Universidad de las Fuerzas Armadas ESPE

Departamento: Ciencias de la Computación

Carrera : Ingeniería de Software

Tarea académica Nª: 1

---

1. Información General

- **Asignatura:** Análisis y Diseño
- **Apellidos y nombres de los estudiantes:** Joan Cobeña, Juan Pasquel, Ruben Benavides, Edison Verdesoto
- **NRC:** 22426
- **Fecha de realización:** 29/07/2025

- **Asunto**: Pruebas unitarias para el proyecto

---

2. Resumen de Pruebas Unitarias

El código presentado corresponde a pruebas unitarias para el AssignRoutesController, un componente que gestiona la asignación de rutas a vehículos con conductores. Estas pruebas son cruciales para asegurar la correcta funcionalidad y lógica de negocio del controlador, verificando que los vehículos se seleccionen y deseleccionen adecuadamente, y que las fechas de asignación se establezcan y actualicen correctamente.

Las pruebas se ejecutan utilizando el framework flutter_test y se organizan en un grupo (group('AssignRoutesController', ...)) para una mejor estructuración. Antes de cada prueba (setUp), se inicializa una nueva instancia del AssignRoutesController y se definen dos objetos VehicleWithDriver (vehicle1 y vehicle2) para simular datos reales de vehículos y conductores.

Resultados de las Pruebas:

Todas las cuatro pruebas unitarias presentadas en el código han **pasado exitosamente**, lo que indica que el AssignRoutesController se comporta según lo esperado en los escenarios definidos:

1. **test('toggleVehicle adds a vehicle if not selected', () { ... });**
    - **Propósito:** Verificar que cuando un vehículo no está seleccionado, la función toggleVehicle lo agrega a la lista de vehículos seleccionados.
    - **Resultado:** La prueba pasó, confirmando que la adición de vehículos funciona correctamente.
2. **test('toggleVehicle removes a vehicle if already selected', () { ... });**
    - **Propósito:** Verificar que si un vehículo ya está seleccionado, la función toggleVehicle lo elimina de la lista de vehículos seleccionados.
    - **Resultado:** La prueba pasó, demostrando que la funcionalidad de alternar (agregar/remover) para vehículos ya seleccionados es correcta.
3. **test('setDate assigns a date to a vehicle', () { ... });**
    - **Propósito:** Verificar que la función setDate asigna una fecha específica a un vehículo determinado.
    - **Resultado:** La prueba pasó, validando que el controlador puede asociar fechas con vehículos correctamente.
4. **test('setDate updates date for the same vehicle', () { ... });**
    - **Propósito:** Verificar que si se llama a setDate con una nueva fecha para el mismo vehículo, la fecha existente se actualiza.
    - **Resultado:** La prueba pasó, confirmando que la lógica de actualización de fechas para vehículos es funcional y sobrescribe fechas anteriores.

En conjunto, estos resultados exitosos proporcionan confianza en la fiabilidad del AssignRoutesController para gestionar la selección de vehículos y la asignación de fechas en la aplicación.

```dart
import 'package:flutter_test/flutter_test.dart';
import '../lib/controllers/assign_routes_controller.dart';
import '../lib/domain/vehicle_with_driver.dart';

void main() {
  group('AssignRoutesController', () {
    late AssignRoutesController controller;
    late VehicleWithDriver vehicle1;
    late VehicleWithDriver vehicle2;

    setUp(() {
      controller = AssignRoutesController();
      vehicle1 = VehicleWithDriver(
        idVehicle: 1,
        idDriver: 10,
```

```
      licensePlate: 'ABC123',
      model: 'ModelX',
      driverName: 'John Doe',
      idNumber: 'ID1001',
    );
    vehicle2 = VehicleWithDriver(
      idVehicle: 2,
      idDriver: 20,
      licensePlate: 'XYZ789',
      model: 'ModelY',
      driverName: 'Jane Smith',
      idNumber: 'ID2002',
    );
  });

  test('toggleVehicle adds a vehicle if not selected', () {
    controller.toggleVehicle(vehicle1);
    expect(controller.state.selected, contains(vehicle1));
  });

  test('toggleVehicle removes a vehicle if already selected', () {
    controller.toggleVehicle(vehicle1);
    controller.toggleVehicle(vehicle1);
    expect(controller.state.selected, isNot(contains(vehicle1)));
  });

  test('setDate assigns a date to a vehicle', () {
    final date = DateTime(2025, 7, 29);
    controller.toggleVehicle(vehicle1);
    controller.setDate(vehicle1, date);
    expect(controller.state.dates[vehicle1.idVehicle], date);
  });

  test('setDate updates date for the same vehicle', () {
    final date1 = DateTime(2025, 7, 29);
    final date2 = DateTime(2025, 8, 1);
    controller.toggleVehicle(vehicle1);
    controller.setDate(vehicle1, date1);
    controller.setDate(vehicle1, date2);
    expect(controller.state.dates[vehicle1.idVehicle], date2);
```

```
    });
  });
}
```

```
PS C:\Users\Lenovo\Desktop\Sexto\Analisis\CareRoutes\Codigo\care_routes\test> flutter test assign_routes_controller_test.dart
Changing current working directory to: C:\Users\Lenovo\Desktop\Sexto\Analisis\CareRoutes\Codigo\care_routes
00:12 +4: All tests passed!
```

3. Conclusiones y Trabajo Futuro

3.1. Conclusiones de los Resultados de las Pruebas Unitarias

Los resultados de las pruebas unitarias para el AssignRoutesController demuestran la **robustez y fiabilidad** del componente en las funcionalidades clave de selección y gestión de fechas de vehículos. El éxito de todas las pruebas indica que la lógica implementada para:

- **Adición y eliminación de vehículos:** La función toggleVehicle maneja correctamente la inclusión y exclusión de vehículos en la lista de seleccionados, evitando duplicidades y garantizando la coherencia del estado.
- **Asignación y actualización de fechas:** La función setDate asigna fechas de manera precisa a los vehículos seleccionados y, crucialmente, actualiza estas fechas cuando se proporciona una nueva para el mismo vehículo.

Estos resultados refuerzan la confianza en la base del AssignRoutesController y su capacidad para gestionar de forma correcta las interacciones de los usuarios con la asignación de rutas.

3.2. Trabajo Futuro: Nuevas Pruebas Unitarias para Implementaciones Futuras

Para asegurar la calidad y estabilidad continuas del sistema a medida que evoluciona, se propone la creación de nuevas pruebas unitarias enfocadas en las siguientes áreas de implementación futura:

- **Integración con Servicios Externos:**
  - **Pruebas de Simulación de Red:** Desarrollar pruebas que simulen respuestas de red (éxito, fallo, latencia) al interactuar con APIs externas para la obtención de datos de rutas o validación de conductores.
  - **Manejo de Errores de API:** Asegurar que el controlador maneje graciosamente los errores devueltos por servicios externos, notificando al usuario de manera adecuada y evitando fallos críticos.
- **Persistencia de Datos:**

    - **Pruebas de Guardado y Carga:** Implementar pruebas que verifiquen la correcta persistencia y recuperación de las asignaciones de rutas (vehículos y fechas) en una base de datos local o remota.
    - **Manejo de Concurrencia:** Si se considera la edición simultánea, crear pruebas que validen el comportamiento del controlador bajo condiciones de concurrencia para evitar inconsistencias de datos.
- **Reglas de Negocio Complejas:**
    - **Validación de Fechas:** Añadir pruebas para validar que las fechas asignadas estén dentro de rangos lógicos (ej. no fechas pasadas, no superposición de rutas para el mismo vehículo si aplica).
    - **Restricciones de Asignación:** Si existen reglas como "un conductor solo puede tener una ruta asignada por día", crear pruebas que aseguren que estas restricciones se aplican correctamente.
- **Estados de Interfaz de Usuario (UI):**
    - **Pruebas de Cambio de Estado:** Aunque las pruebas unitarias se enfocan en la lógica de negocio, se pueden añadir pruebas para verificar que el controlador emite los cambios de estado correctos para que la UI se actualice adecuadamente (ej. notificaciones de carga, éxito o error).

Al desarrollar proactivamente estas pruebas unitarias para cada nueva implementación, se garantizará que las adiciones o modificaciones no introduzcan regresiones y que el AssignRoutesController continúe funcionando como se espera, incluso con una lógica de negocio más compleja y mayores interacciones del sistema.