Async tables in flutter

Repo: https://github.com/V0idSeeker/tableTest.git
Requirements:
Flutter sdk version: 3.31.0-0.1.pre
How to switch to it?
From vs code terminal navigate to your flutter sdk and run the following commands:
git fetch --all --tags
git tag
git checkout 3.31.0-0.1.pre
flutter doctor

Packages:
https://pub.dev/packages/data_table_2
And get for state management

Explanation:

File 1:

Pagination and organizing search results

Create a dart class that extends `AsyncDataTableSource` and it will contain:

- a list of map <String, dynamic>
- attributes

```
String search = "";
int sortIndex = 0;
bool asc = true;
```

- constructor initializes the object's properties
- getRows method

About getRows:
getRows is an **asynchronous function that** will have 2 parameters
startIndex: The index from which to start fetching the data.
count: The number of data items to fetch.

1- Simulates a delay of 2 seconds to mimic a network request or data processing time.

```
    await Future.delayed(Duration(seconds: 2));
```

2- Receive data

```
    List<Map<String, dynamic>> studentsList = dataList;
```

3- Filter Based on the search field is empty no filtering is applied, If not filter based on sortIndex.
sortIndex determines the sorting criteria, where if it equals:

0 → Sort by **ID**
1 → Sort by **First Name**
2 → Sort by **Last Name**
3 → Sort by **Arabic Last Name**
4 → Sort by **Arabic First Name**

4- Handle order based on the value of asc
if true(ascending) if false (descending)

```
log("asc is $asc");
    if (!asc) studentsList = studentsList.reversed.toList();
```

5- Pagination
Retrieve a specific page that contains data

```
    List<Map<String, dynamic>> paginatedData =
        studentsList.skip(startIndex).take(count).toList();
```

pagination in Dart:

- .skip(startIndex): Skips the first startIndex items.
- .take(count): Takes the next count items.

Example:

If start index = 3 and count =5, it means skip the first 3 elements and and start from the 4th element with index 3

Then take 5 items from there.

Why Does startIndex Seem to Take the Value of count?

In pagination, the **startIndex** is calculated as:

startIndex = (pageNum - 1) * count

6- The return:

```
AsyncRowsResponse(
      studentsList.length,
      paginatedData.map((item) {
        return DataRow(
          cells: [
            DataCell(Text(item["id"].toString())),
            DataCell(Text(item["lastName"].toString())),
            DataCell(Text(item["firstName"].toString())),
            DataCell(Text(item["arabicLastName"].toString())),
            DataCell(Text(item["arabicFirstName"].toString())),
          ],
        );
      }).toList(),
    );
```

In a DataTable, the order and number of DataCells in each DataRow must match the order and number of DataColumns.

Why Should They Match?

- **Alignment**: The intersection of a DataColumn and a DataRow represents a specific cell. If the order or count doesn't match, the data displayed will be misaligned.

It's like we are saying go through the paginated table which is actually a map and access the value of the key (whatever) convert it to string and then return it in a text widget for display.

File 2:

Create a Stateless widget class with:
attributes (observable)
Why asc and sortIndex are also reactive?
Because later on in our code we'll make it possible for each one of them to get updated and that's by clicking on columns headers.

```
  RxString search = "".obs;
  RxBool asc = true.obs;// bcs some of the column are in arabic and some
are in french
  RxInt sortIndex = 0.obs;
```

About the return:

- Search index will get updated relatively to user input

```
TextField(
            onSubmitted: (value) {
              search.value = value;
            },
          ),
```

- also

```
Obx(
              () => AsyncPaginatedDataTable2(
//what to show when there is no search results
                empty: Text("Empty),
                rowsPerPage: 10,
                columns: [
                  DataColumn(
/The DataTable assigns an index to each DataColumn based on its position
in the list*/
                    label: Text("Matriculate"),
                    onSort: (index, ascend) {
//Make asc.value switch between true and false.
                      asc.value = asc.isFalse;
                      sortIndex.value = index;//index = 0
                    },
                  ),
```

```
                // add more columns
                  ]
              //MyDataTableSource is the class we defined in file 1
                source: MyDataTableSource(
                  search: search.value,
                  sortIndex: sortIndex.value,
                  asc: asc.value,
                ),

)
```