📦 Camada `Data` – Descrição Completa dos Arquivos

📁 Estrutura

```
Data/
├── DataSources/
│   └── LocalTaskDataSource.swift
├── Repositories/
│   └── TaskRepositoryImpl.swift
```

📄 `Data/DataSources/LocalTaskDataSource.swift`

📌 Função

Simular uma fonte de dados **local**, como se fosse um "banco de dados em memória".

> Em um app real, essa classe seria trocada por `UserDefaults`, `Core Data`, `Realm` ou qualquer outro mecanismo de persistência.

🧱 Conteúdo

```swift
final class LocalTaskDataSource {
    private var tasks: [Task] = []

    func fetchTasks() -> [Task] {
        return tasks
    }

    func addTask(_ task: Task) {
        tasks.append(task)
    }

    func updateTask(_ task: Task) {
        guard let index = tasks.firstIndex(where: { $0.id == task.id }) else
{ return }
        tasks[index] = task
    }
```

```
    func deleteTask(_ task: Task) {
        tasks.removeAll { $0.id == task.id }
    }
}
```

🔗 Conexões

- Chamado por `TaskRepositoryImpl`
- Não conhece `UseCases` nem `ViewModel`
- Usa a `Entity` `Task` da camada Domain

📄 `Data/Repositories/TaskRepositoryImpl.swift`

📌 Função

Implementa o protocolo `TaskRepository` da camada `Domain`.

> É quem de fato realiza as ações nos dados, mas ainda **de forma abstrata** para os UseCases.

🧱 Conteúdo

```
final class TaskRepositoryImpl: TaskRepository {
    private let localDataSource: LocalTaskDataSource

    init(localDataSource: LocalTaskDataSource) {
        self.localDataSource = localDataSource
    }

    func fetchTasks() -> [Task] {
        return localDataSource.fetchTasks()
    }

    func addTask(_ task: Task) {
        localDataSource.addTask(task)
    }

    func updateTask(_ task: Task) {
        localDataSource.updateTask(task)
    }
```

```
    func deleteTask(_ task: Task) {
        localDataSource.deleteTask(task)
    }
}
```

🔗 Conexões

- Implementa `TaskRepository` (interface do domínio)
- Recebe uma instância de `LocalTaskDataSource` via injeção
- **É injetado nos UseCases** pela camada de DI
- Não conhece nada sobre SwiftUI ou ViewModel

🧭 Fluxo geral

```
SwiftUI View
   ↓
ViewModel
   ↓
UseCase
   ↓
TaskRepository (Protocolo)
   ↓
TaskRepositoryImpl (Implementação concreta)
   ↓
LocalTaskDataSource (Dados em memória)
```

✅ Resumo

Arquivo	Função	Conhece quem?
`LocalTaskDataSource.swift`	Simula uma fonte de dados (RAM)	Apenas `TaskRepositoryImpl`

Arquivo	Função	Conhece quem?
`TaskRepositoryImpl.swift`	Implementa `TaskRepository`, acessando os dados via `LocalTaskDataSource`	UseCases e `LocalTaskDataSource`