Informe Pruebas Unitarias

Evaluación 3 – Programación de Aplicaciones Móviles 006V

Integrantes:

* Nestor Gárate
* René Díaz
* Bernardita Rivas

Contenido

[Introducción 3](#_Toc184086779)

[Objetivo 4](#_Toc184086780)

[Configuración de las Pruebas 5](#_Toc184086781)

[Pasos Realizados para Ejecutar las Pruebas 6](#_Toc184086782)

[Casos de Prueba Realizados 7](#_Toc184086783)

[P5 7](#_Toc184086784)

[P6 8](#_Toc184086785)

[P7 9](#_Toc184086786)

[P8 10](#_Toc184086787)

[P9 11](#_Toc184086788)

[P10 12](#_Toc184086789)

[P11 13](#_Toc184086790)

[Conclusiones 14](#_Toc184086791)

[Bibliografía 15](#_Toc184086792)

# Introducción

Este informe presenta un análisis detallado de las funcionalidades desarrolladas para la aplicación móvil *TeLlevoApp*, una solución de transporte colaborativo creada para facilitar los desplazamientos de los estudiantes del Duoc UC. La aplicación permite tanto a conductores como a pasajeros registrarse, programar viajes, y coordinar transporte en función de sus necesidades.

Entre las funcionalidades clave de *TeLlevoApp* se incluyen el registro de usuarios, la programación de viajes para alumnos con movilización propia y la verificación de transporte para aquellos sin vehículo. Adicionalmente, se destacan aspectos importantes del sistema como la gestión de rutas, la asignación de asientos y la determinación de los costos por persona.

Este informe también detalla las pruebas unitarias implementadas en los componentes de la aplicación, como la validación de registros, la programación de viajes y la verificación de transporte, las cuales aseguran la correcta operación de los sistemas antes de su despliegue completo. La ejecución de pruebas unitarias se realiza mediante el uso de frameworks especializados, como Jasmine y Karma, para garantizar la fiabilidad y eficiencia de la aplicación.

A lo largo de este documento, se describen los pasos realizados, los resultados obtenidos y las conclusiones derivadas del proceso de pruebas, buscando ofrecer una visión integral del desarrollo y evaluación de *TeLlevoApp*.

# Objetivo

El objetivo principal de este informe es asegurar la correcta funcionalidad, estabilidad y desempeño de los componentes clave de la aplicación *TeLlevoApp*, específicamente los módulos *api-html* y *genera-qr*, mediante la implementación de pruebas unitarias. Estas pruebas tienen como propósito verificar el comportamiento de dichos componentes en escenarios predefinidos, identificar posibles errores y garantizar su integración eficiente dentro del sistema global de la aplicación. Además, se busca proporcionar una evaluación exhaustiva del proceso de pruebas, incluyendo la documentación de los resultados obtenidos y las acciones correctivas necesarias, con el fin de optimizar la calidad y confiabilidad de la aplicación antes de su despliegue completo.

***Desarrollo***

# Configuración de las Pruebas

Para realizar las pruebas unitarias, se configuró el entorno de desarrollo siguiendo las mejores prácticas recomendadas para aplicaciones Angular.

El proceso incluyó los siguientes pasos:

1. **Uso de TestBed**:

Se utilizó TestBed, una herramienta de Angular que permite configurar y crear instancias de componentes y servicios en un entorno de prueba. Esto facilitó la simulación del comportamiento real de los componentes en un ambiente controlado.

1. **Framework Jasmine**:

Jasmine fue el framework elegido para la escritura de casos de prueba. Este framework permite crear pruebas de forma clara y estructurada mediante expectativas (expect), verificando el comportamiento esperado de los componentes.

1. **Ejecución con Karma**:

Para ejecutar las pruebas, se empleó Karma, una herramienta que lanza un servidor de pruebas y muestra los resultados en tiempo real en el navegador.

# Pasos Realizados para Ejecutar las Pruebas

La configuración y ejecución de las pruebas unitarias se llevó a cabo en el siguiente orden:

1. **Instalación de dependencias**:

Desde la terminal, se ejecutó el siguiente comando para instalar las dependencias del proyecto: **ng test**

# Casos de Prueba Realizados

## P5

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

|  |  |
| --- | --- |
| **Prueba 5** | **Componente api-html** |
| **Propósito** | Confirmar que el componente api-html se crea correctamente |
| **Estructura:** | it('P5 - Verificar la creación del componente', () => {      }); |
| **Evidencia** | Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente |
| **Resultado** | La prueba fue exitosa; el componente se creó correctamente. |

## P6

Interfaz de usuario gráfica, Texto, Aplicación, Carta, Correo electrónico

Descripción generada automáticamente

|  |  |
| --- | --- |
| **Prueba 6** | **Cargar datos exitosamente en ngOnInit** |
| **Propósito** | Validar que el componente carga los datos desde el servicio y actualiza correctamente las propiedades cantidad\_personajes y personajes. |
| **Estructura:** | it('P6 - Debería cargar datos al iniciar', () => {  spyOn(apiService, 'get').and.returnValue(of({  info: { count: 42 },  results: [{ name: 'Rick', status: 'Alive', image: 'url' }]  }));    component.ngOnInit();  expect(component.cantidad\_personajes).toBe(42);  expect(component.personajes.length).toBe(1);  expect(component.personajes[0].name).toBe('Rick');  }); |
| **Evidencia** | Interfaz de usuario gráfica, Texto, Sitio web  Descripción generada automáticamente |
| **Resultado** | La prueba fue exitosa; cantidad\_personajes y personajes se cargaron con los datos esperados. |

## P7

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

|  |  |
| --- | --- |
| **Prueba 7** | **Creación exitosa del componente** |
| **Propósito** | Verificar que el componente genera-qr se crea correctamente. |
| **Estructura:** | it('P7 - Creación exitosa del componente', () => {  expect(component).toBeTruthy();  }); |
| **Evidencia** | Icono, Rectángulo  Descripción generada automáticamente |
| **Resultado** | La prueba fue exitosa; el componente se creó correctamente. |

## P8

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

|  |  |
| --- | --- |
| **Prueba 8** | **Valor inicial de qrData** |
| **Propósito** | Asegurarse de que el valor inicial de qrData es "Texto de base". |
| **Estructura:** | it('P8 - Valor inicial de qrData', () => {  expect(component.qrData).toBe('Texto de base');  }); |
| **Evidencia** | Icono  Descripción generada automáticamente |
| **Resultado** | La prueba fue exitosa; qrData tiene el valor esperado. |

## P9

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

|  |  |
| --- | --- |
| **Prueba 9** | **Generación del código QR al pulsar un botón** |
| **Propósito** | Comprobar que el método generateQRCode() asigna correctamente qrData a createdCode. |
| **Estructura:** | it('P9 - Generación del código QR al pulsar un botón', () => {  component.qrData = 'Prueba QR';  component.generateQRCode();  expect(component.createdCode).toBe('Prueba QR');  }); |
| **Evidencia** | Icono  Descripción generada automáticamente |
| **Resultado** | La prueba fue exitosa; createdCode se actualizó correctamente. |

## P10

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

|  |  |
| --- | --- |
| **Prueba 10** | **Verificar que el código QR no se muestra inicialmente** |
| **Propósito** | Comprobar que el código QR no se renderiza antes de que el usuario lo genere. |
| **Estructura:** | it('P10 - Verificar que el código QR no se muestra inicialmente', () => {  const compiled = fixture.nativeElement;  expect(compiled.querySelector('qrcode')).toBeNull();  }); |
| **Evidencia** | Rectángulo  Descripción generada automáticamente con confianza baja |
| **Resultado** | La prueba fue exitosa; el código QR no aparece inicialmente en el DOM. |

## P11

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

|  |  |
| --- | --- |
| **Prueba 11** | **Actualización de createdCode al llamar a generateQRCode()** |
| **Propósito** | Verificar que generateQRCode() actualiza createdCode correctamente con nuevos datos. |
| **Estructura:** | it('P11 - Actualización de createdCode', () => {  component.qrData = 'Nuevo texto QR';  component.generateQRCode();  expect(component.createdCode).toBe('Nuevo texto QR');  }); |
| **Evidencia** | Icono  Descripción generada automáticamente |
| **Resultado** | La prueba fue exitosa; createdCode se actualizó con el valor esperado. |

# Conclusiones

A partir de la ejecución de las pruebas unitarias, se obtuvieron los siguientes hallazgos principales:

* **Ejecución exitosa de las pruebas**: Todas las pruebas unitarias definidas para los componentes api-html y genera-qr se ejecutaron correctamente, cumpliendo con los resultados esperados en cada caso.
* **Componente api-html**: Se verificó que el componente carga los datos desde el servicio correctamente, actualizando sus propiedades cantidad\_personajes y personajes. Además, se demostró que maneja adecuadamente los errores en caso de fallas al obtener datos del servicio.
* **Componente genera-qr**: Se confirmó que este componente genera códigos QR de manera correcta al ejecutar el método generateQRCode(). También se validó que el estado inicial del componente cumple con los requerimientos, incluyendo la ausencia del elemento <qrcode> antes de generar el código.

Estos resultados respaldan la funcionalidad básica y la confiabilidad de los componentes bajo prueba, demostrando que cumplen con los requisitos funcionales establecidos y que son capaces de manejar los escenarios previstos de manera efectiva.

# Bibliografía

Para configurar y realizar las pruebas unitarias, se utilizaron las siguientes fuentes:

* Documentación oficial de Angular: <https://angular.io>
* Documentación de Jasmine: <https://jasmine.github.io>
* Documentación de Karma: <https://karma-runner.github.io>