Ex 3 : State management

Starting date: @2020년 9월 25일

Deadline: @2020년 10월 9일

Instructions

In this exercise you will have to add states to your project. You will have to add the Favorite page. Once that done create a Popup Widget which you will show when you click on add to my list on Detail Page.
Create the list game store then use it to show your list of the game. Add the user model and store and add a user ID to the game model (it indicate which user create the game in the database : the only person except the admin who can edit the game). Create a dummy user in you user store.

This dummy user should also have list of game (the games that he owns - the one that are shown on the homepage). The user then should be able to add games to his list. (to add a game to the list, click on the + on the detail page then file the form). Once he have add it to the list, the icon + change to - (remove the game from the list).

Add the games, publishers, platforms, genres providers. Add a userId to the game. Get the data from the store on the Settings page.

Also when you click on favorite it change the isFavorite state.

Also manage the completed : when you slide it to the right, the game progression get to 100.

Objectives

- Manage states
- Learn how to use Provider and Consumer
- Install Flutter package (Provider)
- Create a model and use it
- Know how to use Stream

Documentation

You don't have to read all the documentation (because it gonna takes a long time) but you can refers to it when you don't know how to do something or when you have issues.

State management :

- https://flutter.dev/docs/development/data-and-backend/state-mgmt
- https://flutter.dev/docs/development/data-and-backend/state-mgmt/simple

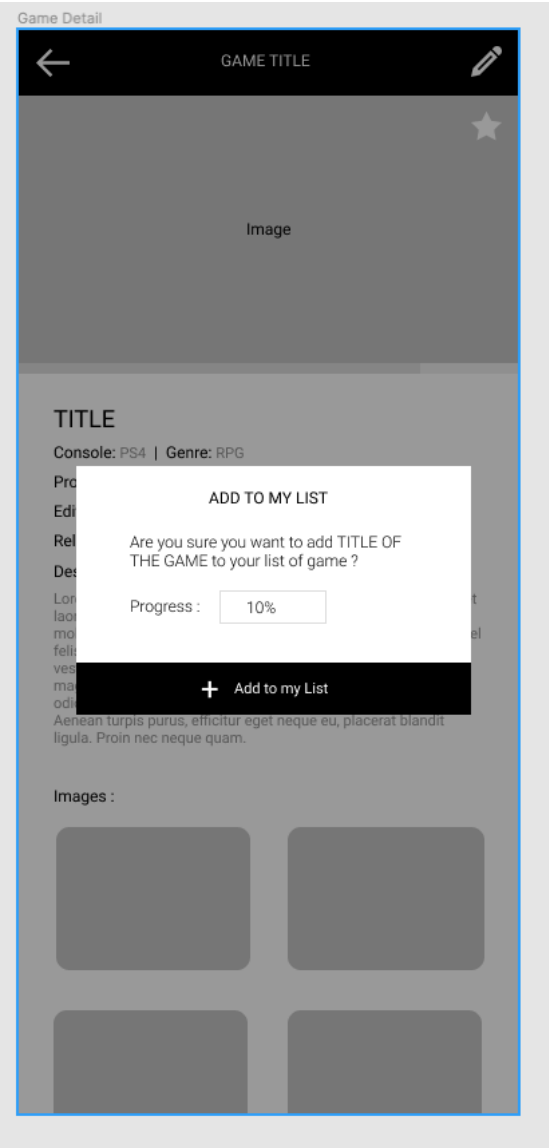
Provider

- https://pub.dev/packages/provider (package)

Stream

- https://api.flutter.dev/flutter/widgets/StreamBuilder-class.html

Screens



- **Detail page** : When you click on the add button on the header, there is a popup that shows which ask you if you want to add the game to the list. It also ask you for your progression in the game.
- **My favorite page**: contains the list all the games the user have marked as favorite. When you click on a card it goes to the detail page when you can remove it as a favorite.
  For the header you can add a filter for this page too if you have time.

Widgets

- For the Modal you can use AlertDialogue widget : https://api.flutter.dev/flutter/material/AlertDialog-class.html

- For the grid you can use GridView widget (it works a bit like ListView) : https://api.flutter.dev/flutter/widgets/GridView-class.html

State management

It can be difficult to pass data via constructors and it can impact the application performance.

When the data change the user interface change (it reflect the data). This data is a **state : Data which affects the UI (User Interface) and which might change over time.**

There is two kind of state :

- App-wide state :
  - Affects the entire app or significant parts of the app
  - Authentication, loaded Products
- Widget (local) state
  - Affects only a widget on its own (does not affect other widgets)
  - Should a loading spinner be displayed, form input

It's important to understand the state concept. If you don't understand after reading that and the documentation, don't hesitate to ask me.

Provider

You have a central store (data container / data provider) in your app which you attach to a widget to your app (doesn't always have to be the root widget of your app). Once you have done that, all child widget can listen to that provider without passing data through their constructor but instead by adding a listener with the of(context). The build where you attached your listener to will run whenever the data / state changes.

Example :

provider

```
import 'package:flutter/material.dart';

import '../models/products.dart';

class Products with ChangeNotifier {
  List<Product> _items = [];

  List<product> get items {
    return [...items];
  }

  void addProduct(value) {
    _items.add(value);
    notifyListeners();
  }
}
```

main.dart

```
import 'package:provider/provider.dart';
import './providers/products.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return ChangeNotifierProvider( // let you listen to a class widget, when it update then the child that listen will rebuild
      create: (ctx) => Product(),
      child: MaterialApp(
```

```
        ...
      )
    )
  }
}
```

For multiProvider :

```
import 'package:provider/provider.dart';
import './providers/products.dart';
import './providers/cart.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MultiProvider(
      providers: [
        ChangeNotifierProvider(create: (_) => Products()),
        ChangeNotifierProvider(create: (_) => Cart()),
      ],
      child: MaterialApp(
        ...
      )
    )
  }
}
```

screen

```
Widget build(BuildContext) {
  // final product = Provider.of<Product>(context, listen: false) - can be used instead of Consumer but Consumer doesn't trigger the w
  return Consumer<Product>(
    builder: (ctx, product, child) => (
      IconButton(
        icon: Icon(
          product.items.isFavorite ? Icons.favorite : Icons.favorite_border
        )
      )
    )
  )
}
```