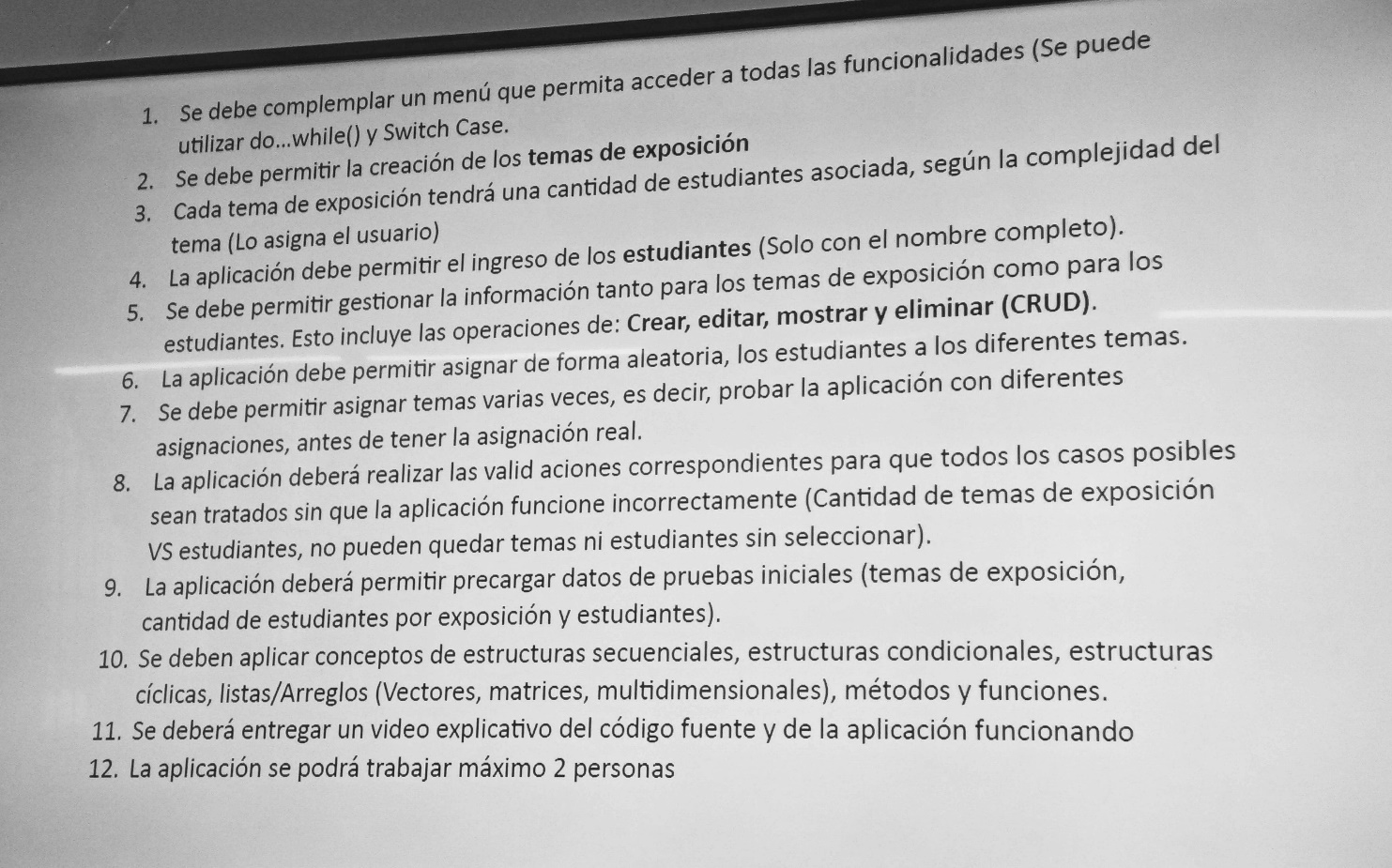
**Evidencias Proyecto**

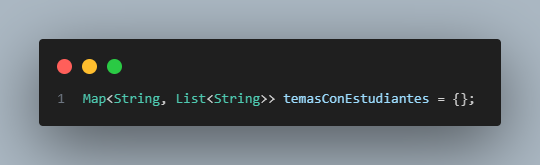
**PROYECTO DART:**

****

**Explicación código:**

****

**INCIALIZACION DEL MAP:**



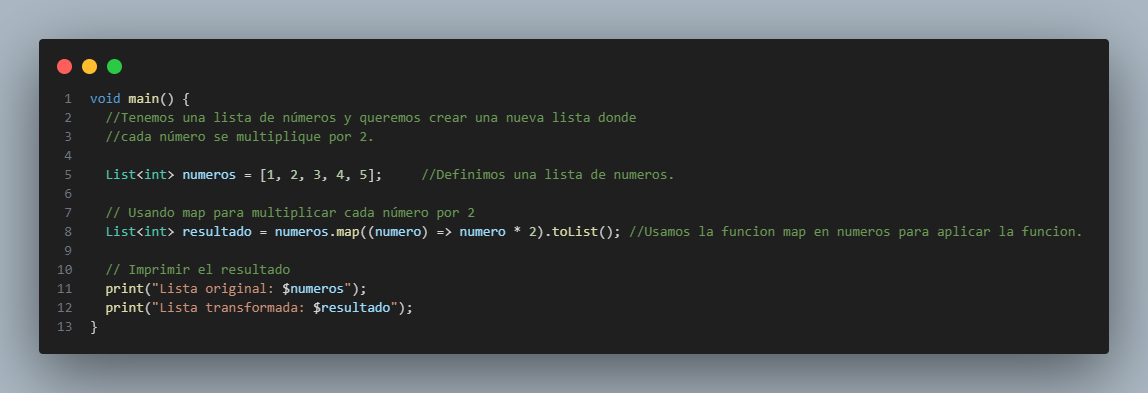
En esta línea de código se temasConEstudiantes es un mapa donde la clave (String) representa el tema y el valor (List<String>) es una lista que contendrá los nombres de los estudiantes asociados a ese tema.

**Método Map:**

El método map iterará sobre cada elemento de un arreglo y devuelve un nuevo arreglo que contiene los resultados de llamar a la función callback en cada elemento.

Se usa básicamente para aplicar una función a cada elemento de una lista interandola y devolver una nueva lista con los resultados.

A continuación, un ejemplo:



En este ejemplo:

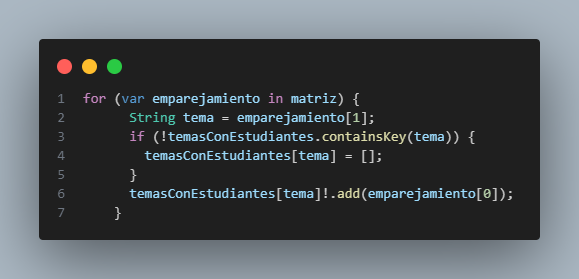
1. Definimos una lista numeros con valores [1, 2, 3, 4, 5].
2. Usamos la función map en numeros para aplicar la función (numero) => numero \* 2 a cada elemento.

* numero representa cada elemento de la lista original.
* numero \* 2 es la operación que se aplica a cada elemento.

1. El resultado se asigna a la lista resultado.
2. Finalmente, imprimimos tanto la lista original como la lista transformada.

En resumen, la función map es muy útil para transformar cada elemento de una lista de manera eficiente y concisa, permitiendo operaciones como mapeo, filtrado o cualquier otra transformación que necesitemos aplicar a los elementos de la lista original.

**ITERACION DE LA MATRIZ EMPAREJAMIENTOS:**



1. En esta líneas de código vemos que matriz es una lista de listas (List<List<String>>) que fue creada anteriormente, donde cada sublista tiene dos elementos: el primero es el nombre del estudiante (emparejamiento[0]) y el segundo es el tema (emparejamiento[1]).

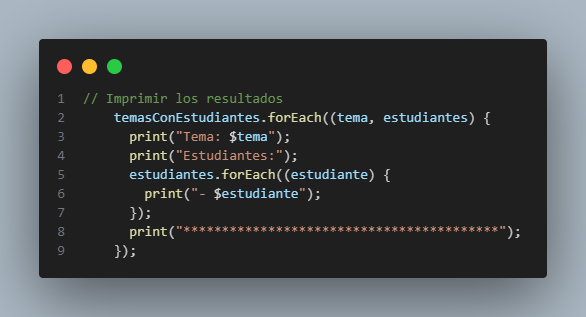
2. El bucle for recorre cada emparejamiento en la lista matriz.

3. Para cada emparejamiento, se obtiene el tema y el estudiante.

4. Se verifica si temasConEstudiantes ya contiene el tema. Si no está presente (!temasConEstudiantes.containsKey(tema)), se inicializa una lista vacía para ese tema.

5. Luego, se añade el estudiante a la lista de estudiantes asociados al tema (temasConEstudiantes[tema]!.add(emparejamiento[0])).

**IMPRESIÓN DE LOS RESULTADOS:**

****

1. temasConEstudiantes.forEach((tema, estudiantes) { ... }): Itera sobre todas las entradas del mapa temasConEstudiantes, donde tema es la clave (nombre del tema) y estudiantes es la lista de estudiantes asociados a ese tema.

2. Dentro del bucle forEach:

- Imprime el nombre del tema.

- Imprime "Estudiantes:".

- Itera sobre la lista de estudiantes.

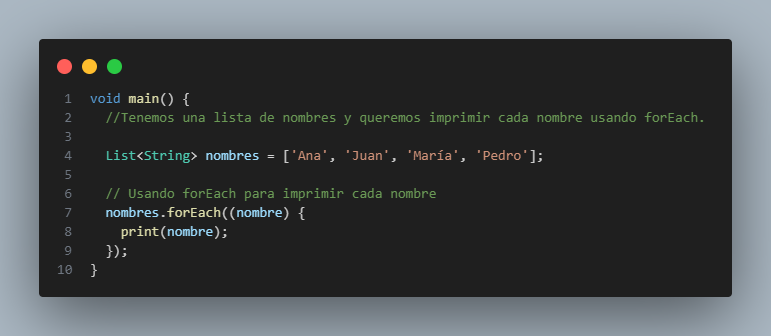
- Imprime cada estudiante precedido por un guión ("- $estudiante").

**Método forEach:**

Se utiliza para iterar sobre los elementos de una lista y aplicar una función a cada elemento de manera secuencial.

La función forEach permite ejecutar una acción (o función o un bloque de código) sobre cada elemento de una lista iterable secuencualmente.

A continuación un ejemplo:



En este ejemplo:

1. Definimos una lista nombres con valores ['Ana', 'Juan', 'María', 'Pedro'].
2. Usamos la función forEach en nombres para aplicar la función (nombre) => print(nombre) a cada elemento.

* nombre representa cada elemento de la lista original.
* print(nombre) es la acción que se ejecuta para cada elemento, que en este caso imprime el nombre en la consola.

1. La función forEach recorre secuencialmente todos los elementos de la lista y ejecuta la acción especificada para cada uno.

En resumen, la función forEach es útil cuando queremos realizar una operación simple o imprimir cada elemento de una lista, sin necesidad de crear una nueva lista o modificar los elementos existentes. Es una forma eficiente y clara de iterar sobre los elementos de un iterable en Dart.

**CONCLUSIONES:**

Esta sección de código realiza los siguientes pasos:

1. Preparación: Inicializa un mapa para almacenar temas y estudiantes.
2. Asignación: Itera sobre una lista de emparejamientos (matriz) para asignar cada estudiante a un tema específico.

1. Presentación: Imprime los resultados organizados por tema, mostrando todos los estudiantes asignados a cada tema.

Esta estructura asegura que cada tema tenga una lista de estudiantes asociados correctamente y presenta los resultados de manera clara y ordenada.