si se había pulsado el botón “votosACero” y si era así llama al método correspondiente de la clase ModeloDatos:

```
        // Si se ha pulsado el botón de reiniciar votos, se reinician
y se redirige a la página principal
        if (req.getParameter("votosACero") != null) {
            bd.reiniciarVotos();
            res.sendRedirect(res.encodeRedirectURL("index.html"));
            return;
        }
```

El método reiniciarVotos() dentro de la clase ModeloDatos es el siguiente:

```
    public void reiniciarVotos(){
        try {
            set = con.createStatement();
            set.executeUpdate("UPDATE Jugadores SET votos=0");
            rs.close();
            set.close();
        } catch (Exception e) {
            // No modifica la tabla
            logger.warning("No modifica la tabla");
            logger.warning(ERROR + e.getMessage());
        }
    }
```

Con esto realizado, cada vez que se pulse el botón se reinician los votos a su valor inicial cero.

PU: actualizarJugador().

Para llevar a cabo la prueba unitaria, una vez implementado el REQ-1, se creó un nuevo test en el archivo “ModeloDatosTest.java”:

```
    @Test
    public void testActualizarJugador() {
        System.out.println("Prueba de actualizarJugador");

        //setear variables de entorno para los tests
        System.setProperty("DATABASE_HOST",
"jdbc:mysql://localhost");
        System.setProperty("DATABASE_PORT", "3306");
        System.setProperty("DATABASE_NAME", "baloncesto");
        System.setProperty("DATABASE_USER", "usuario");
        System.setProperty("DATABASE_PASS", "clave");

        ModeloDatos instance = new ModeloDatos();
```

```
        instance.abrirConexion();

        String nombre = "Llull";
        int votosLlullPrev = instance.getVotosJugador(nombre);

        instance.actualizarJugador(nombre);

        int votosLlullPost = instance.getVotosJugador(nombre);

        assertEquals(votosLlullPrev + 1, votosLlullPost);

        //resetear votos
        instance.reiniciarVotos();
    }
```

Primero, como los test se ejecutan en un entorno especial, las variables de entorno (que tienen los datos de acceso a la base de datos) no se podían acceder a ellas, por lo que se tuvieron que establecer mediante las sentencias “System.setProperty” como se ve al inicio del método.

También se creó el método getVotosJugador(nombre) que devuelve el número de votos del jugador correspondiente:

```
    public int getVotosJugador(String nombre) {
        int votos = 0;
        try {
            set = con.createStatement();
            rs = set.executeQuery("SELECT votos FROM Jugadores WHERE
nombre " + " LIKE '%" + nombre + "%'");
            while (rs.next()) {
                votos = rs.getInt("votos");
            }
            rs.close();
            set.close();
        } catch (Exception e) {
            // No lee de la tabla
            logger.warning("No lee de la tabla");
            logger.warning(ERROR + e.getMessage());
        }
        return (votos);
    }
```

Además, se tuvo que modificar el archivo “main.yaml” para que ejecutase este trabajo en nuestra máquina ubunto y no en la de github, además de que estableciera primero la base de datos:

```
    build:
        runs-on: self-hosted
        steps:
```

```
            - name: descargar repostorio
              uses: actions/checkout@v3
            - name: preparar base de datos de prueba
              run: |
                  mysql -u root < db/baloncesto.sql
            - name: pruebas-unitarias
              run: mvn test
            - name: compilar la aplicación sin repetir pruebas
              run: mvn package -DskipTests=true
```

He de mencionar que también se modificó el testExisteJugador para que se comprobara accediendo a la base de datos:

```
    @Test
    public void testExisteJugador() {
        System.out.println("Prueba de existeJugador");

        //setear variables de entorno para los tests
        System.setProperty("DATABASE_HOST",
"jdbc:mysql://localhost");
        System.setProperty("DATABASE_PORT", "3306");
        System.setProperty("DATABASE_NAME", "baloncesto");
        System.setProperty("DATABASE_USER", "usuario");
        System.setProperty("DATABASE_PASS", "clave");

        String nombre = "Rudy";
        ModeloDatos instance = new ModeloDatos();
        instance.abrirConexion();
        boolean expResult = true;
        boolean result = instance.existeJugador(nombre);
        assertEquals(expResult, result);
        // fail("Fallo forzado.");
    }
```

Finalmente, para que los tests funcionen se modificó el método abrirConexión() de la clase ModeloDatos para que al declarar las variables utilizase las de entorno o las definidas en el test:

```
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");

            // Con variables de entorno
            String dbHost = System.getenv("DATABASE_HOST") == null ?
System.getProperty("DATABASE_HOST") : System.getenv("DATABASE_HOST");
            String dbPort = System.getenv("DATABASE_PORT") == null ?
System.getProperty("DATABASE_PORT") : System.getenv("DATABASE_PORT");
            String dbName = System.getenv("DATABASE_NAME") == null ?
System.getProperty("DATABASE_NAME") : System.getenv("DATABASE_NAME");
```

```
            String dbUser = System.getenv("DATABASE_USER") == null ?
System.getProperty("DATABASE_USER") : System.getenv("DATABASE_USER");
            String dbPass = System.getenv("DATABASE_PASS") == null ?
System.getProperty("DATABASE_PASS") : System.getenv("DATABASE_PASS");


            String url = dbHost + ":" + dbPort + "/" + dbName;
            con = DriverManager.getConnection(url, dbUser, dbPass);

        } catch (Exception e) {
            // No se ha conectado
            logger.severe("No se ha podido conectar");
            logger.severe(ERROR + e.getMessage());
        }
    }
```

REQ-2: Ver votos.

Para implementar el segundo requisito, se siguió un procedimiento similar al anterior, pero como para las posteriores pruebas funcionales se requiere del primer requisito, se hizo primero un merge de la rama REQ-1 a main y luego de esta a la rama REQ-2, ya que al iniciar el sprint ya había creado las dos ramas.

Una vez hecho esto, se creó primero un botón en la página principal “index.html” de la misma manera que en el REQ-1:

```
        <form action="Acb" method="POST">
            <button type="submit" name="verVotos" id="verVotos">
                Ver votos
            </button>
        </form>
```

Del mismo modo, en la clase Acb, en el método service se añadío al inicio de este las siguientes líneas:

```
        //Si se ha pulsado el botón de ver votos
        if (req.getParameter("verVotos") != null) {
            // Llamada a la página jsp que muestra una tabla con los
votos
            List<Jugador> jugadores = bd.obtenerVotos();
            req.setAttribute("jugadores", jugadores);
            req.getRequestDispatcher("VerVotos.jsp").forward(req,
res);
        }
```

En la clase ModeloDatos, se creó el método obtenerVotos que devuelve una lista de objetos de tipo Jugador (creada manualmente por mí):

```java
package main.java;

public class Jugador {
    private String nombre;
    private int votos;

    // Constructor
    public Jugador() {
    }

    // Getters y setters
    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public int getVotos() {
        return votos;
    }

    public void setVotos(int votos) {
        this.votos = votos;
    }
}
```

Método obtenerVotos():

```java
    public List<Jugador> obtenerVotos() {
        List<Jugador> jugadores = new ArrayList<>();
        try {
            set = con.createStatement();
            rs = set.executeQuery("SELECT * FROM Jugadores");
            while (rs.next()) {
                Jugador jugador = new Jugador();
                jugador.setNombre(rs.getString("Nombre"));
                jugador.setVotos(rs.getInt("votos"));
                jugadores.add(jugador);
            }
            rs.close();
            set.close();
        } catch (Exception e) {
            // No lee de la tabla
            logger.warning("No lee de la tabla");
            logger.warning(ERROR + e.getMessage());
        }
```

```
        return jugadores;
    }
```

A continuación, se creó un nuevo archivo llamado VerVotos.jsp donde se muestra una tabla con los votos de todos los jugadores que hay almacenados en la base de datos:

```
<%@ page import="java.util.List,main.java.Jugador" %>
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
        <title>Baloncesto</title>
    </head>
    <body>
        <h1>Resultado de las votaciones:</h1>
        <table border="1" id="tablaVotos">
            <thead>
                <tr>
                    <th>Nombre</th>
                    <th>Votos</th>
                </tr>
            </thead>
            <tbody>
                <%
                List<Jugador> jugadores = (List<Jugador>)
request.getAttribute("jugadores");
                for(Jugador jugador : jugadores) {
                %>
                    <tr>
                        <td><%=jugador.getNombre()%></td>
                        <td><%=jugador.getVotos()%></td>
                    </tr>
                <%}%>
            </tbody>
        </table>
        <br />
        <a href="index.html">Volver a la pagina principal</a>
    </body>
</html>
```

PF-A: Poner votos a cero y ver votos.

Para llevar a cabo esta prueba, se creó un nuevo método en la clase PruebasPhatomjsIT.java, basándonos en el existente con el siguiente código:

```
    @Test
    public void votosACeroYVerVotosTest() {
        DesiredCapabilities caps = new DesiredCapabilities();
        caps.setJavascriptEnabled(true);
        caps.setCapability(PhantomJSDriverService.PHANTOMJS_EXECUTABL
E_PATH_PROPERTY, "/usr/bin/phantomjs");
        caps.setCapability(PhantomJSDriverService.PHANTOMJS_CLI_ARGS,
                new String[] { "--web-security=no", "--ignore-ssl-
errors=yes" });
        driver = new PhantomJSDriver(caps);
        driver.navigate().to("http://localhost:8080/Baloncesto/");

        // Pulsa el botón de poner votos a cero
        driver.manage().timeouts().implicitlyWait(2,
TimeUnit.SECONDS);
        driver.findElement(By.id("votosACero")).click();

        // Pulsa el botón de ver votos
        driver.manage().timeouts().implicitlyWait(2,
TimeUnit.SECONDS);
        driver.findElement(By.id("verVotos")).click();

        // Obtiene todas las celdas de la columna "Votos"
        List<WebElement> votosCells =
driver.findElements(By.xpath("//table[@id='tablaVotos']//td[2]"));

        // Verifica que todas las celdas tengan el valor "0"
        boolean todasCeldasSonCero =
votosCells.stream().allMatch(cell -> cell.getText().equals("0"));

        assertEquals(true,todasCeldasSonCero, "El numero de votos no
es correcto");

        driver.close();
        driver.quit();
    }
```

En los comentarios del código se explica el funcionamiento de este.

PF-B: Votar Otro y comprobar votos.

Para llevar a cabo esta prueba, se creó un nuevo método en la clase PruebasPhatomjsIT.java, basándonos en el existente con el siguiente código:

```
    @Test
    public void votarOtroYComprobarVotosTest() {
        DesiredCapabilities caps = new DesiredCapabilities();
        caps.setJavascriptEnabled(true);
        caps.setCapability(PhantomJSDriverService.PHANTOMJS_EXECUTABL
E_PATH_PROPERTY, "/usr/bin/phantomjs");
        caps.setCapability(PhantomJSDriverService.PHANTOMJS_CLI_ARGS,
                new String[] { "--web-security=no", "--ignore-ssl-
errors=yes" });
        driver = new PhantomJSDriver(caps);
        driver.navigate().to("http://localhost:8080/Baloncesto/");

        //Introducimos el nombre del jugador nuevo
        driver.findElement(By.name("txtOtros")).sendKeys("jugadorTest
");

        //Seleccionamos el radio button de Otros
        driver.findElement(By.id("rdbtnOtros")).click();

        //Pulsamos el boton de votar
        driver.findElement(By.name("btnVotar")).click();

        //Volvemos a la página principal
        driver.manage().timeouts().implicitlyWait(2,
TimeUnit.SECONDS);
        driver.navigate().to("http://localhost:8080/Baloncesto/");

        //Pulsamos el boton de ver votos
        driver.manage().timeouts().implicitlyWait(2,
TimeUnit.SECONDS);
        driver.findElement(By.id("verVotos")).click();

        // Encuentra la celda correspondiente al jugador "Pepe" en la
columna de votos
        driver.manage().timeouts().implicitlyWait(2,
TimeUnit.SECONDS);
        WebElement celdaJugadorTest =
driver.findElement(By.xpath("//table[@id='tablaVotos']//td[text()='ju
gadorTest']/following-sibling::td"));

        //Comparamo el valor de la celda con el valor esperado
        assertEquals("1", celdaJugadorTest.getText(), "El numero de
votos no es correcto");
```

```
        driver.close();
        driver.quit();
    }
```

CONCLUSIONES

Como se puede apreciar, el apartado visual de la aplicación es muy simple y no se ha dedicado tiempo a mejorarlo ya que he considerado que queda fuera del contexto de la asignatura, por lo que me he centrado en llevar a cabo los requisitos solicitados por el profesor.

Finalmente comentar que me ha gustado la realización de esta práctica, ya que me parece muy completa en cuanto a todos los requisitos para completarla, además de la aplicación directa de los conocimientos teóricos impartidos por la asignatura.