RappiPay

Refactor de codigo- Prueba tecnica QA automation RappiPay

RappiPay

COMPONENTE PRÁCTICO (Automation)

Contexto: Se requiere implementar un nuevo script para una prueba automatizada. El caso de prueba involucra tres (3) vistas: **Home del App, Login y el Home de producto**. El flujo que se desea automatizar valida que el usuario Ingrese al **Home del App** donde da un tap al widget del producto para posteriormente ser redireccionado al **Login**. Allí ingresará un código de 6 dígitos y finalmente navegará de forma automática al **Home del producto** donde se encuentra el detalle y acciones que el usuario puede realizar sobre el producto.

A. En la vista de Login se implementa un método para validar que todos los labels definidos en el diseño se encuentren presentes. A continuación verá el código implementado.

```java
public boolean verifyLabelsOnScreen() {
    try {
        // Espera para que la pantalla cargue los elementos
        Thread.sleep(15000);

        boolean flagLabelIniciarSesion = false;
        boolean flagLabelHola = false;
        boolean flagLabelIngresarContrasena = false;
        boolean flagLabelOlvideContrasena = false;

        // Recorre todos los labels en la pantalla
        for (MobileElement label : screenLabels) {
            String text = label.getAttribute("text");

            if (text.equals("Iniciar Sesión")) {
                flagLabelIniciarSesion = true;
            }
            if (text.contains("¡Hola")) {
                flagLabelHola = true;
            }
            if (text.equals("Ingresa tu contraseña de 6 dígitos")) {
                flagLabelIngresarContrasena = true;
            }
            if (text.equals("Olvidé mi contraseña")) {
                flagLabelOlvideContrasena = true;
            }
        }

        // Verifica que todos los labels estén presentes
        if (flagLabelIniciarSesion && flagLabelHola
            && flagLabelIngresarContrasena && flagLabelOlvideContrasena) {
            return true;
        }

    } catch (Exception e) {
        return false;
    }
    return false;
}
```

Instrucción: De acuerdo a su experiencia indique si está de acuerdo con la forma en la que fue implementado el método. En caso de no estar de acuerdo. Indíquenos qué modificaciones haría o reescriba el método para ver reflejados sus cambios y justifique

su respuesta.:

Propuesta de refactorización con justificación con dos enfoques:

Antes de dar los dos enfoques, quiero comentar que me baso en estas soluciones observando que se usa un Thread.sleep como espera, pero esta espera, al no ser explícita, vuelve la prueba lenta e inestable. Además, el método está obligado a recorrer todos los labels de la pantalla y eso suele ser frágil al momento de comparar textos exactos, por posibles problemas de copy o carga parcial.
Aplicaría un refactor del código usando esperas explícitas (WebDriverWait) y agregaría una validación directa de cada label por medio de un elemento no variable, es decir, un locator estable (accessibilityId o resource-id). En caso de que no existan IDs disponibles, haría un fallback por texto usando XPath, lo cual sería mi último recurso por ser más sensible, ya que puede verse afectado por cambios en la estructura de la UI.
También añadiría mensajes claros para identificar los labels faltantes en la pantalla.

Dado este contexto acá espongo mis dos enfoques:

1. Validacion por locators si existen:

```
  public boolean verifyLabelsOnScreen() {
try {
    WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(15));

    // usamos  accessibilityId o resource-id  por cada label
    By lblIniciarSesion = MobileBy.AccessibilityId("login_title");
    By lblHola          = MobileBy.AccessibilityId("login_greeting");
    By lblIngresaPass  = MobileBy.AccessibilityId("login_pin_hint");
    By lblOlvidePass   = MobileBy.AccessibilityId("login_forgot_pin");

    wait.until(ExpectedConditions.visibilityOfElementLocated(lblIniciarSesion));
    wait.until(ExpectedConditions.visibilityOfElementLocated(lblHola));
    wait.until(ExpectedConditions.visibilityOfElementLocated(lblIngresaPass));
    wait.until(ExpectedConditions.visibilityOfElementLocated(lblOlvidePass));

    return true;

} catch (TimeoutException e) {
            //aparte de controlar la excepción de manera especifica podría agregar logs
de lo que falto o mandaría screenshot
    return false;
}
```

```
}
```

justificación: En este método aplicamos esperas explícitas, eliminando el sleep quemado, lo cual permite que la validación de los elementos sea más rápida y estable. Si llega a fallar, el control del error se maneja mediante un timeout (try-catch), lo que nos permite tener el error controlado, a diferencia de la implementación anterior donde no era claro qué estaba fallando. En ese mismo punto agregaría un screenshot y un log indicando qué label o elemento está faltando.

Observación: En este método estamos asumiendo que los locators existen, ya sea por accessibilityId o resource-id, y los utilizados son una suposición únicamente para mostrar cómo quedaría el método refactorizado.

2. Validacion por texto dado el caso no existan locators.
   Para este enfoque, dado que estamos asumiendo que no contamos con locators presentes o estables, se aplicaría un fallback por texto específico, ya sea usando contains o normalize-space en XPath. De igual forma, se mantendría la mejora de usar esperas explícitas para asegurar que la validación se realice cuando los elementos realmente estén presentes en pantalla.

```
public boolean verifyLabelsOnScreen() {
    try {
        WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(15));

        // Espera explicita base/ancla/flag
        wait.until(ExpectedConditions.presenceOfElementLocated(
                MobileBy.xpath("//*[contains(@text,'Iniciar') or contains(@label,'Iniciar')]")
        ));

        // Validaciones por texto (fallback)
        boolean iniciarSesion = isTextVisible(wait, "Iniciar Sesión");
        boolean hola          = isTextVisibleContains(wait, "¡Hola");
        boolean ingresaPin    = isTextVisible(wait, "Ingresa tu contraseña de 6 dígitos");
        boolean olvide        = isTextVisible(wait, "Olvidé mi contraseña");

        return iniciarSesion && hola && ingresaPin && olvide;

    } catch (Exception e) {
        return false;
    }
}

private boolean isTextVisible(WebDriverWait wait, String exactText) {
```

Refactor de codigo- Prueba tecnica QA automation RappiPay

RappiPay

```
    try {
        return wait.until(ExpectedConditions.visibilityOfElementLocated(
                MobileBy.xpath("//*[(@text='" + exactText + "') or (@label='" + exactText +
"')]")
        )) != null;
    } catch (TimeoutException e) {
        return false;
    }
}

private boolean isTextVisibleContains(WebDriverWait wait, String partialText) {
    try {
        return wait.until(ExpectedConditions.visibilityOfElementLocated(
                MobileBy.xpath("//*[contains(@text,'" + partialText + "') or
contains(@label,'" + partialText + "')]")
        )) != null;
    } catch (TimeoutException e) {
        return false;
    }
}
```

justificación: Para este enfoque, el método sigue siendo sensible al copy, pero eliminamos la espera con sleep y utilizamos esperas explícitas, lo que hace la validación más estable. Ya no es necesario usar screenLabels, desacoplamos la validación de los labels mediante dos métodos: uno para textos exactos y otro para textos parciales, lo cual nos permite controlar mejor las validaciones. También aplicamos una excepción específica (TimeoutException), con lo que sabemos que el fallo se debe a que un label no apareció en el tiempo esperado. Adicionalmente, se puede complementar con logs y mensajes claros para identificar qué label faltó.

B. En la vista del Home del App, se necesita crear dos métodos: El primero estará encargado de verificar si el usuario se encuentra o no en el Home del App y el segundo realizará la acción de dar tap en el botón de Rappipay para ingresar al login. A continuación verá la definición de los elementos que se tuvieron en cuenta.

RappiPay

Refactor de codigo- Prueba tecnica QA automation RappiPay

RappiPay

```java
@AndroidFindBy(id = "com.grability.rappi:id/menu_image")
private MobileElement buttonMenu;
@AndroidFindBy(id = "com.grability.rappi:id/support_image")
private MobileElement buttonSupport;
@AndroidFindBy(className = "android.widget.TextView")
private List<MobileElement> widgetElements;
@AndroidFindBy(id = "com.grability.rappi:id/imageView_action")
private MobileElement buttonRappiPay;
@AndroidFindBy(id = "layout_primary_action")
private MobileElement BTN_RAPPIPAY_CO;
@AndroidFindBy(xpath = "//*[contains(@text, 'Restaurante')]")
private MobileElement RESTAURANT;
@AndroidFindBy(xpath = "//*[contains(@text, 'Mercado/Ex')]")
private MobileElement MARKET;
@AndroidFindBy(xpath = "//*[contains(@text, 'Turbo')]")
private MobileElement TURBO;

```

Instrucción: De acuerdo a su experiencia indique si está de acuerdo con la forma en la que se definieron los elementos. En caso de no estar de acuerdo. Indíquenos qué modificaciones haría o reescriba el código para ver reflejados sus cambios y justifique su respuesta.

Propuesta de refactorización:

En este refator mantendré PageFactory (@AndroidFindBy) :

```java
// Este elemento lo podemos usar como ancla/flag/base
@AndroidFindBy(id = "com.grability.rappi:id/menu_image")
private MobileElement btnMenu;

@AndroidFindBy(id = "com.grability.rappi:id/support_image")
private MobileElement btnSupport;

@AndroidFindBy(id = "com.grability.rappi:id/imageView_action")
private MobileElement btnRappiPay;

// le hicimos una modificación dejando el id completo para mantener consistencia
@AndroidFindBy(id = "com.grability.rappi:id/layout_primary_action")
private MobileElement btnRappiPayPrimaryAction;

// Aca localizamos lo que asumimos son categorías (si no existen IDs estables, dejo XPath como fallback por texto)

@AndroidFindBy(xpath = "//*[@text='Restaurante' or contains(@text,'Restauran')]")
private MobileElement categoryRestaurant;

@AndroidFindBy(xpath = "//*[contains(@text,'Mercado')]")
private MobileElement categoryMarket;

@AndroidFindBy(xpath = "//*[@text='Turbo' or contains(@text,'Turbo')]")
private MobileElement categoryTurbo;
```

Refactor de codigo- Prueba tecnica QA automation RappiPay

RappiPay

justificación: Siempre se debe priorizar el uso de ID, porque suele ser el locator más estable y rápido. En este caso no sabemos si existen IDs para todos los elementos, por eso se mantuvieron algunos con XPath, pero se dejó claro que los elementos menu_image, layout_primary_action, support_image e imageView_action, al estar definidos por ID, probablemente serán los más confiables y estables, a diferencia de los que quedaron con XPath.

Se eliminó el TextView genérico, porque es demasiado amplio y poco estable. De la forma como estaba implementado, se extraían “todos los textos” en pantalla, y eso incluye banners, tabs, textos informativos, popups, etc. Si la intención era usarlo como validador de Home (confirmar que estoy en el Home), no es lo más óptimo. Es mejor tener un elemento ancla (como menú o soporte) y validarlo con espera explícita, en lugar de recorrer una lista grande de elementos.

También se agregó consistencia al formato de los IDs. Había un elemento cuyo id no estaba completo (layout_primary_action), y se optó por declararlo con el formato completo (com.grability.rappi:id/...) para que sea consistente y más confiable.

Los XPath se dejaron como fallback y se ajustaron para que sean un poco más tolerantes a cambios de copy (por ejemplo usando contains o combinando exact match + contains), entendiendo que siguen siendo más sensibles porque pueden romperse si cambia el texto o la estructura de la UI.

Se mejoró el nombramiento de variables dando contexto al tipo de elemento (btn, category, etc.), para que el código sea más mantenible y consistente.

Observación: para restaurant, market y turbo se asume que son categorías por el contexto del Home, por eso el naming (categoryRestaurant, etc.). Si luego se amplía la información y resulta que son cards, labels u otro tipo de componente, el cambio de nombre es sencillo y el código queda más claro desde el inicio.