W4 PRACTICE

*SINGLETONS, REPOSITORIES, STATE DRILLING*

## *Learning objectives*

* **Refactor** the Location and Rides services using a **Repository** and a **Singleton Design** pattern
* **Implement** the *ride preferences modal* on Rides screen
* **Observe** the state drilling and the issues related to Flutter states
* **Implement** the *rides filtering modal* on Rides screen

## *How to start?*

* Copy the **START CODE** into your current repository and push it

BLA-100- Start Code Week 4

* Each task of this practice shall be related to commit(s) including the **tasks ID.**

BLA-001 - Create the BlaButton

## *How to submit?*

* Once finished, submit on MS Team
  + Your GitHub repository URL
  + This document if needed

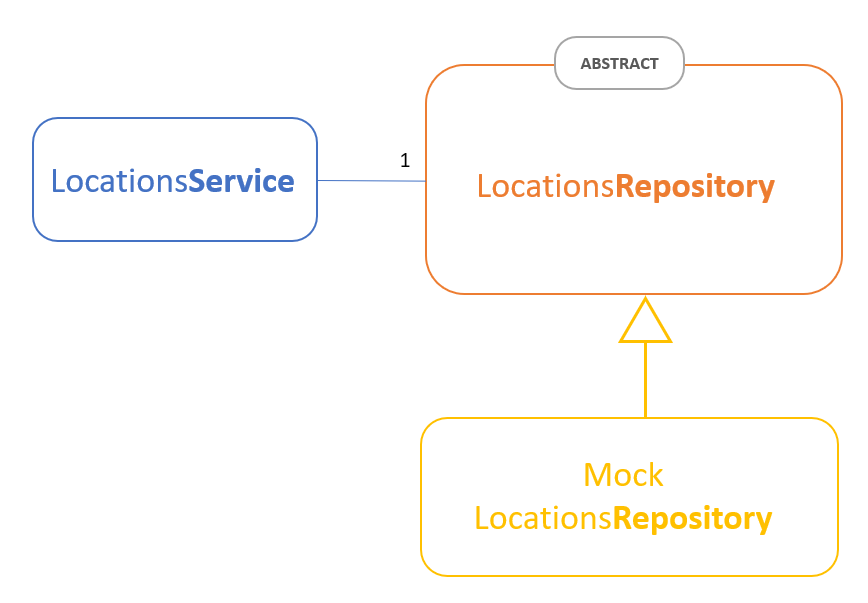


This practice focusses more **on clean code structure** rather than Flutter technical skills.

We will evaluate how well **you follow the coding conventions**, how your name your class, variables etc.

**BLA-101** – Refactor Location Service

The goal of this task is allowing the Location Service to connect to different data source, without changing the service API.

**

**Q1 –** First review the existing RidePreferencesService and the RidePreferencesRepository.

* How do service and repository **interact**?

When there have request come to service then the service processes it and asks the repository for data and the repository fetches data from the database and returns it to the service

* How/Where do we **specify the repository** to be used by the service?

The repository is injected into the service through the constructor. This allows the service to use the repository’s methods

**Q2 –** Then implement the refactoring:

* Create a **LocationsRepository (**abstract)
* Create a mock implementation of this repository (MockLocationsRepository)

|  |  |
| --- | --- |
| City | Country |
| Phnom Penh | Cambodia |
| Siem Reap | Cambodia |
| Battambang | Cambodia |
| Sihanoukville | Cambodia |
| Kampot | Cambodia |

*The mock repository should provide the above locations.*

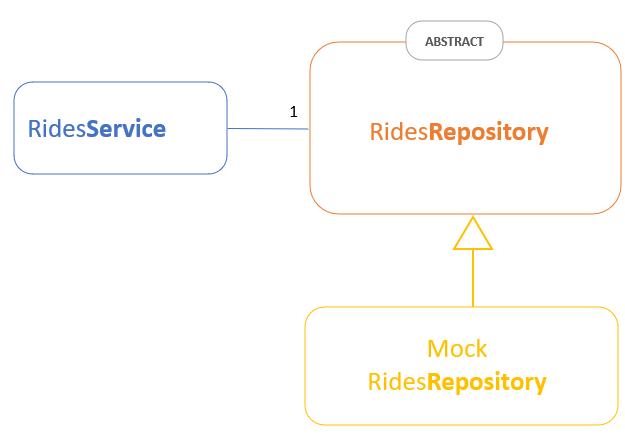
* Create a **LocationsRepository (**abstract), with the following API

List<Location> getLocations()

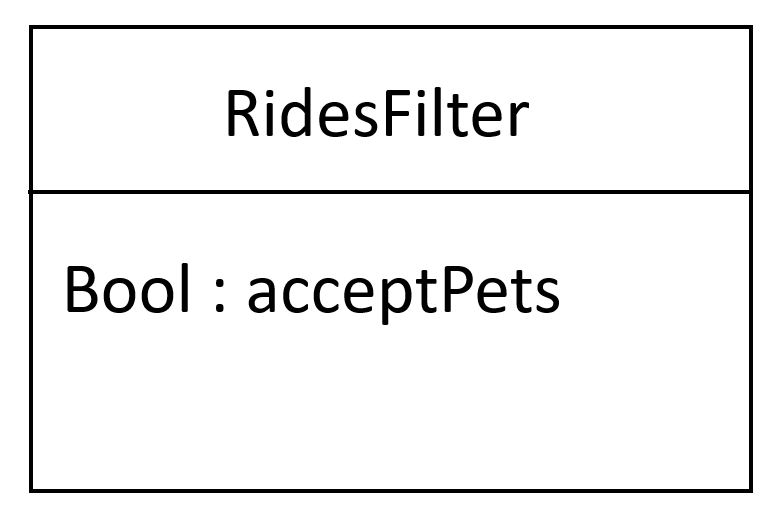
* Update the main code and the views, and **test your refactoring**:
  + The location picker shall display the locations related to the repository

**BLA-102** – Refactor Rides Service

The goal of this task is allowing the Rides Service to connect to different data source, without changing the service API.

**

* Create a class **RidesFilter** in the RidesService file:
  + This class aggregates the specific filters required by the user while researching for rides
  + For now, this class has only 1 attribute (pet accepted) but we will update it later on!



* Create a **RidesRepository** (abstract), with the following API

List<Ride> getRides (RidesPreference preference, RidesFilter? filter)

* Create a **mock implementation** of this repository (MockRidesRepository)
  + The mock repo shall define 5 rides:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Departure | Arrival | Date Dep | Duration | User | Accept Pets | Seats |
| Battambang | SiemReap | Today at 5.30 am | 2 hours | Kannika | No | 2 |
| Battambang | SiemReap | Today at 8 pm | 2 hours | Chaylim | No | 0 |
| Battambang | SiemReap | Today at 5 am | 3 hours | Mengtech | No | 1 |
| Battambang | SiemReap | Today at 8 pm | 2 hours | Limhao | Yes | 2 |
| Battambang | SiemReap | Today at 5 am | 3 hours | Sovanda | No | 1 |

*The mock repository should provide the above locations.*

* Finally update the **RidesService** using a singleton pattern.
  + Include an **initializer method** to specify the repository used by the single instance.
  + Update the service API:

List<Ride> getRides (RidesPreference preference, RidesFilter? filter)

**BLA-103 – BONUS** – Provide the ride with a sorting mode

For a better usage, enhance the service to sort the rides given a specific criterion (as example sorted by time of departure).

List<Ride> getRides (

RidesPreference preference,

RidesFilter? Filter,

RideSortType? sortType)

**BLA-104** – Test the RidesService on **console**

We want to assert the RidesServices works well, using the mock rides repository.

* In the /test folder, create a **rides\_service\_test.dart** with a main()
  + Initialize the RidesServices service with the MockRidesRepository
  + Write the following test and assert they are successful:

|  |  |  |
| --- | --- | --- |
| Test | Operation | Expected result |
| T1 | Create a ride preference   * from Battambang * to SieamReap * today * 1 passenger | Assert 4 results are displayed  Warning: 1 ride is full ! |
| T2 | Create a ride preference   * from Battambang * to SiemReap * today * 1 passenger   Create a ride filter   * pet allowed | 1 result displayed (Mengtech) |

**BONUS** : Write your test [using the Dart/test library](https://pub.dev/packages/test#writing-tests) ! ( test() expect() )

**Example of console output:** (*here the service provides the rides sorted from soonest to latest*)

For your preference (Battambang -> SiemReap, today 1 passenger) we founded 4 rides:

- at 5.00 am with Sovanda (3 hours)

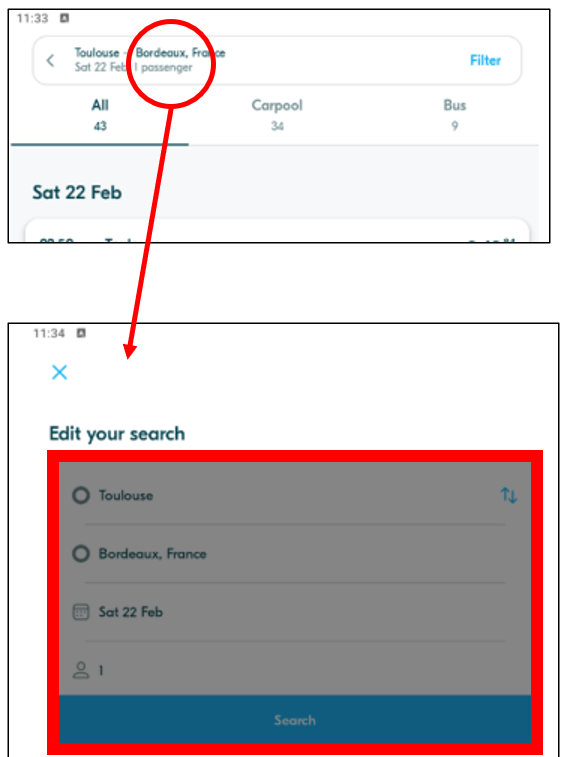
- at 5.00 am with Mengtech (3 hours)

- at 5.30 am with Kannika (2 hours)

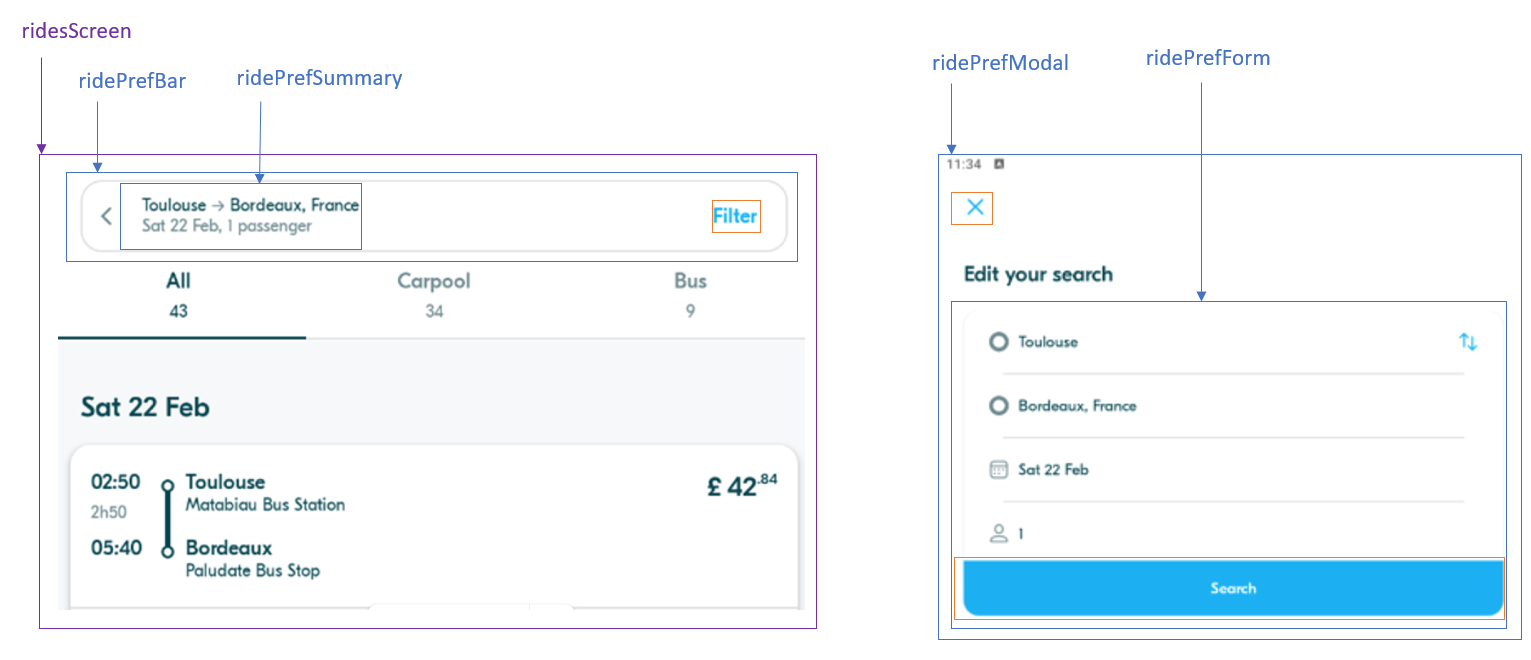
- at 8.00 pm with Limhao (2 hours)

**BLA-105** – Handle the **Preference edition** on Ride Screen

We want to edit the ride preferences directly form the rides screen (see bellow)

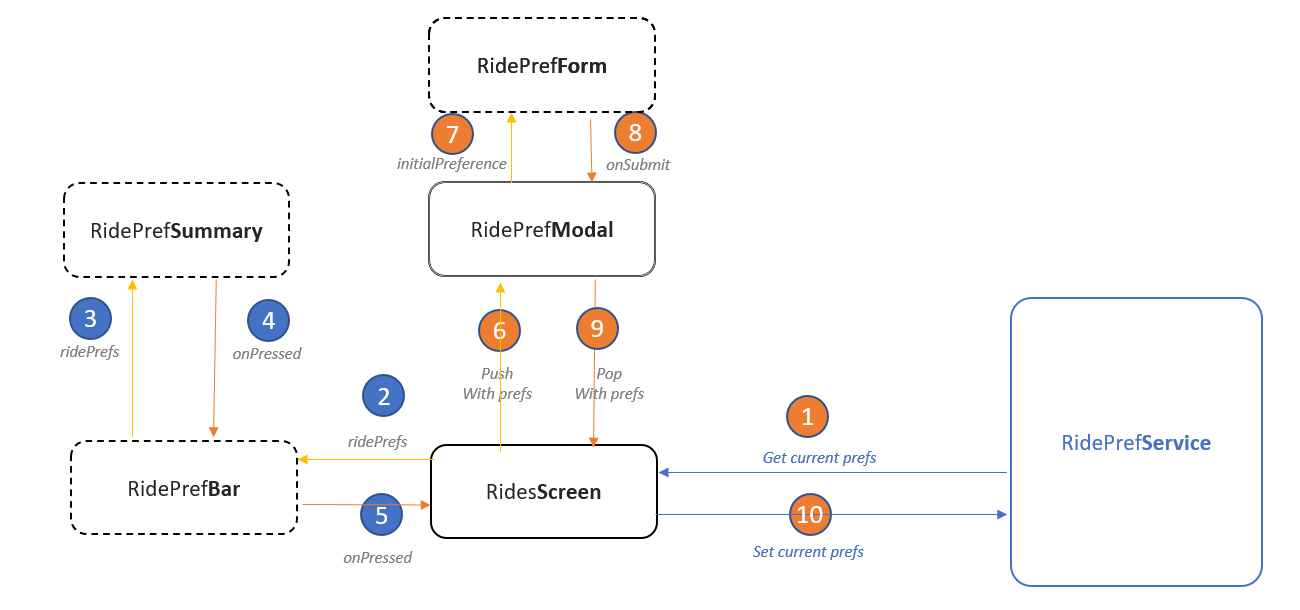


We want to re-use the **RidePreferencesForm** from the **RidePreferencesScreen.** But we need some adjustements, as this is NOT an embeded widget but this time it s a modal dialog:



To ensure a great workflow, it s important to be clear:

* None of the following widget will call the service, they will just pass data or callback : *ridePrefForm, ridePrefModal, ridePrefSummary, ridePrefBar*
* The RidesScreen will then **call the service to update the current pref**
* The RidesScreen will finally force the refresh
* The bellow blue points are already implemented.
* The orange point need to be done :



**Q1 -** Implement this workflow

**Q2 –** Perfrom the following tests:

|  |  |  |
| --- | --- | --- |
| Step | Operation | Expected result |
| 1 | *RidePref Screen*  Departure = Battambang  Arrival = SiemReap  Click on Search | The Rides screen is displayed  4 rides are visible |
| 2 | *Rides Screen*  Click on the ride pref summary on top bar | The RideRef Modal is displayed |
| 3 | *RideRef Modal*  Click on the switch icon ()  Click on Search | The Rides screen is displayed  The summary is updated (SiemReap to Battambang)  0 rides are visible |
| 4 | *Rides Screen*  Click on the BACK button () | The RidePref screen is displayed  The form displays SiemReap to Battambang |

**Q3 –** Did the STEP4 succeeded or failed?

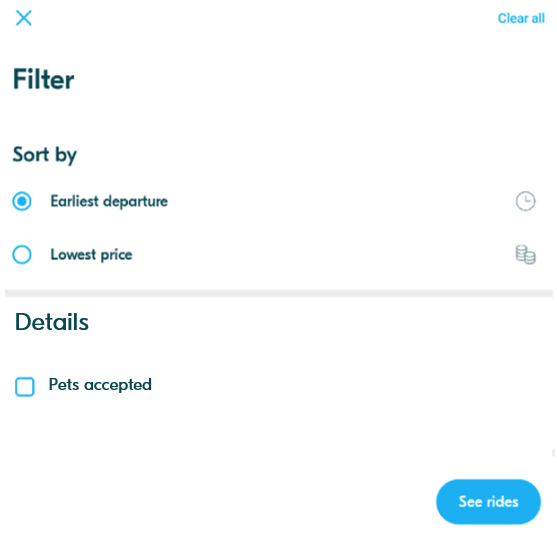
Explain your observation, by understanding **Flutter refresh mecanism.**

**BLA-106** - **BONUS** – Implement the **Filter view**

Implement the **filter view**.

For now, we **focus only on**:

* 1 filter: Pets accepted
* 2 sorts: sort by departure time, sort by price



**Q1 -**Analyze the filter form and complete the table with your widget strategy

|  |  |  |  |
| --- | --- | --- | --- |
| Widget | Screen / Screen Widget /App Widget | Parameters | Callbacks |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Q2 –**Who stores the filter settings? The view? The service? Explain your strategy