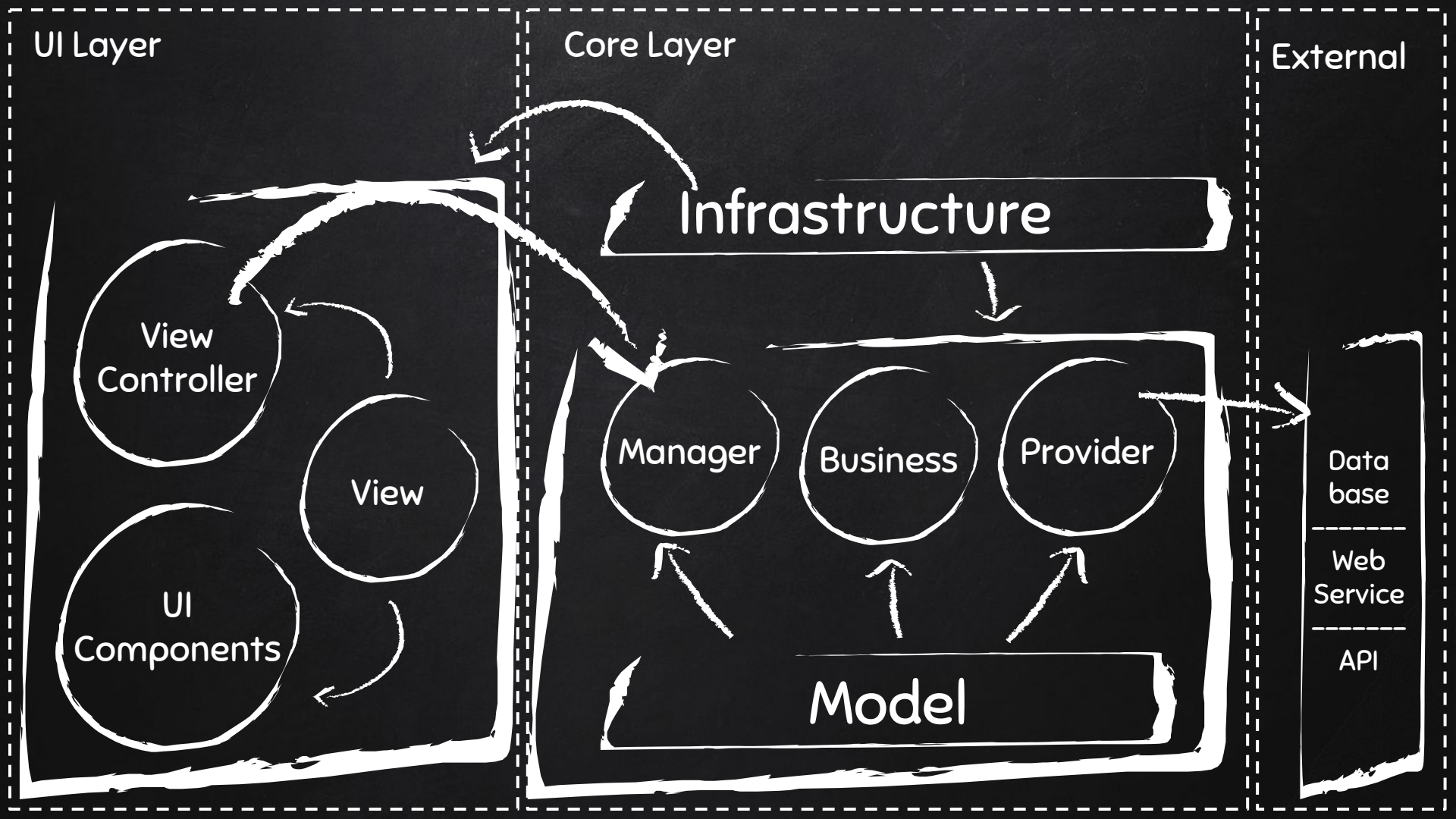
Proposta Arquitetural iOS

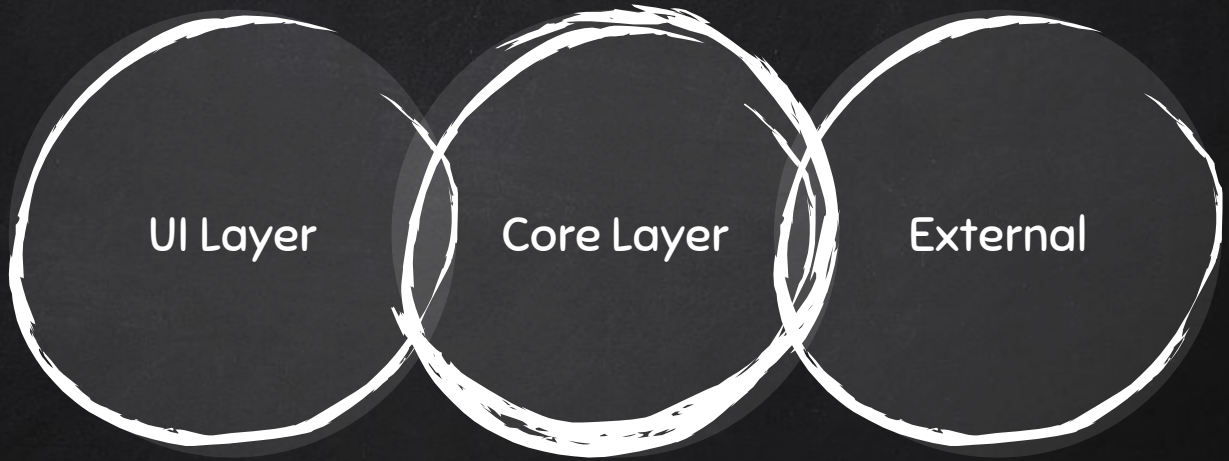
Marcelo Reis - 2017


UI Layer
Core Layer
External
Infrastructure
View Controller
View
UI Components
Manager
Business
Provider
Model
Data base
Web Service
API

O Conceito

Abaixo irei explicar como tudo funciona, acredito que após a leitura, a gráfico acima fará mais sentido

Colunas da arquitetura



UI Layer

A camada UI layer é o aplicativo propriamente dito, onde será desenvolvida as classes ViewController, View e os componentes UI.

View Controller

A camada View Controller atua como uma intermediária entre as camadas View e Manager. ViewControllers são, portanto, um canal responsável por notificar as Views sobre eventuais mudanças de estados dos modelos e vice-versa. Nessa camada também encontraremos as tratativas dos eventos, tais como: IBActions e Delegates / Datasources.

Obs.

1 – Acesso direto aos componentes UI devem ser evitados. Para isso devemos utilizar a camada View.

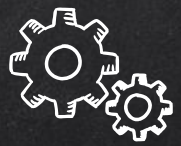
2 – Protocolos deverão ser implementados em extensions.

View

A camada View é o local onde os usuários poderão visualizar e iteragir com o app. Em outras palavras, uma View sabe se exibir e esta apta a responder às ações dos usuários.
Ela será responsável por toda parte de apresentação (UI), ou seja, ela é capaz de exibir animações, componentes customizados e/ou realizar qualquer formatação nos componentes UI quando necessário. Nela também estarão as Views das nossas ViewControllers e os seus IBOutlets.

UI Components

A camada UI Components tem como propósito centralizar e compartilhar componentes customizados e reaproveitáveis utilizados pelo app. Nessa camada podemos ter campos de texto, botões e/ou qualquer outro componente que precise ser customizado e reutilizado pelo app.



Core Layer

A camada Core layer é responsável por encapsular as regras de negócio e chamadas aos diversos provedores de informações.

Manager

A camada Manager deverá conter classes públicas responsáveis por controlar todo o fluxo de requisições entre a camada View Controller e Business. Ela deve ser exclusivamente acessada pela camada View Controller e seu retorno de dados sempre deverá ser realizado na thread principal.

Essa camada pode ser, se necessário, uma instância da classe OperationQueue para controlar operações assíncronas e poder facilitar o cancelamento ou pausa de uma operação. Ela também pode reestabelecer a relação de dependência entre as operações.

Obs. Utilizamos apenas um manager por view controller.

Business

A camada Business deverá conter estruturas internas responsáveis por concentrar todas as regras de negócio. Ela deve ser exclusivamente acessada pela camada Manager e invocar a camada Provider (quando necessário).

Essa camada também será responsável por receber "dados primitivos" (por ex. Dictionary) e converte-los para modelos antes de retorná-los para a camada Manager..
Por fim, (caso necessário) essa camada poderá invocar outros objetos de negócio.

Obs. Utilizamos apenas um business por manager.

Provider

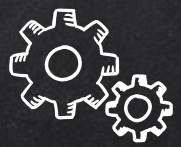
A camada Provider é responsável pela abstração das camadas da aplicação com acesso as fontes de dados e bibliotecas terceiras. Ela deve sempre retornar tipos de "dados primitivos" (por ex. Dictionary) e ser exclusivamente acessada pela camada Business.
Por ser baseada no design pattern Facade, que visa garantir a flexibilidade do código, devemos ter um provider por propósito com assinaturas de métodos genéricos. Dessa forma, caso seja necessário substituirmos uma biblioteca por outra, o impacto será mínimo. Afinal, as demais camadas permanecerão invocando o método com a mesma assinatura.
Outra vantagem dessa camada é o design pattern Strategy, onde podemos utilizar o provider para acessar recursos de mais de uma biblioteca e expormos um método genérico para camada acima (Business).

Model

A camada Model contém entidades que representam os objetos consumidos via serviço ou entidades do banco de dados. Nessa camada também encontraremos, enums, ErrorTypes e métodos para serialização de deserialização dos objetos.

Infrastructure

A camada Infrastructure Core é responsável por centralizar todas as constantes, mensagens e logs utilizados pelo app a fim de abstrair ao máximo todos esses mecanismos dessa camada.



External

Atualmente utilizamos diversas APIs e/ou bibliotecas de empresas e desenvolvedores terceiros. Esta camada representa todas essas libs utilizadas nos projetos.

Valeu!

Ficou alguma dúvida?

Fique a vontade para perguntar:

http://marcelo.cc
me@marcelo.cc