Get flora and fauna data

```dart
import 'dart:async';
import 'dart:convert';


import 'package:http/http.dart' as http;
```

Packages used

```dart
Future<FloraFauna> fetchFloraFauna() async {
  final response =
      await http.get(
'https://apps.des.qld.gov.au/species/?op=getkingdomnames&f=json'
);

  if (response.statusCode == 200) {
    // If the server did return a 200 OK response,
    // then parse the JSON.
    return FloraFauna.fromJson(json.decode(response.body));
  } else {
    // If the server did not return a 200 OK response,
    // then throw an exception.
    throw Exception('Failed to load FloraFauna');
  }
}
```

get response from URL

If the server returned a 0K response, return the body of the respose
Else, throw an exception

```dart
class FloraFauna {

  final List kingdom;


  FloraFauna({ this.kingdom});

  factory FloraFauna.fromJson(Map<String, dynamic> json) {
    return FloraFauna(
      kingdom: json['Kingdom'],

    );
  }
}
```

The FloraFauna Class. Parses JSON data.

Initialise database

```dart
initDB() async {
  Directory documentsDirectory = await getApplicationDocumentsDirectory();
  String path = join(documentsDirectory.path, "TestDB.db");
  return await openDatabase(path, version: 1, onOpen: (db) {},
      onCreate: (Database db, int version) async {
    await db.execute("CREATE TABLE Stories ("
        "id INTEGER PRIMARY KEY,"
        "heading TEXT,"
        "context TEXT"
        ");");
    await db.execute("CREATE TABLE Tasks ("
        "id INTEGER PRIMARY KEY,"
        "heading TEXT,"
        "context TEXT,"
        "due INTEGER,"
        "status INTEGER"
        ");");
    await db.execute("CREATE TABLE Changes ("
        "id INTEGER PRIMARY KEY,"
        "storyid INTEGER,"
        "datetime INTEGER,"
        "newValue TEXT,"
        "oldValue TEXT"
        ");");
  });
}
```

Runs after creation of database

Gets the directory that the apps data will be stored in.
This works on android, ios, and desktop

creates the path where the database will be stored using the directory from the previos line

SQLite queries
Create tables in the database

`async` Means the code in the function will run asynchronously.

`await` Program will await completion of this task before continueing.
can only be used in flucitons marked as async

UI CODE EXAMPLE

Tasks Card code

This is the card that is generated for each task. Stories use a similar card

```
class _TaskCardState extends State<TaskCard> {
  bool test = false;
  Task item;

  Color checkDueDate(dueDate) {
    // determines if the card should be red or blue depending on its completion status
    if (dueDate > DateTime.now().millisecondsSinceEpoch / 1000) {
      return Colors.blueAccent;
    } else if (item.status) {
      return Colors.blueAccent;
    } else {
      return Colors.red;
    }
  }

  _TaskCardState(this.item);
  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.all(8),
      child: InkWell(
        onTap: () {
          //TODO: modify for tasks needs to be built
          // Navigator.push(
          //   context,
          //   MaterialPageRoute(
          //     builder: (context) => Modify(item),
          //   ),
          // );
        },
        child: Container(
          width: 220,
          child: Card(
            color: checkDueDate(item.due),
            child: Row(
              children: [
                Padding(
                  padding: const EdgeInsets.all(8.0),
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: <Widget>[
                      Container(
                        width: 180,
                        child: Text(
                          item.heading,
                          style: TextStyle(
                            fontSize: 18,
                            fontWeight: FontWeight.bold,
                            color: Colors.white,
                          ),
                        ),
                      ),
                      Container(
                        width: 180,
                        child: Text(item.context),
                      ),
                      Container(
                        child: Checkbox(
                          value: item.status,
                          activeColor: Colors.lightBlueAccent,
                          onChanged: (bool newValue) {
                            DBProvider.db.changeStatus(item);
                            setState(
                              () {
                                item.status = newValue;
                              },
                            );
                          },
                        ),
                      ),
                    ],
                  ),
                ),
              ],
            ),
          ),
        ),
      );
    }
  }
```

Variables

Function that determines if the card should be red or blue depending on its completion status. Returns a colour

Adds padding around the card to make it look cleaner

Inkwell makes any child widget a button. in this case it makes the whole card a button that should open the modify page

Gets the colour of the card depending on its completion status. If it's complete, it should return blue. If it's past its due date and also uncomplete, It will return red

Aligns children of column at start of row

This container constrains the width of the tasks heading/context so it doesn't overflow the card

The tasks heading

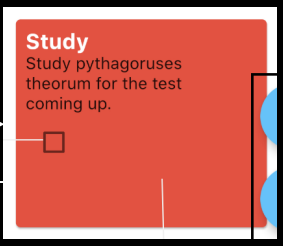
Style of heading

tasks context

Checkbox that shows the completion status of the task.

changes the status in the database

setState rebuilds the card with the changed status



this is what this code would return if:

item.heading = "Study"
item.context = "Study Pythagorases is theorem for the test coming up"
item.status = false
item.due = < current DateTime

New task code

```
newTask(Task newTask) async {
    final db = await database;
    //get the biggest id in the table
    var table = await db.rawQuery(
"SELECT MAX(id)+1 as id FROM Tasks");
    int id = table.first["id"];
    //insert to the table using the new id
    var raw = await db.rawInsert(

"INSERT Into Tasks (id,heading,context,due,status)"
        " VALUES (?,?,?,?,?)",
        [
          id,
          newTask.heading,
          newTask.context,
          newTask.due,
          newTask.status,
        ]);
    return raw;
  }
```

Gets database

Gets the id fot the new task.
current biggest ID + 1

SQLite query

Insert new task

Values to insert.
They replace the ? in the query