Selection View Go Run Terminal Help

est.dart 9+ ×

ers > Alunos > Desktop > AV07 > db_test.dart > main

```
      ); // Dog
    }); // List.generate
  }

  Future<void> updateDog(Dog dog) async {
    final db = await database;

    await db.update(
      'dogs',
      dog.toMap(),
      where: 'id = ?',
      whereArgs: [dog.id],
    );
  }

  Future<void> deleteDog(int id) async {
    final db = await database;

    await db.delete(
      'dogs',
      where: 'id = ?',
      whereArgs: [id],
    );
  }

  var fido = const Dog(
    id: 0,
    name: 'Fido',
    age: 35,
  );

  await insertDog(fido);

  print(await dogs());

  fido = Dog(
    id: fido.id,
    name: fido.name,
    age: fido.age + 7,
  );
  await updateDog(fido);

  print(await dogs());

  await deleteDog(fido.id);

  print(await dogs());
```

Selection View Go Run Terminal Help

st.dart 9+ ×

rs > Alunos > Desktop > AV07 > db_test.dart > main

```
      return db.execute(
        'CREATE TABLE dogs(id INTEGER PRIMARY KEY, name TEXT, age INTEGER)',
      );
    },
    version: 1,
  );

  Future<void> insertDog(Dog dog) async {
    final db = await database;

    await db.insert(
      'dogs',
      dog.toMap(),
      conflictAlgorithm: ConflictAlgorithm.replace,
    );
  }

  Future<List<Dog>> dogs() async {
    final db = await database;

    final List<Map<String, dynamic>> maps = await db.query('dogs');

    return List.generate(maps.length, (i) {
      return Dog(
        id: maps[i]['id'] as int,
        name: maps[i]['name'] as String,
        age: maps[i]['age'] as int,
      ); // Dog
    }); // List.generate
  }

  Future<void> updateDog(Dog dog) async {
    final db = await database;

    await db.update(
      'dogs',
      dog.toMap(),
      where: 'id = ?',
      whereArgs: [dog.id],
    );
  }

  Future<void> deleteDog(int id) async {
    final db = await database;

    await db.delete(
      'dogs',
      where: 'id = ?',
      whereArgs: [id],
```