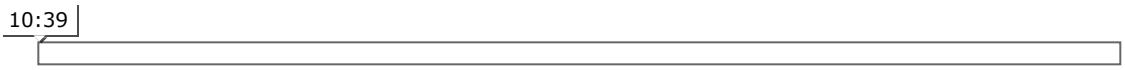
Watch: ⚡ **La estructura de nuestro test: Arrange, Act, Assert**



Puedes encontrar el código de este vídeo en GitHub.

Cuando escribimos un test, tener en cuenta patrones como Arrange, Act, Assert o Given, When, Then nos puede ayudar a detectar que estamos demasiadas cosas en un test y cuando puede ser buena idea separarlo en dos o más tests.

Por ejemplo, si analizamos el primer test que vemos en el vídeo según Arrange, Act, Assert, vemos que estamos haciendo un Act-Assert extra:

```
it("should handle validation successfully", async () => {
    // Arrange
    createUser.mockImplementation(async (data) => await data);

    render(<CreateProfile />);
```

```
    // Act
    const nameInput = screen.getByLabelText(/name/i);
    userEvent.type(nameInput, "Jane");

    const emailInput = screen.getByLabelText(/email/i);
    userEvent.type(emailInput, "jane");

    const button = screen.getByRole("button", { name: /submit/i
});
    userEvent.click(button);

    // Assert
    expect(await screen.findByText(/email is
invalid/i)).toBeInTheDocument();

    // Act
    userEvent.type(emailInput, "jane@example.com");
    userEvent.click(button);

    // Assert
    expect(await screen.findByText(/Thank you
Jane/)).toBeInTheDocument();

    createUser.mockReset();
});
```

Si analizamos por qué, vemos que al hacer dos asserts estamos testeando dos cosas: que sale el mensaje de error, y que el formulario hace el submit después de arreglarlo.

Si lo separamos y aplicamos el patrón de forma estricta, en realidad nos saldrían hasta tres tests:

- Un test para validar que sale el mensaje de error cuando el usuario escribe el mail incorrectamente.
- Un test para validar que el mensaje de error desaparece cuando el usuario arregla el email.
- Un test para validar que sale un mensaje de éxito cuando el usuario rellena el form correctamente.

Puede parecer un enfoque demasiado purista, pero si nos falla uno de los tests vemos rápidamente cuál es el error, mientras que con el enfoque anterior tendríamos que mirar el detalle de qué expect ha

fallado. Si queremos reducir la repetición de código, podemos usar el patrón page object (visto en el vídeo Testeando lo que de verdad importa) o simplemente crear una función en el propio test que rellene el form, por ejemplo.

Aún así, dependiendo del contexto realmente puede tener sentido hacer dos asserts por test. Tenemos que valorar en cada caso que nos puede compensar más y tomarnos estos patrones como una guía, más que como una norma rígida.