Mobile Application Architecture

Mobile Application Architecture

Introduction

Jumail Bin Taliba

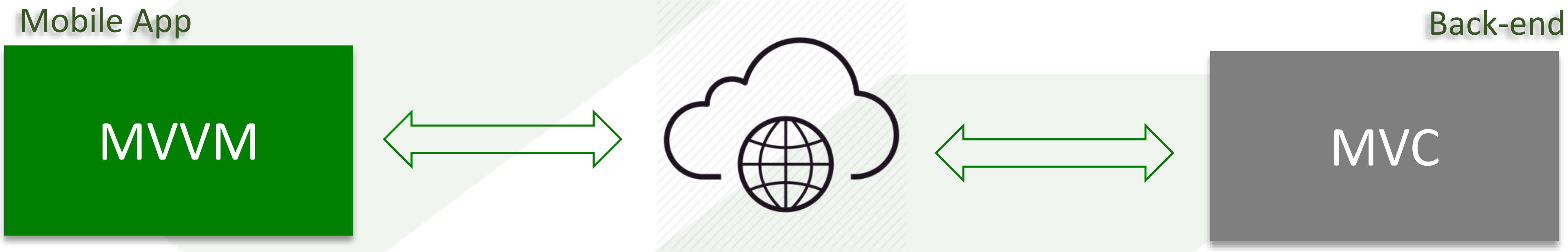
School of Computing, UTM

July 2020

The Plan

- Dependency Injection
- Provider
- MVVM Architectural Pattern

Preview on Architecture



Mobile App

MVVM

Back-end

MVC

- Views
- View Models
- Services
- Models

Mobile Application Architecture

Dependency Injection

Lecture and Demo

Jumail Bin Taliba
School of Computing, UTM

July 2020

Agenda

- Introduction
- Global DI
- Constructor DI
- Inherited Widget DI
- Service Locator DI

Introduction to Dependency Injection (DI)

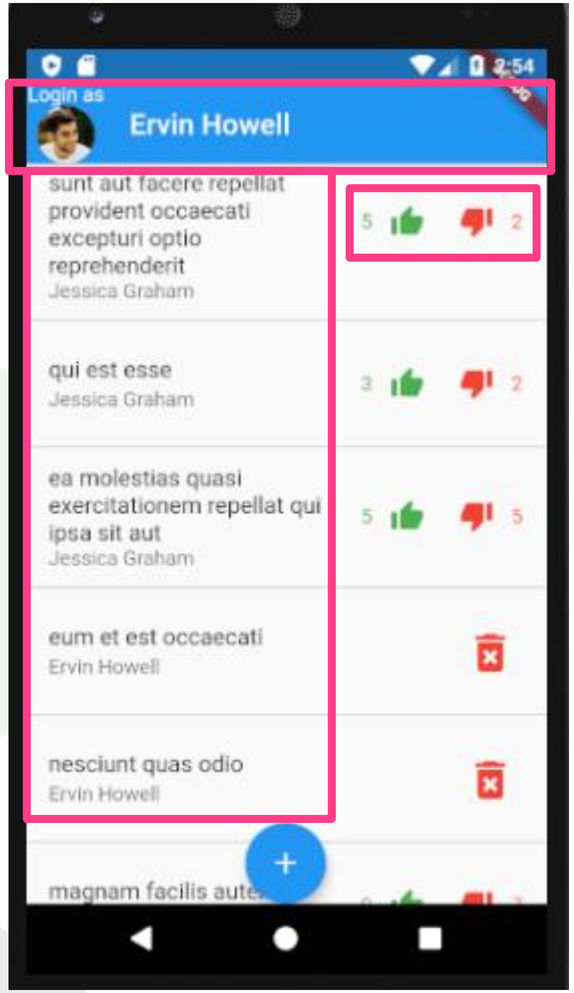
What is dependency?

- Code relies on other codes to accomplish its tasks.

What is dependency injection?

- How to supply or provide the dependencies to the requiring code.

- Several approaches:
  - Global
  - Constructor
  - Inherited widget
  - Service locator



Login as

Ervin Howell

sunt aut facere repellat provident occaecati excepturi optio reprehenderit
Jessica Graham

qui est esse
Jessica Graham

ea molestias quasi exercitationem repellat qui ipsa sit aut
Jessica Graham

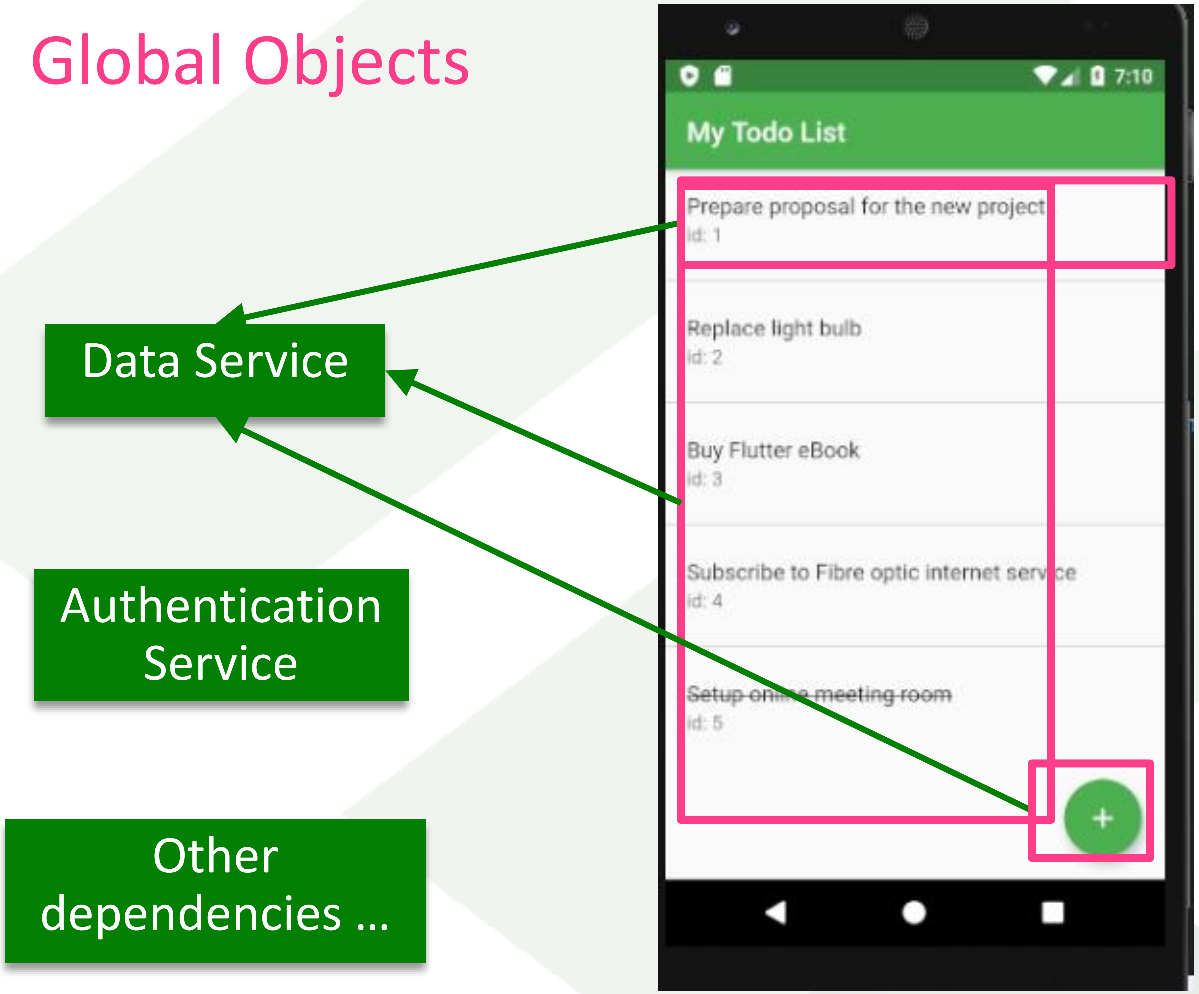
eum et est occaecati
Ervin Howell

nesciunt quas odio
Ervin Howell

magnam facilis aute

Global DI

Global Objects



Data Service

Authentication Service

Other dependencies ...

7:10

My Todo List

Prepare proposal for the new project
id: 1

Replace light bulb
id: 2

Buy Flutter eBook
id: 3

Subscribe to Fibre optic internet service
id: 4

~~Setup online meeting room~~
id: 5

+

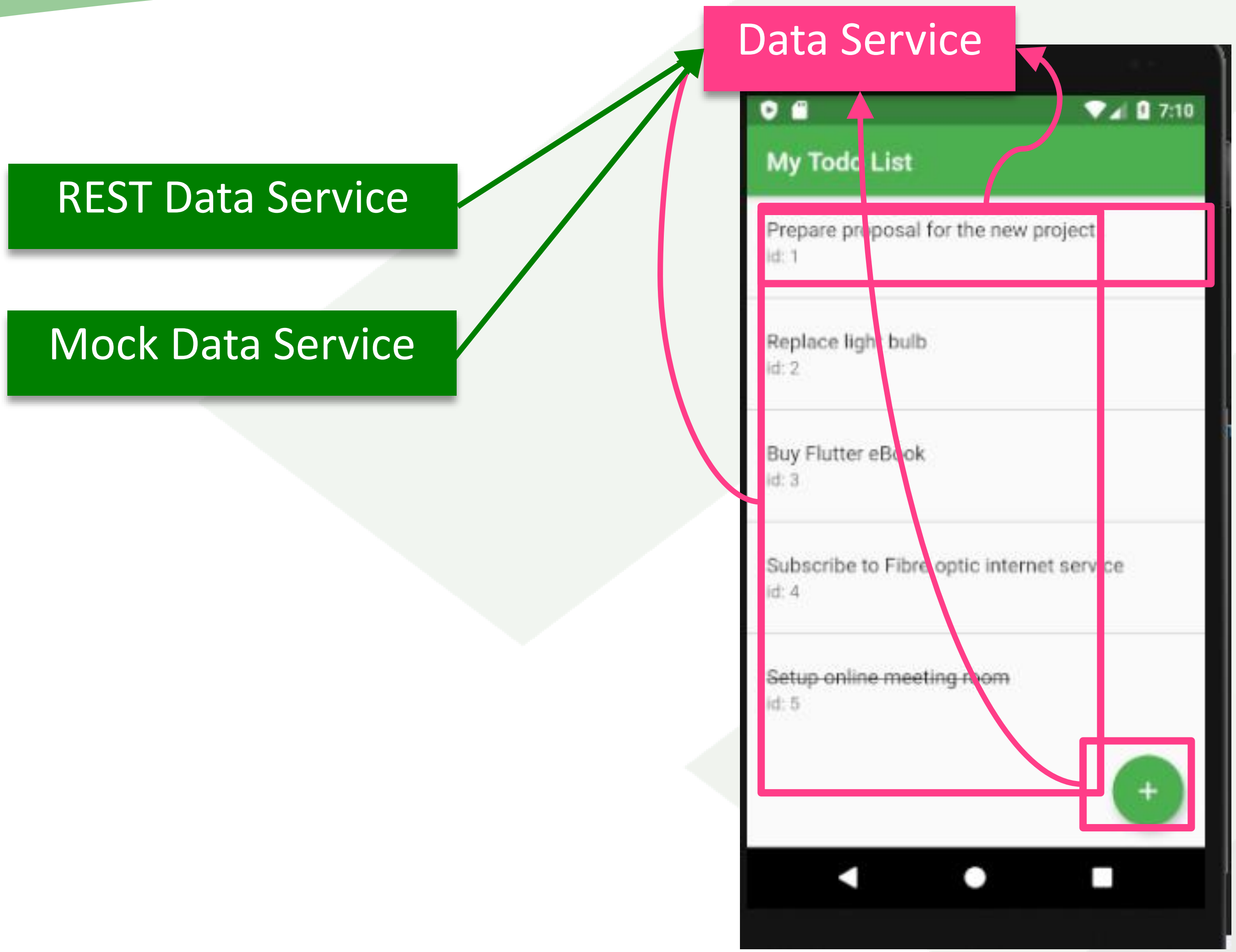
Demo

Global DI

Project Source Code

https://github.com/jumail-utm/architecture_dependency

Constructor DI



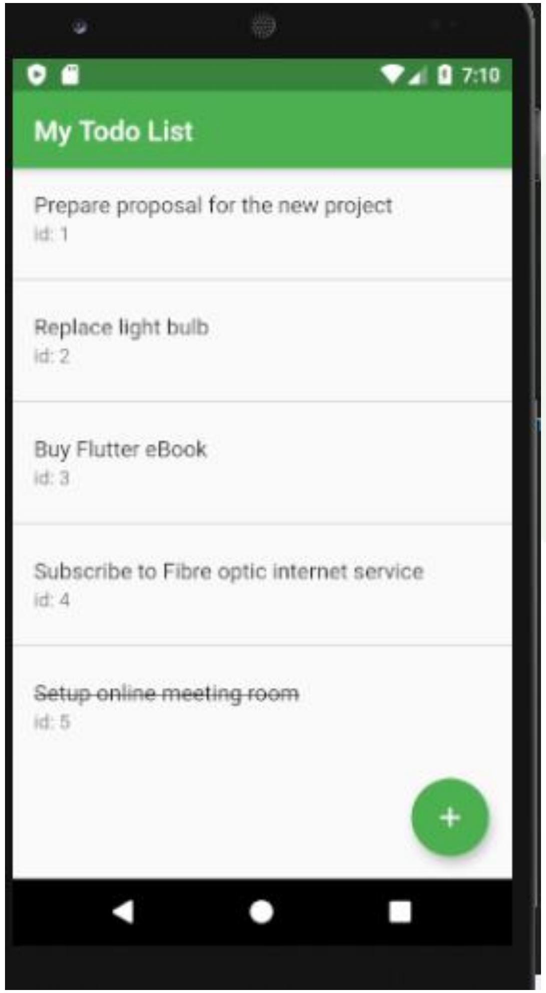
- If a child widget is built inside the parent widget, there should not be any problem accessing the dependencies.
- If a child widget is defined in different class from the parent widget, the parent needs to forward or pass the dependencies via the child's constructor.
- This approach is not suitable for depth widget trees.

Demo

Constructor DI

Inherited Widget DI

User Interface



My Todo List

Prepare proposal for the new project
id: 1

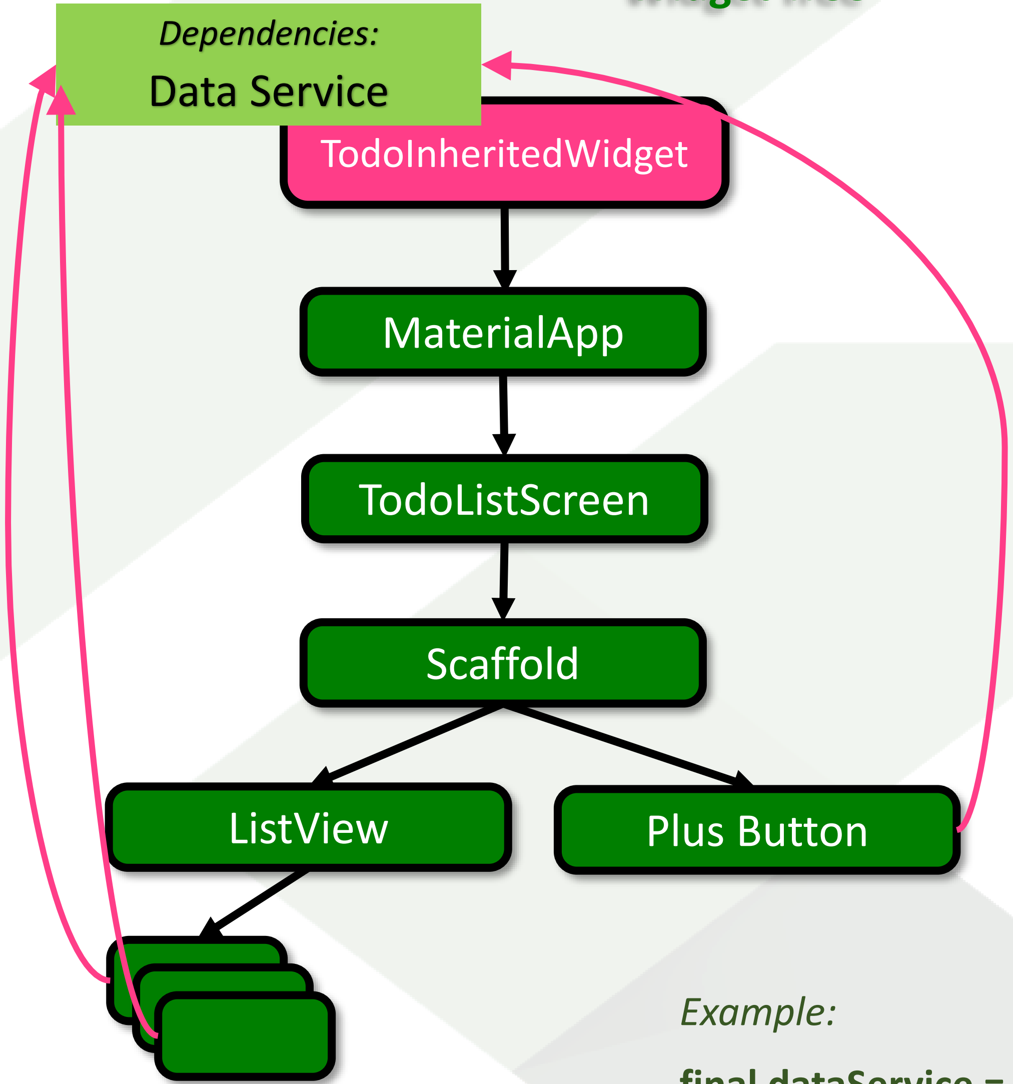
Replace light bulb
id: 2

Buy Flutter eBook
id: 3

Subscribe to Fibre optic internet service
id: 4

~~Setup online meeting room~~
id: 5

Widget Tree



- This approach solves the problem faced by constructor DI on a depth widget tree.
- All child widgets under the tree are able to access the top dependencies in the top widget (TodoInheritedWidget)
- A child widget accesses the top widget using the `of()` method.

Example:

final dataService = TodoInheritedWidget.of(context).dataService

Inherited Widget DI (2)

This approach is commonly used in Flutter:

- `Navigator.of(context).push(…)`
- `MediaQuery.of(context).orientation`
- `Theme.of(context).primaryColor`
- `Scaffold.of(context).openDrawer()`
- `Scaffold.of(context).showSnackBar(mySnackBbar)`

Demo

Inherited Widget DI

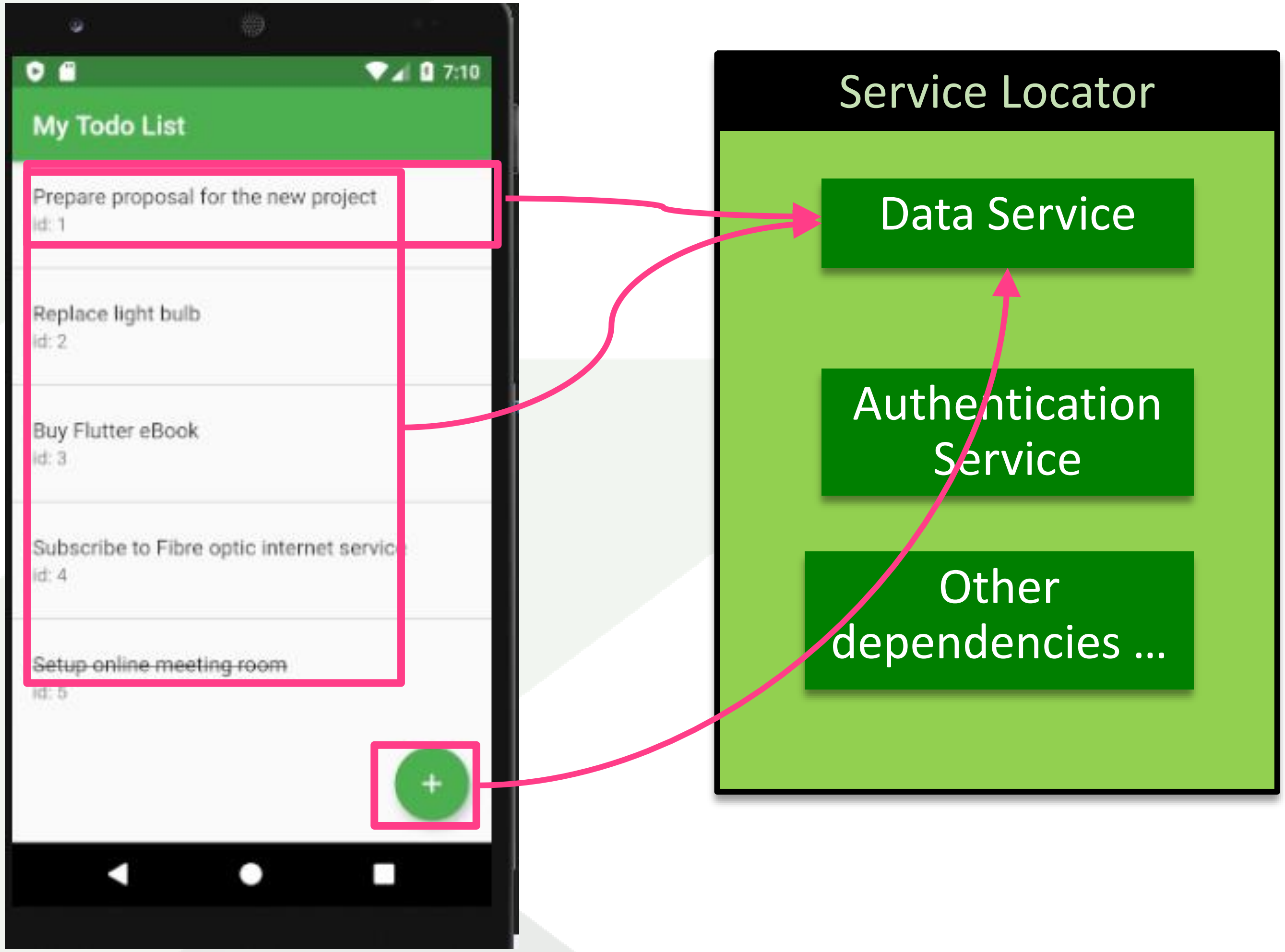
Service Locator DI

- A global approach
- All dependencies are held in a central registry called service locator
- Consumer code access the dependencies via the service locator

Example:

DataService dataService = serviceLocator();

- We'll use the get_it package
- Different ways of DI registration:
  - Factory – SL always gives a new instance
  - Singleton – SL gives the same instance
  - LazySingleton – Similar to Singleton except registration is not done immediately



7:10

My Todo List

Prepare proposal for the new project
id: 1

Replace light bulb
id: 2

Buy Flutter eBook
id: 3

Subscribe to Fibre optic internet servic
id: 4

~~Setup online meeting room~~
id: 5

+

Service Locator

Data Service

Authentication Service

Other dependencies …

Demo

Service Locator DI